MC

Tłumaczenie standardów i rekomendacji
w zakresie cyberbezpieczeństwa

Zabezpieczenia sprzętowe:

Wprowadzenie do warstwowego podejścia do bezpieczeństwa platform dla zastosowań w przetwarzaniu chmurowym i brzegowym

NIST IR 8320_wer. 1.0_PL

NIST IR 8320_wer. 1.0_PL

11 marca 2025

Zabezpieczenia sprzętowe: *Wprowadzenie do warstwowego podejścia do bezpieczeństwa platform dla zastosowań w przetwarzaniu chmurowym i brzegowym*

Publikacja dostępna pod adresem:

Rekomendacje cyberbezpieczeństwa

NIST IR 8320

Hardware-Enabled Security:

Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases

Michael Bartock
Murugiah Souppaya
Ryan Savino
Tim Knoll
Uttam Shetty
Mourad Cherfaoui
Raghu Yeluri
Akash Malhotra
Don Banks
Michael Jordan
Dimitrios Pendarakis
J. R. Rao
Peter Romness
Karen Scarfone

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320

National Institute of Standards and Technology
U.S. Department of Commerce

NIST IR 8320

Hardware-Enabled Security:

Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases

Michael Bartock
Murugiah Souppaya *Computer Security Division*
Information Technology Laboratory

Ryan Savino Tim
Knoll Uttam
Shetty
Mourad Cherfaoui
Raghu Yeluri
Intel Data Platforms Group
Santa Clara, CA

Don Banks
Arm Architecture and Technology Group
San Jose, CA

Michael Jordan
Dimitrios Pendarakis
J. R. Rao
IBM Systems and IBM Research
Poughkeepsie and Yorktown Heights, NY

Peter Romness
Cisco
McLean, VA

Akash Malhotra
AMD Product Security and Strategy Group
Austin, TX

Karen Scarfone
Scarfone Cybersecurity
Clifton, VA

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8320

May 2022

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

O PUBLIKACJI

Niniejsze opracowanie **NIST IR 8320_wer. 1.0_PL**, ***Zabezpieczenia sprzętowe: Wprowadzenie do warstwowego podejścia do bezpieczeństwa platform dla zastosowań w przetwarzaniu chmurowym i brzegowym***, stanowi tłumaczenie publikacji NIST IR 8320, *Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud and Edge Computing Use Cases*, *i* opracowane zostało za zgodą National Institute of Science and Technology.

Przytaczane i cytowane w publikacji przepisy, okólniki, rozporządzenia wykonawcze, dyrektywy, normy, standardy, polityki, memoranda itp. odnoszą się, o ile nie zaznaczono inaczej, do prawodawstwa i rynku amerykańskiego. Jeżeli cytowany fragment ma przełożenie lub odpowiednik w polskim porządku prawnym lub normalizacyjnym, wówczas informacje te wskazane są bezpośrednio w tekście lub w przypisach.

W publikacji posłużono się pojęciami zdefiniowanymi w oryginalnej (angielskiej) wersji dokumentu, na podstawie którego powstały niniejsze zalecenia.

Tam, gdzie to było możliwe i nie budziło kontrowersji, nazwy ról i kluczowych uczestników procesu zarządzania ryzykiem zostały podane w języku polskim. Pozostałe role i funkcje zostały przedstawione w języku angielskim[1]. Do wszystkich tych ról / funkcji zastosowano akronimy terminologii angielskiej.

Terminologia angielska i akronimy występujące w publikacji zdefiniowane są w dokumencie ***Słownik kluczowych pojęć z zakresu cyberbezpieczeństwa*.**

Podmioty, urządzenia lub materiały o charakterze komercyjnym prezentowane są w niniejszym dokumencie w celu odpowiedniego opisania procedury lub koncepcji eksperymentalnej. Celem ich wskazania nie jest nakłanianie do korzystania z ww. podmiotów, urządzeń lub materiałów lub ich poparcie. Wskazanie ich nie ma również na celu sugerowania, że te podmioty, materiały lub sprzęt są najlepsze z dostępnych w danej dziedzinie. Taka identyfikacja nie stanowi rekomendacji, poparcia ani nie ma na celu sugerowania, że dane podmioty, materiały lub urządzenia są bezwzględnie najlepsze z dostępnych dla osiągnięcia danego celu.

[1] Kluczowi uczestnicy zarządzania ryzykiem – patrz: Narodowe Standardy Cyberbezpieczeństwa

SPIS TREŚCI

SPIS ILUSTRACJI

STRESZCZENIE

W dzisiejszych centrach danych w chmurze i w przetwarzaniu brzegowym, powierzchnie podatne na ataki uległy zmianie, a w niektórych przypadkach znacznie się zwiększyły. Jednocześnie hakerstwo stało się powszechne, a większość implementacji środków bezpieczeństwa nie jest spójna ani koherentna. Podstawą każdej strategii bezpieczeństwa centrum danych lub przetwarzania brzegowego powinno być zabezpieczenie platformy, na której dane i procesy będą wykonywane i dostępne. Platforma fizyczna stanowi pierwszą warstwę dla każdego warstwowego podejścia do bezpieczeństwa i zapewnia wstępną ochronę, która umożliwia zaufanie zabezpieczeniom wyższego poziomu.
W niniejszej publikacji przedstawiono techniki i technologie zabezpieczeń sprzętowych, które umożliwiają poprawę bezpieczeństwa platform i ochronę danych w centrach danych w chmurze i podczas przetwarzania brzegowego.

SŁOWA KLUCZOWE

przetwarzanie poufne (*ang. confidential computing*); kontener (*ang. container*); zabezpieczenia sprzętowe (*ang. hardware-enabled security*); sprzętowy moduł zabezpieczeń (*ang. hardware security module - HSM*); bezpieczna enklawa (*ang. secure enclave*); zaufane środowisko wykonawcze (*ang. trusted execution environment TEE*); moduł TPM (*ang. trusted platform module - TPM*); wirtualizacja (*ang. virtualization*).

ZASTRZEŻENIE

Wszelkie wzmianki o produktach komercyjnych lub odniesienia do organizacji komercyjnych mają charakter wyłącznie informacyjny. Nie oznaczają one rekomendacji ani poparcia z naszej strony T, ani też nie oznaczają, że wspomniane produkty są bezwzględnie najlepszymi dostępnymi produktami do danego celu

ODBIORCY

Głównymi odbiorcami tego raportu są specjaliści ds. bezpieczeństwa, tacy jak inżynierowie bezpieczeństwa i architekci; administratorzy systemów i inni specjaliści ds. technologii informatycznych dla dostawców usług w chmurze, a także deweloperzy sprzętu (*ang. hardware*), oprogramowania układowego (*ang. firmware*) i oprogramowania (*ang. software)*, którzy mogą być w stanie wykorzystać sprzętowe

techniki i technologie bezpieczeństwa w celu poprawy bezpieczeństwa platform dla centrów danych w chmurze i przetwarzania brzegowego.

INFORMACJA O ZNAKACH TOWAROWYCH

Wszystkie zastrzeżone znaki towarowe lub znaki towarowe należą do odpowiednich organizacji.

INFORMACJA O UJAWNIENIU PATENTÓW

UWAGA: Laboratorium informatyczne (ang. Information Technology Laboratory – ITL) zwróciło się do właścicieli zastrzeżeń patentowych, których wykorzystanie może być konieczne w celu zapewnienia zgodności z wytycznymi lub wymogami niniejszej publikacji, o ujawnienie ITL takich zastrzeżeń patentowych. Jednak posiadacze patentów nie są zobowiązani do odpowiadania na wezwania ITL dotyczące patentów, a ITL nie podjęło się wyszukiwania patentów w celu zidentyfikowania, które z nich, jeśli w ogóle, mogą mieć zastosowanie w niniejszej publikacji.

Na dzień publikacji i po wystosowaniu wezwania (wezwań) do wskazania zastrzeżeń patentowych, których wykorzystanie może być konieczne w celu zapewnienia zgodności z rekomendacjami lub wskazówkami niniejszej publikacji, ITL nie zidentyfikowało żadnych takich zastrzeżeń patentowych.

ITL nie gwarantuje ani nie sugeruje, że w celu uniknięcia naruszenia patentów przy korzystaniu z niniejszej publikacji nie są konieczne licencje.

1. WPROWADZENIE

W centrach danych i przetwarzaniu brzegowym istnieją trzy istotne czynniki, które wpływają na bezpieczeństwo: (1) wprowadzenie miliardów podłączonych urządzeń i szersze wykorzystanie chmur obliczeniowych znacznie zwiększyło powierzchnie podatne na ataki; (2) hakerstwo stało się zjawiskiem powszechnym dzięki zaawansowanym i ewoluującym technikom naruszania bezpieczeństwa danych; oraz (3) rozwiązania składające się z wielu technologii pochodzących od różnych dostawców skutkują brakiem spójnych i konsekwentnych implementacji środków bezpieczeństwa. Z uwagi na te czynniki, podstawą strategii bezpieczeństwa centrum danych lub przetwarzania brzegowego powinno być skonsolidowane podejście do kompleksowego zabezpieczania całych systemów, w tym platform sprzętowych, na których wykonywane są procesy i do których uzyskiwany jest dostęp.

W niniejszym dokumencie *platformą sprzętową* jest serwer (np. serwer aplikacji, serwer pamięci masowej, serwer wirtualizacji) w centrum danych lub w ośrodku przetwarzania brzegowego. Platforma sprzętowa serwera, zwana również *platformą serwera*, stanowi pierwszą część warstwowego modelu bezpieczeństwa. *Zabezpieczenia sprzętowe* – zabezpieczenia oparte na platformie sprzętowej – mogą stanowić solidniejszą podstawę niż zabezpieczenia zapewniane przez oprogramowanie lub oprogramowanie układowe, które mają większą powierzchnię podatną na ataki i mogą być stosunkowo łatwo modyfikowane. Sprzętowe źródło zaufania (*ang. Root of Trust – RoT*) może stanowić mniejszą powierzchnię podatną na ataki, jeśli zostanie zaimplementowane przy użyciu małej bazy kodu. Istniejące implementacje zabezpieczeń mogą zostać ulepszone poprzez zapewnienie warstwy bazowej, stabilnego i niezmiennego modułu sprzętowego, który łączy weryfikacje oprogramowania i oprogramowania układowego począwszy od sprzętu aż do przestrzeni aplikacji lub określonego środka bezpieczeństwa. W ten sposób można jeszcze bardziej zaufać istniejącym mechanizmom bezpieczeństwa, aby osiągnąć swoje cele w zakresie bezpieczeństwa bez kompromitacji, nawet w przypadku braku zabezpieczeń fizycznych lub ataków pochodzących z warstwy oprogramowania.

W niniejszym dokumencie omówiono sprzętowe techniki i technologie zabezpieczeń, które umożliwiają poprawę bezpieczeństwa platformy serwera i ochronę danych w centrach danych w chmurze i w podczas przetwarzania brzegowego. Pozostała część niniejszej publikacji obejmuje następujące tematy:

- W rozdziale 2 przedstawiono przegląd zabezpieczeń platformy sprzętowej.
- W rozdziale 3 omówiono pomiary i weryfikację integralności platformy.
- W rozdziale 4 omówiono ataki na środowisko uruchomieniowe oprogramowania i mechanizmy ochrony przed nimi.
- W rozdziale 5 omówiono ochronę danych w użyciu, znaną również jako przetwarzanie poufne.
- W rozdziale 6 przeanalizowano usługi wystawiania atestacji zdalnego, które umożliwiają zestawianie pomiarów integralności platformy w celu ułatwienia weryfikacji integralności.
- W rozdziale 7 opisano szereg scenariuszy wykorzystania chmury obliczeniowej, w ramach których zastosowano zabezpieczenia sprzętowe.
- W rozdziale 8 określono kolejne działania z oraz sposób, w jaki inne podmioty mogą wnieść swój wkład.
- W rozdziale „Referencje” znajduje się lista źródeł cytowanych w niniejszym raporcie.
- W załączniku A opisano przykłady technologii niezależnych od dostawcy.
- W załącznikach od B do F opisano przykłady technologii, odpowiednio, firm Intel, *AMD*, Arm, Cisco i *IBM*.
- W załączniku G znajduje się lista akronimów i skrótów użytych w raporcie.
- W załączniku H znajduje się słownik wybranych terminów użytych w publikacji.

W miarę rozwoju technologii i możliwości w zakresie bezpieczeństwa, stale poszukiwane są opinie społeczności na temat poruszany w publikacji i oczekiwane są dodatkowe przykłady technologii od innych firm.

Niniejszy dokument nie odnosi się do innych platform, takich jak laptopy, komputery stacjonarne, urządzenia mobilne lub urządzenia w ramach Internetu rzeczy (*ang. Internet of Things – IoT*), jednak praktyki zawarte w tym dokumencie można dostosować do tych platform i związanych z nimi zastosowań.

2. PRZEGLĄD ZABEZPIECZEŃ PLATFORMY SPRZĘTOWEJ

Zagrożenia dotyczące centrów danych ewoluowały w ostatnich latach, wraz z powstaniem bardziej zaawansowanych powierzchni podatnych na atak i bardziej trwałych mechanizmów ataków. Wraz ze wzrostem uwagi poświęcanej wysokiemu poziomowi bezpieczeństwa oprogramowania, atakujący schodzą coraz niżej w stosie platformy, zmuszając administratorów bezpieczeństwa do zajmowania się różnymi atakami, które zagrażają oprogramowaniu układowemu i sprzętowi platformy. Zagrożenia te mogą powodować:

- Nieautoryzowany dostęp i potencjalnie zdobycie wrażliwych danych platformy lub użytkowników, w tym bezpośredni fizyczny dostęp do modułów pamięci DIMM (*ang. Dual In-Line Memory Module – DIMM*).
- Modyfikację oprogramowania układowego platformy, np. interfejsu UEFI (*ang. Unified Extensible Firmware Interface – UEFI*) / podstawowego systemu wejścia/wyjścia (*ang. Basic Input/Output System – BIOS*), kontrolera BMC (*ang. Board Management Controller – BMC*), układu ME (*ang. Manageability Engine – ME*), połączenia PCI Express (*ang. Peripheral Component Interconnect Express – PCIe*) i różnych kart akceleratorów.
- Przechwytywanie łańcucha dostaw poprzez fizyczną zamianę oprogramowania układowego lub sprzętu na złośliwe wersje.
- Dostęp do danych lub wykonywanie kodu poza granicami geopolitycznymi lub innymi granicami podlegającymi regulacjom prawnym.
- Obejście mechanizmów bezpieczeństwa opartych na oprogramowaniu lub oprogramowaniu układowym.

Na przykład wirus LoJax, odkryty w sierpniu 2018 r., jest złośliwym programem atakującym interfejs UEFI, co umożliwia mu ciągłe utrzymywanie się w warstwie oprogramowania układowego pomimo ponownych instalacji systemu operacyjnego, a tym samym pozostaje niewidoczny dla standardowych skanerów antywirusowych opartych na jądrze systemu [1]. Ataki te mogą być niszczycielskie dla

środowisk opartych na chmurze, ponieważ często wymagają przebudowy lub wymiany serwera po serwerze, co może trwać tygodniami. Ataki te, choć wciąż nie są nagminne, stają się coraz częstsze, ponieważ atakujący stają się coraz bardziej wyrafinowani.

Obciążenia (*ang. workloads*)[2] podlegające określonym regulacjom lub zawierające wrażliwe dane stanowią dodatkowe wyzwania w zakresie bezpieczeństwa w przypadku chmur obsługujących wielu użytkowników. Wirtualizacja i kontenery znacznie zwiększają wydajność, zdolność adaptacji i skalowalność, jednak technologie te powodują koncentrację obciążeń na mniejszej liczbie platform fizycznych i wprowadzają dynamiczną migrację obciążeń i danych między tymi platformami. Dlatego przyjęcie rozwiązania opartego na chmurze skutkuje utratą przez użytkowników widoczności i sterowania platformami obsługującymi zwirtualizowane obciążenia i dane, a także wprowadza konieczność korzystania z zewnętrznych administratorów infrastruktury. Dostawcy i użytkownicy chmury stosują model współodpowiedzialności, w którym każda ze stron odpowiada za różne aspekty ogólnej implementacji. Dostawcy usług w chmurze mogą udostępniać informacje związane z bezpieczeństwem infrastruktury i możliwościami platformy, aby zapewnić swoim najemcom gwarancje bezpieczeństwa. Co więcej, dostawcy usług w chmurze często mają centra danych, które przekraczają wiele granic geopolitycznych, narażając właścicieli obciążeń na konieczność zapewnienia zgodności ze skomplikowanymi przepisami ustawowymi i wykonawczymi obowiązującymi w wielu krajach. Dzieje się tak zwłaszcza w przypadku architektury chmury hybrydowej, w której wykorzystywanych jest wielu dostawców infrastruktury, a każdy z nich ma własną konfigurację infrastruktury i sposób zarządzania nią.

Bez fizycznej kontroli nad funkcjami poufnego przetwarzania lub korzystania z nich, a także bez wglądu w konfiguracje platformy, wdrożenie konwencjonalnych najlepszych praktyk w zakresie bezpieczeństwa i wymogów regulacyjnych staje się

[2] Obciążenia to procesy obliczeniowe, które działają w różnych środowiskach i realizują określone zadania. Obciążenia działają zarówno na serwerach fizycznych, jak i wirtualnych i mogą dynamicznie przenosić się między środowiskami w zależności od potrzeb obliczeniowych.

trudne lub niemożliwe. Ze względu na przepisy wprowadzające wysokie kary za ich nieprzestrzeganie, wiedza o tym, gdzie można uzyskać dostęp do danych i kontrola nad tym, jest ważniejsza niż kiedykolwiek wcześniej. Główne wyzwania stojące przed specjalistami ds. bezpieczeństwa obejmują ochronę obciążeń przed ogólnymi zagrożeniami dla bezpieczeństwa, ryzyko utraty lub ujawnienia danych w przypadku ich naruszenia oraz zachowanie zgodności z przepisami.

Istniejące środki łagodzące zagrożenia dla serwerów w chmurze są często oparte na oprogramowaniu układowym lub oprogramowaniu (aplikacjach), co czyni je podatnymi na te same strategie ataku. Na przykład, jeśli podatności oprogramowania układowego można skutecznie wykorzystać, środki bezpieczeństwa oparte na tym oprogramowaniu można najprawdopodobniej obejść w ten sam sposób. Techniki zabezpieczeń oparte na sprzęcie mogą pomóc w ograniczeniu tych zagrożeń poprzez ustanowienie i utrzymanie zaufania do platformy – gwarancji integralności podstawowej konfiguracji platformy, w tym sprzętu, oprogramowania układowego i oprogramowania. Zapewniając taką gwarancję, administratorzy bezpieczeństwa mogą uzyskać poziom widoczności i kontroli nad tym, gdzie dozwolony jest dostęp do wrażliwych obciążeń i danych. Technologie zabezpieczające platformy, które ustanawiają zaufanie do niej, mogą wysyłać powiadomienia o wykrytych błędach integralności, a nawet samodzielnie je korygować. Konfiguracje platformy mogą być automatycznie przywracane do zaufanego stanu, zapewniając platformie odporność na ataki.

Aby osiągnąć niezbędne środki bezpieczeństwa, jako punkt wyjścia można wykorzystać sprzętowe źródło zaufania (*ang. Root of Trust – RoT*), które jest domyślnie zaufane. Sprzętowe środki bezpieczeństwa mogą stanowić podstawę do zapewnienia integralności platformy. Połączenie tych funkcji z możliwością tworzenia weryfikowalnych dowodów na to, że mechanizmy kontroli integralności zostały wdrożone i pomyślnie użyte, stanowi podstawę do stworzenia zaufanej platformy. Minimalizacja zakresu RoT oznacza zmniejszenie liczby modułów lub technologii, które muszą być domyślnie zaufane. Znacznie zmniejsza to powierzchnię podatną na atak.

Platformy, w których zabezpieczono bazowe oprogramowanie układowe i konfigurację, zapewniają możliwość rozszerzenia zaufania na wyższe poziomy stosu oprogramowania. Zweryfikowane oprogramowanie układowe platformy może z kolei zweryfikować moduł ładujący rozruch systemu operacyjnego, który może następnie sprawdzić inne komponenty oprogramowania, aż do samego systemu operacyjnego i warstw środowiska uruchomieniowego funkcji hypervisor lub kontenera. Opisane tutaj zaufanie przechodnie jest zgodne z koncepcją łańcucha zaufania (*ang. Chain of Trust – CoT*) – metodą, w której każdy moduł oprogramowania w procesie uruchamiania systemu musi dokonać sprawdzenia następnego modułu przed przekazaniem sterowania.

Oparcie integralności platformy i zaufania na sprzętowych środkach bezpieczeństwa może wzmocnić i uzupełnić rozszerzenie koncepcji CoT na kategorię dynamicznego oprogramowania. Tam koncepcję CoT można rozszerzyć jeszcze bardziej, aby uwzględnić ochronę danych i obciążeń. Zabezpieczenia sprzętowe oparte na mechanizmach technologii CoT mogą stanowić warstwową strategię bezpieczeństwa do ochrony danych i obciążeń, gdy są one przenoszone do środowisk dostępnych dla wielu użytkowników w centrum danych w chmurze lub w obiekcie, w którym odbywa się przetwarzanie brzegowe.

Ponadto istnieją inne technologie zabezpieczające platformy sprzętowe, które mogą chronić dane w spoczynku, podczas przesyłania i użytkowania, zapewniając akcelerowane sprzętowo szyfrowanie dysków z lub izolację pamięci opartą na szyfrowaniu. Wiele z tych funkcji może pomóc w ograniczeniu zagrożeń związanych z wykonywaniem spekulatywnym[3] i atakami typu side-channel[4]. Wykorzystanie sprzętu

[3] Wykonywaniem spekulatywne - zdolność mikroprocesorów, przetwarzających potokowo instrukcje maszynowe programu, do wykonywania instrukcji znajdujących się już po skoku warunkowym, co do którego jeszcze nie wiadomo, czy nastąpi, a więc czy (formalnie) kolejne instrukcje zostaną kiedykolwiek wykonane.

[4] Atak side-channel to dowolny atak oparty na dodatkowych informacjach, które można zebrać ze względu na fundamentalny sposób implementacji protokołu lub algorytmu komputerowego, a nie wady w projekcie samego protokołu lub algorytmu (np. wady wykryte w kryptoanalizie algorytmu kryptograficznego) lub drobne, ale potencjalnie niszczycielskie błędy lub niedopatrzenia w implementacji.

do wykonania tych zadań zmniejsza powierzchnię podatną na atak, uniemożliwiając bezpośredni dostęp do oprogramowania układowego lub jego modyfikację. Odseparowanie tych mechanizmów szyfrowania w dedykowanym sprzęcie może pozwolić na poprawę wydajności oraz oddzielenie od innych procesów systemowych. Przykład izolacji sprzętowej został omówiony w dalszej części dokumentu.

3. WERYFIKACJA INTEGRALNOŚCI PLATFORMY

Kluczową koncepcją w ramach zaufanego przetwarzania danych jest weryfikacja integralności służącej do tego platformy. Ocena integralności platformy składa się zazwyczaj z dwóch części:

- **Kryptograficzna sygnatura oprogramowania i oprogramowania układowego.** W niniejszym raporcie termin *sygnatura* odnosi się do obliczania kryptograficznego skrótu pliku (*ang. hash*) wykonywalnego oprogramowania lub oprogramowania układowego, pliku konfiguracyjnego lub innego obiektu. W przypadku jakiejkolwiek zmiany w obiekcie, nowa sygnatura wygeneruje inną wartość skrótu niż pierwotna [2]. Dzięki obliczeniu sygnatury oprogramowania i oprogramowania układowego przed wykonaniem, integralność zmierzonych modułów i konfiguracji może zostać zweryfikowana przed uruchomieniem platformy albo przed uzyskaniem dostępu do danych lub obciążeń. Sygnatury te mogą również działać jako kryptograficzny dowód w ramach audytów zgodności.
- **Weryfikacja oprogramowania układowego i konfiguracji.** Po obliczeniu *sygnatury* oprogramowania układowego i konfiguracji można przeprowadzić lokalne lub zdalne procesy atestowania w celu sprawdzenia, czy żądane oprogramowanie układowe jest rzeczywiście uruchomione i czy konfiguracje zostały autoryzowane [3]. Proces atestowania może również służyć jako podstawa dla dalszych decyzji dotyczących polityk, które będą odpowiadać różnym implementacjom zabezpieczeń w chmurze. Przykładowo, klucze szyfrujące mogą być udostępniane obciążeniom klienta, jeśli zostanie przeprowadzony dowód, że platforma jest zaufana i zgodna z obowiązującymi politykami.

W niektórych przypadkach do oceny integralności platformy dodawana jest trzecia część:

- **Odzyskiwanie oprogramowania układowego i konfiguracji.** Jeśli etap weryfikacji zakończy się niepowodzeniem (tj. w ramach procesu atestacji nie dopasowano odpowiednich sygnatur), oprogramowanie układowe i konfiguracja mogą zostać automatycznie przywrócone do znanego

prawidłowego stanu, np. poprzez przywrócenie oprogramowania układowego do zaufanej wersji. Proces wdrażania tych technik wpływa na ogólną siłę zapewnienia, że zmierzone i zweryfikowane komponenty nie zostały przypadkowo zmienione lub złośliwie zmodyfikowane. Technologie odzyskiwania umożliwiają zachowanie odporności platform na ataki związane z oprogramowaniem układowym oraz na przypadkowe błędy udostępniania [4].

Istnieje wiele sposobów obliczania sygnatur integralności platformy. Większość technologii opiera się na wspomnianej wcześniej koncepcji CoT. W wielu przypadkach sprzętowy moduł zabezpieczeń jest wykorzystywany do przechowywania *sygnatur*, które zostaną wykorzystane w procesie atestacji w późniejszym czasie. W pozostałej części niniejszego rozdziału omówiono sprzętowe moduły zabezpieczeń i różne implementacje technologii CoT.

3.1. SPRZĘTOWY MODUŁ ZABEZPIECZEŃ (*HARDWARE SECURITY MODULE – HSM*)

Sprzętowy moduł zabezpieczeń (*ang. Hardware Security Module – HSM*) to „fizyczne urządzenie komputerowe, które zabezpiecza klucze kryptograficzne i zarządza nimi oraz obsługuje przetwarzanie danych kryptograficznych” [5]. Operacje kryptograficzne, takie jak: szyfrowanie, deszyfrowanie i generowanie/weryfikacja podpisów, są zwykle wykonywane na urządzeniu HSM, a wiele implementacji zapewnia sprzętowe mechanizmy akceleracji operacji kryptograficznych.

Moduł TPM (*ang. Trusted Platform Module – TPM*) jest specjalnym rodzajem urządzenia HSM, które może generować klucze kryptograficzne i chronić niewielkie ilości poufnych informacji, takich jak hasła, klucze kryptograficzne i sygnatury skrótów kryptograficznych [3] . Moduł TPM jest autonomicznym urządzeniem, które można zintegrować z platformami serwerów, urządzeniami klienckimi i innymi produktami. Jednym z głównych zastosowań modułu TPM jest przechowywanie sygnatur skrótu oprogramowania układowego i konfiguracji platformy podczas procesu rozruchu. Sygnatura każdego modułu oprogramowania układowego polega na wygenerowaniu skrótu, który jest następnie rozszerzany do rejestru konfiguracji platformy (*ang. Platform Configuration Register – PCR*) modułu TPM. Wiele modułów oprogramowania

układowego można rozszerzyć do tego samego rejestru PCR, a wytyczne dotyczące tego, które sygnatury oprogramowania układowego są objęte danym rejestrem PCR, zawarte są w specyfikacjach dla poszczególnych platform, takich jak *PC Client Firmware Profile* organizacji Trusted Computing Group (TCG) [6].

Moduły TPM obsługują również funkcje generowania kluczy powiązania i podpisywania, które są unikalne dla każdego z nich i przechowywane w nieulotnej pamięci o dostępie swobodnym (*ang. Non-Volatile Random-Access Memory – NVRAM*) modułu TPM. Prywatna część tej pary kluczy jest odszyfrowywana wewnątrz modułu TPM, dzięki czemu jest dostępna tylko dla samego urządzenia lub jego oprogramowania układowego. Może to stworzyć unikalną relację między kluczami wygenerowanymi w module TPM a systemem platformy, ograniczając operacje dotyczące kluczy prywatnych do oprogramowania układowego platformy, i ma dostęp do określonego modułu TPM. Klucze powiązania są używane do szyfrowania/ deszyfrowania danych, natomiast klucze podpisywania — do generowania/ weryfikacji podpisów kryptograficznych. Moduł TPM zapewnia generator liczb losowych (*ang. Random Number Generator – RNG*) jako chronioną funkcję bez kontroli dostępu. Generator liczb losowych jest wykorzystywany w krytycznych funkcjach kryptograficznych jako źródło entropii dla identyfikatorów jednorazowych, do generowania kluczy i zapewnienia losowości w podpisach [6].

Istnieją dwie wersje modułów TPM: 1.2 oraz 2.0. Wersja 2.0 obsługuje dodatkowe funkcje i algorytmy bezpieczeństwa [6]. Moduły TPM spełniają również kryteria walidacji amerykańskiego rządowego standardu przetwarzania informacji 140 (*ang. Federal Information Processing Standard – FIPS*) Narodowego Instytutu Standaryzacji i Technologii (ang. *National Institute of Standards and Technology – NIST*) i obsługują zatwierdzone przez NIST algorytmy kryptograficzne [7].

3.2. ŁAŃCUCH ZAUFANIA

Łańcuch zaufania (*ang. Chain of Trust – CoT*) to metoda utrzymywania prawidłowych granic zaufania poprzez zastosowanie zasady zaufania przechodniego. Każdy moduł oprogramowania układowego w procesie uruchamiania systemu musi przed przekazaniem

sterowania sprawdzić sygnaturę następnego modułu. Po sprawdzeniu zaleca się natychmiastowe przekazanie wartości sygnatury do pamięci sprzętowego źródła zaufania, takiego jak rejestr HSM, na potrzeby procesu atestacji w późniejszym czasie [6]. Zastosowanie metody CoT można rozszerzyć na domenę aplikacji, umożliwiając oznaczanie i poświadczanie plików, katalogów, urządzeń, urządzeń peryferyjnych itp.

Stosowanie metody CoT zawsze zaczyna się od modułu RoT. Może się on składać z różnych komponentów sprzętowych i oprogramowania układowego. W przypadku kilku technologii zapewniania integralności platformy, główny moduł oprogramowania układowego RoT znajduje się w kodzie niemodyfikowalnej pamięci tylko do odczytu (*ang. Read-Only Memory – ROM*). Jednak moduł RoT nie działa w ten sposób w przypadku wszystkich technologii [6]. Moduł RoT jest zazwyczaj podzielony na komponenty służące do weryfikacji i obliczania sygnatur. Podstawowy moduł RoT do weryfikacji (*ang. Core RoT for Verification – CRTV*) jest odpowiedzialny za weryfikację pierwszego komponentu przed przekazaniem do niego sterowania. Podstawowy moduł RoT do obliczania sygnatur (*ang. Core RoT for Measurement – CRTM*) jest pierwszym komponentem, który jest wykonywany w ramach metody CoT i przekazuje pierwszą sygnaturę do modułu TPM. Moduł CRTM można podzielić na część statyczną (*ang. Static CRTM – SCRTM*) i dynamiczną (*ang. Dynamic CRTM – DCRTM*). Moduł SCRTM składa się z elementów, które mierzą oprogramowanie układowe w czasie uruchamiania systemu, tworząc niezmienny zestaw sygnatur, które pozostaną spójne po ponownym uruchomieniu, z wyjątkiem zmiennych atrybutów, takich jak data i godzina. Moduł DCRTM umożliwia ustanowienie łańcucha zaufania bez ponownego uruchamiania systemu, pozwalając na dynamiczne przywrócenie modułu RoT i wykorzystanie go do obliczania sygnatury.

Moduł RoT, który jest oparty na zabezpieczeniach sprzętowych, będzie trudniejszy do zmiany, natomiast moduł RoT, który jest oparty wyłącznie na oprogramowaniu układowym, można przeinstalować i zmodyfikować. Stały sprzętowy moduł RoT jest obarczony pewnym ryzykiem ze względu na brak możliwości instalowania poprawek. Celem powinno być posiadanie jak najmniejszego modułu RoT. W kontekście niniejszej publikacji oprogramowanie układowe jest specyficzną klasą

oprogramowania komputerowego, które zapewnia sterowanie niskiego poziomu nad określonymi elementami sprzętowymi urządzenia.

Łańcuchy zaufania mogą być tworzone przez różne technologie zapewniające integralność platformy. Więcej informacji można znaleźć w następujących przykładach technologii opisanych w załącznikach:

- UEFI Secure Boot (SB)
- Trusted Execution Technology (TXT) firmy Intel
- Boot Guard firmy Intel
- Platform Firmware Resilience (PFR) firmy Intel
- Podsumowanie przykładów technologii firmy Intel
- Platform Secure Boot (AMD PSB) firmy *AMD*
- TrustZone Trusted Execution Environment (TEE) dla architektury Armv8-A firmy Arm
- Secure Boot i Chain of Trust (CoT) firmy Arm
- Platform Roots of Trust firmy Cisco
- Chain of Trust (CoT) firmy *IBM*

3.3. OCHRONA ŁAŃCUCHA DOSTAW

Organizacje są coraz częściej narażone na ryzyko naruszenia bezpieczeństwa łańcucha dostaw, zarówno celowego, jak i niezamierzonego. Zarządzanie ryzykiem związanym z cyber łańcuchem dostaw wymaga między innymi zapewnienia integralności, jakości i odporności łańcucha dostaw, jego produktów i usług. Zagrożenia związane z cyber łańcuchem dostaw mogą obejmować podrabianie, nieautoryzowaną produkcję, manipulację, kradzież i dodawanie złośliwego oprogramowania lub w inny sposób nieoczekiwanego oprogramowania i sprzętu, a także niewłaściwe praktyki produkcyjne i rozwojowe w cyber łańcuchu dostaw [8], [9], [10].

Opracowano specjalne technologie, które umożliwiają ustalenie autentyczności i integralności sprzętu platformy, w tym jej oprogramowania układowego i konfiguracji. Technologie te umożliwiają zagwarantowanie, że platformy nie zostały naruszone lub zmodyfikowane od momentu ich zmontowania w siedzibie producenta do momentu ich dostarczenia do centrum danych klienta w stanie gotowym do instalacji. Weryfikacja tych atrybutów platformy jest jednym z elementów zabezpieczających łańcuch dostaw[5]. Niektóre technologie obejmują dodatkową funkcję blokowania procesu uruchamiania lub dostępu do tych platform, dopóki nie zostanie podany tajny klucz, który znają tylko klient i producent.

Więcej informacji można znaleźć w następujących przykładach technologii opisanych w załącznikach:

- Transparent Supply Chain (TSC) firmy Intel
- Rozwiązanie PFR z zabezpieczeniem Protection in Transit (PIT) firmy Intel
- Supply Chain Protection firmy Cisco

[5] Więcej informacji na temat bezpieczeństwa łańcucha dostaw można znaleźć na stronie instytucji National Cybersecurity Center of Excellence (*NCCoE*) poświęconej projektowi *Supply Chain Assurance* pod adresem https://www.nccoe.nist.gov/supply-chain-assurance.

4. MECHANIZMY OCHRONY ŚRODOWISKA URUCHOMIENIOWEGO OPROGRAMOWANIA

W niniejszym rozdziale omówiono różne ataki na środowisko uruchomieniowe oprogramowania i mechanizmy ochrony przed nimi.

4.1. ATAKI TYPU *ROP* ORAZ *COP/JOP*

Ataki typu *ROP* (*ang. Return Oriented Programming – ROP*) polegają na wykorzystaniu przepełnienia bufora i ukierunkowanym nadpisywaniu pamięci adresów zwrotnych w stosie. Atakujący przekierowują przepływy powrotne, uszkadzając adresy w stosie danych, aby wskazywały lokalizacje w już wykonywalnym kodzie. Te małe wybrane sekwencje kodu zwane *gadżetami* (*ang. gadgets*) powodują złośliwe modyfikacje systemu lub wywołanie nieautoryzowanych operacji. Typowym przykładem jest wywołanie pliku wykonywalnego powłoki w interfejsie systemowym [11].

Ataki typu *COP*/JOP (*ang. Call/Jump Oriented Programming – COP/JOP*) są podobne do ataków typu *ROP* i opierają się na blokach konstrukcyjnych gadżetów. Ich celem są pośrednie instrukcje skoku na końcu gadżetu, z których wiele jest celowo wysyłanych przez kompilator. Jednak gadżet skoku wykonuje jednokierunkowy transfer przepływu sterowania do celu, w przeciwieństwie do ataku typu *ROP*, w przypadku którego gadżety zwracają sterowanie z powrotem na stos. Może to utrudnić atakującym odzyskanie sterowania po uruchomieniu nieautoryzowanego oprogramowania, ale zaczynają pojawiać się rozwiązania tego problemu, takie jak to przedstawione w publikacji [11].

Aplikacje mogą wykorzystywać równoległy stos, znany jako *stos w tle* (*ang. shadow stack*), aby ograniczyć ataki na oprogramowanie, które próbują zmodyfikować przepływ sterowania. Przy wykorzystaniu specjalnego sprzętu, stos w tle jest używany do przechowywania kopii adresów zwrotnych. Adres jest porównywany z normalnym stosem programu podczas operacji powrotu. Jeśli zawartość się różni, generowany jest wyjątek, który może zapobiec przejęciu sterowania systemem przez złośliwy kod przy użyciu technik takich jak *ROP*. W ten sposób sprzętowy stos w tle może przyczyniać się do łagodzenia skutków niektórych z najbardziej powszechnych i możliwych do wykorzystania rodzajów błędów w oprogramowaniu.

W branży opracowano szereg środków obronnych i zapobiegawczych w celu przeciwdziałania atakom typu *ROP* i *COP*/JOP, w tym:

- Control-Flow Enforcement Technology firmy Intel (Intel CET)
- Pointer Authentication Code (*PAC*) firmy Arm
- Branch Target Identification (*BTI*) firmy Arm
- Zabezpieczenia przed atakami typu *ROP* i *COP/JOP* firmy *IBM*

4.2. ATAKI WYKORZYSTUJĄCE TRANSLACJĘ ADRESÓW

Komercyjne systemy operacyjne opierają się na modelach ochrony pamięci wirtualnej realizowanych poprzez stronicowanie wymuszane przez jednostkę zarządzania pamięcią (*ang. Memory Management Unit – MMU*) procesora. Systemy operacyjne izolują pamięć procesów i jądra za pomocą tabel stron zarządzanych przez oprogramowanie systemowe, z uprawnieniami dostępu, takimi jak użytkownik/administrator i odczyt/ zapis/wykonanie (*ang. Read/Write/Execute – RWX*). Dostęp do pamięci procesów i jądra odbywa się za pośrednictwem adresów wirtualnych, które są mapowane na adresy pamięci fizycznej za pomocą struktur translacji adresów. Struktury te używane do translacji adresów mają kluczowe znaczenie dla wymuszania modelu izolacji.

Nowoczesne systemy operacyjne mają jądra z pojedynczą przestrzenią adresową (w przeciwieństwie do mikrojąder), które zapewniają dobrą wydajność, ale mają dużą powierzchnię podatną na atak. Podatność w jądrze lub sterowniku może zostać wykorzystana do eskalacji uprawnień złośliwego procesu. Elementy podstawowe odczytu/zapisu (*ang. Read/Write – RW*) jądra mogą być wykorzystywane z użyciem podatności typu Write-What-Where wykrytych w kodzie jądra i/lub sterownikach.

Heurystyczne mechanizmy obronne, takie jak randomizacja tabeli stron, mogą zostać ominięte dzięki wyciekom informacji uzyskanym za pomocą złośliwych elementów podstawowych RW. Takie wycieki informacji są generowane poprzez utworzenie łańcucha wywołań systemowych (*ang. syscalls*). Na przykład, jedno wywołanie systemowe może przydzielić pamięć puli RWX, a drugie może wykorzystać arbitralny

zapis pamięci do nadpisania struktur translacji adresów. Metoda ta może być wykorzystywana do dwóch rodzajów ataków. W pierwszym z nich atakujący może przekierować używany adres wirtualny do zawartości kontrolowanej przez siebie (często znajdującej się w pamięci przestrzeni użytkownika). W drugim atakujący może utworzyć złośliwe mapowanie aliasu, które odwołuje się do żądanej pamięci fizycznej z uprawnieniami wybranymi przez atakującego (np. dostęp RW do strony za pośrednictwem mapowania aliasu, który pierwotnie był tylko do odczytu). Ważne jest, aby mechanizmy ochrony translacji adresów blokowały oba te rodzaje ataków.

Oprócz ochrony integralności struktur translacji adresów, procesory mogą również wykrywać i blokować wszelkie konfiguracje wykonania lub dostępu do danych przez kod o niższych uprawnieniach z dostępu o wyższych uprawnieniach. Zabezpieczenia te ustanawiają granice, wymagając, aby kod był wykonywany tylko z niezbędnymi uprawnieniami i w razie potrzeby wymuszając żądania z wyższymi uprawnieniami.

W branży opracowano szereg środków obronnych i zapobiegawczych w celu przeciwdziałania atakom wykorzystującym translacje adresów, w tym:

- Hypervisor Managed Linear Address Translation (HLAT) firmy Intel
- Supervisor Mode Execution Prevention (SMEP) oraz Supervisor Mode Access Prevention (SMAP) firmy Intel
- Supervisor Mode Execution Prevention (SMEP) oraz Supervisor Mode Access Prevention (SMAP) firmy *AMD*

4.3. NARUSZENIA BEZPIECZEŃSTWA PAMIĘCI

Około 70 procent podatności usuwanych każdego roku przez aktualizacje zabezpieczeń to problemy związane z bezpieczeństwem pamięci [12]. Są one szczególnie powszechne w programach napisanych w językach takich jak C i C++, w których wskaźniki są jawne. Takie błędy w kodzie mogą zostać wykorzystane przez atakujących do ujawnienia danych, w tym kluczy i innych sekretów. Naruszenia bezpieczeństwa pamięci mogą być tymczasowe i przestrzenne. Poniżej przedstawiono kilka typowych przykładów obu typów:

- **use-after-free**: program nadal korzysta z przydzielonej pamięci po jej zwolnieniu (*tymczasowe*);
- **use-out-of-scope**: program używa pamięci z zakresu innego programu, która nie znajduje się w jego bieżącym zakresie (*tymczasowe*);
- **use-before-initialization**: program uzyskuje dostęp do pamięci, która została przydzielona, ale nie została jeszcze zainicjowana (*tymczasowe*);
- **bounds violations, buffer overflow**: program uzyskuje dostęp do pamięci poza granicami przydzielonego bufora/struktury danych (*przestrzenne*).

W branży wprowadzono lub opracowuje się technologie sprzętowe mające na celu przeciwdziałanie naruszeniom bezpieczeństwa pamięci. Oba typy wymagają wsparcia oprogramowania pomocniczego.

Pierwszy z nich, *oznaczanie pamięci* (*ang. memory tagging*) - znany również jako kolorowanie, wersjonowanie, tagowanie -, został omówiony w publikacjach [13] i [14]. Jest to probabilistyczne podejście typu „blokada z kluczem" (*ang. lock-and-key*) do wykrywania błędów naruszenia pamięci. Przy rozmiarze znacznika wynoszącym 4 prawdopodobieństwo wykrycia błędu wynosi 94%, a przy rozmiarze znacznika wynoszącym 8 prawdopodobieństwo wynosi 99,6%. Oznaczanie pamięci prawdopodobnie znajdzie ogólne zastosowanie w 64-bitowym oprogramowaniu napisanym w językach C i C++. Oczekuje się również korzyści z jego stosowania w środowiskach mieszanych języków, np. kodu C/C++ i interakcji z językami kompilowanymi lub interpretowanymi „dokładnie na czas" (*ang. Just In-Time – JIT*). Możliwość zastosowania tej metody do oprogramowania w innych językach może być różna. Ponieważ oznaczanie pamięci nie narzuca żadnych zmian w standardowych binarnych interfejsach aplikacji (*ang. Application Binary Interface – ABI*) napisanych w języku C/C++, możliwe jest jego sukcesywne wdrażanie.

Drugi typ to *systemy sprzętowe oparte na zasobach (ang. capability-based hardware systems)*, które umożliwiają oprogramowaniu efektywne implementowanie szczegółowej ochrony pamięci i skalowalnego podziału oprogramowania poprzez zapewnienie silnych, nieprobabilistycznych, wydajnych mechanizmów wspierających

zasady minimalnych uprawnień i celowego użycia w wykonywaniu oprogramowania na wielu poziomach abstrakcji, zapobiegając wykorzystaniu i łagodząc skutki luk w zabezpieczeniach pamięci.

Technologia oparta na zasobach sprzętowych (*ang. hardware capability technology*) łączy odwołania do lokalizacji pamięci, czyli wskaźniki, z ograniczeniami dotyczącymi sposobu, w jaki odwołania te mogą być używane. Ograniczenia te odnoszą się zarówno do zakresów adresów, jak i funkcji, do których można uzyskać dostęp za pośrednictwem odwołań. Takie powiązane informacje nazywane są *zasobem (ang. capability)*. Zasób jest skonstruowany w taki sposób, że nie można go sfałszować przy użyciu oprogramowania. Zastąpienie wskaźników zasobami w programie znacznie poprawia bezpieczeństwo pamięci.

Korzyści płynące z technologii zasobów sprzętowych wykraczają poza bezpieczeństwo pamięci. Wynika to z faktu, że zasoby mogą być wykorzystywane jako blok konstrukcyjny do bardziej szczegółowego podziału (*ang. compartmentalization*) oprogramowania na kompartmenty. Może to skutkować z natury bardziej niezawodnym oprogramowaniem, które jest odporne na ataki. Ważną cechą podziału na kompartmenty jest to, że nawet jeśli jeden przedział zostanie naruszony przez atakującego, nie może on wydostać się z przedziału, aby uzyskać dostęp do innych informacji lub przejąć ogólną kontrolę nad systemem komputerowym.

Oprócz zmian w sprzęcie, bezpieczeństwo oparte na zasobach wymaga zmiany sposobu projektowania kodu. Kod musi być napisany i skompilowany w inny sposób, aby wykorzystać nowe funkcje sprzętowe i osiągnąć wyższy poziom bezpieczeństwa.

W branży opracowano szereg środków obronnych i zapobiegawczych w celu przeciwdziałania naruszeniom bezpieczeństwa pamięci, w tym:

- Privileged Access Never (PAN) firmy Arm
- User (EL0) Execute Never (UXN) oraz Privileged (EL1/EL2) Execute Never (PXN) firmy Arm
- Memory Tagging Extension (MTE) firmy Arm
- Hardware Enforced Capability-Based Architecture (Morello i CHERI) firmy Arm

4.4. ATAKI TYPU SIDE-CHANNEL

Podatność leżąca u podstaw czasowych ataków typu side-channel na pamięć podręczną polega na tym, że wzorzec alokacji w pamięci podręcznej procesora (*ang. CPU — Central Processing Unit*), a w szczególności, które zestawy pamięci podręcznej zostały wykorzystane do alokacji, można określić, mierząc czas potrzebny na uzyskanie dostępu do wpisów, które wcześniej znajdowały się w pamięci podręcznej lub na uzyskanie dostępu do wpisów, które zostały alokowane. Może to spowodować wycieki informacji o wzorcu alokacji pamięci podręcznej, które mogą zostać odczytane przez inne, mniej uprzywilejowane oprogramowanie.

Nowe spekulatywne ataki typu side-channel wykorzystują spekulatywne odczyty pamięci, które są powszechne w wysokowydajnych procesorach. W przypadku wykonywania spekulatywnych odczytów pamięci do buforowalnych lokalizacji poza odgałęzieniem nierozwiązanym architekturowo (lub inną zmianą w przebiegu programu), wyniki tych odczytów można również wykorzystać do utworzenia adresów dalszych spekulatywnych odczytów pamięci. Powodują one alokację w pamięci podręcznej wpisów, których adresy odpowiadają wartościom pierwszego spekulatywnego odczytu. Można to wykorzystać w atakach typu side-channel, jeśli niezaufany kod jest w stanie sterować spekulacją w taki sposób, że powoduje pierwszy spekulatywny odczyt lokalizacji, która w przeciwnym razie nie byłaby dostępna dla tego niezaufanego kodu. Wyniki drugiej alokacji spekulatywnej w pamięci podręcznej mogą być mierzone przez ten niezaufany kod.

Konstrukcje procesorów ewoluowały, aby sprostać tym zagrożeniom. Dodano do nich dodatkowe instrukcje oraz oprogramowanie układowe i oprogramowanie (firmware i software), aby przeciwdziałać tej klasie ataków. W branży opracowano środki obronne i zapobiegawcze w celu przeciwdziałania atakom typu side-channel: Środki ochrony przed atakami typu side-channel firmy Arm

5. OCHRONA DANYCH I POUFNOŚĆ PRZETWARZANIA

Wraz ze wzrostem popularności klienckich usług opartych na chmurze, wirtualizacja stała się koniecznością w chmurze obliczeniowej. Wirtualizacja polega na symulacji sprzętu dla wielu obciążeń w chmurze. Każde obciążenie jest odizolowane od innych, dzięki czemu ma dostęp tylko do własnych zasobów, a także może być całkowicie hermetyzowane w celu zapewnienia przenośności [15] [16]. Konwencjonalne maszyny wirtualne (*ang. Virtual Machine – VM*) mają odizolowaną przestrzeń jądra, w której oprócz jądra uruchamiane są wszystkie aspekty obciążenia. Obecnie zwirtualizowane środowisko zostało rozszerzone o kontenery i w pełni funkcjonalne mechanizmy organizacji obciążeń. Kontenery zapewniają przenośność aplikacji poprzez współdzielenie bazowego jądra, co znacznie zmniejsza zużycie zasobów przez obciążenia i zwiększa wydajność.

Chociaż kontenery mogą zapewniać pewien poziom wygody, podatności w przestrzeni jądra i warstwach współdzielonych mogą być często wykorzystywane do ataków, co sprawia, że bezpieczeństwo platformy bazowej staje się jeszcze ważniejsze. Ze względu na potrzebę dodatkowej ochrony w zwirtualizowanym obszarze roboczym, położono nacisk na szyfrowanie danych zarówno w spoczynku (przechowywanych), jak i podczas ich przetwarzania (użytkowania). *Szyfrowanie danych* w *spoczynku* zapewnia ich ochronę na dysku. Zazwyczaj odnosi się to do niezamontowanego magazynu danych (pamięci masowej) i chroni przed zagrożeniami, takimi jak fizyczne usunięcie dysku. W ramach ochrony i zabezpieczania danych w chmurze podczas ich użytkowania, określanej również jako *poufność przetwarzania*, do izolowania i przetwarzania zaszyfrowanych danych w pamięci wykorzystywane są funkcje sprzętowe, dzięki czemu dane te są mniej narażone na ujawnienie i naruszenie związane ze współbieżnymi obciążeniami lub bazowym systemem i platformą [17]. W niniejszym rozdziale opisano technologie, które można wykorzystać do zapewnienia poufności przetwarzania danych w chmurze i w ramach przetwarzania brzegowego.

Zaufane środowisko wykonawcze (*ang. Trusted Execution Environment – TEE*) to obszar lub enklawa chroniona przez procesor systemowy. Wrażliwe sekrety, takie jak klucze

kryptograficzne, ciągi uwierzytelniające lub dane dotyczące własności intelektualnej i prywatności, mogą być przechowywane w środowisku *TEE*, a operacje obejmujące te sekrety mogą być w nim wykonywane, eliminując w ten sposób potrzebę wydobywania sekretów poza *TEE*. Środowisko *TEE* pomaga również w zapewnieniu, że wykonywane w nim operacje i powiązane z nim dane nie mogą być przeglądane z zewnątrz, nawet przez uprzywilejowane oprogramowanie lub debuggery. Komunikacja ze środowiskiem *TEE* jest możliwa wyłącznie za pośrednictwem określonych interfejsów, a obowiązkiem projektanta/programisty *TEE* jest odpowiednie zdefiniowanie tych interfejsów. Poprawnie zdefiniowany interfejs środowiska *TEE* ogranicza dostęp do absolutnego minimum niezbędnego do wykonania zadania.

5.1. IZOLACJA PAMIĘCI

Istnieje wiele technologii zapewniających ochronę danych poprzez szyfrowanie. Większość z tych rozwiązań koncentruje się na ochronie odpowiednich danych w trakcie spoczynku i nie uwzględnia faktu, że dane te są odszyfrowywane i podatne na ataki podczas użytkowania. Aplikacje działające w pamięci współdzielą tę samą platformę sprzętową i mogą być podatne na ataki ze strony innych obciążeń działających na tym samym sprzęcie lub ze strony administratorów naruszających bezpieczeństwo chmury. Istnieje zdecydowana potrzeba zabezpieczenia własności intelektualnej i zapewnienia, że prywatne dane są szyfrowane i niedostępne w żadnym momencie, szczególnie w centrach danych w chmurze i obiektach przetwarzania brzegowego. Opracowano różne technologie sprzętowe do szyfrowania zawartości przetwarzanej w pamięci platformy.

Więcej informacji można znaleźć w następujących przykładach technologii opisanych w załącznikach:

- TME and Multi-Key Total Memory Encryption (Intel MKTME) firmy Intel
- Secure Memory Encryption (SME)/Transparent Secure Memory Encryption (TSME) firmy *AMD*
- Realm Memory Isolation and Protection firmy Arm

- External Memory (DRAM) Encryption and Integrity z architekturą *CCA* firmy Arm
- Memory Isolation Technology firmy *IBM*

5.2. IZOLACJA APLIKACJI

Izolacja aplikacji to wykorzystanie środowiska *TEE* do ochrony pamięci zarezerwowanej dla poszczególnych aplikacji. Granica zaufania powiązana z aplikacją jest ograniczona tylko do CPU. Przyszłe generacje tych technik umożliwią izolowanie całych aplikacji w ich własnych enklawach, a nie tylko ochronę określonych operacji lub pamięci. Dzięki zastosowaniu oddzielnych enklaw z unikalnymi kluczami dla każdej aplikacji, wrażliwe aplikacje będą mogły być chronione przed ujawnieniem danych, nawet przed złośliwymi działaniami wewnętrznymi wykonywanymi przez osoby mające dostęp do platformy bazowej. Implementacja izolacji aplikacji będzie zazwyczaj obejmować integrację zestawu narzędzi przez programistów aplikacji klienckich w warstwie aplikacji. Ich obowiązkiem jest również zapewnienie bezpiecznego środowiska *TEE*.

Więcej informacji można znaleźć w następujących przykładach technologii opisanych w załącznikach:

- Software Guard Extensions (SGX) firmy Intel
- Confidential Compute Architecture (CCA) firmy Arm
- TrustZone Trusted Execution Environment (TEE) dla architektury Armv8-A firmy Arm
- Realm Memory Isolation and Protection firmy Arm
- Technologia izolacji aplikacji firmy *IBM*

5.3. IZOLACJA MASZYNY WIRTUALNEJ

W miarę pojawiania się nowych technologii izolacji pamięci i operacji, izolowanie całych maszyn wirtualnych staje się coraz bardziej wykonalne. Maszyny wirtualne można już w pewnym stopniu izolować dzięki technologiom takim jak wirtualizacja wspomagana sprzętowo, ale pamięć każdej z nich pozostaje niezaszyfrowana. Niektóre istniejące technologie izolacji pamięci wymagają domyślnego zaufania do

menedżera maszyny wirtualnej (*ang. Virtual Machine Manager – VMM*). W przyszłych generacjach platform technologie izolacji usuną program VMM z granicy zaufania i umożliwią pełne szyfrowanie pamięci maszyn wirtualnych za pomocą unikalnych kluczy dla każdej z nich, chroniąc je nie tylko przed złośliwym oprogramowaniem działającym na hoście funkcji hypervisor, ale także przed oszukańczym oprogramowaniem układowym.

Izolacja maszyn wirtualnych może być wykorzystywana do ochrony obciążeń w środowiskach z wieloma użytkownikami, takich jak chmury publiczne i hybrydowe. Izolacja całych maszyn wirtualnych zapewnia ochronę przed złośliwymi działaniami wewnętrznymi u dostawcy chmury lub narażeniem na złośliwe oprogramowanie i wyciekiem danych do innych użytkowników chmury z obciążeniami działającymi na tej samej platformie. W ramach wielu nowoczesnych wdrożeń w chmurze maszyny wirtualne wykorzystywane są jako węzły robocze kontenerów. Jest to wysoce spójny i skalowalny sposób wdrażania kontenerów niezależnie od bazowych platform fizycznych. Dzięki pełnej izolacji maszyn wirtualnych, wirtualne procesy robocze hostujące kontenery mogą być skutecznie izolowane bez wpływu na korzyści płynące z oddzielenia kontenera od platformy bazowej.

Więcej informacji można znaleźć w następujących przykładach technologii opisanych w załącznikach:

- Trust Domain Extensions (Intel *TDX*) firmy Intel
- Secure Encrypted Virtualization (SEV) firmy *AMD*
- Confidential Compute Architecture (CCA) firmy Arm
- Technologia izolacji maszyny wirtualnej firmy *IBM*

5.4. AKCELERACJA KRYPTOGRAFICZNA

Szyfrowanie szybko staje się coraz bardziej powszechne w zastosowaniach obejmujących centra danych, ponieważ branża przyjmuje coraz więcej standardów i wytycznych dotyczących wrażliwości danych klientów i własności intelektualnej. Ponieważ operacje kryptograficzne mogą obniżać wydajność systemu i zużywać duże ilości zasobów

obliczeniowych, w branży wprowadzono wyspecjalizowane interfejsy sprzętowe zwane akceleratorami kryptograficznymi, które przenoszą zadania kryptograficzne z procesora głównego do oddzielnego układu koprocesora. Akceleratory kryptograficzne często występują w formie podłączanych kart peryferyjnych.

Więcej informacji można znaleźć w poniższych przykładach technologii opisanych w załącznikach lub na stronach internetowych ich dostawców:

- Advanced Encryption Standard New Instructions (AES-NI) firmy Intel
- QuickAssist Technology (QAT) wraz z Key Protection Technology (KPT) firmy Intel
- Advanced Encryption Standard firmy AMD
- Akceleracja kryptograficzna firmy Arm
- Technologia akceleracji kryptograficznej firmy *IBM*

6. USŁUGI ZDALNEJ ATESTACJI

Wykonywanie przeglądów oprogramowania układowego / konfiguracji serwera i rozszerzenie tych pomiarów na źródło zaufania w celu przechowywania i raportowania może pomóc w monitorowaniu, jakie oprogramowanie układowe działa na platformie. Niektóre technologie zapewniające integralność platformy mogą nawet lokalnie zaświadczać[6] i wymuszać użycie oprogramowania układowego i konfiguracji na serwerze. Centra danych składają się jednak zwykle z tysięcy serwerów, a ręczne monitorowanie ich oraz ich oprogramowania układowego byłoby dla operatora przytłaczającym zadaniem. Zdalna usługa może rozwiązać ten problem, zestawiając informacje o serwerze i szczegóły pomiarów. Do zapewnienia integralności przesyłanych danych pomiarowych można wykorzystać podpisy kryptograficzne. Co więcej, usługa zdalna może być wykorzystywana do definiowania polityki list dozwolonych, poprzez określenie, które wersje oprogramowania układowego i pomiary zdarzeń są akceptowalne dla serwerów w danym środowisku centrum danych. Usługa ta weryfikowałaby lub atestowała zebrane dane każdego serwera w odniesieniu do tych polityk, przekazując wyniki do koordynatora zasad w celu raportowania, wysyłania alertów lub wymuszania przestrzegania zasad w oparciu na zdarzeniach.

Usługa zdalnej atestacji może zapewnić dodatkowe korzyści poza weryfikacją oprogramowania układowego serwera. Określenie zasad listy dozwolonych dla określonych wersji oprogramowania układowego może umożliwić administratorom centrów danych łatwe ujawnianie starych wersji i wdrażanie nowych aktualizacji. W niektórych przypadkach pewne technologie sprzętowe i powiązane z nimi funkcje na platformach mogą być wykrywane na podstawie określonych pomiarów dziennika zdarzeń zapisanych w module HSM. Informacje monitorowane w ramach tej usługi zdalnej atestacji mogą być udostępniane za pośrednictwem warstwy administracyjnej centrum danych bezpośrednio użytkownikowi korporacyjnemu.

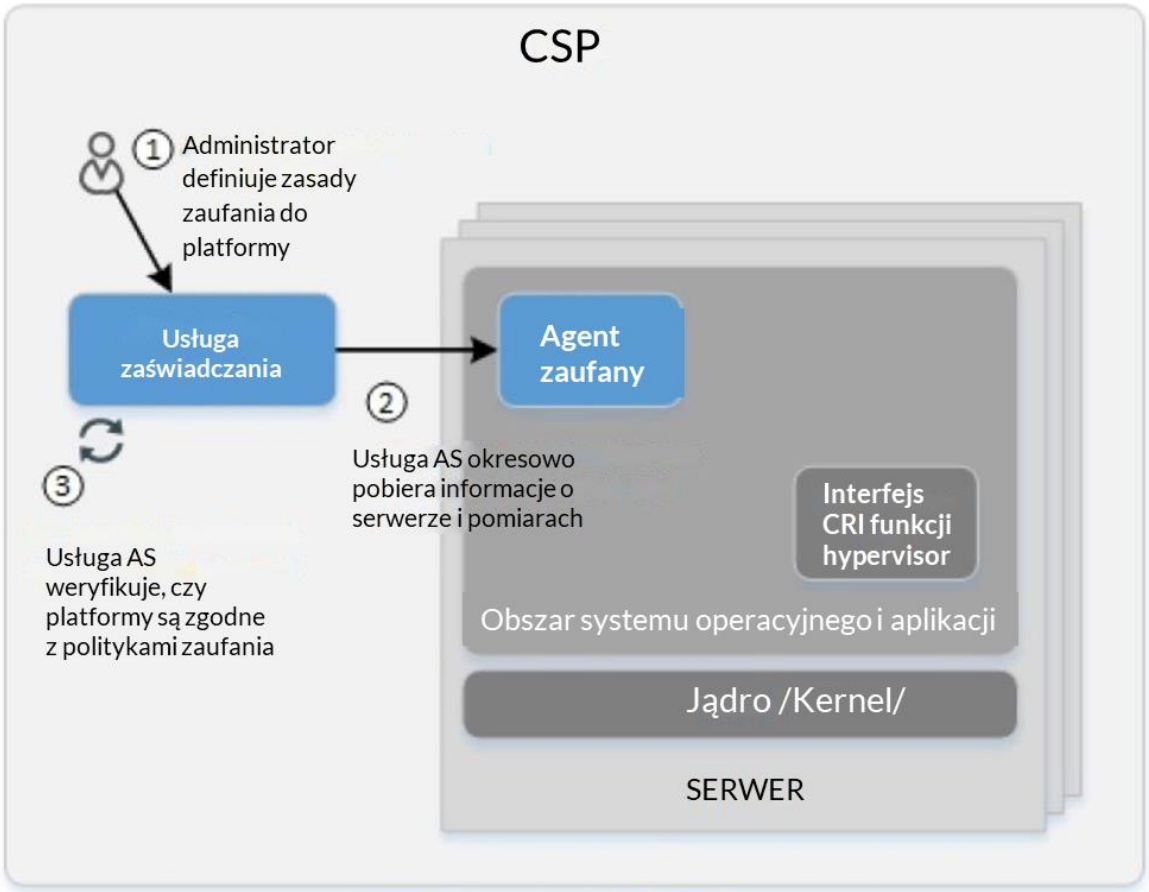
[6] Termin *zaświadczenie*, używany jest zamiennie z określeniem *atestacja*

Zapewniłoby to klientom punktów końcowych widoczność sprzętu i możliwość określenia wymagań dotyczących oprogramowania układowego lub zażądania funkcji platformy dla sprzętu, na którym działają ich usługi.

Kluczową zaletą zdalnej atestacji jest egzekwowanie zgodności we wszystkich systemach sprzętowych w centrum danych. Możliwość weryfikacji w oparciu o zbiorczą listę zezwoleń, w przeciwieństwie do lokalnego systemu egzekwującego zasady łańcucha dostaw, zapewnia operatorom większą elastyczność i sterowanie przy jednoczesnym zabezpieczeniu kryptograficznym. Te mechanizmy egzekwowania mogą być nawet łączone w celu zapewnienia skuteczniejszych polityk bezpieczeństwa.

6.1. ATESTACJA PLATFORMY

Na rysunku 1 przedstawiono usługę zdalnej atestacji (*ang. Attestation Service – AS*) umożliwiającą gromadzenie informacji o konfiguracji platformy i pomiarach integralności z serwerów centrum danych u dostawcy usług w chmurze (*ang. Cloud Service Provider – CSP*) za pośrednictwem usługi agenta zaufania uruchomionej na serwerach platformy. Operator chmury (*ang. cloud operator*) jest odpowiedzialny za zdefiniowanie polityk zaufania dla list dozwolonych. Zasady te powinny obejmować informacje i oczekiwane pomiary dla pożądanych technologii CoT platformy. Zebrane dane hosta są porównywane i weryfikowane na podstawie polityk, a następnie generowany jest raport w celu zapisania odpowiednich informacji o zaufaniu w bazie danych usługi zaświadczania AS.



Rysunek 1: Przykład hipotetycznej usługi zdalnej atestacji

Atestację platformy można rozszerzyć o integralność aplikacji lub pomiar i weryfikację interfejsu środowiska uruchomieniowego kontenera (*ang. container runtime interface - CRI*) funkcji hypervisor i aplikacji zainstalowanych na serwerach dedykowanych. Podczas rozruchu agent aplikacji na serwerze może mierzyć określone przez operatora pliki i katalogi, które są związane z konkretnymi aplikacjami. Można zdefiniować zasady zaufania do list dozwolonych, aby uwzględnić te oczekiwane pomiary, a zasady te mogą zostać uwzględnione w ogólnej ocenie zaufania platformy w ramach zdalnej usługi atestacji. Rozszerzając pomiary na moduł TPM platformy, do CoT można dodać aplikacje działające na serwerze dedykowanym. Komponenty agenta zaufania i agenta aplikacji można dodać do polityk i mierzyć wraz z innymi aplikacjami, aby upewnić się, że podstawowe elementy funkcji nie zostały naruszone. Na przykład typowa

implementacja agenta aplikacji w systemie Linux mogłaby działać wewnątrz pliku *initrd*, a pomiary dokonywane na systemie plików mogłyby zostać rozszerzone na moduł TPM platformy.

Dodatkową funkcją często powiązaną z zaufaniem do platformy jest *tagowanie zasobów (ang asset tagging). Tagi zasobów* (*ang. asset tags*) to proste atrybuty klucz-wartość, które są powiązane z platformą, takie jak lokalizacja, nazwa firmy, oddział lub dział. Takie atrybuty klucz-wartość są śledzone i rejestrowane przez centralną usługę zdalną, taką jak usługa atestacji, i mogą być przekazywane bezpośrednio do serwera za pośrednictwem agenta zaufania. Agent zaufania może następnie zabezpieczyć te powiązania atrybutów z platformą hosta, zapisując dane pomiaru skrótu (hash'a) dla informacji o tagu zasobu w sprzętowym układzie zabezpieczającym, takim jak TPM NVRAM platformy. Skrót tagu zasobu jest następnie pobierany przez usługę atestacji w ramach zapytania TPM i uwzględniany w ocenie raportu zaufania platformy.

Więcej informacji można znaleźć w następujących przykładach technologii opisanych w załącznikach:

- Security Libraries for the Data Center (ISecL-DC) firmy Intel
- Usługa atestacji zdalnej – projekt Veraison (VERificAtIon of atteStatiON)
- Narzędzia do atestacji platformy firmy *IBM*

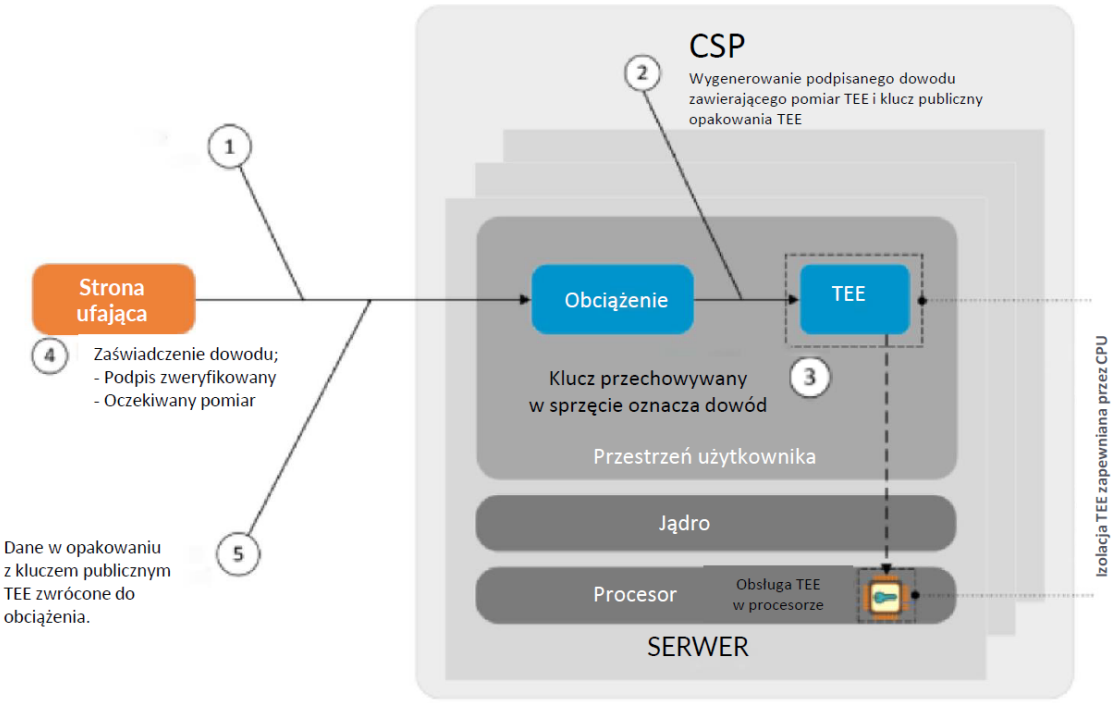
6.2. ZDALNA ATESTACJA ŚRODOWISKA *TEE*

Istnieją przypadki, w których wysoka gwarancja, że wynik przetwarzania w zaufanym środowisku wykonawczym (*ang. Trusted Execution Equipment – TEE*) może być zaufany, powinna zostać przekazana stronie ufającej. Jest to możliwe dzięki procesowi atestacji środowiska *TEE*. *Zdalna atestacja środowiska TEE* obejmuje wygenerowanie weryfikowalnego dowodu kryptograficznego przez to środowisko. Dowód jest następnie wysyłany do strony ufającej, która może zweryfikować podpis dowodu. Jeśli podpis jest prawidłowy, strona ufająca stwierdza, że zdalny kod działa w autentycznym środowisku *TEE*.

Dowód zwykle zawiera pomiar środowiska *TEE*, a także dane związane z jego autentycznością i wersją zgodną z wymogami. Pomiar jest skrótem zawartości środowiska *TEE* (np. kodu i danych statycznych). Pomiar uzyskany w czasie kompilacji jest zazwyczaj znany stronie ufającej i jest porównywany z pomiarem zawartym w dowodach, które są aktywnie pobierane w trakcie działania środowiska uruchomieniowego. Umożliwia to stronie ufającej ustalenie, że nie doszło do ingerencji w zdalny kod. Dowód może również zawierać podpis programisty enklawy i informacje o zaufanej bazie przetwarzania (*ang. Trusted Computing Base - TCB*) platformy.

Dowód może również zawierać część klucza publicznego pary kluczy wygenerowanej w środowisku *TEE* lub bezpieczny skrót klucza publicznego, jeśli istnieje ograniczenie rozmiaru dowodu. W tym drugim przypadku klucz publiczny musi zostać przekazany wraz z dowodem. Klucz publiczny umożliwia stronie ufającej zapakowanie kluczy tajnych, które chce wysłać do środowiska *TEE*. Dzięki tej funkcji strona ufająca może przekazywać klucze tajne bezpośrednio do środowiska *TEE* bez konieczności ufania jakiemukolwiek innemu oprogramowaniu uruchomionemu na serwerze.

Na rysunku 2 przedstawiono przykładowy proces atestacji środowiska *TEE*.



Rysunek 2: Hipotetyczny przykład procesu atestacji środowiska *TEE*

Tłumaczenie

Więcej informacji można znaleźć w następujących przykładach technologii opisanych w załącznikach:

- Secure Boot i Chain of Trust (CoT) firmy Arm
- Continuous Runtime Attestation firmy IBM

7. SCENARIUSZE WYKORZYSTANIA CHMURY OBLICZENIOWEJ, W KTÓRYCH ZASTOSOWANO ZABEZPIECZENIA SPRZĘTOWE

W niniejszym rozdziale opisano szereg scenariuszy wykorzystania chmury obliczeniowej, w których zastosowano zabezpieczenia sprzętowe i funkcje atestacji zaufania zintegrowane z orkiestracją usług operatora, umożliwiające realizację różnych celów związanych z bezpieczeństwem i zgodnością z przepisami.

7.1. WIDOCZNOŚĆ INFRASTRUKTURY BEZPIECZEŃSTWA

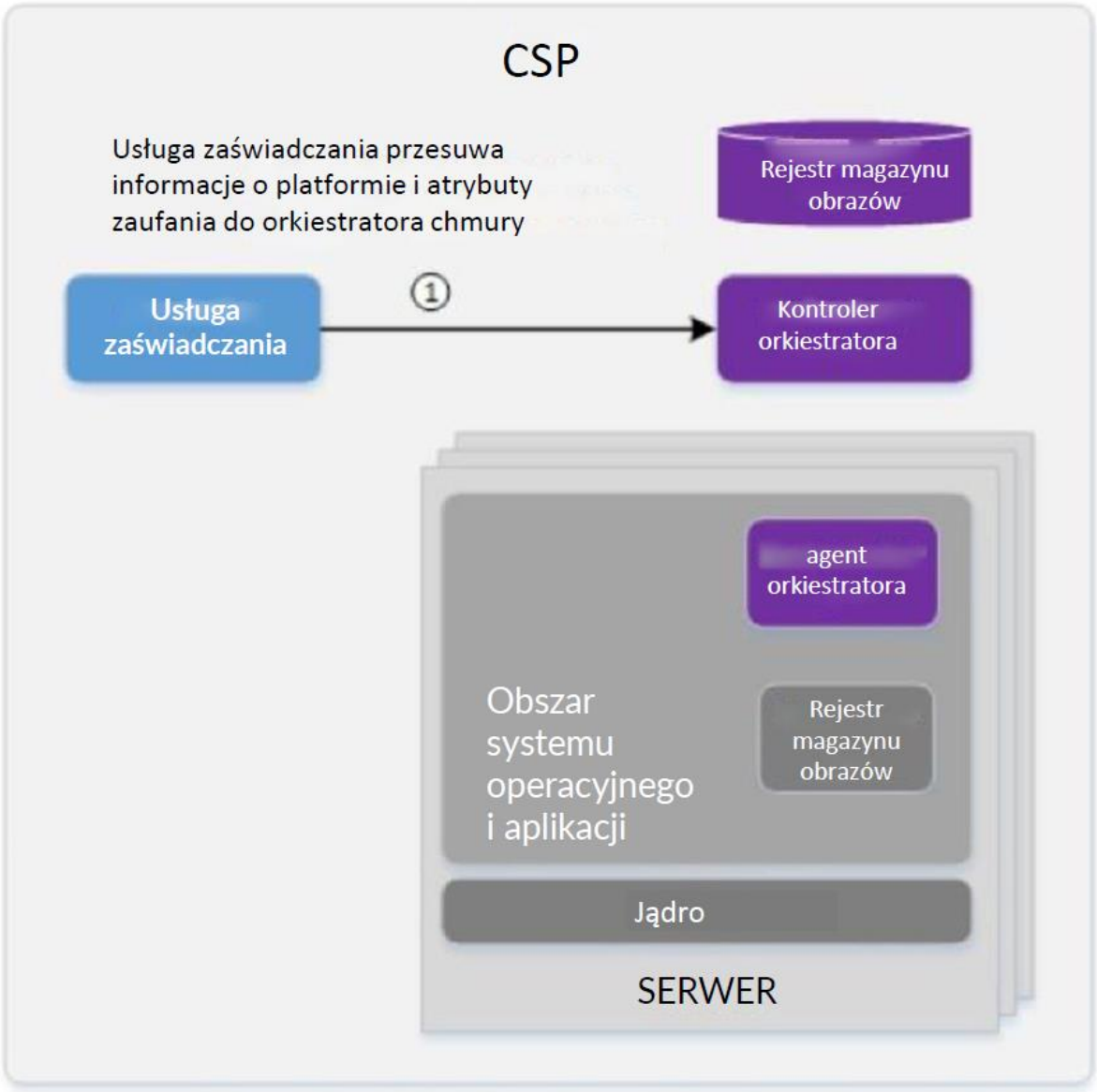
Typowy proces atestacji obejmuje walidację integralności pomiarów oprogramowania układowego platformy. Tego typu pomiary są unikalne dla konkretnej wersji systemu BIOS/UEFI, co oznacza, że raport z atestacji zapewnia wgląd w konkretną aktualnie używaną wersję oprogramowania układowego, a także integralność tego oprogramowania. Atestacja może również obejmować konfigurację sprzętu i informacje o obsłudze funkcji, zarówno poprzez bezpośrednie atestowanie obsługi funkcji, jak i poprzez różne pomiary w zależności od tego, które technologie zapewniające integralność platformy są wykorzystywane.

Weryfikowalne kryptograficznie raporty dotyczące integralności platformy i zawierające szczegółowe informacje o konfiguracji zabezpieczeń (np. wersji systemu BIOS/UEFI, informacje o lokalizacji, wersjach aplikacji) są niezwykle przydatne do audytu zgodności. Takie raporty z atestacji dla platformy fizycznej lub maszyny wirtualnej / procesów w przypadku przetwarzania poufnego mogą być połączone z zasadami uruchamiania obciążeń lub wydawania kluczy, zapewniając możliwość monitorowania i weryfikacji, czy dostęp do danych i obciążeń uzyskano wyłącznie na zgodnym sprzęcie w zgodnych konfiguracjach z włączonymi wymaganymi technologiami zabezpieczeń.

7.2. LOKOWANIE OBCIĄŻEŃ NA ZAUFANYCH PLATFORMACH

Informacje o platformie i zweryfikowane pomiary oprogramowania układowego / konfiguracji przechowywane w ramach usługi atestacji mogą być wykorzystywane do egzekwowania zasad w niezliczonych przypadkach użycia. Jednym z przykładów jest

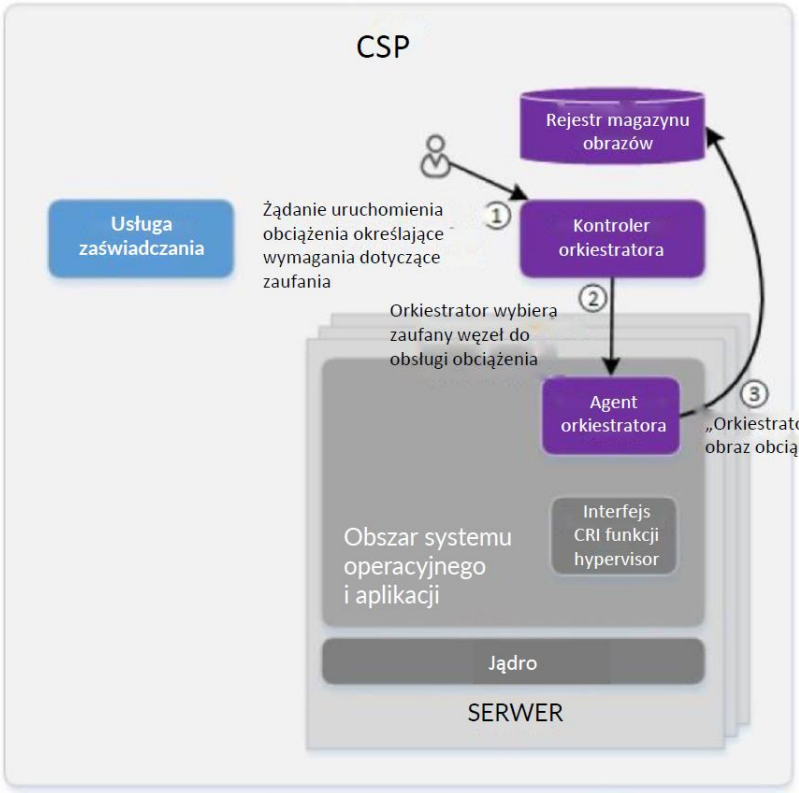
planowanie orkiestracji. Narzędzia do orkiestracji[7] (*ang. orchestrators*) usług w chmurze, takie jak Kubernetes i OpenStack, zapewniają możliwość oznaczania węzłów serwerów w swojej bazie danych za pomocą atrybutów klucz-wartość. Usługa atestacji może publikować atrybuty zaufania i informacyjne w bazach danych orkiestratorów w celu wykorzystania ich w decyzjach dotyczących planowania obciążeń. Pokazano to na rysunku 3.



Rysunek 3: Hipotetyczny przykład etykietowania platformy przez orkiestrator

[7] Inna nazwa narzędzi do orkiestracji – orkiestratory.

W OpenStack można to osiągnąć poprzez etykietowanie węzłów przy użyciu niestandardowych cech. Obrazy obciążeń mogą być przesyłane do magazynu obrazów zawierającego metadane określające wymagane wartości cech, które mają być powiązane z węzłem wybranym przez mechanizm planowania. W oprogramowaniu Kubernetes węzły mogą być etykietowane w magazynie *etcd* za pomocą selektora węzłów lub koligacji węzłów. Istnieje możliwość pisania niestandardowych definicji zasobów (*ang. Custom Resource Definition – CRD*) i podłączania ich do Kubernetes w celu odbierania wartości etykiet z usługi atestacji i kojarzenia ich z węzłami w magazynie *etcd*. Gdy uruchamiane jest wdrożenie lub kontener, atrybuty selektora węzłów lub koligacji węzłów mogą być zawarte w konfiguracyjnym pliku *yaml*, aby nakazać oprogramowaniu Kubernetes wybieranie tylko tych węzłów, które mają określone etykiety. Inne mechanizmy i wersje orkiestratora mogą być modyfikowane w celu dostosowania do podobnego przypadku użycia. Na rysunku 4 przedstawiono, w jaki sposób można skonfigurować orkiestrator, aby uruchamiał obciążenia tylko na zaufanych platformach lub platformach z określonymi atrybutami tagów zasobów.



Rysunek 4: Hipotetyczny przykład planowania orkiestratora

7.3. TAGOWANIE ZASOBÓW I ZAUFANA LOKALIZACJA

Zaufana geolokalizacja jest specyficzną implementacją wspomnianej wcześniej funkcji tagu zaufanego zasobu używanej wraz z usługą atestacji platformy. Wartości atrybutów klucza określające informacje o lokalizacji są używane jako tagi zasobów i dostarczane do komponentów serwera, takich jak moduł TPM. Dzięki temu informacje o lokalizacji mogą być uwzględniane w raportach z atestacji platformy, a tym samym wykorzystywane przez orkiestratory chmury, aplikacje do zarządzania infrastrukturą, silniki zasad i inne podmioty [18]. Orkiestracja przy użyciu tagów zasobów może być wykorzystywana do segregowania obciążeń i dostępu do danych w wielu różnych scenariuszach. Geolokalizacja może być ważnym atrybutem, który należy wziąć pod uwagę w przypadku hybrydowych środowisk w chmurze podlegających kontroli prawnej. Naruszenie takich ograniczeń poprzez umożliwienie dostępu do danych poza określonymi granicami geopolitycznymi może skutkować wysokimi karami.

Oprócz lokalizacji, ta sama zasada może mieć zastosowanie do innych rodzajów informacji zawartych w tagach. Na przykład niektóre serwery mogą być oznaczone jako odpowiednie do przechowywania informacji dotyczących zdrowia. Dane i obciążenia wymagające tego poziomu zgodności powinny być dostępne tylko na platformach skonfigurowanych pod kątem spełnienia takich wymogów. Inne serwery mogą być używane do przechowywania lub przetwarzania informacji i obciążeń niepodlegających tym wymogom. Poza informowaniem o integralności platform, tagi zasobów mogą być wykorzystywane do oznaczania, które serwery są odpowiednie dla danych obciążeń. Mechanizmy atestacji umożliwiają zapewnienie, że informacje w tagach zasobów są autentyczne, zapobiegając dokonaniu łatwego podszycia.

Poza segregacją obciążeń według określonych wymogów prawnych, organizacja może chcieć je posegregować według struktury organizacyjnej. Na przykład informacje dotyczące działu kadr i finansów mogą być ograniczone do platform o różnych profilach bezpieczeństwa, a obciążenia związane z dużymi zbiorami danych mogą wymagać uruchamiania na platformach oznaczonych jako odpowiednie pod względem

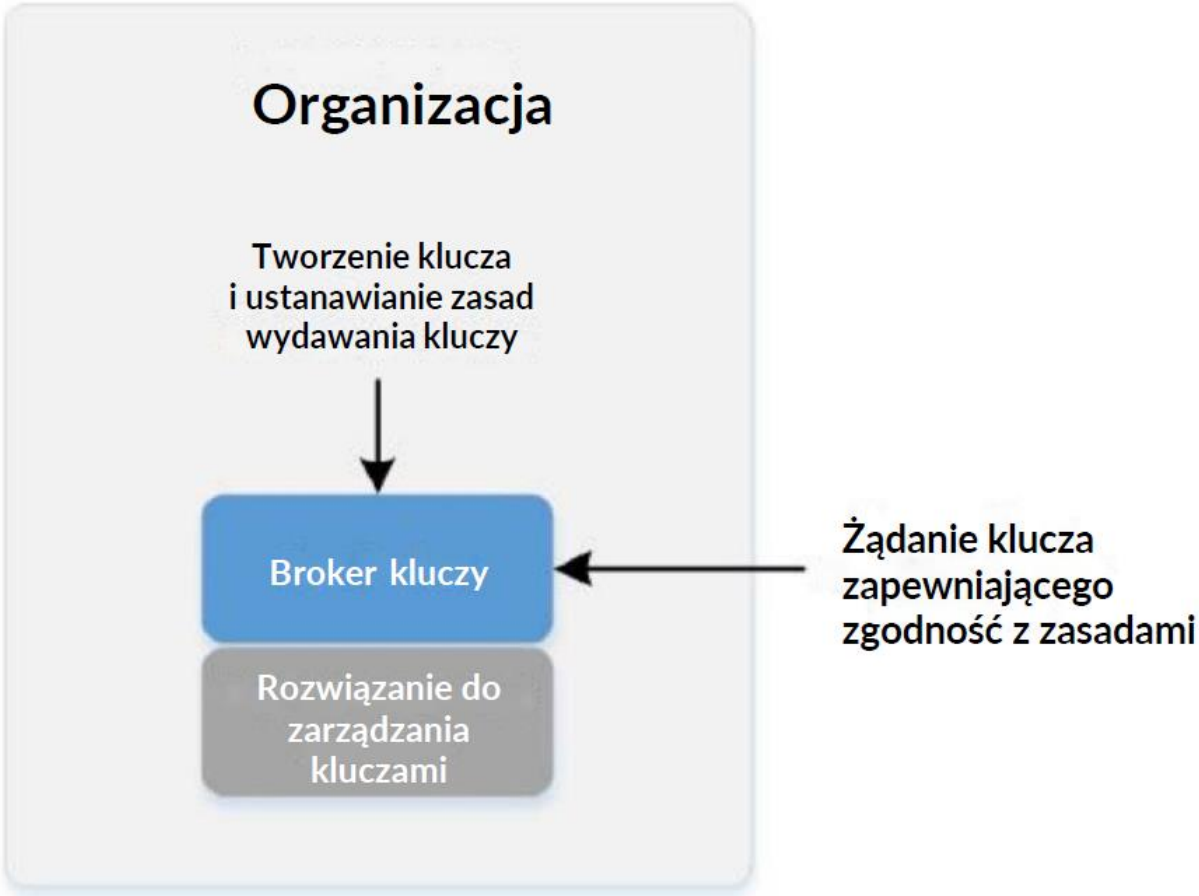
wydajności. W przypadku platform orkiestracji chmury, które natywnie nie obsługują wykrywania lub planowania obciążeń w oparciu o określone funkcje platformy, tagi zasobów mogą zapewnić mechanizm umożliwiający łatwe dodawanie takiej możliwości. Na przykład, w procesie orkiestracji można zaplanować uruchamianie obciążeń wymagających technologii *SGX* firmy Intel wyłącznie na platformach obsługujących tę funkcję, nawet jeśli platforma w chmurze nie wykrywa natywnie obsługi technologii *SGX*. Dostępna dla użytkowników i konfigurowana przez nich funkcja tagowania zasobów umożliwia praktycznie dowolny poziom podziału zasobów według potrzeb biznesowych, bezpieczeństwa lub wymogów prawnych.

7.4. POUFNOŚĆ OBCIĄŻENIA

Klienci, którzy lokują swoje obciążenia w chmurze lub w urządzeniach brzegowych, akceptują fakt, że ich obciążenia są zabezpieczone przez ich dostawców usług, zazwyczaj bez wglądu w stosowane mechanizmy bezpieczeństwa lub wiedzy na ich temat. Możliwość szyfrowania obrazów obciążeń przez użytkowników końcowych może zapewnić izolację kryptograficzną, mającą na celu ochronę magazynowanych danych klientów i własności intelektualnej. Kontrola kluczy jest nieodłącznym elementem procesu szyfrowania obciążenia. Chociaż preferowane jest przeniesienie przechowywania kluczy, zarządzania nimi i ich własności na klienta końcowego, należy zdefiniować odpowiednie zasady wydawania kluczy, które obejmą gwarancję ze strony dostawcy usług, że wykorzystywana platforma sprzętowa i oprogramowanie układowe są zabezpieczone i nienaruszone.

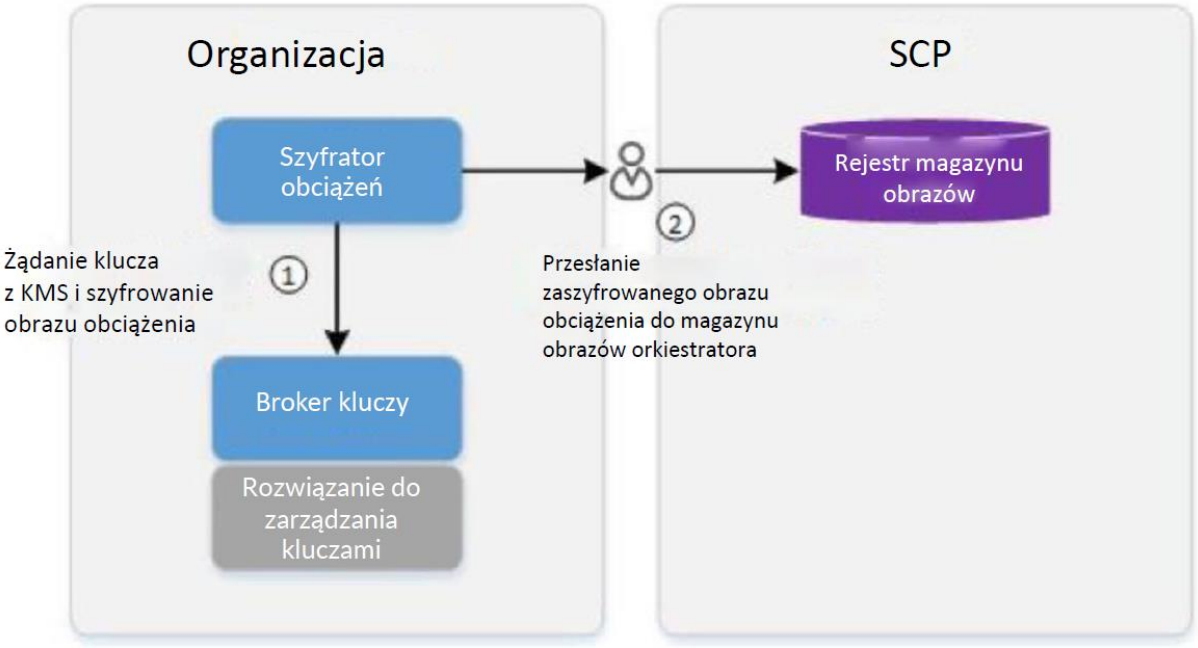
Istnieje kilka rozwiązań do zarządzania kluczami (*ang. Key Management Solution – KMS*), które zapewniają usługi tworzenia i przechowywania kluczy. Wiele z nich jest zgodnych z branżowymi standardami protokołu interoperacyjności zarządzania kluczami (*ang. Key Management Interoperability Protocol – KMIP*) lub standardami kryptografii klucza publicznego (*ang. Public Key Cryptography Standards – PKCS*) nr 11 za pośrednictwem połączenia sieciowego i mogą być wdrażane w organizacjach klientów. Koncepcja polega na zapewnieniu subtelnej warstwy zabezpieczeń oprócz rozwiązania KMS, zwanej *brokerem kluczy (ang. key broker,)*, jak pokazano na rysunku 5,

który stosuje i ocenia zasady w odniesieniu do żądań przychodzących do KMS. Obsługiwane żądania kierowane do brokera kluczy obejmują tworzenie kluczy, kojarzenie zasad uwalniania kluczy i żądanie kluczy poprzez ocenę powiązanych polityk. Zasady uwalniania kluczy mogą być dowolnym zestawem reguł, które muszą zostać spełnione przed uwolnieniem klucza. Zasady uwalniania kluczy są dostępne dla użytkowników i mają być łatwe do rozszerzenia, ale dla celów tej publikacji przyjęto, że są one związane z zaufaniem do platformy.



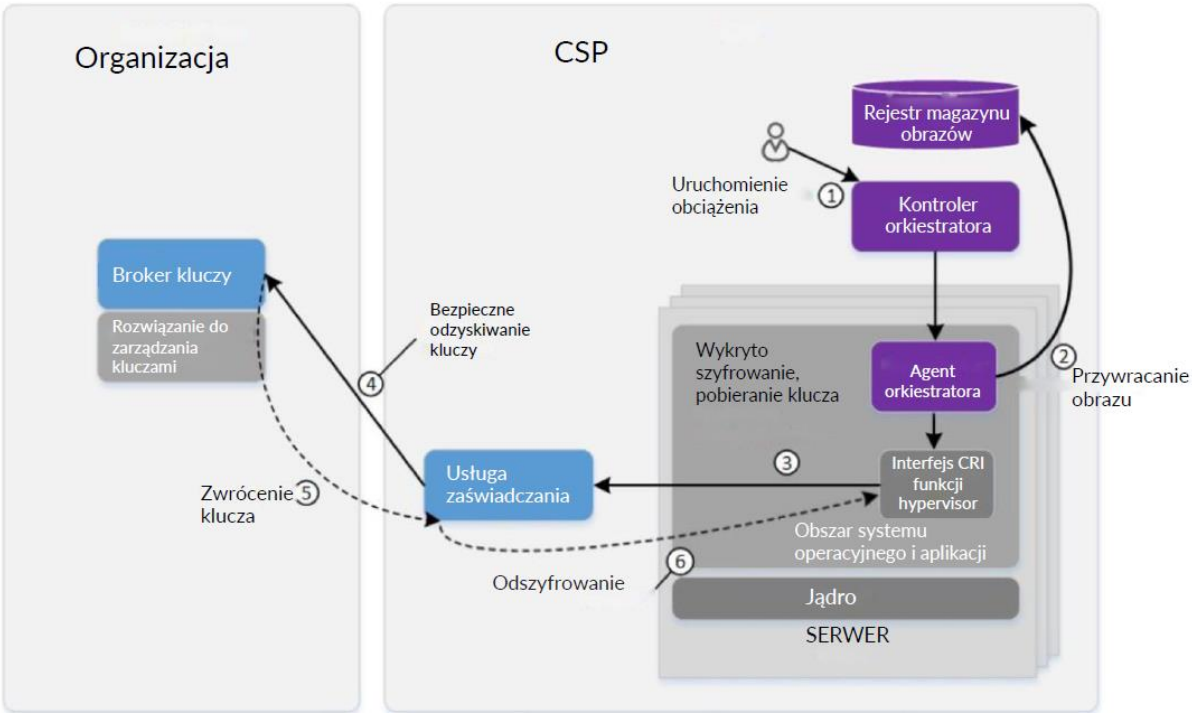
Rysunek 5: Hipotetyczny przykład stosowania brokera kluczy

Po określeniu zasad dotyczących kluczy, klucz utworzony i zarządzany przez rozwiązanie KMS może zostać użyty do zaszyfrowania obrazu obciążenia, jak pokazano na rysunku 6. Użytkownik może następnie przesłać zaszyfrowany obraz do magazynu lub rejestru obrazów orkiestratora *CSP*.



Rysunek 6: Hipotetyczny przykład szyfrowania obrazu obciążenia

Jak pokazuje rysunek 7, proces odzyskiwania klucza i odszyfrowywania jest najbardziej złożonym elementem zapewniania poufności obciążeń. Opiera się on na bezpiecznym transferze kluczy między organizacją a *CSP* na podstawie odpowiednich zasad udostępniania kluczy zarządzanych przez broker kluczy. Klucz jest udostępniany serwerowi obciążenia w celu odszyfrowania zaszyfrowanego obciążenia i jego uruchomienia. Zasady udostępniania kluczy wspomniane powyżej opierają się na zaufaniu do platformy i ważnym dowodzie tego zaufania. Zasady mogą również narzucać wymóg pakowania klucza przy użyciu publicznego klucza pakowania, przy czym prywatna część klucza pakowania jest znana tylko platformie sprzętowej *CSP*.



Rysunek 7: Hipotetyczny przykład odszyfrowywania obciążenia

Gdy usługa węzła środowiska uruchomieniowego otrzyma żądanie uruchomienia, może wykryć, że obraz jest zaszyfrowany, i przesłać żądanie pobrania klucza odszyfrowującego. Żądanie to może zostać przekazane za pośrednictwem usługi atestacji, aby można było przeprowadzić wewnętrzną ocenę zaufania do platformy. Żądanie klucza jest przekazywane do brokera kluczy wraz z dowodem, że platforma przeszła proces atestacji. Broker kluczy może następnie zweryfikować raport z atestacji platformy i zwolnić klucz z powrotem do *CSP* i usług środowiska uruchomieniowego węzła. Wtedy środowisko uruchomieniowe węzła może odszyfrować obraz i kontynuować normalną orkiestrację obciążenia. Podsystem jądra do szyfrowania dysków może zapewniać szyfrowanie obciążeń na platformie w czasie ich magazynowania.

7.5. OCHRONA KLUCZY I SEKRETÓW

Klucze kryptograficzne są zasobami o wysokiej wartości w przypadku obciążeń, zwłaszcza w środowiskach, w których właściciel kluczy nie ma pełnej kontroli nad infrastrukturą, takich jak chmury publiczne, obliczenia brzegowe i wdrożenia wirtualizacji funkcji sieciowych (*ang. Network Function Virtualization – NFV*). W takich

środowiskach klucze są zazwyczaj zapisywane na dysku jako „pliki płaskie" (*ang. flat files*) lub wpisy w plikach konfiguracyjnych. W środowisku uruchomieniowym, obciążenia wczytują klucze do pamięci o dostępie swobodnym (*ang. Random Access Memory – RAM*) i wykorzystują je do wykonywania operacji kryptograficznych, takich jak podpisywanie danych, szyfrowanie/deszyfrowanie lub zakończenie działania protokołu TLS (*ang. Transport Layer Security*).

Klucze na dysku i w pamięci RAM są narażone na konwencjonalne ataki, np. z eskalacją uprawnień, zdalnym wykonywaniem kodu i niewłaściwym zarządzaniem buforem wejściowym. Klucze mogą również zostać skradzione przez nieuczciwych administratorów lub ujawnione z powodu błędów operacyjnych. Na przykład nieprawidłowo chroniona migawka maszyny wirtualnej może zostać wykorzystana przez złośliwego agenta do pozyskania kluczy.

Do serwera można podłączyć moduł HSM, który będzie wykorzystywany przez obciążenia do przechowywania kluczy i wykonywania operacji kryptograficznych. Dzięki temu klucze będą chronione podczas magazynowania i użytkowania. W tym modelu klucze nigdy nie są przechowywane na dysku ani wczytywane do pamięci RAM. Jeśli dołączenie modułu HSM do serwera nie jest możliwe lub jeśli klucze są potrzebne na wielu serwerach jednocześnie, alternatywną opcją jest zastosowanie sieciowego modułu HSM. Obciążenia wysyłają ładunek (*ang. payload*) wymagający przetwarzania kryptograficznego przez połączenie sieciowe do sieciowego modułu HSM, który następnie wykonuje operacje kryptograficzne lokalnie, zwykle w dołączonym module HSM.

W niektórych środowiskach opcja z zastosowaniem modułu HSM jest niemożliwa do wprowadzenia. Właściciele obciążeń mogą nie mieć dostępu do chmury lub środowiska obliczeń brzegowych, aby podłączyć swój moduł HSM do serwera sprzętowego. W przypadku sieciowych modułów HSM problemem mogą być opóźnienia sieciowe, a niektóre obciążenia wymagają zoptymalizowanego czasu reakcji. Ponadto sieciowe moduły HSM są często dostarczane jako usługa przez dostawców chmury, usługi obliczeń brzegowych lub NFV, a opłaty za nie są naliczane na podstawie liczby transakcji. Koszt jest często czynnikiem decydującym o korzystaniu z sieciowego modułu HSM od usługodawcy.

8. DALSZE DZIAŁANIA

Oczekujemy opinii społeczności na temat treści publikacji i prosimy o dodatkowe przykłady technologii od innych firm. Niniejszy raport ma być żywym dokumentem, który będzie często aktualizowany w celu odzwierciedlenia postępu technologicznego oraz dostępności komercyjnych implementacji i rozwiązań. Może się to przyczynić do podniesienia poprzeczki w zakresie bezpieczeństwa platform i ewolucji zastosowań.

NIST pracuje również nad innymi publikacjami dotyczącymi bezpieczeństwa sprzętowego w ramach projektu NCCoE Trusted Cloud. Więcej informacji na temat projektu i łącza do innych publikacji można znaleźć na stronie https://www.nccoe.nist.gov/projects/trusted-cloud-vmware-hybrid-cloud-iaas-environments.

REFERENCJE

NARODOWE STANDARDY CYBERBEZPIECZEŃSTWA[8]	
NSC 199	Standardy kategoryzacji bezpieczeństwa – na podstawie FIPS 199
NSC 200	Minimalne wymagania bezpieczeństwa informacji i systemów informacyjnych podmiotów publicznych – na podstawie FIPS 200
NSC 800-30	Przewodnik dotyczący postępowania w zakresie szacowania ryzyka w podmiotach realizujących zadania publiczne – na podstawie NIST SP 800-30
NSC 800-34	Poradnik planowania awaryjnego – na podstawie NIST SP 800-34
NSC 800-37	Ramy zarządzania ryzykiem w organizacjach i systemach informacyjnych. Bezpieczeństwo i ochrona prywatności w cyklu życia systemu – na podstawie NIST SP 800-37
NSC 800-39	Zarządzanie ryzykiem bezpieczeństwa informacji. Przegląd struktury organizacyjnej, misji i systemu informacyjnego – na podstawie NIST SP 800-39
NSC 800-53	Zabezpieczenia i ochrona prywatności systemów informacyjnych oraz organizacji – na podstawie NIST SP 800-53
NSC 800-53A	Ocenianie środków bezpieczeństwa i ochrony prywatności systemów informacyjnych oraz organizacji. Tworzenie skutecznych planów oceny – na podstawie NIST SP 800-53A
NSC 800-53B	Zabezpieczenia bazowe systemów informacyjnych oraz organizacji – na podstawie NIST SP 800-53B

[8] Narodowe Standardy Cyberbezpieczeństwa - Baza wiedzy - Portal Gov.pl (www.gov.pl)

Tłumaczenie

NARODOWE STANDARDY CYBERBEZPIECZEŃSTWA[8]	
NSC 800-53 MAP	Mapowanie środków bezpieczeństwa: NSC 800-53 wer. 2 – PN-ISO/IEC 27001:2013; PN-ISO/IEC 27001:2013 – NSC 800-53 wer. 2 Patrz: SP 800-53 Rev. 5, *Security and Privacy Controls for Info Systems and Organizations* | CSRC (nist.gov)
NSC 800-60	Wytyczne w zakresie określania kategorii bezpieczeństwa informacji I kategorii bezpieczeństwa systemu informacyjnego – na podstawie NIST SP 800-60
NSC 800-61	Podręcznik postępowania z incydentami naruszenia bezpieczeństwa komputerowego – na podstawie NIST SP 800-61

PUBLIKACJE ANGLOJĘZYCZNE[9]

[1] Barrett B (2018) *Russia 's Elite Hackers Have a Clever New Trick That 's Very Hard to Fix*, Wired. Publikacja dostępna pod adresem https://www.wired.com/story/fancy-bear-hackers-uefi-rootkit

[2] Intel Corporation (2021) *Intel® Trusted Execution Technology (Intel1® TXT) - Software Development Guide - Measured Launched Environment Developer's Guide, Revision 017*. Publikacja dostępna pod adresem https://www.mtel.com/content/www/us/en/software- devel opers/intel-txt-software-devel opment-guide.html

[3] Regenscheid AR (2014) *BIOS Protection Guidelines for Servers.* (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800147B. https://doi.org/10.6028/NIST.SP.800-147B

[4] Regenscheid AR (2018) *Platform Firmware Resiliency Guidelines.* (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-193. https://doi.org/10.6028/NIST.SP.800-193

[5] Barker EB, Barker WC (2019) *Recommendation for Key Management: Part 2 - Best Practices for Key Management Organizations.* (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-57 Part 2, Rev. 1. https://doi.org/10.6028/NIST.SP.800-57pt2r1

[6] Trusted Computing Group (2019) *Trusted Platform Module Library Specification, Family "2.0"*. Publikacja dostępna pod adresem https://trustedcomputinggroup.org/work-groups/trusted-platform-module/

[9] Publikacje angielski zostały podane w celach uzupełniających dla osób zainteresowanych.

[7] National Institute of Standards and Technology (2001) *Security Requirements for Cryptographic Modules*. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 140-2, Change Notice 2 December 03, 2002. https://doi.org/10.6028/NIST.FIPS.140-2

[8] Diamond T, Grayson N, Paulsen C, Polk T, Regenscheid A, Souppaya M, Brown C (2020) *Validating the Integrity of Computing Devices: Supply Chain Assurance*. (National Institute of Standards and Technology, Gaithersburg, MD). Publikacja dostępna pod adresem https://www.nccoe.nist.gov/supply-chain-assurance

[9] Boyens JM, Paulsen C, Moorthy R, Bartol N (2015) *Supply Chain Risk Management Practices for Federal Information Systems and Organizations*. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800161. https://doi.org/10.6028/NIST.SP.800-161

[10] National Institute of Standards and Technology (2022) *Cybersecurity Supply Chain Risk Management*. Publikacja dostępna pod adresem https://csrc.nist.gov/Projects/cyber-supply-chain-risk-management

[11] Bletsch T, Jiang X, Freeh V, Liang Z (2011) *Jump-Oriented Programming: A New Class of Code-Reuse Attack*. Publikacja dostępna pod adresem https://www.comp.nus.edu.sg/~liangzk/papers/asiaccs11.pdf

[12] Arm (2022) *TrustedFirmware-A Project*. Publikacja dostępna pod adresem https://www.trustedfirmware.org/projects/tf-a/

[13] Arm (2022) *Morello*. Publikacja dostępna pod adresem https://developer.arm.com/architectures/cpu- architecture/a-profile/morello

[14] Arm (2022) *Arm Morello System Development Platform Technical Reference Manual*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/102278/latest

[15] Scarfone KA, Souppaya MP, Hoffman P (2011) *Guide to Security for Full Virtualization Technologies*. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-125. https://doi.org/10.6028/NIST.SP.800-125

[16] Intel Corporation (2022) *Intel® Virtualization Technology (Intel® VT)*. Publikacja dostępna pod adresem https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html

[17] Linux Foundation (2022) *What is the Confidential Computing Consortium?* Publikacja dostępna pod adresem https://confidentialcomputing.io

[18] Bartock MJ, Souppaya MP, Yeluri R, Shetty U, Greene J, Orrin S, Prafullchandra H, McLeese J, Scarfone KA (2015) *Trusted Geolocation in the Cloud: Proof of Concept Implementation*. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) 7904. https://doi.org/10.6028/NIST.IR.7904

[19] Debian Wiki (2022) *Secure Boot*. Publikacja dostępna pod adresem https://wiki.debian.org/SecureBoot

[20] Wilkins R, Richardson B (2013) *UEFI Secure Boot in Modern Computer Security Solutions* (Unified Extensible Firmware Interface Forum). Publikacja dostępna pod adresem https://uefi.org/sites/default/files/resources/UEFI Secure Boot in Modern Computer_Security_Solutions_2013.pdf

[21] RedHat (2021) *README* (for shim). Publikacja dostępna pod adresem https://github.com/rhboot/shim

[22] Intel Corporation (2018) *Intel® Trusted Execution Technology (Intel1® TXT) Overview*. Publikacja dostępna pod adresem https://www.intel.com/content/www/us/en/support/articles/000025873/technologies.html

[23] Futral W, Greene J (2013) *Intel® Trusted Execution Technology for Server Platforms* (Apress, Berkeley, CA). Publikacja dostępna pod adresem https://link.springer.com/book/10.1007/978-1- 4302-6149-0

[24] Linux Kernel Organization (2022) *Intel® TXT Overview*. Publikacja dostępna pod adresem httns://www.kernel.org/doc/Documentation/intel txt.txt

[25] Intel Corporation (2022) *Transparent Supply Chain*. Publikacja dostępna pod adresem https://www.intel.com/content/www/us/en/products/docs/servers/transparent-supply-chain.html

[26] Intel Corporation (2020) *Intel Highlights Latest Security Investments at RSA 2020*. Publikacja dostępna pod adresem https://newsroom.intel.com/news-releases/intel-highlights-latest-security-investments-rsa-2020/

[27] Department of Defense (2018) Subpart 246.870, Contractors' Counterfeit Electronic Part Detection and Avoidance Systems, *Defense Federal Acquisition Regulation Supplement*. Publikacja dostępna pod adresem https://www.acq.osd.mil/dpap/dars/dfars/html/current/2468.htm#246.870-2

[28] Intel Corporation (2021) Intel 64 and IA-32 *Architectures Software Developer's Manual Volume 1: Basic Architecture*. Publikacja dostępna pod adresem https://software.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-software-developers-manual-volume-1-basic-architecture.html

[29] Nichols S (2020) *RIP ROP, COP, JOP? Intel to bring anti-exploit tech to market in this year's Tiger Lake chip family*. (The Register, San Francisco, CA). Publikacja dostępna pod adresem https://www.theregister.com/2020/06/15/intel_cet_tiger_lake

[30] Patel BV (2020) *A Technical Look at Intel's Control-flow Enforcement Technology*. (Intel Fellow Client Computing Group, Intel Corporation). Publikacja dostępna pod adresem https://software.intel.com/content/www/us/en/develop/articles/technical-look-control-flow-enforcement-technology.html

[31] Patel BV (2016) *Intel Releases New Technology Specifications to Protect Against ROP attacks*. (Intel Corporation).

[32] Shanbhogue V, Gupta D, Sahita R (2019) *Security Analysis of Processor Instruction Set Architecture for Enforcing Control-Flow Integrity*. (Hardware and Architectural Support for Security and Privacy '19: Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy). https://doi.org/10.1145/3337167.3337175

[33] Intel Corporation (2021) *Intel Architecture Instruction Set Extensions and Future Features Programming Reference.* Publikacja dostępna pod adresem https://software.intel.com/content/www/us/en/develop/download/intel-architecture-instruction-set-extensions-programming-reference.html

[34] Intel Corporation (2018) *Intel Security Features and Technologies Related to Transient Execution Attacks.* Publikacja dostępna pod adresem https://software.intel.com/content/www/us/en/develop/articles/software-security-guidance/best-practices/related-intel-security-features-technologies.html

[35] Anvin HP (2012) *Description x86: Supervisor Mode Access Prevention.* Publikacja dostępna pod adresem https://lwn.net/Articles/517251/

[36] Corbet J (2012) *Supervisor Mode Access Prevention.* Publikacja dostępna pod adresem https://lwn.net/Articles/517475/

[37] Intel Corporation (2022) *Strengthen Enclave Trust with Attestation*. Publikacja dostępna pod adresem https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/attestation-services.html

[38] European Telecommunications Standards Institute (2022) *Network Functions Virtualisation (NFV)*. Publikacja dostępna pod adresem https://www.etsi.org/technologies/nfv

[39] Intel Corporation (2020) *Intel® Trust Domain Extensions*. Publikacja dostępna pod adresem https://software.intel.com/content/dam/develop/external/us/en/documents/tdx-whitepaper-v4.pdf

[40] Intel Corporation (2022) *Intel AES New Instructions (Intel AES-NI)*. Publikacja dostępna pod adresem https://www.intel.com/content/www/us/en/architecture-and-technology/advanced- encryption-standard-aes/data-protection-aes-general-technology.html

[41] Intel Corporation (2012) *Flexible Workload Acceleration on Intel Architecture Lowers Equipment Cost*. Publikacja dostępna pod adresem https://www.intel.fr/content/dam/www/public/us/en/documents/white-papers/communications-quick-assist-paper.pdf

[42] Tadepalli, H (2017) *Intel QuickAssist Technology with Intel Key Protection Technology in Intel Server Platforms Based on Intel Xeon Processor Scalable Family*. (Intel Corporation).

[43] Intel Corporation (2022) *Intel® Security Libraries for Datacenter (Intel® SecL-DC)*. Publikacja dostępna pod adresem https://intel-secl.github.io/docs/4.2/

[44] Kaplan D, Powell J, Woller T (2016) *AMD Memory Encryption*. (Advanced Micro Devices, Inc.). Publikacja dostępna pod adresem https://developer.amd.com/wordpress/media/2013/12/AMD Memory Encryption W hitepaper_v7-Public.pdf

[45] Kaplan D (2017) *Protecting VM Register State with SEV-ES*. (Advanced Micro Devices, Inc.). Publikacja dostępna pod adresem https://www.amd.com/system/files/TechDocs/Protecting%20VM%20Register%20Stat e%20with%20SEV-ES.pdf

[46] Advanced Micro Devices, Inc. (2020) *AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More.* Publikacja dostępna pod adresem https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

[47] Arm (2022) *Arm Architecture Reference Manual for A-profile architecture*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/ddi0487/latest

[48] Arm (2020) *Trusted Base System Architecture for Armv8-A*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/den0021/f/?lang=en

[49] Arm (2022) *TrustZone for AArch64*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/102418/0100/What-is-TrustZone-?lang=en

[50] Parsec Project (2022) *Parsec.* Publikacja dostępna pod adresem https://github.com/parallaxsecond/parsec

[51] Parsec Project (2019) *Authenticators*. Publikacja dostępna pod adresem https://parallaxsecond.github.io/parsec-book/parsec

[52] SPIFFE (2022) *SPIFFE Verifiable Identity Document (SVID)*. Publikacja dostępna pod adresem https://spiffe.io/docs/latest/spiffe-about/spiffe-concepts/#spiffe-verifiable-identity-document-svid

[53] SPIFFE (2022) *SPIRE Concepts*. Publikacja dostępna pod adresem https://spiffe.io/docs/latest/spire-about/spire-concepts/

[54] SPIFFE (2022) *SPIFFE Workload API*. Publikacja dostępna pod adresem https://spiffe.io/docs/latest/spiffe- about/spiffe-concepts/#spiffe-workload-api

[55] Arm (2021) *Base System Architecture - Architecture Compliance Suite*. Publikacja dostępna pod adresem https://github. com/ARM-software/bsa-acs

[56] Ubuntu (2021) *Firmware Test Suite (fwts)*. Publikacja dostępna pod adresem https://wiki.ubuntu.com/FirmwareTestSuite/Reference

[57] Kerrisk M (2021) *Unix sockets peer credential checks.* Publikacja dostępna pod adresem https://man7.org/linux/man-pages/man2/getsockopt2.html

[58] Serebryany K, Stepanov E, Shlyapnikov A, Tsyrklevich V, Vyukov D (2018) *Memory Tagging and how it improves C/C++ memory safety*, Google. Publikacja dostępna pod adresem https://arxiv.org/ftp/arxiv/papers/1802/1802.09517.pdf

[59] Serebryany K (2019) *ARM Memory Tagging Extension and How It Improves C/C++ Memory Safety*. Publikacja dostępna pod adresem https://www.usenix.org/system/files/login/articles/login summer19 03 serebryany.pdf

[60] Watson RNM et al. (2020) *Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 8).* University of Cambridge Computer Laboratory, Technical Report Number 951. Publikacja dostępna pod adresem https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-951.pdf

[61] Arm (2022) *Arm Architecture Reference Manual Supplement - Morello for A-profile Architecture*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/ddi0606/latest

[62] Arm (2020) *Cache Speculation Side-channels, Version 2.5*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/102816/0205/

[63] Arm (2020) *Straight-line Speculation, Version 1.0*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/102825/0100/

[64] Arm (2021) *Arm v8.5-A/v9 CPU updates, Version 1.5*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/102822/0105/

[65] Arm (2022) *Introducing Arm Confidential Compute Architecture Version 2*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/den0125/latest

[66] Arm (2021) *The Realm Management Extension (RME), for Armv9-A*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/ddi0615/aa

[67] Arm (2021) *Arm Realm Management Extension (RME) System Architecture*. Publikacja dostępna pod adresem https://developer.arm.com/documentation/den0129/aa

[68] Lundblade L, Mandyam G, O'Donoghue J (2022) *The Entity Attestation Token (EAT)*. (Internet Engineering Task Force, RATS Working Group), IETF InternetDraft draft-ietf-rats-eat-12. https://datatracker.ietf.org/doc/html/draft-ietf-rats-eat

[69] Trusted Computing Group (2018) *Implicit Identity Based Device Attestation, Version 1.0*. Publikacja dostępna pod adresem https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Arch-Implicit-Identity-Based-Device-Attestation-v1-rev93.pdf

[70] Dyer JG, Lindemann M, Perez R, Sailer R, van Doorn L, Smith SW (2001) "*Building the IBM 4758 secure coprocessor*," in *Computer*, vol. 34, no. 10, pp. 57-66, Oct. 2001 https://ieeexplore.ieee.org/document/955100

[71] IBM (2022) *IBM Cloud Hyper Protect Crypto Services*. Publikacja dostępna pod adresem https://www.ibm.com/cloud/hyper-protect-crypto

[72] Trusted Computing Group (2019) *TPM 2.0 Library*. Publikacja dostępna pod adresem https://trustedcomputinggroup.org/resource/tpm-library-specification/

[73] Trusted Computing Group (2019) *Trusted Platform Module Library Part 1: Architecture*. Publikacja dostępna pod adresem https://trustedcomputinggroup.org/wp-content/uploads/TCGTPM2r1p59_Part1_Architecture_pub.pdf

[74] Engel C (2020) "*POWER9 Introduces Secure Boot to PowerVM*," *IBM* Power Systems Community. Publikacja dostępna pod adresem https://community.ibm.com/community/user/power/blogs/chris-engel1/2020/06/17/power9-introduces-secure-boot-to-powervm

[75] Heller D, Sastry N (2019) *OpenPower Secure and Trusted Boot, Part 2: Protecting System Firmware with OpenPOWER Secure Boot, IBM*. Publikacja dostępna pod adresem https://developer.ibm.com/articles/protect-system-firmware-openpower/

[76] IBM (2019) *z/Architecture Principles of Operation, Thirteenth Edition*. Publikacja dostępna pod adresem http://publibfp.dhe.ibm.com/epubs/pdf/a227832c.pdf

[77] *Security Target for PR/SM for IBM z14 and IBM LinuxONE Systems, Version: 16.12 Revision: 1, Status: RELEASE, Last Update: 2018-06-20*. (Nowsza wersja tego źródła jest dostępna pod adresem https://commoncriteriaportal.org/files/epfiles/1133bpdf.pdf.)

[78] IBM (2022) *IBM Hyper Protect Virtual Servers*. Publikacja dostępna pod adresem https://www.ibm.com/products/hyper-protect-virtual-servers

[79] OpenPOWER Foundation (2021) *How to build and run Secure VM using Ultravisor on an OpenPOWER machine*. Publikacja dostępna pod adresem https://github.com/open-power/ultravisor/wiki/How-to-build-and-run-Secure-VM-using-Ultravisor-on-a-OpenPOWER-machine

[80] OpenPOWER Foundation (2021) *Ultravisor*. Publikacja dostępna pod adresem https://github.com/open- power/ultravisor

[81] Joseph EK (2020) *Technical Overview of Secure Execution for Linux on IBM Z, IBM*. Publikacja dostępna pod adresem https://developer.ibm.com/blogs/technical-overview-of-secure-execution-for-linux-on-ibm-z/

[82] IBM (2006) *Secure Blue - Secure CPU Technology*. Publikacja dostępna pod adresem https://researcher.watson.ibm.com/researcher/view_page.php?id=6904

[83] Williams P, Boivie R (2011) "*Secure Blue++: CPU Support for Secure Executables*," Trust 2011, 4th International Conference on Trusted Computing, June 22-24, 2011, Pittsburgh, Pennsylvania.

[84] Boivie R, Williams P (2011) "*SecureBlue++: CPU Support for Secure Execution*," 2nd Annual NSA Trusted Computing Conference, September 20-22, 2011, Orlando, Florida.

[85] IBM (2019) "*IBM 4767-002 PCIe Cryptographic Coprocessor (HSM).*" Publikacja dostępna pod adresem https://public.dhe.ibm.com/security/cryptocards/pciecc2/docs/4767 PCIe Data Sheet.pdf

[86] IBM (2021) "*IBM CEX7S / 4769 PCIe Cryptographic Coprocessor (HSM).*" Publikacja dostępna pod adresem https://public.dhe.ibm.com/security/cryptocards/pciecc4/docs/4769

[87] IBM (2022) *CEX7S / 4769 Overview: IBM Systems cryptographic hardware products*. Publikacja dostępna pod adresem https://www.ibm.com/security/cryptocards/pciecc4/overview

[88] IBM (2021) *Protected-key CPACF*. Publikacja dostępna pod adresem https://www.ibm.com/docs/en/zos/2.3.0?topic=management-protected-key-cpacf

[89] *IBM TPM Attestation Client Server*. Publikacja dostępna pod adresem https://sourceforge.net/projects/ibmtpm20acs/files/AttestProv.doc/download?use mirror=phoenixnap

[90] *Integrity Measurement Architecture (IMA)*. Publikacja dostępna pod adresem https://sourceforge.net/p/linux-ima/wiki/Home/

[91] Sailer R, Zhang X, Jaeger T, van Doorn L (2004) "*Design and Implementation of a TCG-Based Integrity Measurement Architecture.*" *Proceedings of the 13th USENIX Security Symposium*, USENIX Association, Berkeley, CA, pp. 223-38. Publikacja dostępna pod adresem https://www.usenix.org/legacy/publications/library/proceedings/sec04/tech/fullpapers/sailer/sailerhtml/index.html

ZAŁĄCZNIK A – PRZYKŁADY TECHNOLOGII NIEZALEŻNYCH OD DOSTAWCY

W niniejszym załączniku opisano przykłady technologii niezależnych od dostawców, które odpowiadają kluczowym koncepcjom opisanym w różnych częściach tego dokumentu.

A.1 WERYFIKACJA INTEGRALNOŚCI PLATFORMY

A.1.1. UEFI Secure Boot (SB)

„UEFI Secure Boot (SB) to mechanizm weryfikacji zapewniający, że kod uruchamiany przez oprogramowanie układowe UEFI komputera jest zaufany" [19]. SB uniemożliwia złośliwemu oprogramowaniu „wykorzystanie punktów ataku przed uruchomieniem systemu, w tym samego wbudowanego w system oprogramowania układowego, a także odstępu czasu między inicjacją oprogramowania układowego a załadowaniem systemu operacyjnego" [20].

Podstawową koncepcją stojącą za mechanizmem SB jest podpisywanie plików wykonywalnych przy użyciu protokołu kryptograficznego z kluczem publicznym. Publiczny *klucz platformy* (*ang. Platform Key – PK*) może być przechowywany w oprogramowaniu układowym i używany jako klucz główny. W oprogramowaniu układowym mogą być również przechowywane publiczne klucze wymiany kluczy (*ang. Key Exchange Key – KEK*) w tak zwanej *bazie podpisów* (*ang. signature database)*. Ta baza danych zawiera klucze publiczne, które mogą być stosowane do weryfikacji różnych komponentów wykorzystywanych przez interfejs UEFI (np. sterowników), a także programów ładujących i systemów operacyjnych wczytywanych ze źródeł zewnętrznych (np. dysków, urządzeń USB, sieci). Baza podpisów może również zawierać *zabronione podpisy (ang. forbidden signatures*), które zawierają listę unieważnionych wcześniej kluczy. Baza podpisów powinna zawierać aktualną listę autoryzowanych i zabronionych kluczy utworzoną przez UEFI. Podpis pliku wykonywalnego jest weryfikowany na podstawie bazy podpisów przed jego uruchomieniem, a każda próba uruchomienia niezaufanego programu zostanie uniemożliwiona [19], [20].

Zanim klucz platformy zostanie wczytany do oprogramowania układowego, interfejs UEFI znajduje się w trybie konfiguracji (*ang. setup mode*), który umożliwia każdemu zapisanie klucza PK lub KEK w oprogramowaniu układowym. Zapis klucza PK przełącza oprogramowanie układowe w *tryb użytkownika* (*ang. user mode*). W trybie użytkownika klucze PK i KEK mogą być zapisywane tylko wtedy, gdy są podpisane przy użyciu prywatnej części klucza PK. Zasadniczo klucz PK służy do uwierzytelniania właściciela platformy, natomiast klucze KEK są używane do uwierzytelniania innych składników dystrybucji (distro), takich jak systemy operacyjne [20].

Shim to prosty pakiet oprogramowania przeznaczony do pracy jako program ładujący pierwszego poziomu w systemach UEFI. Jest to powszechnie stosowany fragment kodu, który jest uważany za bezpieczny, dobrze zrozumiały i audytowany, dzięki czemu można mu zaufać i podpisać za pomocą klucza platformy. Oznacza to, że dostawcy urzędowych certyfikatów (*ang. Certificate Authority - CA*) oprogramowania układowego muszą zadbać jedynie o podpisanie pakietu Shim, a nie wszystkich innych programów, które sprzedawcy mogą chcieć obsługiwać [19]. Shim staje się źródłem (RoT) dla wszystkich innych programów UEFI dostarczanych w dystrybucji. Osadza klucz CA specyficzny dla dystrybucji, który sam jest używany do podpisywania dodatkowych programów (np. Linux, GRUB, fwupdate). Umożliwia to płynne przekazanie zaufania. Dystrybucje są następnie odpowiedzialne za podpisanie reszty swoich pakietów. W idealnej sytuacji pakiet Shim nie będzie musiał być często aktualizowany, co powinno zmniejszyć obciążenie centralnych zespołów ds. audytu i certyfikacji [19].

Kluczowym elementem projektu pakietu Shim jest umożliwienie użytkownikom sterowania własnymi systemami. Klucz distro CA jest wbudowany w kod binarny Shim, ale istnieje również dodatkowa baza danych kluczy, którą może zarządzać użytkownik – tak zwany klucz właściciela maszyny (*ang. Machine Owner Key – MOK*). Klucze mogą być dodawane i usuwane z bazy MOK przez użytkownika, całkowicie niezależnie od klucza distro CA. Narzędzie *mokutil* może być używane do zarządzania kluczami z systemu operacyjnego Linux, ale zmiany kluczy MOK mogą być potwierdzane tylko bezpośrednio z konsoli podczas uruchamiania systemu. Pomaga to wyeliminować ryzyko, że złośliwe oprogramowanie systemu

operacyjnego potencjalnie zarejestruje nowe klucze, a tym samym ominie zabezpieczenie SB [19]. W systemach z układem TPM aktywowanym i obsługiwanym przez oprogramowanie układowe systemu, pakiet shim rozszerzy różne rejestry PCR o skróty ładowanych elementów docelowych [21]. Skróty certyfikatów są również rozszerzone na układ TPM, w tym skróty certyfikatów systemowych, dostawcy, MOK i shim, zarówno z zakazanych, jak i dozwolonych list.

A.2 KEYLIME

„Keylime to projekt open source prowadzony przez organizację Cloud Native Computing Foundation (CNCF), niezależne od dostawców forum zrzeszające ponad 145 organizacji użytkowników wykorzystujących technologie natywne dla chmury do tworzenia swoich produktów i usług w ramach wielu projektów open source, w tym na przykład Kubernetes. Keylime zapewnia wysoce skalowalne rozwiązanie do zdalnej atestacji programu rozruchowego (*ang. boot*) i pomiaru integralności środowiska uruchomieniowego. Keylime umożliwia użytkownikom monitorowanie zdalnych węzłów przy użyciu sprzętowego kryptograficznego źródła zaufania.

Projekt Keylime powstał w zespole badawczym ds. bezpieczeństwa w laboratorium MIT Lincoln Laboratory. Keylime zapewnia kompleksowe rozwiązanie do uruchamiania początkowego (*ang. bootstrapping*) ze sprzętowym zabezpieczeniem kryptograficznym dla maszyn zdalnych, a także dostarczania zaszyfrowanych ładunków (payloadów) i monitorowania integralności systemu środowiska uruchomieniowego. Zapewnia również elastyczne ramy dla zdalnej atestacji przy użyciu sprzętowego źródła zaufania opartego na module TPM. Użytkownicy mogą tworzyć własne niestandardowe działania, które będą uruchamiane, gdy urządzenie nie przejdzie pomiarów w ramach procesu atestacji.

Celem projektu Keylime jest uczynienie technologii TPM łatwo dostępną zarówno dla programistów, jak i użytkowników, bez konieczności dogłębnego zrozumienia niższych poziomów jej działania. Spośród wielu scenariuszy, szczególnie dobrze nadaje się dla użytkowników, którzy muszą zdalnie atestować maszyny niebędące pod ich pełną kontrolą (np. dla klienta chmury hybrydowej lub zdalnego urządzenia

brzegowego / IoT w niezabezpieczonej fizycznej lokalizacji podatnej na manipulacje). RozwiązanieKeylime może być wykorzystywane do monitorowania całej floty węzłów roboczych OpenShift i podejmowania natychmiastowych działań w przypadku naruszenia bezpieczeństwa któregokolwiek z nich. Mierzy zaufany rozruch maszyn i integralność środowiska uruchomieniowego przy użyciu architektury do pomiaru integralności jądra systemu Linux (*ang. Linux Kernels Integrity Measurement Architecture*)".

ZAŁĄCZNIK B – PRZYKŁADY TECHNOLOGII FIRMY INTEL

W niniejszym załączniku opisano przykłady technologii firmy Intel, które odpowiadają kluczowym koncepcjom opisanym w różnych częściach tego dokumentu.

B.1 WERYFIKACJA INTEGRALNOŚCI PLATFORMY

B.1.1. Łańcuch zaufania (ang. Chain of Trust – CoT)

B.1.1.1. Trusted Execution Technology (TXT) firmy Intel

Technologia Trusted Execution Technology (TXT) firmy Intel w połączeniu z modułem TPM zapewnia sprzętowe źródło zaufania dostępne na platformach serwerowych i klienckich firmy Intel, które obejmuje „funkcje bezpieczeństwa, takie jak sterowane uruchamianie i chronione wykonywanie" [22]. W technologii TXT wykorzystywane są uwierzytelnione moduły kodu (*ang. Authenticated Code Module - ACM*), które mierzą różne elementy łańcucha zaufania podczas uruchamiania systemu i rozszerzają je na moduł TPM platformy [2], [22]. Moduły ACM TXT to podpisany kod binarny specyficzny dla mikroukładu, który jest wywoływany w celu wykonania funkcji wymaganych do uruchomienia środowiska TXT. Kod ACM jest ładowany i wykonywany z pamięci podręcznej CPU w obszarze zwanym pamięcią RAM uwierzytelnionego kodu (*ang. Authenticated Code RAM – AC RAM*). Mikrokod procesora, który działa jako *główne źródło zaufanych pomiarów* (*ang. Core Root of Trust for Measurement – CRTM*), uwierzytelnia kod ACM poprzez weryfikację dołączonego podpisu cyfrowego względem klucza publicznego producenta z jego skrótem zapisanym na stałe w mikroukładzie. Kod ACM, załadowany do chronionej pamięci wewnątrz procesora, wykonuje różne testy i weryfikacje konfiguracji mikroukładu i procesora.

Moduły ACM potrzebne do zainicjowania środowiska TXT to BIOS i Secure Initialization (SINIT). Oba są zwykle dostarczane w ramach obrazu systemu BIOS platformy. Moduł uwierzytelnionego kodu Secure Initialization (*ang. Secure Initialization Authenticated Code Module – SINIT ACM*) może być również dostarczony na dysku [2], [23]. Kod BIOS ACM jest odpowiedzialny za pomiar oprogramowania układowego BIOS do modułu TPM i wykonuje dodatkowe operacje bezpieczeństwa oparte na systemie BIOS. W najnowszej wersji technologii TXT połączonej z technologią Intel Boot Guard

oznaczono ten kod ACM jako Startup ACM, aby odróżnić go od starszego BIOS ACM. Kod SINIT ACM służy do pomiaru oprogramowania systemowego lub systemu operacyjnego do modułu TPM i „inicjalizuje platformę, aby system operacyjny mógł przejść do bezpiecznego trybu pracy" [23].

Po zakończeniu procedur uruchamiania systemu BIOS sterowanie jest przekazywane do modułu ładującego system operacyjny. W systemie z obsługą technologii TXT moduł ładujący system operacyjny jest instruowany, aby załadować specjalny moduł o nazwie Trusted Boot przed załadowaniem pierwszego modułu jądra [23]. Trusted Boot (tboot) to moduł uruchamiany przed jądrem / menedżerem maszyny wirtualnej (*ang. Virtual Machine Manager – VMM*) typu open-source, który integruje się z technologią TXT w celu wykonania mierzonego uruchomienia jądra systemu operacyjnego / VMM. Kod tboot składa się zazwyczaj z dwóch części: preambuły i zaufanego rdzenia. Preambuła tboot jest najczęściej wykonywana przez moduł ładujący systemu operacyjnego, ale może też zostać załadowana w jego środowisku uruchomieniowym. Preambuła tboot jest odpowiedzialna za przygotowanie parametrów wejściowych modułu SINIT i domyślnie jest niezaufana. Wykonuje instrukcję procesora, która przekazuje sterowanie do mikrokodu procesora. Mikrokod ładuje moduł SINIT do pamięci AC RAM, uwierzytelnia go, mierzy SINIT do modułu TPM i przekazuje mu sterowanie. Moduł SINIT weryfikuje konfigurację platformy i egzekwuje wszelkie istniejące zasady sterowania uruchamianiem, mierząc je i przekazując zaufany rdzeń tboot do modułu TPM. Zaufany rdzeń tboot przejmuje sterowanie i kontynuuje łańcuch zaufania, mierząc jądro systemu operacyjnego i dodatkowe moduły (takie jak initrd) przed przekazaniem sterowania do systemu operacyjnego [24].

Technologia Intel TXT obejmuje funkcję silnika zasad, która zapewnia możliwość określenia znanych dobrych konfiguracji platformy. Takie zasady sterowania uruchamianiem (*ang. Launch Control Policiy – LCP*) określają, które oprogramowanie systemowe może wykonać bezpieczne uruchomienie. Zasady LCP mogą wymuszać użycie określonych konfiguracji platformy i wersji zaufanego rdzenia tboot wymaganych do uruchomienia środowiska systemowego [23].

B.1.1.2. Boot Guard firmy Intel

Technologia Boot Guard firmy Intel obejmuje sprzętowe źródło zaufania do uwierzytelniania systemu BIOS. Producent oryginalnego sprzętu (OEM) umożliwia uwierzytelnianie przy pomocy technologii Boot Guard poprzez trwałe połączenie zasad i klucza publicznego należącego do OEM z układem scalonym na linii produkcyjnej serwerów. Gdy procesor Intel rozpozna, że platforma jest wyposażona w technologię Boot Guard, uwierzytelnia i uruchamia moduł ACM. Moduł ACM ładuje wstępny BIOS lub początkowy blok rozruchowy (*ang. Initial Boot Block – IBB*) do pamięci podręcznej procesora, uwierzytelnia go przy użyciu wbudowanego klucza publicznego OEM i mierzy go do modułu TPM.

Jeśli uwierzytelnienie IBB przebiegnie prawidłowo, weryfikuje on pozostałe oprogramowanie układowe BIOS, ładuje je do pamięci i przekazuje sterowanie wykonaniem. Blok IBB pełni tylko tę ograniczoną funkcję, dzięki czemu ma wystarczająco mały rozmiar, aby zmieścić się w pamięci podręcznej procesora Intel. Jeśli uwierzytelnianie przeprowadzane przez technologię Boot Guard nie powiedzie się, wymusza ona wyłączenie systemu. Po zakończeniu wykonywania Boot Guard, łańcuch zaufania może być kontynuowany dla innych komponentów za pośrednictwem technologii SB. Rozwiązanie TXT może być stosowane w połączeniu z tymi technologiami w celu zapewnienia dynamicznego, zaufanego uruchamiania jądra systemu operacyjnego i oprogramowania.

Ponieważ technologia Boot Guard jest wbudowana w układ elektroniczny i uwierzytelnia startowy BIOS z pamięci podręcznej procesora, zapewnia odporność na niektóre klasy ataków fizycznych. W technologii Boot Guard zabezpieczenia stosuje się również w celu zapewnienia trwałego wycofania naruszonych modułów ACM, obrazów systemu BIOS i zasad wprowadzania danych.

B.1.1.3. Platforma Firmware Resilience (PFR) firmy Intel

Technologia Intel Platform Firmware Resilience (PFR) to rozwiązanie działające na poziomie platformy, które stanowi otwarte źródło zaufania oparte na programowalnym układzie logicznym. Ma ona na celu zapewnienie odporności oprogramowania układowego (zgodnie z publikacją NIST SP 800-193 [4])

i kompleksowej ochrony różnych komponentów oprogramowania układowego platformy, w tym systemu BIOS, oprogramowania układowego usług platformy serwera (*ang. Server Platform Services Firmware – SPS FW*) i kontrolerów BMC (*ang. Board Management Controller – BMC*). Rozwiązanie PFR zapewnia właścicielowi platformy minimalną zaufaną bazę przetwarzania (*ang. Trusted Compute Base – TCB*) znajdującą się pod jego pełną kontrolą. TCB zapewnia uwierzytelnianie kryptograficzne i automatyczne odzyskiwanie oprogramowania układowego platformy, aby umożliwić jej prawidłowe działanie i powrót do znanego dobrego stanu w przypadku złośliwego ataku lub błędu operatora, takiego jak nieudana aktualizacja.

W publikacji NIST SP 800-193 [4] przedstawiono trzy główne wytyczne mające na celu zapewnienie odporności platform na potencjalnie destrukcyjne ataki:

- **Ochrona:** Mechanizmy zapewniające, że kod oprogramowania układowego platformy i krytyczne dane pozostają w stanie integralności i są chronione przed naruszeniem, takie jak proces zapewniania autentyczności i integralności aktualizacji oprogramowania układowego.
- **Wykrywanie:** Mechanizmy wykrywania uszkodzeń kodu oprogramowania sprzętowego platformy i krytycznych danych
- **Przywracanie:** Mechanizmy przywracania kodu oprogramowania układowego platformy i krytycznych danych do stanu integralności w przypadku wykrycia, że taki kod oprogramowania układowego lub krytyczne dane zostały naruszone, lub w przypadku wymuszenia ich przywrócenia za pomocą autoryzowanego mechanizmu. Przywracanie jest ograniczone do możliwości odzyskania kodu oprogramowania układowego i krytycznych danych.

Ponadto, w publikacji NIST SP 800-193 [4] zawarto wytyczne umożliwiające spełnienie tych wymagań przy użyciu trzech głównych funkcji źródła zaufania platformy:

- **Źródło zaufania do aktualizacji**, które jest odpowiedzialne za uwierzytelnianie aktualizacji oprogramowania układowego i zmian krytycznych danych w celu wsparcia funkcji bezpieczeństwa platformy. Obejmuje to weryfikację podpisów aktualizacji oprogramowania układowego, a także zabezpieczenia przed wycofaniem podczas aktualizacji.

- **Źródło zaufania do wykrywania**, które odpowiada za wykrywanie naruszeń oprogramowania układowego i krytycznych danych.
- **Źródło zaufania do przywracania**, które jest odpowiedzialne za przywracanie oprogramowania układowego i krytycznych danych po wykryciu ich naruszenia lub na polecenie administratora.

Rozwiązanie PFR zostało zaprojektowane tak, aby wspierać realizację wytycznych NIST i umożliwiać stworzenie odpornej platformy, która jest w stanie samoczynnie odzyskać sprawność po wykryciu ataku lub naruszenia oprogramowania układowego. Obejmuje to weryfikację całego oprogramowania układowego i konfiguracji platformy podczas jej włączania, aktywną ochronę pamięci nieulotnej platformy w środowisku uruchomieniowym oraz aktywną ochronę szeregowego interfejsu urządzenia peryferyjnego (*ang. Serial Peripheral Interface – SPI flash*) i magistrali zarządzania systemem (*ang. System Management Bus – SMBus*). Technologia PFR umożliwia również monitorowanie postępu rozruchu komponentów platformy i zapewnia automatyczne przywracanie oprogramowania układowego do znanego dobrego stanu po wykryciu jego uszkodzenia lub naruszenia konfiguracji. Cel ten jest osiągany dzięki wykorzystaniu bezpośrednio programowalnej macierzy bramek (*ang. Field-Programmable Gate Array – FPGA*) do ustanowienia źródła zaufania.

Technologia PFR definiuje specjalny tryb przed rozruchem (T-1), w którym aktywny jest tylko moduł PFR FPGA. Procesory Intel Xeon i inne urządzenia, które mogą potencjalnie wpływać na proces rozruchu, takie jak mikroukład Platform Controller Hub (PCH)/Manageability Engine (ME) i kontroler BMC, nie są zasilane. W trybie T-1 przeprowadzana jest kryptograficzna weryfikacja oprogramowania układowego o krytycznym znaczeniu dla rozruchu, takiego jak BIOS, ME i BMC. W przypadku jego naruszenia inicjowane jest zdarzenie odzyskiwania, a uszkodzone oprogramowanie układowe w aktywnych regionach pamięci SPI flash jest usuwane i przywracane za pomocą znanej dobrej kopii do odzyskiwania. Po powodzeniu system uruchamia się w normalnym trybie, wykorzystując technologię Boot Guard jako statyczne źródło zaufania.

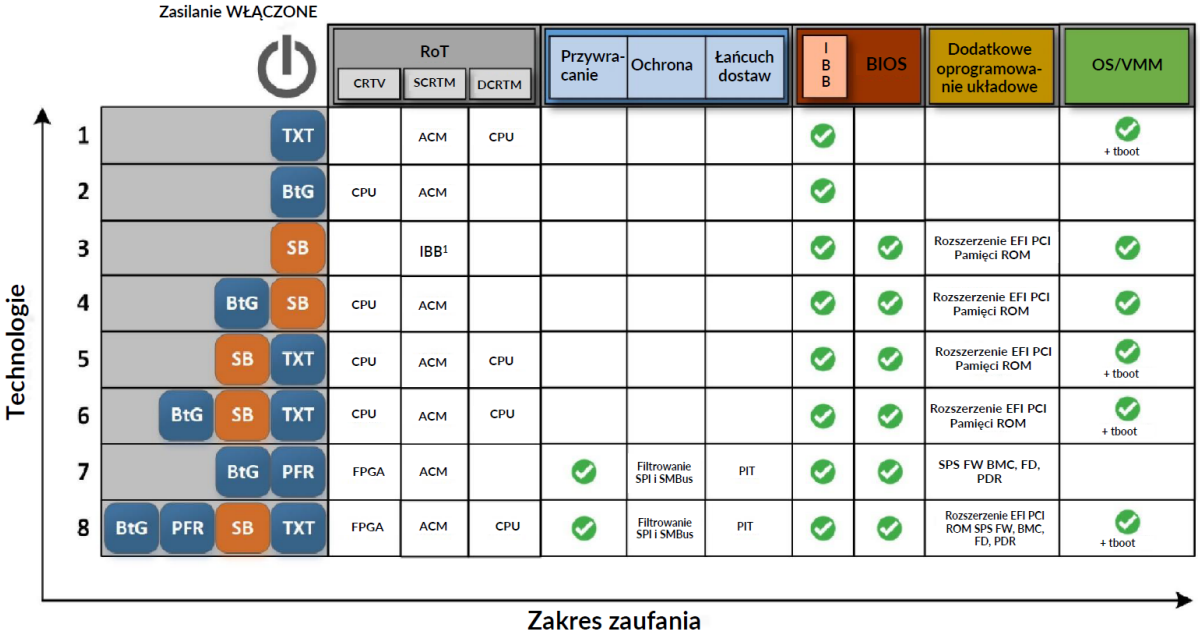
Źródło zaufania PFR FPGA wykorzystuje hierarchię kluczy do uwierzytelniania struktur danych znajdujących się w pamięci SPI flash. Hierarchia kluczy opiera się na zapewnionym kluczu głównym (*ang. Root Key – RK*) przechowywanym w pamięci NVRAM źródła zaufania FPGA oraz strukturze klucza podpisywania kodu (*ang. Code Signing Key – CSK*), która jest zatwierdzana przy użyciu klucza głównego, przechowywana w pamięci flash SPI i używana do podpisywania struktur danych niższego poziomu. Technologia PFR FPGA wykorzystuje klucz CSK do weryfikacji podpisu cyfrowego manifestu oprogramowania układowego platformy, w którym opisano oczekiwane wyniki pomiarów tego oprogramowania. Źródło zaufania PFR FPGA weryfikuje te pomiary przed zezwoleniem na uruchomienie systemu. Gdy konieczne jest odzyskanie oprogramowania, ponieważ pomiary nie są zgodne z oczekiwaną wartością lub ponieważ podczas uruchamiania systemu wykryto zawieszenie, źródło zaufania PFR FPGA wykorzystuje obraz odzyskiwania do przywrócenia oprogramowania układowego. Obraz odzyskiwania i wszelkie obrazy aktualizacji są przechowywane w skompresowanym formacie kapsuły i weryfikowane przy użyciu podpisu cyfrowego.

Każdy magazyn oprogramowania układowego platformy jest podzielony na trzy główne sekcje: aktywną, odzyskiwania i przejściową. Regiony odzyskiwania, a także statyczne części aktywnych regionów, są chronione przed zapisem z innych komponentów platformy przez źródło zaufania PFR FPGA. Region przejściowy umożliwia zapis innym komponentom platformy w celu zapewnienia obszaru do umieszczania cyfrowo podpisanych i skompresowanych kapsuł aktualizacji, które są następnie weryfikowane przez źródło zaufania PFR FPGA przed przekazaniem do regionów aktywnych lub odzyskiwania. Kopia odzyskiwania może zostać zaktualizowana w trybie T-1, jeśli źródło zaufania PFR FPGA zweryfikuje podpis cyfrowy kapsuły aktualizacji i potwierdzi, że potencjalny obraz odzyskiwania jest obrazem rozruchowym.

B.1.1.4. Podsumowanie przykładów technologii firmy Intel

Istnieje kilka technologii zapewniających różne poziomy integralności i zaufania do platformy. Pojedyncze technologie nie zapewniają pełnego łańcucha zaufania.

W połączeniu mogą one zapewniać kompleksową ochronę aż do warstwy systemu operacyjnego i VMM.Na rysunku 8 przedstawiono zakres ochrony oprogramowania układowego i oprogramowania dla każdego istniejącego przykładu technologii łańcucha zaufania.



Rysunek 8: Zakres ochrony oprogramowania układowego i oprogramowania przez istniejące technologie łańcucha zaufania

Na rysunku 8 zidentyfikowano komponenty każdej technologii, które składają się na źródło zaufania w ich własnych łańcuchach, a także pokazano ogólny zakres ochrony oprogramowania układowego i oprogramowania, jaki zapewnia każda technologia.

Ponieważ dostępnych jest wiele technologii, wybór odpowiedniej kombinacji do wdrożenia może być trudny. Na rysunku 8 przedstawiono możliwe kombinacje technologii, które rozszerzają pomiary na moduł TPM w celu atestacji integralności platformy. Należy zauważyć, że każda kombinacja obejmuje co najmniej jedną technologię sprzętową, aby zapewnić implementację źródła zaufania. Można również wdrożyć dodatkową opcję rozszerzenia łańcucha zaufania dalej niż do systemu operacyjnego. Wdrożenie tylko technologii sprzętowych naruszyłoby łańcuch zaufania, zapewniając pomiary integralności tylko dla oprogramowania układowego uruchamianego przed systemem operacyjnym. W przypadku korzystania wyłącznie

z technologii SB jako źródło zaufania używane będzie oprogramowanie sprzętowe, które nie posiada sprzętowych zabezpieczeń i jest znacznie bardziej podatne na ataki. Dzięki uwzględnieniu obu tych części, łańcuch zaufania może zostać rozszerzony ze sprzętowego źródła zaufania na system operacyjny i jeszcze dalej.

Te kombinacje umożliwiają zapewnienie, że odpowiednie pomiary zostaną rozszerzone na TPM w celu atestacji integralności i mogą uniemożliwić uruchomienie serwera, jeśli określone moduły bezpieczeństwa zostaną naruszone. Mechanizmy atestacji zapewniane przez te technologie dostarczają kryptograficznego dowodu integralności mierzonych komponentów, który może być wykorzystany do zapewnienia wglądu w konfiguracje bezpieczeństwa platformy i udowodnienia jej integralności. Należy zwrócić uwagę na połączenie technologii UEFI SB z TXT na rysunku 8. Kombinacja ta zapewnia możliwość weryfikacji podpisu UEFI SB przed pomiarem integralności modułu tboot w warstwie OS/VMM.

Oprócz funkcji atestacji, technologia PFR umożliwia weryfikację oprogramowania układowego platformy i automatyczne przywracanie naruszonego oprogramowania układowego do znanych, dobrych wersji. Rozwiązanie PFR może współpracować z dowolną kombinacją technologii CoT, zapewniając ochronę i odporność na wektory ataków związane z oprogramowaniem układowym. Połączenie technologii sprzętowych zabezpieczeń oprogramowania układowego, takich jak PFR, z konfiguracją łańcucha zaufania opartą na sprzęcie jest częścią wielowarstwowej strategii bezpieczeństwa.

B.1.2. Ochrona łańcucha dostaw

B.1.2.1. Transparent Supply Chain (TSC) firmy Intel

„Intel Transparent Supply Chain (TSC) to zbiór zasad i procedur wdrożonych w fabrykach ODM, które umożliwiają użytkownikom końcowym sprawdzenie, gdzie i kiedy każdy komponent platformy został wyprodukowany” [25]. „Narzędzia TSC firmy Intel umożliwiają producentom platform powiązanie informacji o platformie i pomiarów za pomocą [modułu TPM]. Umożliwia to klientom identyfikowalność i rozliczalność platform dzięki raportowaniu na poziomie komponentów” [26].

Technologia TSC firmy Intel zapewnia następujące kluczowe funkcje [25]:

- Cyfrowo podpisane oświadczenie o zgodności dla każdej platformy.
- Certyfikaty platformy powiązane z dyskretnym modułem TPM, zapewniające identyfikowalność na poziomie systemu.
- Identyfikowalność na poziomie komponentów przy użyciu bezpośredniego pliku danych o platformie, który zawiera informacje o zintegrowanych komponentach, w tym procesorze, pamięci masowej i operacyjnej oraz kartach rozszerzeń.
- Narzędzie Auto Verify, które porównuje migawkę bezpośrednich danych platformy zebranych podczas produkcji z migawką komponentów platformy wykonaną przy pierwszym uruchomieniu.
- Weryfikacja załadowania oprogramowania układowego

B.1.2.2. Rozwiązanie PFR z zabezpieczeniem Protection in Transit (PIT) firmy Intel

Oprócz funkcji ochrony platformy, wykrywania i odzyskiwania, technologia PFR zapewnia również ochronę podczas transportu (*ang. Protection In Transit - PIT*) lub ochronę łańcucha dostaw. Blokada platformy wymaga obecności hasła w obrębie technologii PFR FPGA, a także komponentu wykorzystującego częstotliwość radiową (*ang. Radio Frequency – RF*). Hasło jest usuwane przed wysyłką platformy i musi zostać wymienione przed jej uruchomieniem. W przypadku uszczelniania oprogramowania układowego platformy, technologia PFR FPGA oblicza skróty oprogramowania układowego platformy w podłączonych urządzeniach flash PCH i BMC, w tym w obszarach statycznych i dynamicznych, i przechowuje je w pamięci NVRAM przed wysyłką. Po dostarczeniu, technologia PFR FPGA ponownie oblicza skróty i zgłasza wszelkie niezgodności, aby potwierdzić, że oprogramowanie układowe nie zostało zmodyfikowane podczas transportu systemu.

B.2 MECHANIZMY OCHRONY ŚRODOWISKA URUCHOMIENIOWEGO OPROGRAMOWANIA

B.2.1. Ataki typu ROP (ang. Return Oriented Programming – ROP) oraz COP/JOP (ang. Call/Jump Oriented Programming – COP/JOP)

B.2.1.1. Control-Flow Enforcement Technology firmy Intel (Intel CET)

Intel CET to rozszerzenie zestawu instrukcji służące do implementacji integralności przepływu sterowania i obrony przed atakami typu *ROP* i *COP*/JOP. *ROP*, podobnie jak *COP*/JOP, to dominująca metodologia ataków stosowana przez twórców ukrytych exploitów wykorzystujących luki w zabezpieczeniach programów [28].

Technologia Intel CET zapobiega tej klasie ataków, zapewniając następujące funkcje:

- Stos w tle (*ang. shadow stack*) – ochrona adresu zwrotnego w celu przeciwdziałania atakom typu *ROP*
- Pośrednie śledzenie odgałęzień (*ang. iIndirect branch tracking*) - ochrona wolnych odgałęzień w celu przeciwdziałania atakom typu *COP*/JOP

„Technologia CET wprowadza system stosu w tle w celu wykrywania i udaremniania manipulacji stosem wymaganym do przeprowadzenia ataku typu *ROP*" [29]. Ten drugi stos jest używany wyłącznie do operacji transferu sterowania. Został zaprojektowany tak, że jest chroniony przed dostępem z pamięci kodu aplikacji, i jednocześnie śledzi przechowywane przez procesor kopie adresów zwrotnych [30]. „Gdy technologia CET jest włączona, instrukcja CALL przesuwa adres zwrotny na stos w tle oprócz normalnego działania polegającego na przesunięciu adresu zwrotnego na normalny stos (bez zmian w tradycyjnym działaniu stosu). Instrukcje zwrotu (np. RET) powodują zdjęcie adresu zwrotnego zarówno ze stosu w tle, jak i tradycyjnego, i przekazują sterowanie do zdejmowanego adresu tylko wtedy, gdy adresy zwrotne z obu stosów są zgodne. [...] Zabezpieczenia tabeli stron dla stosu w tle również zostały zaprojektowane z myślą o ochronie integralności stosu w tle i/lub jego przepełnieniu lub niedopełnieniu" [31].

„Technologia CET wprowadza również możliwość pośredniego śledzenia odgałęzień, aby zapewnić oprogramowaniu możliwość przeciwdziałania atakom typu *COP*/JOP"

[30]. Instrukcja ENDBRANCH jest nowym dodatkiem do architektury zestawu instrukcji (*ang. Intel Instruction Set Architecture - ISA*) firmy Intel. Oznacza ona dozwolone cele dla pośredniego rozgałęzienia lub skoku, wymuszając wygenerowanie przez procesor wyjątku dla niezamierzonych lub złośliwych operacji [31].

„Intel aktywnie współpracuje z firmą Microsoft i innymi partnerami branżowymi w celu rozwiązania problemu przejęcia przepływu sterowania. Technologia CET firmy Intel została wykorzystana do rozszerzenia wcześniejszych rozwiązań w zakresie integralności przepływu sterowania opartych wyłącznie na oprogramowaniu. Technologia CET firmy Intel, jeśli jest prawidłowo wykorzystywana przez oprogramowanie, stanowi duży krok w kierunku przeciwdziałania wykorzystywaniu instrukcji przekazywania przepływu sterowania przez atakujących" [31]. Analiza bezpieczeństwa technologii Intel CET została opublikowana w dokumencie [32].

B.2.2. Ataki wykorzystujące translację adresów

B.2.2.1. Hypervisor Managed Linear Address Translation (HLAT) firmy Intel

Liniowa translacja adresów zarządzana przez funkcję hypervisor (*ang. Hypervisor Managed Linear Address Translation – HLAT*) to funkcja umożliwiająca monitorom zabezpieczeń opartym na technologii Intel Virtualization Technology (Intel VT-x) wymuszanie ochrony środowiska uruchomieniowego i zapewnień integralności w tabelach stron zarządzanych przez system operacyjny. Pomaga to chronić zasoby jądra, a także wewnątrzpasmowych agentów bezpieczeństwa i monitorowane przez nich zasoby, przed atakami na tabelę stron systemu operacyjnego.

„Technologia [HLAT] jest przeznaczona do użycia przez funkcję hypervisor/monitor maszyny wirtualnej (*ang. Virtual Machine Monitor – VMM*) w celu wymuszenia liniowej translacji gościa (na fizyczne mapowania gościa). W połączeniu z istniejącą funkcją Extended Page Table (EPT), technologia HLAT umożliwia VMM zapewnienie integralności połączonych liniowych translacji gościa (mapowań i uprawnień) buforowanych przez TLB procesora, poprzez zredukowaną programową TCB zarządzaną przez VMM" [33]. W ten sposób wymuszone przez VMM translacje gościa

są lepiej chronione przed zmianami dokonywanymi przez niezaufanych atakujących, których celem jest oprogramowanie systemowe [33].

Funkcja ta ma na celu rozszerzenie funkcji bezpieczeństwa dla typu monitora VMM, który może wykorzystywać starsze bity uprawnień EPT do odczytu/zapisu/wykonania XWR) (bity 2:0 EPTE), a także nowy bit dostępu użytkownik-wykonanie (XU) (bit 10 EPTE) w celu zapewnienia integralności kodu/danych rezydujących w pamięci fizycznej gościa przypisanej do systemu operacyjnego gościa. Uprawnienia EPT są również używane w ramach monitorów VMM do izolowania pamięci, na przykład do hostowania bezpiecznego jądra (*ang. Secure Kernel – SK*), które może zarządzać właściwościami bezpieczeństwa dla jądra ogólnego przeznaczenia (*ang. General Purpose Kernel – GPK*). W przypadku takich zastosowań ważne jest, aby monitor VMM zapewniał, że liniowe mapowania adresów gościa, które są używane przez jądro ogólnego przeznaczenia (GPK) do odwoływania się do monitorowanych przez EPT fizycznych stron gościa, są również kontrolowane pod względem dostępu" [33].

„Monitory VMM mogą wymuszać integralność tych specyficznych liniowych mapowań fizycznych gościa (struktur stronicowania) poprzez wykorzystanie starszych uprawnień EPT do oznaczania pamięci fizycznej gościa zawierającej odpowiednie struktury stronicowania gościa jako tylko do odczytu. Celem oznaczenia tych struktur stronicowania gościa jako tylko do odczytu jest zapewnienie, że oprogramowanie gościa nie utworzy nieprawidłowego mapowania. Niemniej jednak powszechnie wiadomo, że takie techniki sterowania edycją tabeli stron powodują bardzo wysokie koszty ogólne ze względu na wymóg monitorowania przez funkcję VMM wszystkich kontekstów stronicowania utworzonych przez system operacyjny (gościa). Technologia HLAT umożliwia funkcji VMM wymuszanie integralności mapowań liniowych gościa bez tak wysokich kosztów ogólnych" [33].

Technologia HLAT wykorzystuje mechanizm procesora, który implementuje alternatywną strukturę stronicowania architektury Intel Itanium (IA) zarządzaną w pamięci fizycznej gościa przez bezpieczne jądro (*ang. Secure Kernel*). Ta struktura stronicowania zawiera liniowe translacje fizyczne gościa, które funkcja VMM / bezpieczne jądro chce wymusić.

Dodatkowo, aby uwzględnić starsze podejścia do monitorowania tabeli stron, technologia HLAT definiuje dwa nowe bity kontrolne EPT we wpisach liści EPT. Bit kontrolny „Paging-Write" określa, które fizyczne strony gościa zawierają struktury stronicowania HLAT lub starsze struktury stronicowania IA. Umożliwia to procesorowi użycie bitu Paging-Write jako uprawnienia do wykonywania zapisów bitów A/D, zamiast programowego uprawnienia w EPTE. Bit kontrolny „Verify Paging-Write" określa, które fizyczne strony gościa powinny być przywoływane tylko przez struktury stronicowania translacji (gościa) oznaczone jako zapisywalne w ramach stronicowania w EPT [33].

B.2.2.2. Supervisor Mode Execution Prevention (SMEP) oraz Supervisor Mode Access Prevention (SMAP) firmy Intel

Technologie Supervisor Mode Execution Prevention (SMEP) i Supervisor Mode Access Prevention (SMAP) to opcjonalne funkcje, które mogą być wykorzystywane przez oprogramowanie systemowe (takie jak jądro) w celu wzmocnienia separacji uprawnień między trybem użytkownika a trybem jądra. Technologie te dodatkowo wymuszają właściwości użytkownika / nadzorcy (*ang. supervisor*) określone za pomocą mechanizmów translacji adresów, przeciwdziałając wykonywaniu złośliwego kodu lub złośliwemu wykorzystaniu konfiguracji danych przez procesy działające w trybie użytkownika.

Rozwiązanie Intel OS Guard, znane również jako SMEP, pomaga zapobiegać wykonywaniu kodu z niezaufanej pamięci aplikacji podczas pracy na bardziej uprzywilejowanym poziomie (nadzorcy). „[Po] jego aktywacji, system operacyjny nie będzie mógł bezpośrednio wykonywać kodu aplikacji, nawet spekulatywnie. Dzięki temu ataki typu BTI (*ang. Branch Target Injection*) na system operacyjny są znacznie trudniejsze, ponieważ atakujący musi znaleźć gadżety w kodzie systemu operacyjnego. Aplikacji trudniej jest również przerobić kod systemu operacyjnego, aby przeskakiwał do gadżetu systemu operacyjnego. We wszystkich najpopularniejszych systemach operacyjnych obsługa funkcji SMEP jest domyślnie włączona" [34].

SMAP to funkcja bezpieczeństwa, która uniemożliwia nieautoryzowane wykorzystanie przez jądro danych dostępnych dla obszaru użytkownika [35]. Włączenie bitu SMAP w rejestrze kontrolnym CR4 spowoduje wyzwolenie błędu strony przy próbie dostępu do pamięci przestrzeni użytkownika podczas pracy w trybie uprzywilejowanym. Gdy

istnieje konieczność uzyskania dostępu do pamięci przestrzeni użytkownika przez jądro, włączana jest osobna flaga AC, aby zezwolić na wymagany dostęp [36]. „Dwie nowe instrukcje (STAC i CLAC) umożliwiają stosunkowo szybkie manipulowanie tą flagą". Gdy w normalnych warunkach pracy flaga AC jest ustawiona w tryb ochrony, SMAP blokuje całą klasę ataków, w ramach których jądro jest wprowadzane w błąd w celu odczytu (lub zapisu) z pamięci przestrzeni użytkownika. Funkcja SMAP umożliwia również wczesne wykrywanie błędów jądra, w których programiści dokonują dereferencji[10] wskaźników przestrzeni użytkownika bezpośrednio z jądra [36].

B.3 OCHRONA DANYCH I POUFNOŚĆ PRZETWARZANIA

B.3.1. Izolacja pamięci

B.3.1.1. Technologie Intel TME oraz Intel Multi-Key TME (Intel MKTME)

Technologia Intel Total Memory Encryption (Intel TME) zapewnia możliwość szyfrowania całej pamięci fizycznej systemu. Funkcja ta jest zwykle włączana na bardzo wczesnych etapach procesu rozruchu dzięki niewielkiej zmianie w systemie BIOS. Po skonfigurowaniu i zablokowaniu tej zmiany wszystkie dane na zewnętrznych magistralach pamięci procesora i wszelkich dodatkowych modułów DIMM będą szyfrowane przy użyciu 128-bitowych kluczy wykorzystujących standardowy algorytm NIST o nazwie AES-XTS. Klucz szyfrowania używany w ramach technologii Intel TME wykorzystuje sprzętowy generator RNG zaimplementowany w procesorze Intel, a klucze nie są dostępne za pośrednictwem oprogramowania ani zewnętrznych interfejsów do procesora. Architektura jest elastyczna i w przyszłości będzie obsługiwać dodatkowe schematy ochrony pamięci. Technologia Intel TME jest przeznaczona do obsługi niezmodyfikowanego istniejącego oprogramowania systemu i aplikacji. Ogólny wpływ technologii TME na wydajność będzie prawdopodobnie stosunkowo niewielki i w dużym stopniu zależny od obciążenia.

Technologia Intel Multi-Key Total Memory Encryption (Intel MKTME) bazuje na Intel TME i dodaje obsługę wielu kluczy szyfrowania. Implementacja procesora obsługuje

[10] Dereferencja – zamiana referencji lub wskaźnika (identyfikatora obiektu lub adresu zmiennej) na wartość przechowywaną wewnątrz tego obiektu lub tej zmiennej.

stałą liczbę kluczy szyfrowania, a oprogramowanie może skonfigurować procesor do korzystania z podzbioru dostępnych kluczy. Oprogramowanie zarządza wykorzystaniem kluczy i może użyć każdego z dostępnych kluczy do zaszyfrowania dowolnej strony pamięci. Dzięki temu technologia Intel MKTME umożliwia szczegółowe szyfrowanie stron pamięci. Domyślnie technologia Intel MKTME stosuje klucz szyfrowania Intel TME, chyba że oprogramowanie wyraźnie określi inaczej.

Oprócz obsługi klucza efemerycznego generowanego przez procesor (niedostępnego przez oprogramowanie ani przy użyciu zewnętrznych interfejsów do procesora), technologia Intel MKTME obsługuje również klucze dostarczane przez oprogramowanie. Klucze dostarczane przez oprogramowanie są szczególnie przydatne, gdy są używane z pamięcią nieulotną, mechanizmami atestacji lub stosowane w połączeniu z usługami dostarczania kluczy. Włączenie systemu operacyjnego może zapewnić dodatkowe korzyści z technologii Intel MKTME, zarówno w środowiskach natywnych, jak i zwirtualizowanych. Po prawidłowym włączeniu technologia Intel MKTME jest dostępna dla każdego systemu operacyjnego gościa w zwirtualizowanym środowisku i może on korzystać z niej w taki sam sposób, jak natywny system operacyjny.

B.3.2. Izolacja aplikacji

B.3.2.1. Software Guard Extensions (SGX) firmy Intel

Intel Software Guard Extensions (SGX) to zestaw instrukcji, które zwiększają bezpieczeństwo kodu aplikacji i danych. Programiści mogą wydzielić wrażliwy kod i dane do *enklawy SGX*, która jest wykonywana w regionie chronionym przez procesor. Programista tworzy i uruchamia na platformach serwerów enklawy *SGX*, w których tylko procesor jest zaufany, w celu zapewnienia atestacji i chronionych środowisk wykonawczych dla kodu i danych enklawy. Rozwiązanie *SGX* zapewnia również mechanizm zdalnej atestacji enklawy. Mechanizm ten umożliwia zdalnemu dostawcy weryfikację następujących elementów [37]:

1. Enklawa działa na rzeczywistym procesorze Intel wewnątrz enklawy *SGX*.
2. Platforma działa z najnowszym poziomem zabezpieczeń (zwanym również wersją TCB).

3. Tożsamość enklawy jest taka, jak deklarowana.
4. Enklawa nie została naruszona.

Gdy wszystkie powyższe elementy zostaną zweryfikowane, dostawca zdalnej atestacji może następnie dostarczyć klucz tajny do enklawy. Wykorzystanie enklawy *SGX* jest zarezerwowane dla aplikacji Ring-3. Nie może jej używać ani system operacyjny ani sterownik/ moduł BIOS.

SGX usuwa oprogramowanie uprzywilejowane (np. system operacyjny, urządzenia VMM, System Management Mode) i nieuprzywilejowane (np. aplikacje Ring-3, maszyny wirtualne, kontenery) z granicy zaufania kodu działającego wewnątrz enklawy, zwiększając bezpieczeństwo wrażliwego kodu aplikacji i danych. Enklawa *SGX* ufa procesorowi w zakresie wykonywania i ochrony pamięci. *SGX* szyfruje pamięć, aby zapewnić ochronę przed snoopingiem magistrali pamięci i atakami typu cold boot na kod enklawy i dane w pamięci dynamicznej o dostępie swobodnym (*ang. Dynamic Random Access Memory – DRAM*) hosta. Rozwiązanie *SGX* obejmuje instrukcje ISA, które mogą być używane do zarządzania stronami Enclave Page Cache w celu tworzenia i inicjalizacji enklaw.

SGX polega na systemowym interfejsie UEFI BIOS i systemie operacyjnym w zakresie początkowego udostępniania, alokacji zasobów i zarządzania. Jednak po uruchomieniu enklawa *SGX* działa w kryptograficznie izolowanym środowisku oddzielonym od systemu operacyjnego i BIOS-u.

SGX może zezwolić dowolnej aplikacji (całej lub jej części) na działanie wewnątrz enklawy i umożliwia twórcom aplikacji sterowanie bezpieczeństwem ich własnych aplikacji. Zaleca się jednak, aby programiści utrzymywali jak najmniejszą bazę kodową *SGX*, przeprowadzali walidację całego systemu (w tym odporność oprogramowania na ataki typu side channel) i postępowali zgodnie z innymi wytycznymi dotyczącymi bezpiecznego tworzenia oprogramowania.

Enklawy *SGX* mogą być wykorzystywane do różnych zastosowań, od ochrony kluczy prywatnych i zarządzania poświadczeniami zabezpieczeń po świadczenie usług zabezpieczeń. Ponadto w branżowych standardach bezpieczeństwa, takich jak

Bezpieczeństwo wizualizacji funkcji sieciowych (*ang. Network Functions Virtualization – NFV*) Europejskiego Instytutu Norm Telekomunikacyjnych (*ang. European Telecommunications Standards Institute – ETSI*) – ETSI NFV SEC [38], zdefiniowano i opublikowano wymagania dotyczące enklaw HMEE (*ang. Hardware Mediated Execution Enclaves*) do celów NFV, 5G i bezpieczeństwa przetwarzania brzegowego. *SGX* jest enklawą HMEE.

B.3.3. Izolacja maszyny wirtualnej

B.3.3.1. Trust Domain Extensions (Intel *TDX*) firmy Intel

Rozwiązanie Intel Trust Domain Extensions (Intel *TDX*) wprowadza nowe elementy architektoniczne do wdrażania izolowanych sprzętowo maszyn wirtualnych zwanych zaufanymi domenami (*ang. Trust Domain – TD*). Rozwiązanie Intel *TDX* służy do izolowania maszyn wirtualnych od funkcji VMM/hypervisor i wszelkiego oprogramowania innego niż oprogramowanie TD na platformie w celu ochrony zaufanych domen przed szerokim wyborem oprogramowania. Rozwiązanie *TDX* jest oparte na kombinacji rozszerzeń maszyn wirtualnych (*ang. Virtual Machine Extension – VMX*), rozszerzeń ISA, technologii MKTME i modułu oprogramowania atestowanego przez procesor, zwanego modułem *TDX*-SEAM. *TDX* izoluje maszyny wirtualne od wielu zagrożeń sprzętowych i większości zagrożeń programowych, w tym od VMM i innego oprogramowania *CSP*. *TDX* umożliwia dzierżawcy chmury sprawowanie kontroli nad bezpieczeństwem własnych danych i ochroną własności intelektualnej. Spełnia tę funkcję przy jednoczesnym zachowaniu roli *CSP* polegającej na zarządzaniu zasobami i integralnością platformy w chmurze.

Rozwiązanie *TDX* zapewnia następujące możliwości zaufanym domenom, aby mogły one sprostać wyzwaniom związanym z bezpieczeństwem:

- Poufność i integralność stanu pamięci i procesora, aby chronić wrażliwe dane IP i obciążenia przed większością ataków bazujących na oprogramowaniu i wieloma atakami opartymi na sprzęcie. Obciążenie jest wyposażone w narzędzie, które obsługuje wykluczenie oprogramowania układowego, oprogramowania, urządzeń i operatorów platformy w chmurze z TCB.

Obciążenia mogą korzystać z tego narzędzia w celu zapewnienia bezpieczniejszego dostępu do instrukcji procesora i innych jego funkcji. Obciążenie może mieć tę możliwość niezależnie od infrastruktury chmury użytej do jego wdrożenia.

- Zdalna atestacja umożliwia stronie ufającej (właścicielowi obciążenia lub użytkownikowi usług dostarczanych przez obciążenie) ustalenie, że obciążenie działa na platformie obsługującej *TDX* znajdującej się w zaufanej domenie przed udostępnieniem danych tego obciążenia. Zdalna atestacja ma na celu umożliwienie właścicielom i klientom usługi cyfrowej określenie wersji TCB, na której polegają w celu zabezpieczenia swoich danych. Funkcja VMM pozostaje menedżerem zasobów platformy, a zaufana domena nie powinna powodować odmowy usługi dla VMM. Obrona zaufanej domeny przed odmową świadczenia usług przez funkcję VMM nie jest celem.

Rozwiązanie *TDX* zwiększa również ochronę zaufanej domeny przed ograniczonymi formami ataków, w ramach których wykorzystywany jest fizyczny dostęp do pamięci platformy, takimi jak ataki offline, analiza pamięci DRAM (np. ataki typu cold-boot) i aktywne ataki na interfejsy pamięci DRAM, w tym przechwytywanie, modyfikowanie, przenoszenie, łączenie i aliasowanie zawartości pamięci [39]. Funkcja VMM nadal pełni rolę menedżera zasobów, a bezpieczne domeny nie mają uprawnień do odmowy świadczenia usług na rzecz VMM.

B.3.4. Akceleracja kryptograficzna

B.3.4.1. Advanced Encryption Standard New Instructions (AES-NI) firmy Intel

Intel AES New Instructions (Intel AES-NI) to zestaw instrukcji szyfrowania, który poprawia wydajność sprzętową algorytmu zaawansowanego standardu szyfrowania (*ang. Advanced Encryption Standard – AES*) i przyspiesza szyfrowanie danych. Intel AES-NI składa się z siedmiu nowych instrukcji, które przyspieszają szyfrowanie i odszyfrowywanie oraz usprawniają generowanie kluczy i manipulowanie macierzami, jednocześnie pomagając w mnożeniu bez przenoszenia (*ang. carry-less multiplication*). Minimalizuje to problemy z wydajnością aplikacji związane z konwencjonalnym

przetwarzaniem kryptograficznym i pomaga zapewnić zwiększone bezpieczeństwo poprzez przeciwdziałanie atakom typu side-channel na algorytm AES związanym z konwencjonalnymi programowymi metodami wyszukiwania w tabelach [40].

AES jest najczęściej stosowanym standardem ochrony ruchu sieciowego, danych osobowych i korporacyjnej infrastruktury informatycznej. Dzięki zaimplementowaniu pewnych zaawansowanych kroków podrzędnych algorytmu AES w sprzęcie, technologia Intel AES-NI wzmacnia i przyspiesza wykonywanie aplikacji AES [40].

B.3.4.2. QuickAssist Technology (QAT) wraz z Key Protection Technology (KPT) firmy Intel

Technologia Intel QuickAssist (QAT) to wysokowydajny akcelerator sprzętowy do wykonywania operacji z zakresu kryptografii, bezpieczeństwa i kompresji. Aplikacje takie jak maszyny wirtualne, kontenery i funkcja jako usługa wywołują Intel QAT przy użyciu standardowych interfejsów OpenSSL, TLS i Internet Protocol Security (IPsec) w celu odciążenia symetrycznych i asymetrycznych operacji kryptograficznych. Technologia QAT najlepiej sprawdza się w przypadku infrastruktury i aplikacji w chmurze, z wieloma użytkownikami, NFV, do przetwarzania brzegowego i 5G dla wszystkich rodzajów obciążeń, w tym programowalnych sieci komputerowych, sieci dostarczania zawartości, multimediów i pamięci masowej [41].

Technologia Intel Key Protection Technology (KPT) umożliwia klientom zabezpieczenie swoich kluczy, które mają być używane z QAT, poprzez zastosowanie funkcji korzystania z własnego klucza (*ang. bring-your-own-key*). Technologia KPT umożliwia klientom dostarczanie własnych kluczy kryptograficznych do urządzenia QAT na platformie docelowej, na której działa ich obciążenie. Klucze chronione przez KPT nigdy nie stają się jawne w pamięci DRAM hosta lub podczas przesyłania. Klienci szyfrują swój klucz obciążenia (np. klucz prywatny RSA dla Nginx) za pomocą KPT wewnątrz swoich modułów HSM. Taki zaszyfrowany klucz obciążenia jest dostarczany do docelowej platformy QAT, gdzie jest odszyfrowywany bezpośrednio przed użyciem. KPT zapewnia ochronę kluczy w trakcie ich przechowywania, przesyłania i użytkowania [42].

B.3.5. Podsumowanie przykładów technologii

Infrastruktura chmury zapewnia poprawę wydajności, elastyczności i skalowalności obciążeń centrum danych poprzez oddzielenie sprzętu od warstwy aplikacji. Jednak stwarza to nowe zagrożenia, ponieważ wielu użytkowników ma dostęp do obciążeń, powierzchnie podatne na atak stają się współużytkowane, a administratorzy infrastruktury ze strony operatora chmury uzyskują dostęp do platform bazowych. Odpowiedzią na te problemy są techniki izolacji, które dodają ochronę maszyn wirtualnych, aplikacji i danych podczas wykonywania, i stanowią kluczowy poziom warstwowego podejścia do bezpieczeństwa w architekturze bezpieczeństwa centrum danych.

Istnieją różne techniki izolacji, które można wykorzystać do różnych potrzeb związanych z bezpieczeństwem. Pełna izolacja pamięci chroni platformę przed technikami ekstrakcji pamięci fizycznej, a ta sama technologia rozszerzona o wiele kluczy umożliwia poszczególnym maszynom wirtualnym lub dzierżawcom platformy unikalne szyfrowanie pamięci. Kolejne generacje tych technologii umożliwią pełną izolację pamięci maszyn wirtualnych, chroniąc je przed złośliwymi intruzami działającymi w infrastrukturze, złośliwym oprogramowaniem atakującym wielu dzierżawców i innymi technikami ataku. Techniki izolacji aplikacji umożliwiają poszczególnym aplikacjom tworzenie odizolowanych enklaw, które wymagają jedynie domyślnego zaufania do procesora platformy i które mają możliwość dostarczenia dowodu enklawy innym aplikacjom przed wysłaniem danych.

B.4 USŁUGI ZDALNEJ ATESTACJI

B.4.1. Security Libraries for the Data Center (ISecL-DC) firmy Intel

ISecL-DC to implementacja zestawu bloków konstrukcyjnych typu open-source do zdalnej atestacji, które wykorzystują funkcje Intel Security do wykrywania, atestacji i włączania krytycznych zabezpieczeń i poufnego przetwarzania. Ta technologia oprogramowania pośredniczącego zapewnia jednolity zestaw interfejsów API (*ang. Application Programming Interface – API*) w celu ułatwienia integracji z oprogramowaniem do zarządzania chmurą oraz narzędziami do monitorowania i egzekwowania zabezpieczeń. ISecL-DC stosuje podstawy zdalnej atestacji opisane

w niniejszym rozdziale oraz standardowe specyfikacje w celu utrzymania usługi gromadzenia danych platformy i efektywnego silnika weryfikacji do przeprowadzania kompleksowych ocen zaufania. Te oceny zaufania mogą być wykorzystywane do zarządzania różnymi politykami zaufania i bezpieczeństwa stosowanymi do dowolnego obciążenia, tak jak w przypadku planowania obciążeń opisanego w podrozdziale 7.2. W przyszłych generacjach produkt zostanie rozszerzony o możliwość atestacji środowiska *TEE*, aby zapewniać jego wiarygodność i ważność w celu umożliwienia poufnego przetwarzania danych [43].

B.4.2. Podsumowanie technologii

Atestacja platformy zapewnia audytowalne raporty podstawowe dotyczące integralności oprogramowania układowego i oprogramowania serwera i może zostać rozszerzone o lokalizację innych informacji o tagach zasobów przechowywanych w module TPM, a także weryfikację integralności aplikacji zainstalowanych na serwerze. Raporty te zapewniają wgląd w konfiguracje zabezpieczeń platformy i mogą być wykorzystywane do sterowania dostępem do danych i obciążeń. Atestacja platformy jest przeprowadzana dla poszczególnych serwerów i zwykle jest wykorzystywana przez narzędzia do orkiestracji w chmurze lub szeroką gamę platform do zarządzania infrastrukturą.

Atestacja środowiska *TEE* zapewnia mechanizm, za pomocą którego użytkownik lub aplikacja może — przed udostępnieniem sekretów lub kodu w środowisku *TEE* — zweryfikować, czy faktycznie używana jest prawdziwa enklawa *TEE* z akceptowalną bazą TCB. Tworzenie enklawy *TEE* odbywa się na poziomie aplikacji, a atestacje *TEE* są zwykle wykorzystywane przez użytkownika lub aplikację wymagającą dowodu bezpieczeństwa enklawy przed przekazaniem sekretów.

Wymienione różne techniki atestacji służą uzupełniającym się celom we wdrożeniu chmury obliczeniowej w centrum danych lub w obiekcie do przetwarzania brzegowego.

ZAŁĄCZNIK C – PRZYKŁADY TECHNOLOGII FIRMY *AMD*

W niniejszym załączniku opisano przykłady technologii firmy *AMD*, które odpowiadają kluczowym koncepcjom opisanym w różnych częściach tego dokumentu.

C.1 WERYFIKACJA INTEGRALNOŚCI PLATFORMY

C.1.1. Platform Secure Boot (AMD PSB) firmy AMD

Technologia *AMD* Platform Secure Boot (AMD PSB) zapewnia sprzętowe źródło zaufania do uwierzytelniania początkowego kodu BIOS platformy podczas procesu uruchamiania serwera. Producenci systemów serwerowych, takich jak OEM lub producenci oryginalnych urządzeń (*ang. Original Device Manufacturer – ODM*), włączają funkcje *AMD* PSB do swojego procesu produkcyjnego, trwale łącząc polityki z układem scalonym.

Ostateczny obraz systemu BIOS producenta OEM lub ODM zawiera klucz publiczny *AMD* i klucz publiczny do podpisywania systemu BIOS producenta OEM (podpisany kluczem prywatnym *AMD*). Po włączeniu systemu, podsystem *AMD* Security Processor (*ang. AMD Security Processor – ASP*) rozpoczyna wykonywanie niezmiennej, wbudowanej pamięci Boot ROM. Uwierzytelnia on i wczytuje wieloetapowe moduły ładujące rozruch ASP z pamięci flash SPI/LPC do swojej pamięci wewnętrznej, która inicjalizuje układ i pamięć systemową.

Po zainicjowaniu pamięci systemu moduły ładujące rozruch ASP wczytują i uwierzytelniają klucz publiczny podpisujący system BIOS OEM, a następnie uwierzytelniają początkowy kod systemu BIOS. Po pomyślnej weryfikacji ASP odblokowuje rdzeń x86 w celu wykonania uwierzytelnionego początkowego kodu BIOS. System BIOS może rozszerzyć łańcuch zaufania CoT na inne komponenty za pośrednictwem technologii SB. Jeśli uwierzytelnianie przeprowadzane przez technologię PSB nie powiedzie się, wymusza ona wyłączenie systemu.

Technologia *AMD* PSB obsługuje ochronę przed odwołaniem i wycofaniem obrazów BIOS-u za pośrednictwem identyfikatora wersji klucza do podpisywania systemu BIOS OEM oraz ochrony przed wycofaniem.

C.2 OCHRONA DANYCH I POUFNOŚĆ PRZETWARZANIA

C.2.1. Izolacja pamięci

C.2.1.1. Secure Memory Encryption (SME) / Transparent Secure Memory Encryption (TSME) firmy *AMD*

AMD Secure Memory Encryption (SME) to technologia szyfrowania pamięci firmy *AMD*, która umożliwia ochronę danych w pamięci DRAM poprzez szyfrowanie zawartości pamięci systemowej [44]. Jej włączenie powoduje, że zawartość pamięci jest szyfrowana za pomocą specjalnych komponentów sprzętowych w kontrolerach pamięci podręcznej procesora. Każdy kontroler zawiera wysokowydajny silnik AES, który szyfruje dane podczas zapisu do pamięci DRAM i odszyfrowuje je podczas odczytu. Szyfrowanie danych odbywa się za pomocą klucza szyfrującego w trybie, w którym wykorzystywane jest dodatkowe dostrojenie oparte na adresie fizycznym w celu ochrony przed atakami polegającymi na przenoszeniu bloków szyfrogramu.

Klucz szyfrowania używany przez silnik AES z technologią SME jest generowany losowo przy każdym resecie systemu i nie jest widoczny dla żadnego oprogramowania działającego na rdzeniach procesora. Klucz ten jest zarządzany w całości przez *AMD* Secure Processor, który działa jako dedykowany podsystem bezpieczeństwa zintegrowany z czipem (*ang. System-on-Chip – SoC*) firmy *AMD*. Klucz jest generowany przy użyciu wbudowanego sprzętowego RNG zgodnego z wymogami publikacji NIST SP 800-90, jest przechowywany w dedykowanych rejestrach sprzętowych i nigdy nie jest ujawniany poza SoC.

Obsługiwane są dwa tryby szyfrowania pamięci do różnych zastosowań. Najprostszym trybem jest Transparent Secure Memory Encryption (TSME), który jest opcją systemu BIOS i umożliwia automatyczne szyfrowanie pamięci przy wszystkich dostępach do niej. Tryb TSME działa w tle i nie wymaga interakcji z oprogramowaniem. Innym obsługiwanym trybem jest Secure Memory Encryption (SME) zarządzany przez system operacyjny, w którym poszczególne strony pamięci mogą być oznaczone do szyfrowania za pośrednictwem tabel stron procesora. Tryb SME zapewnia dodatkową elastyczność, jeśli musi być zaszyfrowany tylko podzbiór pamięci, ale wymaga wsparcia ze strony odpowiedniego oprogramowania.

Szyfrowana pamięć zapewnia silną ochronę przed atakami typu cold boot, DRAM interface snooping i podobnymi naruszeniami bezpieczeństwa.

C.2.2. Izolacja maszyny wirtualnej

C.2.2.1. Secure Encrypted Virtualization (SEV) firmy *AMD*

Funkcja *AMD* Secure Encrypted Virtualization (SEV) została opracowana w celu odizolowania maszyn wirtualnych od funkcji hypervisor. Po włączeniu funkcji *SEV* poszczególne maszyny wirtualne są szyfrowane za pomocą klucza szyfrowania AES. Gdy komponent taki jak funkcja hypervisor próbuje odczytać pamięć gościa, jest w stanie zobaczyć dane tylko w postaci zaszyfrowanej. Zapewnia to silną izolację kryptograficzną między maszynami wirtualnymi, a także między maszynami wirtualnymi a funkcją hypervisor.

Aby chronić gości obsługujących *SEV*, oprogramowanie układowe *SEV* wspomaga egzekwowanie trzech głównych właściwości zabezpieczeń: autentyczności platformy, atestacja uruchomienia gościa i poufności danych gościa.

Uwierzytelnianie platformy uniemożliwia złośliwemu oprogramowaniu lub oszukańczemu urządzeniu podszywanie się pod prawidłową platformę. Autentyczność platformy jest potwierdzana za pomocą jej klucza tożsamości. Klucz ten jest podpisany przez *AMD*, aby wykazać, że platforma jest autentyczną platformą *AMD* z funkcją *SEV*.

Atestacja uruchomienia gościa dowodzi jego właścicielowi, że jego gość został bezpiecznie uruchomiony z włączoną funkcją *SEV*. Podpis różnych komponentów stanu gościa związanego z *SEV*, w tym początkowej zawartości pamięci, jest dostarczany przez oprogramowanie układowe właścicielowi gościa w celu zapewnienia możliwości weryfikacji, czy gość znajduje się w oczekiwanym stanie. Dzięki temu atestowaniu właściciel gościa może upewnić się, że funkcja hypervisor nie ingerowała w inicjalizację funkcji *SEV* przed przesłaniem poufnych informacji do gościa.

Poufność gościa jest zapewniana poprzez szyfrowanie pamięci za pomocą klucza do szyfrowania pamięci, który jest znany tylko oprogramowaniu układowemu *SEV*. Interfejs zarządzania *SEV* nie zezwala na eksport klucza do szyfrowania pamięci ani żadnego innego sekretu *SEV* poza oprogramowanie układowe bez odpowiedniego uwierzytelnienia.

Funkcja *AMD SEV* ma dwa dodatkowe tryby:

- SEV With Encrypted State (SEV-ES): W tym trybie rejestry maszyny wirtualnej są szyfrowane i chronione przed odczytem lub modyfikacją przez przejętą przez atakującego funkcją hypervisor lub maszyną wirtualną [45].
- SEV with Secure Nested Paging (SEV-SNP): Tryb ten dodaje silną ochronę integralności pamięci, aby przeciwdziałać złośliwym atakom opartym na funkcji hypervisor, takim jak data replay i memory remapping [46].

ZAŁĄCZNIK D – PRZYKŁADY TECHNOLOGII FIRMY ARM

W niniejszym załączniku opisano przykłady technologii firmy Arm, które odpowiadają kluczowym koncepcjom opisanym w różnych częściach tego dokumentu.

D.1 WERYFIKACJA INTEGRALNOŚCI PLATFORMY

Aby zrozumieć środowisko Secure Boot, należy najpierw zrozumieć stany zabezpieczeń i poziomy wyjątków zaimplementowane przez technologię TrustZone. W poniższych podrozdziałach opisano technologie dostępne dla procesorów Arm A-profile[11] [47], [48].

D.1.1. TrustZone Trusted Execution Environment (TEE) dla architektury Armv8-A firmy Arm

D.1.1.1. Świat normalny (niezabezpieczony) i świat bezpieczny

Technologia TrustZone [49] zapewnia dwa środowiska wykonawcze wbudowane w procesor z wymuszoną sprzętowo izolacją między nimi w całym systemie. Istnieją dwa stany bezpieczeństwa: bezpieczny i niezabezpieczony. Są one mapowane odpowiednio do świata bezpiecznego (*ang. Secure World – SW*) i normalnego (*ang. Normal World – NW*), czasami określanego również jako świat niezabezpieczony. Każdy procesor implementuje oba światy, ale może działać tylko w jednym z nich w danym momencie, niezależnie od tego, w którym świecie działa każdy z pozostałych procesorów w implementacji wieloprocesorowej. Na przykład, rdzeń 0 może działać w NW, podczas gdy rdzeń 1 działa w SW, rdzeń 2 działa w NW itd. — wszystkie jednocześnie. Koncepcja SW i NW wykracza poza procesor i obejmuje pamięć, oprogramowanie, transakcje magistralowe, przerwania i urządzenia peryferyjne w ramach SoC.

NW obsługuje bogate środowisko wykonawcze (*ang. Rich Execution Environment – REE*), które zazwyczaj obejmuje dużą liczbę aplikacji, złożony system operacyjny (np. Linux) i często funkcję hiperwizor. REE stanowi rozległą powierzchnię podatną na atak. SW zapewnia środowisko *TEE*, które obsługuje mniejszy i prostszy stos oprogramowania niż

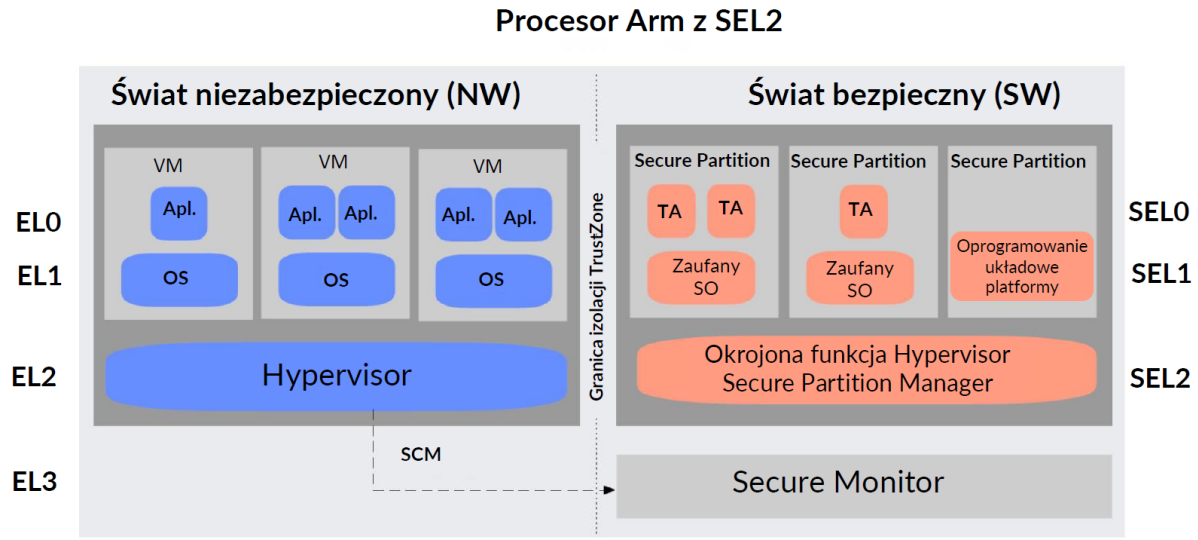
[11] Opis ten dotyczy również zapowiedzianej niedawno architektury Armv9-A.

REE. Środowisko *TEE* może obejmować kilka zaufanych usług, okrojone jądro i, jeśli procesor obsługuje wyjątek bezpieczeństwa poziomu 2 (*ang. Secure Exception Level 2 – SEL2*, wyjaśniono w załączniku D.1.1.2), prostą funkcję hypervisor. Środowisko *TEE* ma znacznie mniejszą powierzchnię podatną na atak i nie obsługuje dowolnych kodów, dzięki czemu jest znacznie mniej podatne na ataki w porównaniu do REE. Ponadto, poziomy SEL zapewniają dodatkową ochronę w środowisku *TEE*.

Jak przedstawiono na rysunku 9, pomiędzy NW i SW istnieje wymuszona sprzętowo granica izolacji. W przypadku procesorów klasy A, NW żąda bezpiecznej usługi od SW, wywołując Secure Monitor Call (SMC), aby wykonać przejście z NW do SW i z powrotem za pośrednictwem funkcji Secure Monitor, która działa w SW przy najwyższym poziomie wyjątku, EL3. Funkcja Secure Monitor przekazuje informacje do funkcji Secure Partition Manager (SPM) wykonywanej w poziomie SEL2, która wywołuje bezpieczną usługę w zaufanej aplikacji (*ang. Trusted Application – TA*) działającej w bezpiecznej partycji (*ang. Secure Partition – SP*).

Przestrzenie adresowe SW i NW można podzielić na kilka regionów. Każdy region jest określony jako bezpieczny lub niezabezpieczony. Rejestry kontrolujące partycjonowanie przestrzeni adresowej są dostępne tylko dla SW, dzięki czemu tylko oprogramowanie SW może partycjonować pamięć. Nie przewiduje się, aby partycjonowanie pamięci SW i NW zmieniało się dynamicznie w środowisku uruchomieniowym, ponieważ jest ono konfigurowane tylko raz podczas uruchamiania systemu[12], które zawsze odbywa się w stanie bezpiecznym.

[12] Przed wprowadzeniem architektury *CCA*, TF-A nie definiowało dynamicznego transferu pamięci między dwoma światami. Architektura *CCA* obsługuje tę operację (patrz załącznik D.3).



Rysunek 9: Procesor Arm z technologią TrustZone

Architektura zapewnia dwie fizyczne przestrzenie adresowe (*ang. Physical Address Space – PAS*): bezpieczną (SW) i niezabezpieczoną (NW). W stanie niezabezpieczonym adresy wirtualne są zawsze tłumaczone na niezabezpieczone adresy fizyczne. Oprogramowanie w stanie niezabezpieczonym może uzyskać dostęp tylko do niezabezpieczonych zasobów. W stanie bezpiecznym oprogramowanie sprzętowe działające na poziomie wyjątku 3 (EL3) i SEL2 może uzyskać dostęp zarówno do bezpiecznych, jak i niezabezpieczonych fizycznych przestrzeni adresowych. SEL0 i SEL1 mogą uzyskać dostęp do niezabezpieczonego adresu fizycznego, jeśli został on zmapowany do odpowiedniego wpisu w tabeli strony przez SEL2 dla SP. Podczas wykonywania w SW kod nigdy nie jest pobierany ani wykonywany z przestrzeni adresowej w NW.

Urządzenia wejścia/wyjścia (*ang. Input/Output – I/O*) mogą być przypisane do SW lub NW. Urządzenia mapowane w pamięci podlegają tym samym zasadom dostępu, które opisano dla dostępu do pamięci.

D.1.1.2. Poziomy wyjątków

W NW istnieją trzy poziomy wyjątków (*ang. Exception Level – EL*):

- **EL0** to poziom wyjątku, w którym aplikacje są wykonywane w NW. Mają wgląd w przestrzeń adresową aplikacji utworzoną przez system operacyjny działający w EL1. EL0 jest najmniej uprzywilejowanym poziomem wyjątku.

- **EL1** jest poziomem wyjątku, w którym wykonywany jest system operacyjny. System operacyjny zarządza przestrzeniami adresowymi aplikacji, które tworzy w EL0 i jest właścicielem całej przypisanej do nich pamięci. System operacyjny może być systemem operacyjnym typu „bare-metal", który działa bezpośrednio na sprzęcie, lub może znajdować się w maszynie wirtualnej utworzonej przez funkcję hypervisor.
- **EL2** jest poziomem wyjątku, w którym wykonywana jest funkcja hypervisor. Jest ona właścicielem pamięci NW i zarządza nią, a także zarządza maszynami wirtualnymi. Maszyna wirtualna składa się z systemu operacyjnego działającego w EL1 i tworzonych przez system aplikacji działających w EL0. Można do niej przypisać również niezabezpieczone urządzenia.

Wyższe poziomy wyjątków (tj. z większą liczbą oznaczającą poziom) mają przywilej dostępu do rejestrów, które sterują niższymi poziomami. Przy ogólnym działaniu systemu, uprzywilejowane EL zazwyczaj kontrolują swoją własną konfigurację. Jednak bardziej uprzywilejowane poziomy wyjątków czasami uzyskują dostęp do rejestrów powiązanych z niższymi poziomami, na przykład w celu odczytu i zapisu zestawu rejestrów w ramach operacji zapisywania i przywracania podczas przełączania kontekstu lub operacji zarządzania energią. Poziomy EL1 i EL0 współużytkują tę samą konfigurację jednostki MMU, a sterowanie jest ograniczone do uprzywilejowanego kodu działającego w EL1.

Poziomy wyjątków normalnego świata, od EL0 do EL2, mają swoje odpowiedniki w bezpiecznym świecie i służą one podobnym celom:

- **SEL0** to poziom wyjątku, w którym zaufane aplikacje są wykonywane w SW. Zapewniają one bezpieczne usługi dla NW (np. usługi kryptograficzne Platform Security Architecture [PSA], zarządzanie prawami cyfrowymi (*ang. Digital Rights Management*), sprzętowy moduł TPM, bezpieczny interfejs do współdzielonego urządzenia sprzętowego, takiego jak dyskretny moduł TPM lub HSM). Zaufane aplikacje mają wgląd w przestrzeń adresową aplikacji utworzoną przez zaufany system operacyjny (*ang. Trusted Operating System – TOS*), np. OP-TEE,

w momencie jej powstania. Jedna zaufana aplikacja z SEL0 nie może uzyskać dostępu do żadnej innej pamięci bezpiecznej aplikacji z SEL0, chyba że pamięć ta jest specjalnie zmapowana jako współdzielona przez TOS z SEL1. SEL0 jest najmniej uprzywilejowanym bezpiecznym poziomem wyjątku w SW.

- **SEL1** to bezpieczny poziom wyjątku, w którym wykonywany jest TOS, zaprezentowany w SP na rysunku 9. TOS zarządza przestrzeniami adresowymi aplikacji i jest właścicielem całej pamięci przypisanej do SP. Alternatywnie, oprogramowanie sprzętowe platformy dostarczone przez dostawcę może być wykonywane we własnej bezpiecznej partycji w SEL1. W takim wypadku nazywa się ją bezpieczną partycją typu bare-metal. Każda bezpieczna partycja stanowi oddzielną bezpieczną domenę w SW i jest zasadniczo odpowiednikiem maszyny wirtualnej z NW w SW.
- **SEL2** jest poziomem wyjątku, w którym wykonywana jest funkcja SPM. Jest to prosta funkcja hypervisor, która zarządza bezpiecznymi partycjami i ich przestrzeniami adresowymi. Kieruje ona wiadomości do i z bezpiecznych partycji. Rozdzielenie przestrzeni adresowych umożliwia również przeniesienie dostarczonego przez producenta oprogramowania układowego platformy, które w poprzedniej implementacji bez SEL2 działało w EL3, do izolowanej bezpiecznej partycji działającej w SEL1 (jak pokazano na rysunku 9).
- **EL3** to specjalny poziom wyjątku [49]. EL3 jest zawsze wykonywany w SW. Zarządza przejściem między światami i obsługuje oprogramowanie sprzętowe, które zapewnia usługi Secure Monitor, w tym:

✓ Wykonanie rozruchu początkowego (BL2).

✓ Silnik do przechwytywania i wysyłania wywołań SMC, który obsługuje przychodzące wywołania SMC i przekierowuje je.

✓ Utrzymanie izolacji i pamięci między środowiskiem niezabezpieczonym i bezpiecznym.

✓ Power State Coordination Interface – zarządzanie energią na niskim poziomie.

- ✓ System Control and Management Interface – niezależne od systemu operacyjnego interfejsy oprogramowania używane do zarządzania systemem.
- ✓ Powiadamianie o błędach dotyczących niezawodności, dostępności i przydatności.
- ✓ Software Delegated Exception Interface – zapewnia mechanizm dostarczania zdarzeń o wysokim priorytecie, który ma wyższy priorytet niż przerwania ukierunkowane na systemy operacyjne i funkcje hypervisor.

Trusted Firmware zapewnia obsługę oprogramowania układowego pod kątem własności intelektualnej (*ang. Intellectual Property – IP*) systemu Arm, takiej jak połączenia wzajemne. Silicon Providers (SiP) zapewniają obsługę oprogramowania układowego w celu obsługi IP klientów i firm zewnętrznych. Obejmuje to zarządzanie energią specyficzne dla SoC.

Informacje na temat rozszerzeń dodanych do tej architektury w ramach *CCA* znajdują się w załączniku D.3.1.

D.1.1.3. Zaufane aplikacje i bezpieczne usługi

Zaufane aplikacje w SP mają działać tylko przez krótkie okresy, aby nie blokować wykonywania NW przez długi czas na danym procesorze. SW nie jest przeznaczony do uruchamiania aplikacji ogólnego przeznaczenia, ale raczej bezpiecznych usług, takich jak te opisane wcześniej dla poziomu SEL1. Najczęściej te bezpieczne usługi będą usługami globalnymi oferowanymi aplikacjom NW i będą działać przez cały okres rozruchu. Jednak ich użycie może być ograniczone do określonych aplikacji NW poprzez wdrożenie mechanizmu uwierzytelniania w SW w celu zapewnienia, że żądający z NW jest upoważniony do korzystania z usługi.

Planowanie działania SW odbywa się dla każdego procesora i jest realizowane za pośrednictwem bezpiecznego przerwania obsługiwanego przez Secure Monitor na poziomie EL3. Bezpośrednie wywołania usług, takich jak na przykład Platform Management Communications Infrastructure, są zazwyczaj blokowane na tym procesorze. Sterowanie jest zwracane do stanu niezabezpieczonego dopiero po zakończeniu żądanej operacji. Takie wywołania są jednak krótkie i rzadkie. Wywołania SMC bezpiecznych usług TA są planowane przez system operacyjny lub funkcję

hypervisor hosta. Większość żądań bezpiecznej usługi będzie krótka. Jeśli jednak zaufana aplikacja musi działać przez dłuższy czas, konwencja wywoływania SMC pozwala TA zwrócić sterowanie do NW, gdzie system operacyjny hosta (np. Linux) może zmienić harmonogram SW poprzez ponowne wydanie polecenia SMC w dogodnym czasie. SW powróci do poprzedniego stanu. To kooperacyjne podejście do planowania pozwala NW sterować planowaniem zgodnie z potrzebami.

D.1.1.4. Debugowanie, śledzenie i profilowanie

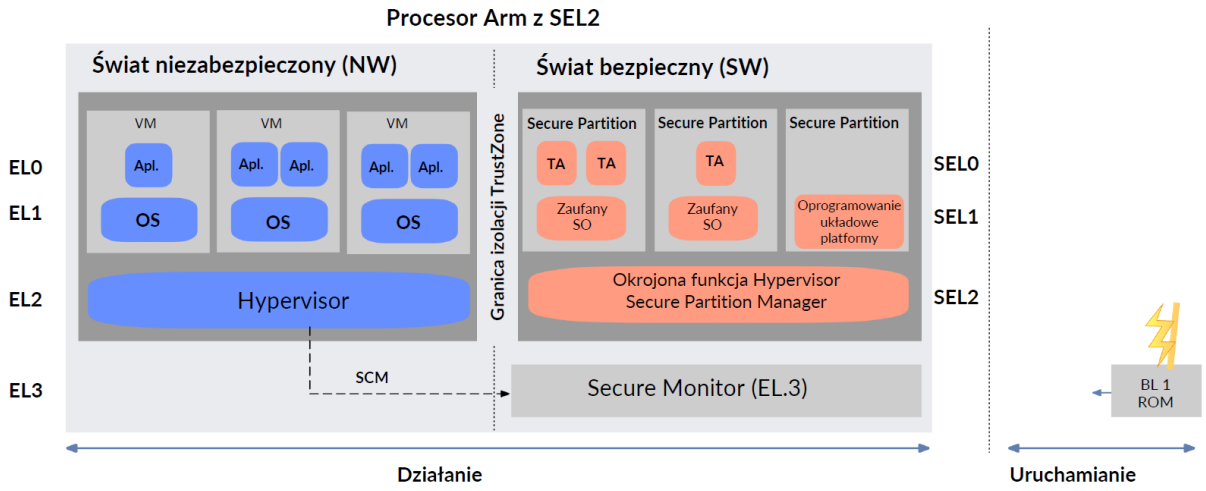
Systemy Arm są wyposażone w zaawansowane funkcje umożliwiające debugowanie, śledzenie i profilowanie. Czip SoC musi być odpowiednio skonfigurowany, aby nie można było wykorzystać tych funkcji do naruszenia bezpieczeństwa systemu. Różnym programistom powierza się debugowanie różnych części systemu na różnych etapach cyklu życia jego zabezpieczeń. Aby spełnić te wymagania, dostępne są sygnały szybkości do sterowania korzystaniem z funkcji w stanie bezpiecznym i niezabezpieczonym.

D.1.2. Technologia bezpiecznego rozruchu i łańcucha zaufania (CoT) firmy Arm

Rozruchowe oprogramowanie układowe może być zaufane tylko wtedy, gdy zaufane są wszystkie składniki oprogramowania uruchamiane w ramach procedury rozruchu. Jest to tak zwany łańcuch zaufania (*ang. Chain of Trust – CoT*). Projekt Trusted Firmware-A[13] (TF-A) [12] umożliwia implementację rozruchowego oprogramowania układowego, które wczytuje, uwierzytelnia i weryfikuje każdy ładunek kodu rozruchowego przed przekazaniem mu sterowana. Weryfikacja zapewnia integralność rozruchowego oprogramowania układowego i krytycznych danych, umożliwiając potwierdzenie, że nie zostały one w żaden sposób uszkodzone lub zmodyfikowane i są autentyczne. W ten sposób rozruchowe oprogramowanie układowe ustanawia kompletny kaskadowy łańcuch zaufania (CoT).

[13] W ramach projektu TF-A opracowano referencyjną implementację oprogramowania bezpiecznego świata. Jest to otwarty projekt społecznościowy zarządzany przez organizację Linaro. Obsługa procesorów Arm A-Profile (Cortex i Neoverse) jest obecnie dostępna jako oprogramowanie open source pod adresem https://git.trustedfirmware.org/TF-A/trusted-firmware-a.git/. Funkcja skupia się na zaufanym rozruchu i małym, zaufanym środowisku uruchomieniowym (kod EL3).

Na platformie Arm, bezpieczny rozruch[14] odbywa się w SW zapewnianym przez implementację technologii TrustZone w ramach architektury Arm Processor Element (PE) - patrz załącznik D.3.1.3. Po wystąpieniu zdarzenia włączenia/zresetowania procesora system rozpoczyna wykonywanie kodu rozruchowego w pamięci ROM[15] w poziomie EL3 na rdzeniu rozruchowym[16]. Niezmienna pamięć ROM stanowi sprzętowe źródło zaufania dla zespołu procesora i jest domyślnie zaufana.



Rysunek 10: Oprogramowanie układowe odpowiedzialne za rozruch i środowisko uruchomieniowe

Podczas produkcji, SiP i OEM lub ODM zapewniają kod w pamięci ROM (BL1 na rysunku 10) i pierwszy modyfikowalny ładunek kodu rozruchowego (BL2). Kod rozruchowy w pamięci ROM jest zazwyczaj niewielki i prosty. Jego główną funkcją jest

[14] W tym podrozdziale odniesienia do „bezpiecznego rozruchu" obejmują zweryfikowany rozruch, w którym każdy moduł kodu rozruchowego jest uwierzytelniany po załadowaniu go z nośnika rozruchowego, zazwyczaj pamięci flash, aby upewnić się, że nie został zmodyfikowany lub uszkodzony, przed przekazaniem do niego sterowania, oraz zmierzony rozruch, w którym każdy moduł kodu rozruchowego jest mierzony przy użyciu funkcji skrótu, zarówno dla kodu, jak i danych konfiguracyjnych, a każdy pomiar jest rozszerzany na rejestr PCR. Zweryfikowany rozruch i zmierzony rozruch uzupełniają się. Bezpieczny rozruch UEFI, który kontynuuje weryfikację i pomiary, również został uwzględniony w SRTM dla systemów Arm.

[15] Rozruchowe pamięci ROM są zazwyczaj implementowane jako maskowane pamięci ROM lub wbudowane pamięci flash z obsługą sprzętu, aby zapewnić, że nie można ich zmienić po zaprogramowaniu. Projekt niezmiennego pierwszego ładunku rozruchowego nie jest ograniczony do konkretnych implementacji. Musi być spełniony jedynie wymóg niezmienności dotyczący architektury.

[16] Pozostałe rdzenie pozostają zresetowane do momentu zakończenia rozruchu. Taka serializacja pozwala uniknąć luk w zabezpieczeniach, które mogłyby powstać w wyniku współbieżnego wykonywania kodu rozruchowego na wielu procesorach.

załadowanie kodu rozruchowego drugiego etapu (BL2), pierwszego modyfikowalnego kodu rozruchowego, z pamięci nieulotnej do pamięci, w której jest przechowywane oprogramowanie układowe, zazwyczaj pamięci flash, i zweryfikowanie go przy użyciu niezmiennego klucza publicznego źródła zaufania dostarczonego podczas produkcji. Konkretny algorytm klucza publicznego używany do uwierzytelniania jest definiowany przez implementację. Proces ten zapewnia integralność pierwszej instrukcji. Jeśli kod BL2 pomyślnie przejdzie weryfikację, kod BL1 przekazuje mu sterowanie. Następnie kod BL2 przeprowadza pewne operacje inicjalizacji systemu platformy, takie jak na przykład konfigurowanie kontrolera pamięci dla niewbudowanej pamięci DRAM. Następnie kod BL2 ładuje i kryptograficznie weryfikuje wszystkie kolejne ładunki rozruchowe, w tym oprogramowanie układowe EL3 rozruchu i środowiska uruchomieniowego (BL3-1), oprogramowanie układowe na poziomie SEL2, SEL1 i SEL0 (BL3-2) oraz pierwszy niezabezpieczony ładunek rozruchowy (BL3-3 – np. UEFI, U-Boot), który jest wykonywany w poziomie EL2 po przekazaniu mu sterowania. Kod BL2 używa samopodpisanych certyfikatów X.509 Key i Content do weryfikacji ładunków BL3-x. Kod BL3-3 to ostatni element łańcucha rozruchu, reprezentuje łańcuch zaufania rozruchu.

W ramach procesu rozruchu opcjonalnie można mierzyć całe rozruchowe oprogramowanie układowe do kodu BL3-3 włącznie. Kod BL3-3, w trakcie wykonywania, może również zapewniać pomiary dla wczytywanego oprogramowania układowego (np. sterowników środowiska uruchomieniowego). Jest to funkcja opcjonalna. Proces opisany powyżej zostaje zmieniony w taki sposób, że rozruchowe oprogramowanie układowe jest wczytywane, weryfikowane, a następnie mierzone przed przekazaniem mu sterowania. W przypadku architektury Arm pomiary są rozszerzane o rejestry PCR: PCR0 (całe podpisane oprogramowanie układowe i dane) i PCR1 (wszystkie krytyczne dane konfiguracyjne – np. ustawienia portu debugowania). Rodzaj implementacji rejestru PCR jest definiowany w zależności od platformy i jest ona ukryta za pomocą interfejsu API oprogramowania układowego, który zapewnia tylko operacje odczytu i rozszerzenia. Implementacja rejestru PCR musi jedynie gwarantować przestrzeganie zachowania architektury i bezpieczeństwo

pomiarów PCR. Na przykład, rejestr PCR może być zaimplementowany w układzie scalonym w chronionej lokalizacji pamięci, w implementacji oprogramowania układowego TPM, lub w urządzeniu zewnętrznym, takim jak dyskretny moduł TPM lub bezpieczny element (SE). Rejestry PCR0 i PCR1 są zerowane podczas resetowania.

Pomiary rozruchowe mogą być wykorzystane do atestacji oprogramowania sprzętowego, które zostało uruchomione w systemie. Klucz atestacji, znany również jako klucz poręczenia (*ang. Endorsement Key – EK*), kryptograficznie potwierdza tożsamość urządzenia, a tym samym zaufanie do niego, na potrzeby podmiotów zewnętrznych. Klucz atestacji urządzenia może być wykorzystywany w wielu różnych schematach atestacji (np. technologii uwierzytelniania stowarzyszenia Fast Identity Online [FIDO], TCG TPM, Platform Security Architecture [PSA] i *CCA*). Za pomocą zdalnego weryfikatora można porównać atestowany pomiar z listą znanych „dobrych" pomiarów, aby zdecydować, czy system znajduje się w prawidłowym bezpiecznym stanie. Następnie można wykorzystać zasady, aby zdecydować, jakie działania należy podjąć, jeśli porównanie się nie powiedzie.

Wykorzystywany jest monotonicznie rosnący licznik, aby zapobiec wycofaniu oprogramowania układowego, w tym danych konfiguracyjnych, do poprzedniej wersji, ponieważ ta poprzednia wersja oprogramowania może zawierać luki, które mogą zostać wykorzystane. Jednak opcja wycofania aktualizacji w celu przywrócenia poprzedniego stanu może być dozwolona, jeśli jej użycie zostanie autoryzowane.

D.1.3. Funkcyjne interfejsy API technologii Platform Security Architecture (PSA)

Funkcyjne interfejsy API PSA definiują podstawy, na których budowane są usługi bezpieczeństwa, dzięki czemu urządzenia mogą być bezpieczne z założenia. Trzy interfejsy API (do kryptografii, przechowywania i atestacji) zapewniają spójne środowisko programistyczne dla twórców oprogramowania systemowego i aplikacji, umożliwiając interoperacyjność w przypadku różnych implementacji sprzętowego źródła zaufania.

D.1.3.1. Kryptograficzny interfejs API technologii PSA (Crypto API)

PSA Crypto API to przenośny interfejs do operacji kryptograficznych na szerokiej gamie sprzętu. Interfejs zapewnia dostęp do elementów podstawowych niskiego

poziomu wykorzystywanych w nowoczesnej kryptografii. Nie wymaga, aby użytkownik miał dostęp do materiału klucza, zamiast tego wykorzystuje identyfikatory kluczy nieprzezroczystych. Definiuje interfejs dla usług kryptograficznych, w tym kryptograficznych elementów podstawowych i funkcji przechowywania kluczy. Interfejs został zaprojektowany tak, aby był zarówno skalowalny, jak i modułowy, umożliwiając urządzeniom implementację tylko tego, czego potrzebują.

Implementacje mogą izolować procesor kryptograficzny od aplikacji wywołującej, a także izolować wiele aplikacji wywołujących, jedną od drugiej. Implementacja może być podzielona na fronton i zaplecze. W odizolowanej implementacji zaplecze znajduje się w odizolowanym środowisku, które jest chronione przed frontonem. Ochronę mogą zapewniać różne technologie, np.:

- Izolacja procesów systemu operacyjnego.
- Izolacja partycji za pomocą maszyny wirtualnej lub środowiska *TEE*, takiego jak TrustZone.
- Fizycznie oddzielne urządzenia sprzętowe.

Definiowany jest niskopoziomowy interfejs kryptograficzny, w którym wywołujący jawnie wybiera algorytm i parametry bezpieczeństwa, których chce użyć. Wszystkie funkcje kryptograficzne działają zgodnie z algorytmem określonym przez wywołującego. Ogólne interfejsy wyższego poziomu, w których implementacja wybiera najlepszy algorytm do danego celu, nie są określane. Na bazie interfejsu PSA Crypto API można jednak tworzyć biblioteki wyższego poziomu.

Możliwe przypadki użycia interfejsu PSA Crypto API są następujące:

- TLS.
- Szyfrowanie bezpiecznej pamięci masowej.
- Poświadczenia sieciowe (np. X.509).
- Parowanie urządzeń (np. parowanie tokenów w komunikacji bliskiego zasięgu [*Near Field Communication – NFC*] lub protokoły uzgadniania kluczy Bluetooth).

- Zweryfikowany rozruch (integralność i uwierzytelnianie oprogramowania układowego).
- Atestacja (zapewniane elementy podstawowe są odpowiednie do zastosowania w procesie atestacji).
- Atestacja fabryczna (te interfejsy API mogą być używane do generowania unikalnych kluczy tożsamości urządzeń do wykorzystania w fabryce).

Dostępne są interfejsy dla następujących typów symetrycznych operacji kryptograficznych:

- Skróty komunikatów, powszechnie znane jako funkcje skrótu (hasze).
- Kody uwierzytelniania komunikatów (*ang. Message Authentication Code – MAC*).
- Szyfry symetryczne.
- Uwierzytelnione szyfrowanie z powiązanymi danymi (*ang. Authenticated Encryption with Associated Data – AEAD*).

Zarówno para jednoczęściowych funkcji (np. szyfrowanie, odszyfrowywanie), jak i seria funkcji, które umożliwiają operacje wieloczęściowe, są zdefiniowane dla każdego typu symetrycznej operacji kryptograficznej (np. alokacja, inicjalizacja, konfiguracja, aktualizacja i zakończenie).

D.1.3.2. Interfejs PSA Storage API

Interfejsy PSA Storage API zapewniają interfejsy pamięci masowej klucz/wartość do użytku z pamięcią chronioną sprzętowo. Opisują one interfejs dla pamięci zapewnianej przez źródło RoT w ramach architektury PSA (interfejs API wewnętrznego zaufanego magazynu [*ang. Internal Trusted Storage – ITS*] w ramach PSA), a także interfejs dla zewnętrznej chronionej pamięci (interfejs API chronionego magazynu [*ang. Protected Storage – PS*] w ramach PSA). Możliwe są dwa przypadki użycia: bezpieczne przechowywanie sekretów urządzenia (ITS) i ochrona danych magazynowanych (PS). ITS jest bardziej wyspecjalizowanym interfejsem API i ma na celu zapewnienie integralności i/lub prywatności danych. Na przykład klucz tożsamości urządzenia wymaga zarówno poufności, jak i integralności, natomiast klucz publiczny to dane publiczne wymagające integralności, ale nie prywatności. PS to interfejs API ogólnego

przeznaczenia, który jest używany najczęściej i jest przeznaczony do ochrony większych zbiorów danych przed atakami fizycznymi. Zapewnia możliwość przechowywania danych na zewnętrznej pamięci flash, z gwarancją ochrony magazynowanych danych, w tym szyfrowania związanego z urządzeniem, integralności i ochrony przed odtwarzaniem. Możliwe jest wybranie odpowiedniego poziomu ochrony, np. tylko szyfrowanie, tylko integralność lub wszystkie trzy, w zależności od modelu zagrożeń urządzenia i charakteru jego wdrożenia.

Spójne interfejsy API umożliwiające dostęp do pamięci pozwalają na pisanie oprogramowania w sposób niezależny od platformy, zwiększając możliwość przenoszenia na różne platformy z architekturą PSA.

D.1.3.3. Interfejs PSA Attestation API

PSA Attestation API to standardowy interfejs zapewniany przez Rot w ramach architektury PSA, który jest zdefiniowany w modelu zabezpieczeń PSA. Interfejs API może być używany zarówno do bezpośredniego podpisywania danych, jak i do budowania zaufania w innych schematach atestacji. Architektura PSA zapewnia ramy i minimalne ogólne funkcje bezpieczeństwa umożliwiające producentom OEM i dostawcom usług integrację różnych schematów atestacji z RoT w ramach PSA. RoT PSA zgłasza informacje (oświadczenia), które można wykorzystać do określenia dokładnej implementacji PSA RoT i jego stanu bezpieczeństwa. Jeśli PSA RoT ładuje inne komponenty, dołącza również informacje o nich. Inne komponenty spoza PSA RoT mogą dodawać informacje do raportu, wywołując udostępniony interfejs API, który dołączy i podpisze dodatkowe informacje. PSA RoT podpisuje raporty z atestacji przy użyciu wstępnego klucza atestacji.

Każda instancja urządzenia zawiera chroniony klucz atestacji, który może być użyty do udowodnienia, że jest to konkretna certyfikowana implementacja. W procesie atestacji tożsamość może zostać zweryfikowana i porównana z informacjami w certyfikacie. Pod koniec procesu weryfikator może ustanowić bezpieczne połączenie z atestowanym punktem końcowym i dostarczyć dane uwierzytelniające. Połączenie jednostki sprzętowej i oprogramowania zainstalowanego na tej jednostce może być certyfikowane jako zgodne z pewnym opublikowanym poziomem

bezpieczeństwa. Strona może chcieć sprawdzić otrzymaną listę roszczeń z bazą danych znanych pomiarów dla każdego komponentu, aby zdecydować, który poziom zaufania należy zastosować.

Wstępna atestacja wymaga trzech usług:

- Usługi weryfikowania rejestracji, wymuszającej zasady podczas rejestrowania urządzenia do usługi.
- Usługi weryfikowania podczas produkcji (OEM), zapewniającej stan produkcyjny tożsamości podczas atestacji.
- Usługi weryfikowania certyfikacji (zewnętrznej), weryfikującej, czy wszystkie atestowane komponenty są aktualne, poprawnie podpisane i certyfikowane do współdziałania.

Interfejs API musi być dostarczony przez implementację.

D.1.4. Platform AbstRaction for SECurity (Parsec)

Parsec [50] (Platform AbstRaction for SECurity), to inicjatywa open-source mająca na celu zapewnienie wspólnego interfejsu API do zabezpieczania usług w sposób niezależny od platformy. Zapewnia mikrousługę, która dopasowuje łatwe w użyciu interfejsy API bezpieczeństwa, w wybranym języku, do elementów podstawowych bezpieczeństwa występujących w różnych implementacjach sprzętowych. Jest ona częścią piaskownicy CNCF.

Parsec ma na celu zdefiniowanie standardu oprogramowania do interakcji z bezpiecznym magazynem obiektów i usługami kryptograficznymi, tworząc wspólny sposób komunikacji z funkcjami, które tradycyjnie były dostępne za pośrednictwem bardziej wyspecjalizowanych interfejsów API. *Parsec* zapewnia system bibliotek w różnych językach programowania. Każda biblioteka została zaprojektowana tak, aby była łatwa w użyciu. Ekosystem ten zapewnia programistom dostęp do bezpiecznych rozwiązań w szerokim zakresie zastosowań w infrastrukturze obliczeniowej, przetwarzaniu brzegowym i IoT.

Platformy obliczeniowe ewoluowały i oferują teraz szereg udogodnień w zakresie bezpiecznego przechowywania i bezpiecznych operacji. Są to rozwiązania wspierane sprzętowo, takie jak moduły HSM i TPM, usługi oprogramowania układowego działające w środowisku *TEE*, a także oparte na chmurze usługi *Parsec*. Funkcje bezpieczeństwa mogą być zapewniane wyłącznie przez oprogramowanie, a wówczas są one chronione przez mechanizmy przewidziane w systemie operacyjnym. Inicjatywa *Parsec* opiera się na architekturze PSA. Podstawowym komponentem inicjatywy *Parsec* jest usługa bezpieczeństwa *Parsec*. Jest to proces działający w tle, który działa na platformie hosta i zapewnia łączność z bezpiecznymi obiektami tego hosta oraz udostępnia protokół przewodowy oparty na funkcyjnych interfejsach API PSA.

Usługa *Parsec* nasłuchuje na odpowiednim nośniku transportu. Technologia transportowa jest jednym z wielu możliwych do podłączenia komponentów *Parsec*, nie jest wymagany żaden pojedynczy transport. Wybór transportu zależy od systemu operacyjnego i wdrożenia. W systemach Linux, w których aplikacje klienckie działają w kontenerach (izolacja ze współdzielonym jądrem), transport może być oparty na gniazdach systemu Unix. Aplikacje klienckie nawiązują połączenia z usługą, wysyłając żądania API do punktu końcowego transportu. Zwykle odbywa się to za pośrednictwem biblioteki klienta, która ukrywa szczegóły zarówno protokołu przewodowego, jak i transportu.

Pojedyncza instancja usługi *Parsec* jest wykonywana na każdym hoście fizycznym. W środowiskach zwirtualizowanych usługa *Parsec* może znajdować się na specjalnie przypisanym gościu lub potencjalnie działać w ramach funkcji hypervisor. Inną opcją jest uruchomienie usługi *Parsec* na każdym pojedynczym gościu, z obsługą zaplecza działającą w środowisku *TEE* / bezpiecznej enklawie. Usługa *Parsec* nie obsługuje zdalnych aplikacji klienckich. Każdy fizyczny host lub węzeł musi mieć własną instancję usługi. Usługa może jednak inicjować wychodzące wywołania zdalne innych usług, takich jak hostowane w chmurze usługi HSM.

Usługa *Parsec* jest również odpowiedzialna za pośredniczenie w dostępie do podstawowych zabezpieczeń między wieloma aplikacjami klienckimi w środowisku z wieloma użytkownikami. Usługa *Parsec* jest w stanie obsługiwać scenariusze z wieloma

użytkownikami, wykorzystując tożsamość klienta, która jest ciągiem lub tokenem, który każda aplikacja kliencka przekazuje do usługi *Parsec* przy każdym wywołaniu interfejsu API. Nie narzuca ona żadnego konkretnego formatu ani semantyki dla tych ciągów tożsamości klienta. Mogą one być przekazywane nieprzezroczyście do usługi jako sekwencje oktetów przy użyciu protokołu przewodowego. Jedynym wymogiem dotyczącym ciągu tożsamości klienta jest to, że musi on być w jakiś sposób weryfikowalny przez usługę *Parsec*. Komponent, który umożliwia tę weryfikację, nazywany jest dostawcą tożsamości.

Dostawca tożsamości nie jest częścią usługi *Parsec*. Może to być inny komponent lub usługa, działająca lokalnie lub zdalnie. Może to być orkiestrator kontenerów lub inna usługa zarządzania środowiskiem uruchomieniowym, a nawet funkcja systemu operacyjnego. Oczekuje się, że usługa *Parsec* będzie mogła ustanowić zaufanie do dostawcy tożsamości za pomocą odpowiednich środków. Usługa *Parsec* obejmuje podłączany mechanizm komunikacji z różnymi dostawcami tożsamości.

Zarządzanie tożsamością klienta może obejmować następujące przykłady:

- Tożsamość klienta może być po prostu jednostką systemu operacyjnego, taką jak identyfikator procesu, identyfikator użytkownika lub identyfikator grupy procesu aplikacji klienckiej. Jeśli do transportu protokołu przewodowego wykorzystywane są gniazda systemu Unix, usługa *Parsec* może zweryfikować tożsamość, wykonując wywołania systemowe w celu sprawdzenia danych uwierzytelniających równorzędnego elementu [51]. W tym przypadku dostawcą tożsamości jest właściwie jądro systemu operacyjnego.
- Tożsamość klienta może być ustalana na podstawie weryfikowanego dokumentu tożsamości SPIFFE (*ang. SPIFFE Verifiable Identity Document – SVID*) zgodnego ze standardami Secure Production Identity Framework for Everyone (SPIFFE) [52]. W tym przypadku dostawcą tożsamości byłby komponent środowiska uruchomieniowego SPIFFE, taki jak lokalna usługa SPIFFE Runtime Environment (SPIRE) [53], z którą usługa *Parsec* komunikuje się za pomocą interfejsu API SPIFFE Workload API [54].

Możliwe są również inne metody weryfikacji. Niezależnie od tego, jak tożsamość klienta zostanie zweryfikowana, usługa *Parsec* wykorzysta wynikowy ciąg tożsamości jako przestrzeń nazw dla wszystkich kluczy i innych przechowywanych zasobów. Usługa *Parsec* gwarantuje, że każda aplikacja kliencka ma widoczność i dostęp tylko do własnej przestrzeni nazw.

Usługa *Parsec* zaspokaja zapotrzebowanie na nową abstrakcję platformy, która oferuje wspólną paletę elementów podstawowych bezpieczeństwa za pośrednictwem interfejsu oprogramowania, która jest zarówno niezależna od możliwości bazowego sprzętu, jak i zdolna do obsługi wielu aplikacji klienckich na tym samym hoście, niezależnie od tego, czy znajdują się one w kontenerach, czy maszynach wirtualnych.

Więcej informacji można również znaleźć w publikacjach [55], [56], [57].

D.2 MECHANIZMY OCHRONY ŚRODOWISKA URUCHOMIENIOWEGO OPROGRAMOWANIA

D.2.1. Ataki typu ROP (ang. Return Oriented Programming) oraz JOP (ang. Jump Oriented Programming)

D.2.1.1. Pointer Authentication Code (*PAC*) firmy Arm

Celem ataków jest często zakłócenie przepływu sterowania w oprogramowaniu. *PAC [48]* to funkcja wprowadzona w celu blokowania tego typu ataków. Daje ona możliwość uwierzytelniania adresu zawartości rejestru, zanim zostanie on użyty jako cel pośredniego rozgałęzienia lub załadowany. Funkcja *PAC* jest skuteczną obroną przed atakami typu *ROP*, które polegają na próbie spowodowania, by funkcja powróciła do niewłaściwej lokalizacji.

Dzięki technologii *PAC* sprzęt gwarantuje, że powrót następuje do właściwej lokalizacji, poprzez zachowanie oryginalnej wartości wskaźnika. Górne bity 64-bitowego wskaźnika są używane do przechowywania *PAC*, który jest podpisem kryptograficznym na wartości wskaźnika i pewnym dodatkowym określonym kontekście. Wprowadzono specjalne instrukcje, aby dodać podpis *PAC* do wskaźnika, zweryfikować podpis *PAC* uwierzytelnianego wskaźnika i przywrócić oryginalną wartość wskaźnika. W ramach

operacji uwierzytelniania podpis *PAC* jest generowany ponownie i porównywany z wartością przechowywaną we wskaźniku. Jeśli uwierzytelnianie powiedzie się, zwracany jest wskaźnik bez podpisu *PAC*. W przypadku niepowodzenia uwierzytelniania zwracany jest nieprawidłowy wskaźnik. Jeśli wskaźnik zostanie później użyty, zgłaszany jest wyjątek. Daje to systemowi sposób na zapewnienie gwarancji o silnym podłożu kryptograficznym w odniesieniu do prawdopodobieństwa, że określone wskaźniki zostały naruszone przez atakujących.

D.2.1.2. Branch Target Identification (BTI) firmy Arm

Gdy atakujący znajdzie lukę do wykorzystania, jego kolejnym krokiem jest wykonanie kodu w celu przejęcia sterowania maszyną, do której uzyskał dostęp. Techniki wykorzystywane do modyfikacji przepływu sterowania obejmują ataki typu *ROP* i *JOP*. W ramach tych technik znajdowane są małe sekcje (zwane gadżetami) podatnych na ataki programów, które można połączyć w łańcuch, aby osiągnąć cel atakującego.

Systemy obsługujące technologię Branch Target Identification (*BTI*) [48] mogą wymusić, aby pośrednie rozgałęzienia były kierowane tylko do tych lokalizacji kodu, które zaczynają się od jednej z akceptowanych instrukcji *BTI*. Strony mogą być oznaczone jako zawierające instrukcje *BTI*. Pośrednie rozgałęzienia mogą prowadzić tylko do lokalizacji zidentyfikowanych jako posiadające instrukcje *BTI*, co zmniejsza zdolność atakującego do wykonania dowolnego kodu. Funkcja *BTI* współpracuje z podpisem *PAC*, co znacznie zmniejsza liczbę gadżetów dostępnych dla atakującego.

D.2.2. Naruszenia bezpieczeństwa pamięci

D.2.2.1. Privileged Access Never (PAN) firmy Arm

Technologia *PAN* pomaga zapobiegać wykorzystaniu jądra systemu operacyjnego lub funkcji hypervisor w celu uzyskania niezamierzonego dostępu do pamięci przydzielonej aplikacji w trybie użytkownika (EL0). Jeśli technologia *PAN* jest włączona, wszelkie próby uzyskania dostępu do strony kontrolowanej przez atakującego w trybie użytkownika przez jądro lub funkcję hypervisor zostaną zablokowane. Dostęp spowoduje błąd uprawnień i nie doprowadzi do buforowania danych lub instrukcji. W niektórych przypadkach system operacyjny lub funkcja

hypervisor musi uzyskać dostęp do nieuprzywilejowanych regionów, na przykład w celu zapisu do bufora należącego do aplikacji. Aby obsłużyć taką operację, zestaw instrukcji zapewnia nieuprzywilejowane ładowania i pamięć, które nie są blokowane przez technologię *PAN* (*LDTR* i *STTR*). Są one sprawdzane pod kątem uprawnień poziomu EL0, nawet jeśli są wykonywane przez system operacyjny na poziomie EL1 lub EL2. Kod aplikacji musi być wykonywalny w przestrzeni użytkownika (EL0), jednak nigdy nie powinien być wykonywany z uprawnieniami jądra (EL1/EL2), więc technologia *PAN* kontroluje taki dostęp.

D.2.2.2. User (EL0) Execute Never (UXN) oraz Privileged (EL1/EL2) Execute Never (PXN) firmy Arm

Technologie User (EL0) Execute Never (UXN) i Privileged (EL1/EL2) Execute Never (PXN) zapewniają ochronę przed atakami typu „stack smashing", w ramach których złośliwe oprogramowanie próbuje zapisać nowe kody operacji w pamięci, a następnie próbuje je wykonać. Zazwyczaj atak ten jest ukierunkowany na pamięć stosu. Takie uprawnienia do wykonywania są przechowywane we wpisach tabeli stron. Każda próba rozgałęzienia do adresu wewnątrz zaznaczonej strony powoduje błąd uprawnień. Atrybuty tabeli translacji i elementy sterujące zapisem mogą blokować wykonanie z dowolnej lokalizacji, do której złośliwy kod mógłby dokonać zapisu.

D.2.2.3. Memory Tagging Extension (MTE) firmy Arm

Tagowanie pamięci umożliwia programistom identyfikację przestrzennych i czasowych naruszeń bezpieczeństwa pamięci w ich programach (np. użycie pamięci po zwolnieniu, użycie pamięci spoza zakresu, użycie pamięci przed inicjalizacją, naruszenia powiązań). Technologia *MTE* [48], [58], [59] została zaprojektowana w celu szybkiego wykrywania naruszeń bezpieczeństwa pamięci i zapewnienia odporności na ataki, które próbują obejść kod. *MTE* jest okrojoną, probabilistyczną wersją systemu z "blokadą z kluczem", w którym jedna z ograniczonego zestawu wartości blokady może być powiązana z lokalizacjami pamięci tworzącymi część alokacji, a równoważny klucz jest przechowywany w nieużywanych wysokich bitach adresów używanych jako odwołania do tej alokacji. Przy każdym użyciu odwołania przed uzyskaniem dostępu następuje sprawdzenie, czy klucz pasuje do blokady. Jeśli klucz pasuje do blokady, dostęp do

pamięci jest dozwolony. W przeciwnym razie nieprawidłowy dostęp może zostać zapisany w celu późniejszego odwołania (a wykonanie może być kontynuowane) lub może zostać przerwany (a wykonanie zostanie zatrzymane). Po zwolnieniu alokacji wartość blokady powiązana z każdą lokalizacją jest zmieniana na jedną z pozostałych wartości blokady, dzięki czemu dalsze użycie odwołania ma rozsądne prawdopodobieństwo niepowodzenia. Dzięki temu można łatwiej wykryć i wyeliminować trudne do wychwycenia błędy bezpieczeństwa pamięci, co zwiększa niezawodność i poprawia bezpieczeństwo produktu.

Technologia *MTE* zapewnia wsparcie architektoniczne dla środowiska uruchomieniowego, stałe wykrywanie różnych klas błędów pamięci i może wspomagać debugowanie w celu wyeliminowania luk w zabezpieczeniach, zanim zostaną one wykorzystane. Zapewnia sprzętową obsługę dostępu do pamięci i sprawdzanie wartości blokady. Oprogramowanie wybiera i ustawia wartości podczas alokowania i zwalniania. Dodano instrukcje — do wykorzystania przez oprogramowanie ogólnego przeznaczenia — do ustawiania tagów adresów logicznych, manipulowania tagami adresów logicznych oraz do przechowywania i ładowania tagów alokacji z pamięci. Dodano dodatkowe instrukcje do wykorzystania przez oprogramowanie systemowe i zewnętrznych agentów debugowania w celu wydajnego przesyłania tagów alokacji do i z pamięci. Oczekuje się, że rozszerzenie to będzie miało ogólne zastosowanie do 64-bitowego oprogramowania napisanego w językach C i C++, które nie wykorzystuje bitów tagu adresu logicznego do innych celów. Oczekuje się również korzyści z jego stosowania w środowiskach mieszanych języków, np. kodowania C/C++ w interakcji z językami kompilowanymi lub interpretowanymi „dokładnie na czas. Możliwość zastosowania tej metody do oprogramowania w innych językach może być różna. Ponieważ technologia *MTE* nie narzuca żadnych zmian w standardowych binarnych interfejsach aplikacji (*ang. Application Binary Interface – ABI*) kodowania C/C++, możliwe jest przyrostowe wdrażanie w ramach różnych poziomów wyjątków.

Technologie *MTE* i *PAC* mogą być włączone w tym samym czasie.

D.2.2.4. Hardware Enforced Capability-Based Architecture (Morello i CHERI) firmy Arm

Firma Arm i Uniwersytet Cambridge współpracują nad rozwojem architektury Capability Hardware Enhanced RISC Instructions (CHERI), która zapewnia podejście do bezpieczeństwa pamięci oparte na zasobach [60]. Firma Arm opracowała prototypową architekturę, Morello [13], [14], [61], która dostosowuje koncepcje sprzętowe architektury CHERI. Morello to program badawczy prowadzony przez Arm we współpracy z partnerami i finansowany przez fundację UK Research and Innovation (UKRI) w ramach brytyjskiego rządowego programu Digital Security by Design (DSbD). To nowe podejście do cyberbezpieczeństwa wymaga znaczącej zmiany w sposobie projektowania architektury sprzętowej, a także w sposobie tworzenia oprogramowania działającego na urządzeniach obsługujących architekturę opartą na zasobach CHERI, aby można było korzystać z nowych funkcji.

Architektura Morello ma na celu poprawę niezawodności i bezpieczeństwa systemów poprzez realizację dwóch celów projektowych: szczegółowej ochrony pamięci prowadzącej do zwiększenia jej bezpieczeństwa oraz skalowalnego podziału. Aby osiągnąć te cele, architektura Morello wprowadza zasady zdefiniowane przez CHERI, w tym zasadę minimalnych uprawnień i celowego użycia. Architektura Morello jest kompatybilna ze starszymi wersjami architektury Armv8-A i stanowi jej uzupełnienie.

Architektura CHERI CPU dodaje 128-bitowe „zasoby” plus bit tagu pamięci. Zasób zawiera adres, informacje o ograniczeniach, informacje o uprawnieniach i typ obiektu. Bit tagu pamięci to metadane, które odróżniają zasób od normalnych danych i zapobiegają „fałszowaniu” zasobów. Zasób może być używany zamiast zwykłego wskaźnika w niektórych lub wszystkich sytuacjach. Samo zastąpienie wszystkich wskaźników zasobami daje możliwość silnej przestrzennej ochrony pamięci. Ładowania i pamięć używające zasobów jako adresów są sprawdzane pod kątem zgodności z zakresem adresów zasobów i dostarczonymi uprawnieniami. Ograniczenia nie mogą zostać dowolnie zwiększone, uprawnienia nie mogą zostać złagodzone, a tag nie może zostać zmieniony.

Trzy przykładowe przypadki użycia to:

- **Kontrola dostępu do jądra:** Ze względu na tagowanie pamięci i ograniczoną możliwość manipulowania deskryptorami segmentów zasobów, proces nie może utworzyć deskryptora segmentu zasobu opisującego pamięć, dla której nie posiada jeszcze deskryptora.
- **Sandboxing:** Zasoby mogą być również wykorzystywane do „piaskowania" procesu w podprzestrzeniach adresowych, umożliwiając procesowi posiadanie własnych zasad ochrony pamięci dla części programu.
- **Bezpieczeństwo wskaźnika:** Zasoby mogą być również wykorzystywane do zapewnienia bezpieczeństwa wskaźników przy użyciu automatycznego sprawdzania granic.

Morello System Development Platform (SDP) to prototypowa płytka demonstracyjna zawierająca czip Morello SoC. SDP stanowi prototyp platformy technologicznej DSbD dla architektury Morello. Zasoby zostały wprowadzone do profilu architektury Arm v8-A jako rozszerzenie stanu Arm v8 AArch64, zgodnie z zasadami zaproponowanymi w wersji 8 instrukcji CHERI ISA. Mat to na celu zapewnienia sprzętowego wsparcia dla szczegółowej ochrony i tworzenia bloków konstrukcyjnych w celu bezpiecznego, skalowalnego podziału.

Dostępny jest również wstępny model systemu Morello Fixed Virtual Platform (FVP). Umożliwia on rozwój oprogramowania bez konieczności posiadania prototypowego sprzętu. Modele Arm FVP wykorzystują technologię translacji binarnej, aby zapewnić szybkie symulacje systemu opartego na architekturze Arm. Morello FVP stanowi funkcjonalnie dokładny model implementacji Morello SoC IP. Model FVP umożliwia partnerom przemysłowym i akademickim testowanie nowej prototypowej architektury opartej na zasobach w rzeczywistych zastosowaniach. Oprogramowanie Arm Development Studio: Morello Edition obsługuje Morello FVP, który zawiera modele programowe rdzeni Rainier. Płytka Morello SDP jest ściśle oparta na płytce Arm® Neoverse™ N1 System Development Platform. Dostępne są specyfikacje, modele i płytki demonstracyjne Morello.

D.2.3. Środki ochrony przed atakami typu side-channel firmy Arm

Atakujący mogą wykorzystać niepożądane efekty uboczne wykonywania poza kolejnością i wykonywania spekulatywnego w nowoczesnych procesorach, aby naruszyć rozdział między systemem operacyjnym a procesami oraz między poszczególnymi procesami w celu kradzieży danych. Do przeprowadzenia ataku często nie jest konieczny fizyczny dostęp do systemu. Atakujący może potencjalnie naruszyć typową separację procesów i uprawnień, używając specjalnie zaprojektowanego oprogramowania do zbierania poufnych informacji z innego oprogramowania działającego w tym samym systemie.

Firma Arm wdrożyła szereg środków przeciwdziałających atakom typu side-channel. W chwili obecnej zidentyfikowano cztery potencjalne warianty mechanizmów, a w każdym z nich wykorzystuje się spekulacje procesora do wpływania na to, które wpisy w pamięci podręcznej zostaną alokowane w taki sposób, aby wydobyć pewne informacje, które w przeciwnym razie nie byłyby dostępne dla oprogramowania. Dodano instrukcje zaporowe, które umożliwiają przeciwdziałanie wariantom Spectre i Meltdown 1 (CVE-2017-5753), 2 (CVE-2017-5715), 3 (CVE-2017-5754) i 3a (CVE-2018-3640), a także sterowanie uściślaniem pamięci w celu przeciwdziałania wariantowi 4 (CVE-2018-3639). Pełniejszy opis znajduje się w publikacjach [48], [62], [63], [64].

D.2.3.1. Bariery spekulacji

Do architektury dodano nową barierę, barierę spekulacji (*ang. Speculation Barrier*), aby kontrolować spekulacje pamięcią. Dopóki bariera nie zakończy działania, wykonanie jakiejkolwiek instrukcji pojawiającej się w programie później niż bariera nie może być spekulatywne — w zakresie, w jakim taka spekulacja może być obserwowana przez kanały boczne w wyniku spekulacji przepływu sterowania lub spekulacji wartości danych, ale może być spekulatywnie wykonywana w wyniku przewidywania, że potencjalnie generująca wyjątek instrukcja nie wygenerowała wyjątku.
W szczególności nie może ona powodować spekulacyjnej alokacji do żadnej struktury buforującej, w której alokacja tego wpisu mogłaby wskazywać na jakąkolwiek wartość danych obecną w pamięci lub w rejestrach. Instrukcja może zostać wykonana, gdy

wiadomo, że nie jest spekulatywna, a wszystkie wartości danych wygenerowane przez instrukcje pojawiające się w kolejności programu przed barierą spekulacji mają potwierdzone przewidywane wartości.

D.2.3.2. Unieważnienia przewidywania

W przypadku wszystkich zasobów przewidywania wykonania, które przewidują adresy lub wartości rejestrów, architektura wymaga, aby wykonanie spekulatywne w jednym kontekście zdefiniowanym sprzętowo było oddzielone w trudny do określenia sposób od przewidywań przetestowanych w innym kontekście zdefiniowanym sprzętowo. Do celów niniejszej definicji kontekst zdefiniowany sprzętowo jest określany przez:

- poziom wyjątku;
- stan bezpieczeństwa;
- identyfikator maszyny wirtualnej (*ang. Virtual Machine IDentifier – VMID*) w przypadku wykonywania na poziomie EL1;
- identyfikator przestrzeni; adresowej (*ang. Address Space IDentifier – ASID*) i identyfikator VMID w przypadku wykonywania na poziomie EL0.

D.2.3.3. Bariery synchronizacji

Architektura Arm jest „słabo" uporządkowaną architekturą pamięci, która obsługuje uzupełnianie poza kolejnością (*ang. out-of-order*). *Bariera pamięci* (*ang. Memory barrier*) to ogólny termin stosowany w odniesieniu do instrukcji lub sekwencji instrukcji, które wymuszają synchronizację zdarzeń przez PE względem wycofywanych instrukcji ładowania/przechowywania. Bariery pamięci zdefiniowane przez architekturę Armv8 zapewniają szereg funkcji, w tym porządkowanie i wykonywanie instrukcji ładowania/magazynowania oraz synchronizację kontekstu. Nowe instrukcje zapewniają barierę synchronizacji dla instrukcji, danych, śledzenia, błędów i profilowania.

D.2.3.4. Arm Trusted Firmware (TF-A) i Linux

Firma Arm wprowadziła aktualizacje do projektu typu open-source Trusted Firmware (TF-A) [12] i opracowała poprawki dla jądra systemu Linux i Android, które wykorzystują te zaimplementowane aktualizacje.

Opublikowano poprawki zarówno dla architektury AArch64, jak i AArch32. Dostępne są zabezpieczenia dla wariantów 1, 2, 3 i 4 zarówno dla architektury AArch64, jak i AArch32. W oprogramowaniu Trusted Firmware zaimplementowano nowe wywołanie SMC w celu obsługi niektórych z tych zabezpieczeń. Zabezpieczenia programowe opisane w publikacji *Cache Speculation Side-channels* [62] powinny być wdrażane tam, gdzie model zagrożeń wymaga ochrony przed złośliwymi aplikacjami. W architekturze Armv8.5-A firma Arm wprowadziła funkcje procesora [47], które można zaimplementować z architektury Armv8.0, zapewniające odporność na tego typu ataki.

D.2.3.5. Instrukcje przetwarzania danych niezależne od czasu

Nowa opcja może być skonfigurowana w taki sposób, aby zapewnić czas przetwarzania niezależny od danych (*ang. Data Independent Timing)* dla większości klas instrukcji przetwarzania danych – tj. czas potrzebny na wykonanie instrukcji jest niezależny od wartości danych dostarczonych do któregokolwiek z ich rejestrów. Ponadto reakcja tych instrukcji na asynchroniczne wyjątki nie różni się w zależności od wartości podanych w dowolnym z ich rejestrów. Dotyczy to następujących instrukcji:

- Wszystkie instrukcje kryptograficzne.
- Zestaw instrukcji korzystających z pliku rejestru ogólnego przeznaczenia.
- Zestaw instrukcji korzystających z techniki SIMD (*ang. Single Instruction, Multiple Data*) i pliku rejestru zmiennoprzecinkowego.
- Wszystkie ładowania i pamięć niewrażliwe pod względem czasu na wczytywaną lub zapisywaną wartość.

Uniemożliwia to atakującemu wywnioskowanie czegokolwiek na temat przetwarzanych danych na podstawie instrukcji lub zestawu instrukcji.

D.3 OCHRONA DANYCH I POUFNOŚĆ PRZETWARZANIA

D.3.1. Confidential Compute Architecture (CCA) firmy Arm

Technologia Arm Confidential Compute Architecture (CCA) [65] zapewnia architekturę chronionego środowiska wykonawczego (*ang. Protected Execution Environment*), która obejmuje mechanizmy wykorzystywane do tworzenia środowiska,

w którym uprawnienie nie oznacza żadnego prawa dostępu. Technologia *CCA* izoluje od siebie środowiska wykonawcze niezależnie od poziomu uprawnień. To rozdzielenie zasad ma wiele głębszych konsekwencji dla architektury, relacji zaufania i kontraktów.

Technologia *CCA* firmy Arm chroni przetwarzane dane, wykonując obliczenia w bezpiecznym środowisku wspieranym sprzętowo i weryfikowanym zdalnie. Chroni kod, dane i wykonanie przed obserwacją i modyfikacją przez innych agentów programowych i sprzętowych. Dzięki technologii *CCA* właściciel chronionego środowiska wykonawczego nie musi ufać innym programom na tym samym hoście ani uprzywilejowanym agentom sprzętowym, takim jak urządzenia nadrzędne z bezpośrednim dostępem do pamięci. Mechanizmy zapewniające chronione środowisko wykonawcze są bezpośrednio mierzone i raportowane przy użyciu procesu atestacji w celu określenia ich wiarygodności.

D.3.1.1. Obiekty Realm firmy Arm

Technologia *CCA* zapewnia obsługę architektury dynamicznie tworzonych obiektów zwanych Realm. Obiekt Realm [66], [67] zawiera zarówno kod i dane użytkownika (EL0), jak i przestrzeni jądra (EL1). Nadrzędnym obiektem zarządzającym zasobami Realm jest funkcja hypervisor w NW. Dzierżawcy obiektów Realm nie muszą ufać ani funkcji hypervisor, ani istniejącemu kodowi w SW. Obiekty Realm są chronione przed sobą nawzajem, żaden z nich nie musi ufać innym obiektom Realm.

Źródło zaufania *CCA* wymusza autentyczność platformy *CCA* poprzez atestację stanu rozruchu i stanu bezpieczeństwa obiektu Realm, autentyczność jego zawartości poprzez weryfikację i pomiary oraz poufność danych gościa poprzez ochronę i szyfrowanie przestrzeni adresów fizycznych (*ang. Physical Address Space – PAS*).

Świat obiektów Realm to świat oddzielony od NW i SW w środowisku TrustZone. Świat obiektów Realm jest przeznaczony do użytku wyłącznie przez te obiekty. Obiekt Realm chroni zawarte w nim informacje przed innymi podmiotami w systemie. Oprogramowanie o wyższych uprawnieniach zachowuje kontrolę nad przydzielaniem i zarządzaniem zasobami wykorzystywanymi przez obiekty Realm, ale nie ma dostępu do ich zawartości. Oprogramowanie o wyższych uprawnieniach zachowuje również

kontrolę nad planowaniem w swoich obiektach Realm, ale nie może w inny sposób sterować zachodzącym w nich procesem wykonania ani bezpośrednio go obserwować.

Dokładniej rzecz ujmując, architektura *CCA* zapewnia:

- Dodatkową kontrolę dostępu do pamięci, działającą równolegle do istniejącej kontroli wymuszanej za pomocą tabel translacji.
- Ochronę stanu wykonania.
- Wiarygodny pomiar (atestację) stanu początkowego.
- Gwarancję, że konfiguracja systemu (na przykład, czy dozwolone jest zewnętrzne debugowanie) nie ulegnie zmianie w czasie użytkowania obiektu Realm (niezmienność).

Dane obiektów Realm pozostają poufne nawet po zniszczeniu obiektu lub zresetowaniu systemu.

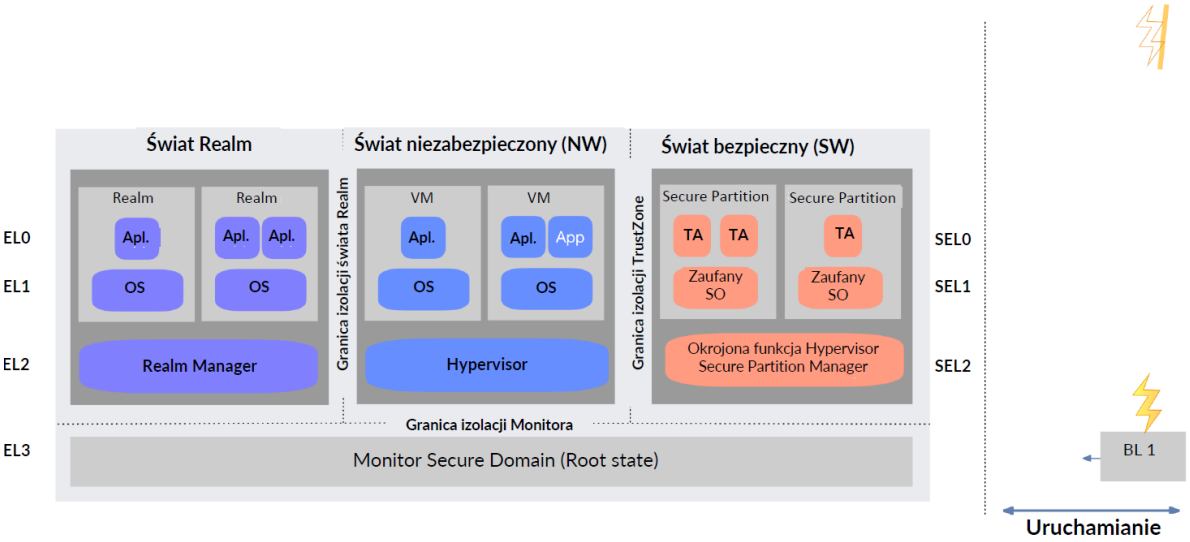
Obiekty Realm zostały zaprojektowane tak, by można je było tworzyć i niszczyć na żądanie. Niezabezpieczona funkcja hypervisor może utworzyć nowy obiekt Realm w dowolnym momencie, podobnie jak może utworzyć nową maszynę wirtualną. Funkcja hypervisor może dodawać lub usuwać strony z obiektu Realm w dowolnym momencie, w taki sam sposób, w jaki zarządza stronami innych maszyn wirtualnych. Jest to istotna różnica w stosunku do architektury TrustZone, która nie została zaprojektowana do obsługi dynamicznego tworzenia chronionych środowisk wykonawczych.

Obiekty Realm zostały zaprojektowane do obsługi złożonych schematów zarządzania pamięcią przy istniejących systemach operacyjnych i funkcjach hypervisor i minimalnie inwazyjnych zmianach. Co za tym idzie, konieczny jest mechanizm sterujący dostępem do pamięci wykorzystywanej do implementacji obiektów Realm. Sterowanie dostępnością pamięci z danego świata musi być szczegółowa (z rozdzielczością 4K) i dynamiczna. Cała pamięć fizyczna w systemie *CCA* jest podzielona na moduły granularne. Każdy taki moduł ma zestaw właściwości definiujących ograniczenia, w ramach których można uzyskać dostęp do odpowiednich adresów. Naruszenie tych ograniczeń powoduje wystąpienie błędu.

D.3.1.2. Rozszerzenie do zarządzania obiektami Realm (*ang. Realm Management Extension – RME*) firmy Arm

Technologia Realm Manager jest odpowiedzialna za zarządzanie obiektami Realm. Obiekt Realm to maszyna wirtualna składająca się z jądra systemu operacyjnego (działającego w EL1) i zestawu aplikacji (wykonywanych w EL0). Oprócz dwóch stanów bezpieczeństwa obsługiwanych przez architekturę TrustZone (stan bezpieczny i stan niezabezpieczony), architektura *CCA* wprowadza dwa dodatkowe stany bezpieczeństwa obsługiwane przez rozszerzenie Realm Management Extension (RME): stan Realm i stan Root. W ramach rozszerzenia RME, poziom EL3 zostaje przeniesiony ze stanu bezpiecznego do własnego stanu zabezpieczonego – stanu Root. Rozszerzenie RME zapewnia izolację poziomu EL3 od wszystkich innych bezpiecznych stanów. Stan Realm, niezabezpieczony i bezpieczny, nie muszą ufać poziomowi EL3. Stan bezpieczny i stan Realm mogą sobie wzajemnie nie ufać, ponieważ stan bezpieczny jest odizolowany sprzętowo zarówno od stanu niezabezpieczonego, jak i stanu obiektu Realm, podczas gdy stan Realm jest odizolowany sprzętowo zarówno od stanu niezabezpieczonego, jak i stanu bezpiecznego. Wszystkie inne stany bezpieczeństwa obejmują poziomy EL2, EL1 i EL0 zarówno w świecie Realm, jak i NW. Na rysunku 11 zaprezentowano dodanie świata Realm do NW i SW.

Domena zabezpieczeń Monitor wykonuje oprogramowanie układowe środowiska uruchomieniowego, które zarządza przełączaniem stanów zabezpieczeń i przypisywaniem zasobów do poszczególnych stanów bezpieczeństwa. Zgodność z poprzednimi zastosowaniami, w których wykorzystywano architekturę TrustZone, jest możliwa dzięki zachowaniu stanu bezpiecznego. W ramach takich istniejących zastosowań można jednak korzystać z nowych funkcji rozszerzenia RME, takich jak dynamiczne przydzielanie pamięci.



Rysunek 11: Świat Root (Monitor), świat Realm i granice izolacji

Atestacja architektury *CCA* umożliwia użytkownikowi usługi świadczonej przez obiekt Realm – stronie zależnej – określenie wiarygodności tego obiektu i implementacji architektury *CCA*. Strona zależna może być lokalna lub zdalna.

Dynamiczna architektura TrustZone wykorzystuje rozszerzenie RME w celu zapewnienia mechanizmu przypisywania stron pamięci do niezabezpieczonych i bezpiecznych przestrzeni adresowych w czasie wykonywania. Będące częścią *CCA*, rozszerzenie RME w architekturze Armv9-A umożliwia dynamiczne przenoszenie stron pamięci z NW do SW i z powrotem. W oparciu o rozszerzenie RME, architektura *CCA* zapewnia następujące dodatkowe funkcje, które wzbogacają dynamiczne rozwiązanie TrustZone:

- Podział oprogramowania układowego i izolacja monitora EL3, wykorzystywane do zapewnienia silniejszego źródła zaufania (RoT) i usług atestacji.
- Szyfrowanie wszystkich danych w bezpiecznej, przypisanej pamięci DRAM za pomocą mechanizmu ochrony pamięci.

D.3.1.3. Izolacja i ochrona pamięci obiektów Realm firmy Arm

Jak pokazano na rysunku 11, architektura TrustZone zapewnia dwa stany bezpieczeństwa, z których każdy jest powiązany z własną przestrzenią PAS: bezpieczną przestrzenią PAS i niezabezpieczoną przestrzenią PAS. Rozszerzenie RME

dodaje dwie dodatkowe przestrzenie PAS: przestrzeń Realm PAS i przestrzeń Root PAS. Zapewniają one izolację danych należących do każdej z przestrzeni PAS. Każda przestrzeń PAS ma swój własny tryb translacji adresów. Dostęp urządzenia do pamięci podlega gwarancjom izolacji przestrzeni RME PAS.

D.3.1.4. Szyfrowanie i integralność pamięci zewnętrznej (DRAM) z architekturą *CCA* firmy Arm

Wiele zasobów obiektów Realm i architektury *CCA* jest przechowywanych w pamięci zewnętrznej. Szyfrowanie pamięci za pomocą architektury *CCA* ma na celu zapewnienie dodatkowej izolacji i prywatności między światem Realm, Root i SW oraz w ramach przestrzeni Realm PAS. Dzięki architekturze *CCA* pamięć jest unikalnie szyfrowana dla każdego świata i lokalizacji przy użyciu oddzielnego kontekstu szyfrowania dla każdej przestrzeni PAS (świata Realm, Root i bezpiecznego) przy użyciu innej poprawki adresu dla każdego bloku szyfrowanych danych, aby zapewnić izolację przestrzenną. Szyfrowanie pamięci jest losowo ponownie stosowane podczas uruchamiania systemu, aby cała istniejąca zaszyfrowana zawartość pamięci była niemożliwa do odszyfrowania po zresetowaniu systemu.

Dane są szyfrowane sprzętowo przed zapisaniem w pamięci zewnętrznej lub współdzielonej pamięci podręcznej, która znajduje się poza punktem fizycznego aliasu.

Architektura *CCA* chroni przed wieloma różnymi rodzajami ataków na pamięć, w tym:

- **Bezpośredni dostęp do pamięci:** Nieautoryzowany agent w tym samym systemie próbuje uzyskać bezpośredni dostęp do zawartości pamięci przydzielonej do innego świata lub obiektu Realm.
- **Sondowanie:** Atakujący próbuje uzyskać fizyczny dostęp do zawartości pamięci przydzielonej do świata lub obiektu Realm – na przykład za pomocą sond sprzętowych lub urządzeń rejestrujących, takich jak moduł pamięci NVDIMM (Non-Volatile Dual In-Line Memory Module).
- **Wyciek:** Atakujący uzyskuje fizyczny dostęp do zawartości pamięci przydzielonej do świata lub obiektu Realm, na przykład poprzez błędną konfigurację lub błędy w implementacji platformy *CCA*.

D.3.1.5. Rozruch oprogramowania układowego w architekturze Arm *CCA*

Oprogramowanie sprzętowe świata Root w ramach architektury *CCA* jest uruchamiane przed oprogramowaniem układowym opisanym dla SW w załączniku D.3.1.4. Zaczyna się od niezmiennego początkowego kodu rozruchowego, który może znajdować się na wbudowanej pamięci ROM lub w zablokowanej pamięci na chipie. W zabezpieczonym systemie niezmienny początkowy kod rozruchowy jest z natury zaufany i nie jest weryfikowany ani mierzony. Jest on uważany za część domeny zabezpieczeń sprzętowych *CCA* i identyfikowany na podstawie identyfikatora atestacji platformy *CCA*. Wszelkie podlegające aktualizowaniu oprogramowanie sprzętowe *CCA*, w tym oprogramowanie układowe domeny zabezpieczeń zarządzania obiektami Realm, jest zarówno weryfikowane, jak i mierzone i jest to raportowane w atestacji platformy *CCA*.

Obiekty Realm są mierzone przez oprogramowanie układowe domeny zabezpieczeń zarządzania obiektami Realm podczas tworzenia obiektu Realm i zgłaszane w atestacji obiektu Realm powiązanym z bieżącym atestowaniem platformy *CCA*.

Łańcuch rozruchu SW może zostać rozpoczęty przez oprogramowanie układowe Monitor. Łańcuch rozruchowy NW jest uruchamiany przez SW, gdy tworzy on wystąpienie modułu ładującego rozruch (np. UEFI) w NW. Są one kontynuowane niezależnie od architektury *CCA*.

Technologia *CCA* Hardware Enforced Security (HES) jest hostowana w zaufanym podsystemie i implementuje podstawowe usługi bezpieczeństwa architektury *CCA*, takie jak:

- Stan rozruchu platformy *CCA*.
- Usługi atestacji platformy *CCA*.
- Parametry platformy *CCA* Root.
- Cykl życia zabezpieczeń systemu *CCA*.
- Zapewnianie bezpieczeństwa architektury *CCA*.
- Zestawianie stanu rozruchu sprzętu *CCA*.

Oprogramowanie układowe *CCA* obejmuje:

- Na urządzeniu HES:
  - ✓ Oprogramowanie układowe urządzenia HES.
  - ✓ Zaufane oprogramowanie układowe podsystemu dla hosta HES.

- Na hoście procesora aplikacji (PE):
  - ✓ Oprogramowanie sprzętowe PE aplikacji dla domen zabezpieczeń Monitor i Realm Management.
  - ✓ Oprogramowanie układowe zaufanego podsystemu dla wszystkich innych zaufanych podsystemów w domenie zabezpieczeń systemu *CCA*.

D.3.1.6. Rozszerzenie Arm RME oraz ochrona przed debugowaniem, śledzeniem, profilowaniem i monitorowaniem wydajności

Systemy Arm są wyposażone w zaawansowane funkcje umożliwiające debugowanie, śledzenie i profilowanie. Rozszerzenie RME zapewnia elementy sterujące, które mogą być użyte do ograniczenia tego, które części systemu mogą być debugowane. Dostępne są sygnały włączające różne funkcje debugowania, śledzenia i profilowania, które pomagają zarządzać tymi funkcjami. Zewnętrzne sygnały debugowania są zwykle podłączone do zabezpieczeń lub modułu uwierzytelniania, w którym debugowanie dla każdego stanu można włączyć lub wyłączyć, po jednym dla każdego z czterech sygnałów. Są one zazwyczaj zarządzane i wyłączane w zależności od bieżącego stanu cyklu życia urządzenia. Zewnętrzny dostęp do jednostek monitorowania wydajności jest regulowany przez rozszerzenie RME.

Rejestry sprzętowe kontrolują, kiedy własne usługi śledzenia debugowania, profilowania i monitorowania wydajności są przełączane kontekstowo podczas przełączania między stanami bezpieczeństwa lub obiektami Realm.

D.3.1.7. Usługa atestacji zdalnej – projekt Veraison (VERificAtIon of atteStatiON)

W ramach tej inicjatywy tworzone jest otwarte oprogramowanie, które może być wykorzystywane do budowania usług weryfikacji atestacji urządzeń, które mogą obsługiwać wiele architektur. Aby umożliwić obsługę architektury Arm *CCA*, w ramach projektu Veraison opracowane zostaną wtyczki implementujące model atestacji architektury Arm *CCA*. Oprogramowanie Veraison obsługuje weryfikację (obejmuje implementacje referencyjne dla tokenu atestacji obiektu [68], profilu EAT PSA i mechanizmu komponowania identyfikatora urządzenia [69]) oraz inicjowanie obsługi (interfejs API umożliwiający dostarczanie wartości referencyjnych / poświadczeń z łańcucha dostaw, odwoływanie obiektów, separację danych wielu dzierżawców, a także ścieżkę audytu). Jest ono przewidziane jako wdrożenie referencyjne, które może być hostowane samodzielnie lub jako usługa (*platform-as-a-service*).

D.3.2. Akceleracja kryptograficzna firmy Arm

D.3.2.1. Rozszerzenia kryptograficzne

W ramach rozszerzeń kryptograficznych architektury ARMv8A [47] dodano 32 nowe instrukcje Advanced SIMD, które działają w oparciu o plik rejestru wektorowego. Można je wykorzystać do przyspieszenia wykonywania algorytmów kryptograficznych wymienionych poniżej:

- Bezpieczny algorytm wyznaczania wartości skrótu 1 (*Secure Hash Algorithm 1 – SHA1*), SHA256 i AES: instrukcje te zostały dodane do zestawów instrukcji A32 i A64.
- SHA512, SHA3, SM3 i SM4: te instrukcje zostały dodane wyłącznie do zestawu instrukcji A64.

Zapewniają one od trzech do dziesięciu razy szybsze szyfrowanie programowe. Są one przydatne do odszyfrowywania i szyfrowania małych modułów granularnych, które są zbyt małe, aby przenoszenie ich do zewnętrznego akceleratora sprzętowego było efektywne.

D.3.2.2. Generowanie rzeczywistych liczb losowych (True Random Number Generation – TRNG)

Generator rzeczywistych liczb losowych (*True Random Number Generator – TRNG*) zapewnia entropię w postaci liczb losowych z próbkowanego wyniku

nieprzewidywalnego procesu fizycznego, a nie za pomocą algorytmu, jak w przypadku generatora liczb pseudolosowych (*Deterministic Random Bit Generator – DRBG*). Głównym zastosowaniem rzeczywistych liczb losowych jest kryptografia, gdzie są one wykorzystywane do generowania losowych kluczy kryptograficznych, np. pary kluczy, inicjatora i identyfikatora jednorazowego w algorytmie podpisu cyfrowego krzywej eliptycznej (*Elliptic Curve Digital Signature Algorithm – ECDSA*), symetrycznych kluczy MAC, w celu bezpiecznego przechowywania i przesyłania danych (np. w ramach protokołów szyfrowania, takich jak TLS). Klucze wygenerowane przy użyciu TRNG są nieprzewidywalne, a zatem są wysoce odporne na ataki polegające na zgadywaniu oparte na zrozumieniu implementacji algorytmu.

Dodano dwie nowe instrukcje dotyczące liczb losowych, RNDR i RNDRRS [47]. Zwracają one 64-bitową liczbę losową do rejestru ogólnego przeznaczenia. Odczyt do rejestru RNDRRS spowoduje ponowne wygenerowanie inicjatora przed wygenerowaniem i zwróceniem nowej liczby losowej.

Generator DRBG tworzy liczby losowe na podstawie kryptograficznie zabezpieczonego algorytmu i otrzymuje inicjatory z TRNG. Generator TRNG jest zgodny z kilkoma standardami, w tym NIST SP 800-90B, NIST SP 800-22, FIPS 140-2 i AIS-31 organizacji British Standards Institution (BSI). Algorytm DRBG jest zgodny ze standardem NIST SP 800-90A w wersji 1. Cały proces generowania liczb losowych jest zgodny ze standardem NIST SP 800-90C.

ZAŁĄCZNIK E – PRZYKŁADY TECHNOLOGII FIRMY CISCO

W niniejszym załączniku opisano przykłady technologii firmy Cisco, które odpowiadają kluczowym koncepcjom opisanym w różnych częściach tego dokumentu. Podane przykłady koncentrują się na rozwiązaniu Unified Computing System (UCS) firmy Cisco, otwartym serwerze, który zawiera procesory x86 innych dostawców wymienionych w załącznikach i umożliwia użytkownikom dodawanie kolejnych technologii wymienionych w tym dokumencie.

E.1 WERYFIKACJA INTEGRALNOŚCI PLATFORMY

E.1.1. Źródła zaufania platformy firmy Cisco

Aby zapewnić najwyższy możliwy stopień integralności, w platformach obliczeniowych Cisco stosuje się metodologię obrony w głąb, aby zaprojektować platformę, która zapobiega zarówno powszechnym, jak i wyrafinowanym atakom, wykorzystując do tego celu technologię opartą na różnych komponentach sprzętowych. W produktach Cisco UCS znajdują się dwa źródła zaufania platformy (*Platform Root of Trust – PRoT*), zwykle określane w niniejszym dokumencie jako sprzętowe moduły zabezpieczeń. Kontroler Cisco Integrated Management Controller jest PRoT na serwerach Cisco, a kontroler Cisco Chassis Management Controller (CMC) jest PRoT dla Cisco IO Module, Cisco Intelligent Fabric Manager i obudowy mainframe.

Oba źródła PRoT na platformach obliczeniowych Cisco implementują sprzętowe źródło zaufania, służące do zapewniania integralności i autentyczności oprogramowania układowego. Są one oparte na niezmiennej pamięci (np. ROM), we wbudowanym czipie SoC lub innym układzie sprzętowym.

Cisco rozszerza integralność platformy, aby uwzględnić jej autentyczność, zakorzenioną w module Cisco Trust Anchor Module, takim jak dyskretna technologia Anti-Counterfeit Technology 2 lub moduł TPM do użytku tylko na poziomie platformy. Ten zastrzeżony, odporny na manipulacje chip znajduje się w wielu produktach Cisco i zapewnia nieulotną bezpieczną pamięć, bezpieczny unikalny identyfikator urządzenia oraz usługi kryptograficzne, w tym generowanie liczb losowych, bezpieczny magazyn, zarządzanie kluczami i usługi kryptograficzne dla

uruchomionego systemu operacyjnego i aplikacji. Autentyczność platformy zapewnia, że platforma nie jest podrobiona i umożliwia zagwarantowanie krytycznych parametrów bezpieczeństwa podczas inicjowania obsługi.

Funkcje zapewniające integralność platformy firmy Cisco zapobiegają również niektórym atakom fizycznym. Płytki drukowane (*Printed Circuit Board – PCB*) serwerów Cisco są wytwarzane w taki sposób, że sygnały krytyczne są przesyłane w wewnętrznych warstwach płytki, nie są dostępne w jej górnych lub dolnych warstwach, w przypadku których łatwo może dojść do sondowania i modyfikacji integralności elektrycznej. Dodatkowo, Cisco wykorzystuje konstrukcję pakowania typu pin-side-down (np. Ball Grid Array) dla elementów lutowanych, aby zapobiec łatwej manipulacji elektrycznej tymi pinami.

Ponieważ serwery zawierają zazwyczaj wymienne komponenty, podzespoły, części lub urządzenia, platformy obliczeniowe Cisco zapewniają uwierzytelnianie dla urządzeń obsługujących protokół bezpieczeństwa i model danych (*Security Protocol and Data Model – SPDM*), np. kontroler pamięci masowej, lub innych komponentów Cisco wymienianych u klienta (*ang. Field Replacement Units – FRU*), np. wirtualna karta interfejsu Cisco. SPDM to standard zabezpieczania komunikacji i uwierzytelniania urządzeń w ramach platformy.

E.1.2. Łańcuch zaufania (Chain of Trust – CoT) firmy Cisco

Holistyczne podejście Cisco do integralności platformy koncentruje się na jej ochronie w trakcie działania. Integralność i autentyczność oprogramowania układowego (powszechnie znane jako bezpieczny rozruch lub łańcuch zaufania) zaczynają się od sprzętowego źródła zaufania (RoT). Integralność oprogramowania układowego ma kluczowe znaczenie, ponieważ steruje ono określonym urządzeniem (np. kontroler sieci) lub różnymi urządzeniami i operacjami na platformie, takimij jak Integrated Management Controller i CMC firmy Cisco.

Źródło PRoT firmy Cisco ustanawia łańcuch zaufania od sprzętu przez oprogramowanie układowe po integralność oprogramowania, zapewniając, że początkowy rozruch procesorów hosta ma dodatkową ochronę. Źródło PRoT dla serwerów firmy Cisco

przeprowadza dodatkową weryfikację kodu BIOS i sprawdza integralność płytki drukowanej, porównując oczekiwane wartości z wartościami rzeczywistymi. Zapewnia to dodatkową ochronę przed określonymi atakami polegającymi na wymianie pojedynczych komponentów. Wszystkie komponenty platform firmy Cisco implementują bezpieczny rozruch i CoT. Więcej informacji można znaleźć tutaj.

E.2 OCHRONA ŁAŃCUCHA DOSTAW FIRMY CISCO

Kompleksowy program firmy Cisco mający na celu zarządzanie ryzykiem w łańcuchu dostaw i zapewnienie ochrony klientom został opisany w dokumencie Best Practices In Cyber Supply Chain Risk Management. Ten program ochrony łańcucha dostaw obejmuje procesy sprawdzania przez Cisco autentyczności i integralności sprzętu platformy podczas instalowania krytycznych parametrów bezpieczeństwa Cisco na źródłach Cisco PRoT. Te parametry bezpieczeństwa są oparte na źródle PRoT, ustanawiając bezpieczną tożsamość, na podstawie której można uwierzytelnić CoT.

E.3 MECHANIZMY OCHRONY ŚRODOWISKA URUCHOMIENIOWEGO OPROGRAMOWANIA FIRMY CISCO

Nowoczesne platformy zawierają wiele inteligentnych urządzeń, na których działa jakaś forma kodu. Chociaż kod ten nie jest uważany za sprzęt, odgrywa on ważną rolę we wpływie na ogólną integralność platformy i ma kluczowe znaczenie dla zapewnienia, że platforma działa na najwyższym poziomie bezpieczeństwa. Niezależnie od tego, czy jest to kod FPGA, który przekazuje konfigurację do regulatora napięcia, czy złożone oprogramowanie działające na źródle PRoT, Cisco stosuje dodatkowe kontrole integralności, aby zagwarantować, że kod lub krytyczne parametry bezpieczeństwa nie zostały złośliwie zmodyfikowane.

W architekturze oprogramowania układowego PRoT firmy Cisco wykorzystano wiele funkcji zabezpieczeń na poziomie sprzętowym, odpowiednich do ograniczeń i wymagań dotyczących bezpieczeństwa projektu. Na przykład firma Cisco zapewnia izolację stron kodu i stron danych, dzięki czemu kod nie może być uruchamiany ze stron danych. Strony aplikacji są randomizowane poprzez losowe generowanie układu przestrzeni adresowej (*Address Space Layout Randomization – ASLR*). Jak opisano

powyżej, oprogramowanie układowe wykorzystuje zabezpieczenia sprzętowe do ochrony sekretów lub wrażliwych danych podczas ich przechowywania, przesyłania i przetwarzania. W niektórych platformach kod pamięci ROM jest wykorzystywany do zapewnienia autentyczności oprogramowania układowego podczas procesu rozruchu. W przypadku krytycznych aplikacji i produktów Cisco korzysta z usług zewnętrznego lub wewnętrznego podmiotu zajmującego się bezpieczeństwem informacji, którego specjalnością jest analizowanie kodu źródłowego pod kątem problemów związanych z bezpieczeństwem z uwzględnieniem podejścia atakującego. Oprogramowanie układowe Cisco poddaje się również certyfikacji FIPS i Common Criteria, aby zapewnić użytkownikowi określony poziom bezpieczeństwa.

Po zakończeniu uruchamiania komponentu, gdy jego oprogramowanie układowe już działa, kolejnym ważnym aspektem ogólnej integralności platformy i ochrony łańcucha dostaw jest integralność oprogramowania układowego w środowisku uruchomieniowym. Komponenty oprogramowania układowego i ich zawartość są kontrolowane i weryfikowane. Oprogramowanie typu open-source lub zakupione komponenty są analizowane wewnętrznie przed ich integracją. Analiza obejmuje wyszukiwanie wywołań systemowych powszechnych w języku C lub równoważnych językach, aby się upewnić, że nie ma złośliwych wstrzyknięć do wiersza poleceń, że kod zapewnia oczyszczanie danych wejściowych, weryfikację kodu pod kątem analizy statycznej oraz odpowiednie rozwiązywanie wszelkich błędów i ostrzeżeń z analizy statycznej. Wewnętrznie Cisco zapewnia, że infrastruktura i serwery programistyczne spełniają rygorystyczne wymagania dotyczące zabezpieczeń i chronią przepływ oprogramowania od bardzo wczesnych etapów rozwoju do oficjalnego wydania użytkownikom, co zapobiega wstrzykiwaniu złośliwego kodu w łańcuchu dostaw oprogramowania. Stosowane metody obejmują dodatkowe kontrole integralności przechowywanego kodu. Proces ten jest stale weryfikowany i ulepszany. Źródło PRoT firmy Cisco dla niektórych platform i jednostki FRU udostępniają pomiary oprogramowania układowego użytkownikom końcowym w celu zweryfikowania stanu tego komponentu. Ponadto źródła PRoT firmy Cisco wykorzystują wewnętrzne procesy, które samodzielnie monitorują krytyczne zasoby lub krytyczne parametry bezpieczeństwa, aby się upewnić, że nie zostały one naruszone.

E.4 OCHRONA DANYCH I POUFNOŚĆ PRZETWARZANIA FIRMY CISCO

Innym aspektem integralności oprogramowania układowego w środowisku uruchomieniowym jest zużycie i przetwarzanie danych. Źródło PRoT firmy Cisco wykorzystuje bezpieczną pamięć opartą na sprzęcie, aby zapewnić ochronę przechowywanych danych. Zabezpieczenie to jest możliwe dzięki sprzętowym kryptograficznym kotwicom zaufania, podobnym do zaufanych środowisk wykonawczych (*Trusted Execution Environment – TEE*), które bezpiecznie przechowują tajne dane, takie jak klucze szyfrowania i chronią przed niektórymi atakami typu side channel, takimi jak ataki fizyczne przy użyciu technik z wykorzystaniem częstotliwości radiowej (*Radio Frequency – RF*). Przetwarzanie danych przez sieć jest zabezpieczone przy użyciu standardowych protokołów bezpieczeństwa, takich jak TLS, z tożsamościami umocowanymi w module Cisco Trust Anchor Module.

E.5 ATESTACJA PLATFORMY FIRMY CISCO

Źródło PRoT firmy Cisco stale monitoruje komponenty systemu w celu wykrycia degradacji integralności. Może wysyłać ostrzeżenia lub podejmować działania naprawcze, włącznie z aktywacją trybu blokady, który uniemożliwia korzystanie z urządzenia do czasu usunięcia usterki. Źródło PRoT firmy Cisco dostarcza alerty lub powiadomienia za pośrednictwem wielu protokołów: Simple Network Management Protocol, Simple Mail Transfer Protocol, przekazywanie dziennika systemowego (*ang. syslog*), Redfish oraz xAPI firmy Cisco. Źródło PRoT monitoruje inne komponenty na platformie i, w zależności od ich statusu i roli, może podejmować działania naprawcze lub ostrzegać użytkownika. Jeśli degradacja integralności jest wystarczająco poważna, przechodzi w tryb blokady i uniemożliwia dalsze użytkowanie do czasu naprawienia problemu.

E.6 WIDOCZNOŚĆ INFRASTRUKTURY BEZPIECZEŃSTWA FIRMY CISCO

Platforma Cisco UCS zapewnia weryfikowalne kryptograficznie raporty dotyczące integralności platformy i szczegółów bezpieczeństwa, w tym pomiary BIOS dla źródła PRoT.

E.7 LOKOWANIE OBCIĄŻEŃ NA ZAUFANYCH PLATFORMACH FIRMY CISCO

Cisco Intersight to oparte na chmurze lub lokalne rozwiązanie do zarządzania, które sprawdza sprzęt platformy UCS pod kątem autentyczności przy użyciu źródła Cisco PRoT, zanim zostanie ona przejęta lub zacznie być zarządzana za pomocą rozwiązania Intersight.

ZAŁĄCZNIK F – PRZYKŁADY TECHNOLOGII FIRMY *IBM*

W niniejszym załączniku opisano przykłady technologii firmy *IBM*, które odpowiadają kluczowym koncepcjom opisanym w poprzednich częściach tego dokumentu.

F.1 WERYFIKACJA INTEGRALNOŚCI PLATFORMY

F.1.1. Sprzętowy moduł zabezpieczeń (Hardware Security Module – HSM)

Moduły HSM zawierają koprocesory kryptograficzne (takie jak *IBM* 4758 i nowsze modele), które dodatkowo zapewniają bardzo silne funkcje wykrywania manipulacji, zapobiegania i reagowania na ataki fizyczne. Moduły *IBM* HSM są dostępne jako funkcje systemów *IBM* Z, LinuxONE i *IBM* POWER [70]. Moduły HSM oferują dodatkowe warstwy ochrony w przypadku wdrożeń w chmurze [71].

F.1.2. Łańcuch zaufania (Chain of Trust – CoT) firmy IBM

Systemy *IBM* Z i *IBM* POWER są wyposażone w sprzętowe źródło zaufania, które umożliwia korzystanie z kluczowych funkcji bezpieczeństwa, w tym zaufanego rozruchu i zwiększonej poufności danych.

Moduł TPM jest specjalnym typem modułu HSM. Serwery POWER9 są wyposażone w moduł TPM 2.0. Specyfikację modułu TPM 2.0 można znaleźć na stronie internetowej TCG [72][73].

W systemach POWER 9 Enterprise zastosowano ulepszenia sprzętowe i oprogramowania układowego, dzięki czemu są one jeszcze bezpieczniejsze w przypadku wdrożeń w chmurze hybrydowej. Jednym z ulepszeń jest proces Secure Initial Program Load (IPL) lub Secure Boot, który umożliwia uruchamianie w systemie wyłącznie oprogramowania układowego podpisanego przez producenta platformy, związanego z oprogramowaniem Hostboot i POWER Hypervisor, w tym oprogramowania układowego partycji. Kolejnym ulepszeniem jest struktura do obsługi zdalnej atestacji stosu oprogramowania układowego systemu za pośrednictwem sprzętowego modułu TPM.

Funkcja Secure Boot implementuje oparty na procesorze łańcuch zaufania oparty na sprzęcie procesora POWER9 i aktywowany przez stos oprogramowania układowego

POWER9. Funkcja Secure Boot zapewnia zaufaną bazę oprogramowania układowego w celu zwiększenia poufności i integralności danych klienta w zwirtualizowanym środowisku. Funkcja POWER9 Trusted Boot zapewnia pomiary konfiguracji systemu i kodu ścieżki IPL (*ang. Initial Program Load*), które mogą być później wykorzystane jako dowód dla strony trzeciej poprzez atestowanie początkowej konfiguracji ścieżki IPL systemu. Aby utworzyć moduł CRTM, używany jest przepływ funkcji Secure Boot, który dodaje kontrole kryptograficzne do każdej fazy procesu IPL, aż do ustanowienia komunikacji z modułem TPM. Przepływ ten ma na celu zapewnienie integralności całego oprogramowania układowego, które ma być wykonywane na procesorach rdzeniowych, zapobiegając w ten sposób uruchomieniu nieautoryzowanego lub złośliwie zmodyfikowanego oprogramowania układowego. Niepowodzenie weryfikacji komponentu oprogramowania układowego uniemożliwi ukończenie procesu IPL, jeśli komponent jest uznawany za krytyczny dla prawidłowego działania systemu. Jeśli komponent nie jest podstawową funkcją krytyczną, jego obraz po nieudanej weryfikacji nie zostanie wykonany, proces IPL będzie mógł zostać ukończony i zostaną wygenerowane odpowiednie powiadomienia.

Szczegółowe informacje na temat implementacji bezpiecznego i zaufanego rozruchu w systemie *IBM* POWER można znaleźć w publikacjach [74] i [75]. Szczegółowe informacje na temat implementacji bezpiecznego i zaufanego rozruchu w systemach *IBM* Z i LinuxONE można znaleźć w publikacji [76].

F.2 MECHANIZMY OCHRONY ŚRODOWISKA URUCHOMIENIOWEGO OPROGRAMOWANIA

F.2.1. Zabezpieczenia przed atakami typu ROP i COP/JOP firmy IBM

W ramach platformy POWER dodano cztery instrukcje (hashst, hashchk, hashstp, hashchkp) do przeciwdziałania atakom typu *ROP* w Power ISA 3.1B, począwszy od procesora Power10.

F.3 OCHRONA DANYCH I POUFNOŚĆ PRZETWARZANIA

W obecnych środowiskach przetwarzania danych aplikacje polegają na oprogramowaniu systemowym podczas świadczenia usług, takich jak zarządzanie dostępem do zasobów systemu komputerowego. Oprogramowanie systemowe obejmuje (co najmniej) system operacyjny, ale może również obejmować funkcję hypervisor. Zazwyczaj oprogramowanie systemowe musi być zaufane, ponieważ ma pełną kontrolę nad aplikacjami i ich danymi. System operacyjny lub funkcja hypervisor mogą uzyskać dostęp do danych dowolnej aplikacji lub zmodyfikować je, a także potencjalnie manipulować zabezpieczeniami zaimplementowanymi przez aplikację bez wykrycia takiego działania. Dlatego oprogramowanie bazowe musi być częścią zaufanej bazy przetwarzania (*ang. Trusted Computing Base – TCB*).

W środowiskach współdzielonych klienci są zmuszeni ufać, że jednostki, które opracowują, konfigurują, wdrażają oprogramowanie systemowe i sterują nim, nie są złośliwe. Klienci muszą również ufać, że oprogramowanie systemowe jest odporne na ataki, które powodują eskalację uprawnień oraz naruszenie poufności i integralności aplikacji klientów. Ten wymóg szerokiego zaufania jest często trudny do spełnienia i stwarza znaczne ryzyko, zwłaszcza dla klientów, którzy korzystają z usług w chmurze publicznej. Firma *IBM* przewodziła wysiłkom podejmowanym w całej branży w celu rozwiązania takich problemów poprzez zmniejszenie rozmiaru bazy TCB i wprowadzenie technologii, które sprawiają, że dane klientów są niedostępne dla administratorów systemu i chmury.

F.3.1. Technologia izolacji pamięci firmy IBM

Przetwarzanie poufne (przykłady w tekście na temat izolacji maszyn wirtualnych poniżej) i szyfrowanie zawartości w pamięci są skutecznymi technologiami bezpieczeństwa umożliwiającymi osiągnięcie izolacji pamięci, jednak dopiero korzystanie z nich w połączeniu z innymi technologiami izolacji może zapewnić solidny, wielowarstwowy schemat ochrony.

Jednym z przykładów takiej technologii izolacji jest Z Processor Resource/System Manager (PR/SM) firmy *IBM*. PR/SM to funkcja hypervisor typu 1 zintegrowana ze

wszystkimi modelami *IBM* Z, która przekształca zasoby fizyczne w wirtualne, dzięki czemu wiele partycji logicznych (*ang. Logical Partition – LPAR*) może współdzielić te same zasoby fizyczne. Zapewnia możliwość podziału fizycznych zasobów systemu (dedykowanych lub współdzielonych) na izolowane partycje logiczne. Każda partycja logiczna działa jak niezależny system z własnym środowiskiem operacyjnym. Technologia PR/SM umożliwia każdej partycji logicznej posiadanie dedykowanych lub współdzielonych procesorów i wejść/wyjść oraz dedykowanej pamięci (której konfigurację można dynamicznie zmieniać w razie potrzeby). Technologia PR/SM zapewnia administratorowi bezpieczeństwa możliwość zdefiniowania całkowicie bezpiecznej konfiguracji systemu. Gdy system jest zdefiniowany w taki sposób, uzyskuje się całkowitą separację partycji logicznych, zapobiegając w ten sposób uzyskaniu przez dowolną partycję wiedzy o działaniu innej partycji. Ta technologia izolacji została oceniona zgodnie z normą Common Criteria i osiągnęła poziom weryfikacji formalnej (*ang. Evaluation Assurance Level – EAL*) 5+ [77].

F.3.2. Technologia izolowania aplikacji firmy IBM

IBM Hyper Protect Virtual Servers to nowa technologia oparta na środowisku *IBM* Secure Service Containers do ochrony obciążeń w systemach *IBM* Z i LinuxONE w całym cyklu życia aplikacji. *IBM* Secure Service Container (SSC) to kontenerowe środowisko uruchomieniowe umożliwiające szybkie i bezpieczne wdrażanie urządzeń programowych na serwerach. Urządzenie programowe to integracja systemu operacyjnego, oprogramowania pośredniczącego i komponentów oprogramowania, które działają autonomicznie i zapewniają podstawowe usługi i infrastruktury, które koncentrują się na możliwości wykorzystania i bezpieczeństwie.

Celem technologii SSC jest zapewnienie środowiska uruchomieniowego dla obciążeń, w tym dostępu do sieci, pamięci i adapterów kryptograficznych. Głównym celem jest ochrona wszelkich danych utworzonych przez obciążenie i zapewnienie dostępu do danych tylko określonej instancji obciążenia. Oznacza to, że nawet drugie obciążenie tego samego typu nigdy nie uzyska dostępu do żadnych danych utworzonych przez inną instancję. Technologia SSC działa tylko wewnątrz partycji logicznej typu SSC.

Oprogramowanie układowe wyłącza dostęp do pamięci dla partycji logicznych typu SSC, a specjalny moduł ładujący rozruch/łańcuch rozruchu zapewnia, że tylko podpisane urządzenia oparte na technologii SSC mogą być uruchamiane w takiej partycji. Nie ma dostępu do urządzenia hostingowego za pośrednictwem protokołu SSH, a działania mogą być inicjowane wyłącznie za pomocą dobrze zdefiniowanych i przetestowanych interfejsów API, zawsze z założeniem, że tylko obciążenie ma dostęp do swoich danych.

Szczegółowe informacje na temat technologii *IBM* Hyper Protect Virtual Servers można znaleźć w publikacji [78].

F.3.3. Technologia izolacji maszyny wirtualnej firmy IBM

Zarówno *IBM* POWER Protected Execution Facility (PEF), jak i *IBM* Secure Execution dla systemu Linux (IBM Z i LinuxONE) są przykładami środowiska *TEE* zapewniającego izolację maszyn wirtualnych/pamięci. Technologia *IBM* POWER PEF umożliwia obsługę bezpiecznych maszyn wirtualnych (*ang. Secure Virtual Machine – SVM*) w systemach *IBM* Power. Chroni ona maszyny SVM przed innym oprogramowaniem, gdy są one przechowywane, w trakcie przesyłania i podczas działania. Maszyny SVM są obsługiwane przez nowy tryb w architekturze *IBM* Power Architecture o nazwie Ultravisor, który ma wyższe uprawnienia niż tryb hypervisor. Maszyna SVM może działać tylko w systemach obsługujących technologię PEF i zweryfikowanych przez klienta, który utworzył maszynę SVM. Każdy system obsługujący technologię PEF ma parę kluczy publiczny/prywatny, w której klucz prywatny jest znany tylko systemowi (nie jest ujawniany właścicielowi systemu). Więcej szczegółowych informacji na temat technologii PEF można znaleźć w publikacji [78]. Instrukcje dotyczące konfiguracji stosu oprogramowania z obsługą technologii PEF można znaleźć w publikacji [79]. Kod PEF/Ultravisor jest dostępny jako oprogramowanie typu open source pod adresem [80].

IBM Secure Execution dla systemu Linux to sprzętowa technologia bezpieczeństwa wbudowana w systemy *IBM* z15T i LinuxONE III generacji. Została zaprojektowana w celu zapewnienia skalowalnej izolacji dla poszczególnych obciążeń, aby umożliwić ich ochronę nie tylko przed atakami zewnętrznymi, ale także przed zagrożeniami

wewnętrznymi. Technologia Secure Execution zapewnia izolację między hostem funkcji hypervisor maszyny wirtualnej opartej na jądrze (*ang. Kernel-based Virtual Machine – KVM*) a gośćmi w środowiskach wirtualnych. Ten poziom izolacji pionowej ma na celu wyeliminowanie możliwości uzyskania przez administratorów pełnego wglądu we wrażliwe obciążenia hostowane w maszynach wirtualnych i poszczególnych kontenerach. Technologia Secure Execution zapewnia zaostrzone ograniczenia dostępu w celu ochrony własności intelektualnej i zastrzeżonych tajemnic, jednocześnie umożliwiając administratorom zarządzanie i wdrażanie obciążeń jako czarnych skrzynek i kontynuowanie normalnej pracy. Technologia Secure Execution umożliwia również przedsiębiorstwom izolację pomiędzy poszczególnymi obciążeniami dla wielu użytkowników działającymi na współdzielonej partycji logicznej. Szczegółowe informacje na temat technologii *IBM* Secure Execution dla systemu Linux można znaleźć w publikacji [81].

Technologie *IBM* POWER PEF i *IBM* Secure Execution dla systemu Linux wykorzystują kilka kluczowych innowacji opracowanych przez *IBM* Research i mających na celu umożliwienie zaufanego wykonywania, takich jak technologie SecureBlue [82] i SecureBlue++ [83], [84], które położyły podwaliny pod bezpieczną izolację aplikacji.

F.3.4. Technologia akceleracji kryptograficznej firmy IBM

Systemy *IBM* Z, LinuxONE i *IBM* Power są wyposażone w zintegrowane moduły HSM podłączane przez złącze PCIe. Przykłady obejmują koprocesor *IBM* 4767 oferowany w ramach systemów *IBM* Z i LinuxONE z kartą CryptoExpress5s, systemy *IBM* Power z koprocesorami EJ32 i EJ33 oraz koprocesor *IBM* 4769 oferowany w ramach systemów *IBM* Z i LinuxONE z kartą CryptoExpress7s, który wkrótce będzie oferowany w systemach *IBM* Power [85], [86].

Funkcje CryptoExpress w systemach *IBM* Z i LinuxONE zapewniają elastyczność obsługi różnych typów obciążeń i mogą być skonfigurowane jako akcelerator kryptograficzny, koprocesor kryptograficzny *IBM* Common Cryptographic Architecture lub koprocesor kryptograficzny *IBM* Enterprise Public Key Cryptography Standards (PKCS) #11 (EP11) [87].

Systemy *IBM* Z i LinuxONE udostępniają również instrukcje CP Assist for Cryptographic Functions (CPACF) do wysokowydajnej akceleracji kryptograficznej typu on-core dla symetrycznych i asymetrycznych operacji kryptograficznych. W połączeniu z adapterem *IBM* CryptoExpress, klucze znajdujące się w module HSM mogą być eksportowane w sposób zaszyfrowany i używane w ramach instrukcji CPACF, dzięki czemu klucz nigdy nie jest jawny w pamięci funkcji hypervisor, systemu operacyjnego lub aplikacji [88].

F.4 USŁUGI ATESTACJI ZDALNEJ

F.4.1. Narzędzia do atestacji platformy firmy IBM

IBM TPM Attestation Client Server Framework od *IBM* Research to narzędzie typu open-source do przeprowadzania atestacji platformy [89].

F.4.2. Stałe atestowanie środowiska uruchomieniowego firmy IBM

Ciągłe monitorowanie agenta aplikacji jest możliwe dzięki rozszerzeniu pomiarów na całe środowisko uruchomieniowe, a nie wykonywanie ich tylko w trakcie rozruchu. Na przykład, po uruchomieniu, architektura Integrity Measurement Architecture (IMA) w jądrze systemu Linux będzie stale rozszerzać pomiary na środowisko uruchomieniowe [90]. Pomiary te mogą być okresowo atestowane w ramach usługi weryfikacji, która nie tylko sprawdza nieoczekiwane zmiany w agencie aplikacji, ale także monitoruje jego dynamiczne zachowanie [91].

ZAŁĄCZNIK G – AKRONIMY I SKRÓTY

Wybrane akronimy i skróty użyte w niniejszej publikacji zostały rozwinięte poniżej.

Dodatkowo patrz: NSC 7298, *Słownik kluczowych pojęć z zakresu cyberbezpieczeństwa*

Akronim	Terminologia angielska	Terminologia polska
ABI	Application Binary Interface	Binarny interfejs aplikacji
AC RAM	Authenticated Code Random Access Memory	Pamięć RAM uwierzytelnionego kodu
ACM	Authenticated Code Module	Uwierzytelniony moduł kodu
AEAD	Authenticated Encryption with Associated Data	Uwierzytelnione szyfrowanie z powiązanymi danymi
AES	Advanced Encryption Standard	Zaawansowany standard szyfrowania
AMD PSB	AMD Platform Secure Boot	Technologia *AMD* PSB
API	Application Programming Interface	Interfejs programistyczny aplikacji (interfejs API)
AS	Attestation Service	Usługa atestacji (zaświadczania)
ASID	Address Space IDentifier	Identyfikator przestrzeni adresowej
ASLR	Address Space Layout Randomization	Losowe generowanie układu przestrzeni adresowej
ASP	AMD Security Processor	Podsystem *AMD* Security Processor

BIOS	Basic Input/Output System	Podstawowy system wejścia/wyjścia
BMC	Board Management Controller	Kontroler BMC
BSI	British Standards Institution	Brytyjska instytucja normalizacyjna
BTI	Branch Target Identification	Branch Target Identification
CA	Certificate Authority	Urząd certyfikacji
CCA	(Arm) Confidential Compute Architecture	Arm Confidential Compute Architecture
CHERI	Capability Hardware Enhanced RISC Instructions	Capability Hardware Enhanced RISC Instructions
CMC	(Cisco) Chassis Management Controller	Chassis Management Controller (Cisco)
CNCF	Cloud Native Computing Foundation	Cloud Native Computing Foundation
COP	Call Oriented Programming	Atak typu *COP*
CoT	Chain of Trust	Łańcuch zaufania
CPACF	CP Assist for Cryptographic Functions	CP Assist for Cryptographic Functions
CPU	Central Processing Unit	Procesor
CRD	Custom Resource Definition	Niestandardowa definicja zasobu
CRI	Container Runtime Interface	Interfejs środowiska

		uruchomieniowego kontenera
CRTM	Core Root of Trust for Measurement	Główne źródło zaufania do pomiarów
CRTV	Core Root of Trust for Verification	Główne źródło zaufania do weryfikacji
CSK	Code Signing Key	Klucz podpisywania kodu
CSP	Cloud Service Provider	Dostawca usług w chmurze
DCRTM	Dynamic Core Root of Trust for Measurement	Dynamiczne główne źródło zaufanych pomiarów
DICE	Device Identifier Composition Engine	Mechanizm komponowania identyfikatora urządzenia
DIMM	Dual In-Line Memory Module	Moduł pamięci DIMM
DRAM	Dynamic Random-Access Memory	Pamięć dynamiczna o dostępie swobodnym
DRBG	Deterministic Random Bit Generator	Generator liczb pseudolosowych
DSbD	Digital Security by Design	Digital Security by Design
EAL	Evaluation Assurance Level	Poziom weryfikacji formalnej
EAT	Entity Attestation Token	Token atestacji (zaświadczania) obiektu
ECDSA	Elliptic Curve Digital Signature Algorithm	Algorytm podpisu cyfrowego krzywej eliptycznej

EL	Exception Level	Poziom wyjątku
EPT	Extended Page Table	Extended Page Table
ETSI	European Telecommunications Standards Institute	Europejski Instytut Norm Telekomunikacyjnych
ETSI NFV	European Telecommunications Standards Institute Network Functions	European Telecommunications Standards Institute Network Functions
SEC	Virtualization Security	Virtualization Security
FIDO	Fast Identity Online (Alliance)	Stowarzyszenie FIDO
FIPS	Federal Information Processing Standard	Federalny standard przetwarzania informacji
FPGA	Field Programmable Gate Array	Bezpośrednio programowalna macierz bramek
FRU	Field Replacement Unit	Część wymieniana u klienta)
FVP	Fixed Virtual Platform	Ustalona plaforma wirtualna
HES	Hardware Enforced Security	Zabezpieczenia wymuszane sprzętowo
HLAT	Hypervisor Managed Linear Address Translation	Technologia HLAT
HMEE	Hardware Mediated Execution Enclave	Hardware Mediated Execution Enclave
HSM	Hardware Security Module	Sprzętowy moduł zabezpieczeń

I/O	Input/Output	Wejście/wyjście
IA	Intel Itanium Architecture	Architektura Intel Itanium
IBB	Initial Boot Block	Początkowy blok rozruchowy
IMA	Integrity Measurement Architecture	Architektura pomiarów integralności
Intel AES-NI	Intel Advanced Encryption Standard New Instructions	Intel Advanced Encryption Standard New Instructions
Intel CET	Intel Control-Flow Enforcement Technology	Intel Control-Flow Enforcement Technology
Intel MKTME	Intel Multi-Key Total Memory Encryption	Intel Multi-Key Total Memory Encryption
Intel *TDX*	Intel Trust Domain Extensions	Intel Trust Domain Extensions
Intel TME	Intel Total Memory Encryption	Intel Total Memory Encryption
Intel TSC	Intel Transparent Supply Chain	Intel Transparent Supply Chain
Intel VT-x	Intel Virtualization Technology	Intel Virtualization Technology
IoT	Internet of Things	Internet rzeczy
IPL	Initial Program Load	Ładowanie początkowego programu
IPsec	Internet Protocol Security	Internet Protocol Security
IR	NIST Interagency or Internal Report	Sprawozdanie międzyresortowe lub wewnętrzne NIST

ISA	Instruction Set Architecture	Architektura zestawu instrukcji
ISecL-DC	Intel Security Libraries for the Data Center	Biblioteki zabezpieczeń Intel dla centrów danych
IT	Information Technology	Technologia informacyjna
ITL	Information Technology Laboratory	Information Technology Laboratory
ITS	Internal Trusted Storage (API)	------
JIT	Just-in-Time	Dokładnie na czas
JOP	Jump Oriented Programming	Atak typu *JOP*
KEK	Key Exchange Key	Klucz wymiany kluczy
KMIP	Key Management Interoperability Protocol	Protokół interoperacyjności zarządzania kluczami
KMS	Key Management Service	Usługa zarządzania kluczami
KPT	Key Protection Technology	Technologia ochrony kluczy
KVM	Kernel-Based Virtual Machine	Maszyna wirtualna oparta na jądrze
LCP	Launch Control Policy	Zasady sterowania uruchamianiem
LPAR	Logical Partition	Partycja logiczna
MAC	Message Authentication Code	Kod uwierzytelniania wiadomości

ME	Manageability Engine	Silnik zarządzania
MMU	Memory Management Unit	Jednostka zarządzania pamięcią
MOK	Machine Owner Key	Klucz właściciela maszyny
MTE	Memory Tagging Extension	Rozszerzenie znaczników pamięci
NCCoE	National Cybersecurity Center of Excellence	National Cybersecurity Center of Excellence
NFC	Near Field Communication	Komunikacja bliskiego zasięgu
NFV	Network Functions Virtualization	Wirtualizacja funkcji sieciowych
NIST	National Institute of Standards and Technology	Narodowy Instytut Standaryzacji i Technologii
NVDIMM	Non-Volatile Dual In-Line Memory Module	Moduł pamięci NVDIMM
NVRAM	Non-Volatile Random-Access Memory	Pamięć nieulotna o dostępie swobodnym
NW	Normal World, Non-Secure World	Świat normalny, świat niezabezpieczony
ODM	Original Device Manufacturer	Producent oryginalnego urządzenia
OEM	Original Equipment Manufacturer	Producent oryginalnego sprzętu

OS	Operating System	System operacyjny
PAC	Pointer Authentication Code	Kod uwierzytelniania wskaźnika
PAN	Privileged Access Never	Privileged Access Never
Parsec	Platform AbstRaction for SECurity	Platform AbstRaction for SECurity
PAS	Physical Address Space	Fizyczna przestrzeń adresowa
PCB	Printed Circuit Board	Płytka drukowana
PCH	Platform Controller Hub	Mikroukład PCH
PCIe	Peripheral Component Interconnect Express	Złącze PCIe
PCR	Platform Configuration Register	Rejestr konfiguracji platformy
PE	Processor Element	Element procesora
PEF	(IBM POWER) Protected Execution Facility	IBM POWER
PFR	Platform Firmware Resilience	Platform Firmware Resilience
PIT	Protection in Transit	Ochrona w tranzycie
PK	Platform Key	Klucz platformy
PKCS	Public Key Cryptography Standards	Standardy kryptografii klucza publicznego
PR/SM	(IBM Z) Processor Resource/System Manager	Processor Resource/System Manager (IBM Z)

PRoT	Platform Root of Trust	Platforma źródła zaufania
PS	Protected Storage (API)	Protected Storage (Interfejs API)
PSA	Platform Security Architecture	Platform Security Architecture
PXN	Privileged Execute Never	Privileged Execute Never
QAT	QuickAssist Technology	QuickAssist Technology
RAM	Random Access Memory	Pamięć o dostępie swobodnym
REE	Rich Execution Environment	Rich Execution Environment
RF	Radio Frequency	Częstotliwość radiowa
RK	Root Key	Klucz główny
RME	Realm Management Extension	Realm Management Extension
RNG	Random Number Generator	Generator liczb losowych
ROM	Read-Only Memory	Pamięć tylko do odczytu
ROP	Return Oriented Programming	Atak typu *ROP*
RoT	Root of Trust	Źródło zaufania
RTU	Root of Trust for Update	Źródło zaufania do aktualizacji
RW	Read/Write	Odczyt/zapis
RWX	Read/Write/Execute	Odczyt/zapis/wykonanie
SB	UEFI Secure Boot	Bezpieczny rozruch UEFI

SCRTM	Static Core Root of Trust for Measurement	Część statyczna głównego źródła zaufanych pomiarów
SDEI	Software Delegated Exception Interface	Software Delegated Exception Interface
SDP	(Morello) System Development Platform	Platforma programistyczna (Morello)
SEL	Secure Exception Level	Poziom wyjątku bezpieczeństwa
SEV	Secured Encrypted Virtualization	Bezpieczna szyfrowana wirtualizacja
SEV-ES	Secured Encrypted Virtualization with Encrypted State	Secured Encrypted Virtualization with Encrypted State
SEV-SNP	Secured Encrypted Virtualization with Secured Nested Paging	Zabezpieczona szyfrowana wirtualizacja z zabezpieczonym zagnieżdżonym stronicowaniem
SGX	Software Guard Extensions	Software Guard Extensions
SHA	Secure Hash Algorithm	Bezpieczna funkcja skrótu
SIMD	Single Instruction, Multiple Data	Single Instruction, Multiple Data (Technika SIMD)
SINIT ACM	Secure Initialization Authenticated Code Module	Secure Initialization Authenticated Code Module
SiP	Silicon Provider	Silicon Provider
SK	Secure Kernel	Bezpieczne jądro

SMAP	Supervisor Mode Access Prevention	Supervisor Mode Access Prevention
SMBus	System Management Bus	Magistrala zarządzania systemem
SMC	Secure Monitor Call	Secure Monitor Call
SME	Secure Memory Encryption	Secure Memory Encryption
SMEP	Supervisor Mode Execution Prevention	Supervisor Mode Execution Prevention
SMM	System Management Mode	System Management Mode
SoC	System-on-Chip	System na chipie
SP	Special Publication, Secure Partition	Publikacja specjalna, Bezpieczna partycja
SPDM	Security Protocol and Data Model	Protokół bezpieczeństwa i model danych
SPI	Serial Peripheral Interface	Szeregowy interfejs urządzenia peryferyjnego
SPIFFE	Secure Production Identity Framework for Everyone	Secure Production Identity Framework for Everyone
SPIRE	SPIFFE Runtime Environment	SPIFFE Runtime Environment
SPM	Secure Partition Manager	Secure Partition Manager
SPS FW	Server Platform Services Firmware	Oprogramowanie układowe (firmware) usług platformy serwera

SSC	(IBM) Secure Service Container	Secure Service Container (IBM)
SVID	SPIFFE Verifiable Identity Document	Weryfikowalny dokument tożsamości SPIFFE
SVM	Secure Virtual Machine	Bezpieczna maszyna wirtualna
SW	Secure World	Świat bezpieczny
TA	Trusted Application	Zaufana aplikacja
TCB	Trusted Compute Base, Trusted Computing Base	Zaufana baza przetwarzania
TCG	Trusted Computing Group	Trusted Computing Group
TD	Trust Domain	Zaufana domena
TEE	Trusted Execution Environment	Zaufane środowisko wykonawcze
TF-A	Trusted Firmware-A	Trusted Firmware-A
TLS	Transport Layer Security	Bezpieczeństwo warstwy transportowej
TOS	Trusted Operating System	Zaufany system operacyjny
TPM	Trusted Platform Module	Moduł TPM
TRNG	True Random Number Generator	Generator prawdziwych liczb losowych
TSME	Transparent Memory Encryption	Transparent Memory Encryption
TXT	Trusted Execution Technology	Trusted Execution Technology

UCS	(Cisco) Unified Computing System	Unified Computing System (Cisco)
UEFI	Unified Extensible Firmware Interface	Interfejs UEFI
UKRI	UK Research and Innovation	UK Research and Innovation
USB	Universal Serial Bus	Uniwersalna magistrala szeregowa
UXN	User Execute Never	User Execute Never
Veraison	VERificAtIon of atteStatiON	VERificAtIon of atteStatiON
VM	Virtual Machine	Maszyna wirtualna
VMID	Virtual Machine IDentifier	Identyfikator maszyny wirtualnej
VMM	Virtual Machine Manager, Virtual Machine Monitor	Menedżer maszyny wirtualnej, monitor maszyny wirtualnej
VMX	Virtual Machine Extensions	Rozszerzenia maszyny wirtualnej
XTS	xor-encrypt-xor (XEX) Based Tweaked-Codebook Mode with Ciphertext Stealing	xor-encrypt-xor (XEX) Based Tweaked-Codebook Mode with Ciphertext Stealing (Tryb XTS)

ZAŁĄCZNIK H – SŁOWNIK

Dodatkowo patrz: NSC 7298, *Słownik kluczowych pojęć z zakresu cyberbezpieczeństwa*

Terminologia angielska	Terminologia polska	Definicja
Asset Tag	Tag zasobu	Proste atrybuty klucz-wartość, które są powiązane z platformą (np. lokalizacja, nazwa firmy, oddział lub dział).
Attestation	Atestacja (zaświadczanie)	Proces zapewniania podpisu cyfrowego dla zbioru danych i/lub wyników obliczeń bezpiecznie przechowywanych w sprzęcie, a następnie weryfikacja podpisu i zbioru wyników obliczeń przez wnioskodawcę.
Chain of Trust (CoT)	Łańcuch zaufania	Metoda utrzymywania prawidłowych granic zaufania poprzez zastosowanie zasady zaufania przechodniego, w której każdy moduł oprogramowania w procesie uruchamiania systemu musi zmierzyć następny moduł przed przekazaniem sterowania.
Confidential Computing	Przetwarzanie poufne	Funkcje sprzętowe, które izolują i przetwarzają zaszyfrowane dane w pamięci, dzięki czemu są one mniej narażone na ujawnienie i naruszenie związane ze współbieżnymi obciążeniami lub bazowym systemem i platformą.

Terminologia angielska	Terminologia polska	Definicja
Cryptographic Accelerator	Akcelerator kryptograficzny	Wyspecjalizowany układ koprocesora oddzielony od centralnej jednostki przetwarzania, na który przenoszone są zadania kryptograficzne w celu zwiększenia wydajności.
Hardware-Enabled Security	Bezpieczeństwo sprzętowe	Bezpieczeństwo, którego podstawą jest platforma sprzętowa.
Platform Trust	Zaufanie do platformy	Gwarancja integralności podstawowej konfiguracji platformy, w tym sprzętu, oprogramowania układowego i oprogramowania.
Root of Trust (RoT)	Źródło zaufania	Punkt startowy, który jest domyślnie zaufany.
Shadow Stack	Stos w tle	Równoległy stos sprzętowy, który aplikacje mogą wykorzystywać do przechowywania kopii adresów zwrotnych, które są porównywane z normalnym stosem programu podczas operacji powrotu.
Trusted Execution Environment (TEE)	Zaufane środowisko wykonawcze	Obszar lub enklawa chroniona przez procesor systemowy.