

BitUnlocker

Leveraging Windows Recovery
to Extract BitLocker Secrets



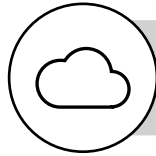
Who are we?

Security **T**esting & **O**ffensive **R**esearch at **M**icrosoft (STORM)

Alon Leviev (@alon_leviev)
Security Researcher @ Microsoft

Netanel Ben Simon (@NetanelBenSimon)
Senior Security Researcher @ Microsoft

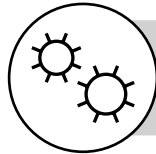
Agenda



Research Background



WinRE Overview

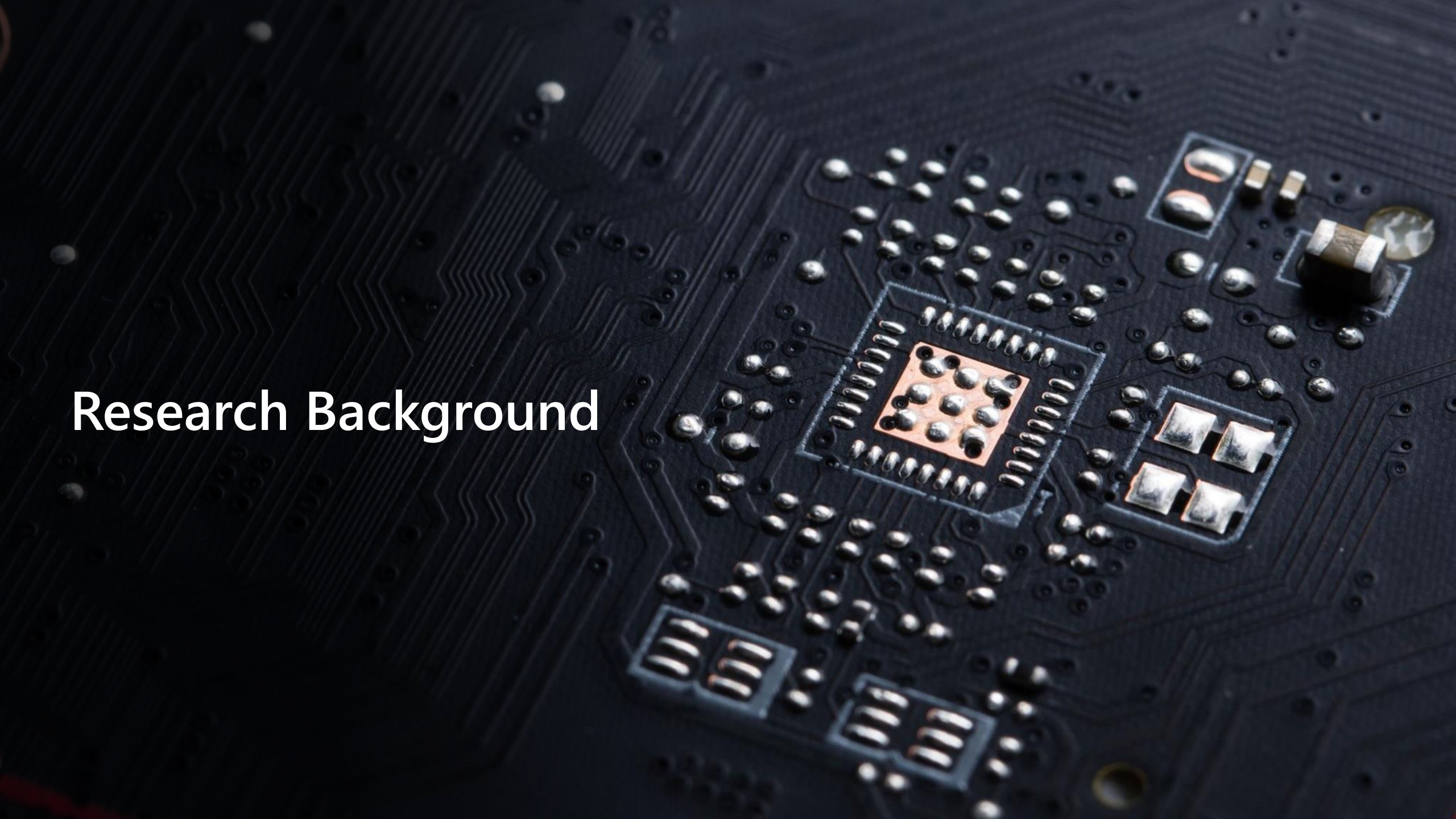


Vulnerabilities and Exploitation

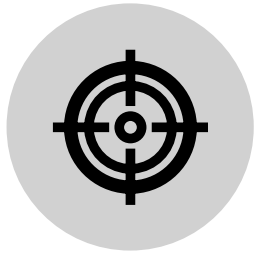


Closing Remarks

Research Background



Data at Rest Protection



Defend your sensitive data
against theft scenarios



Data at Rest Protection – Why Should You Care?



According to research, **a laptop is stolen every 53 seconds** – but how prepared are you for what comes next? *The Guardian*



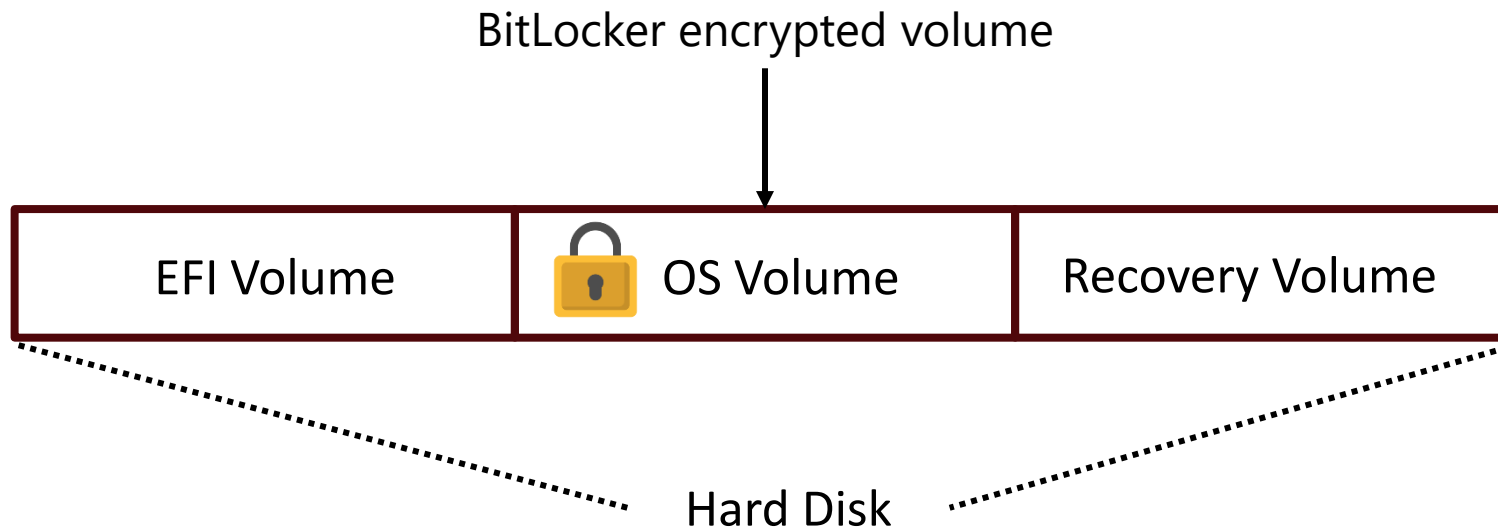
Did you know that **laptop computer have a 1-in-10 chance of being stolen**, which means there is a 10% chance for you to be the victim of laptop theft. *Prey Project*



The average value of a lost laptop is \$49,246. This value is based on seven cost components: replacement cost, detection, forensics, data breach, lost IP costs, lost productivity, and legal, consulting and regulatory expenses. *Intel*

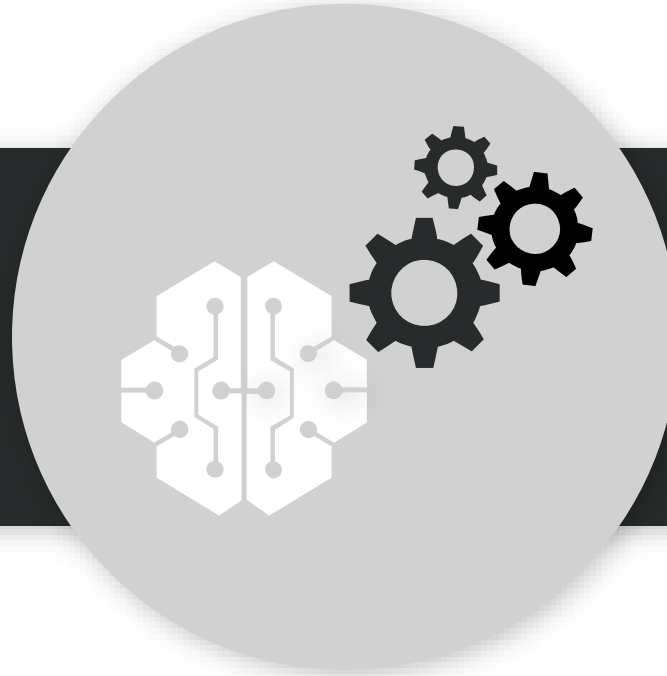
BitLocker – Windows's Data Protection Cornerstone

BitLocker is a Full Volume Encryption (FVE) technology



BitLocker Threat Model

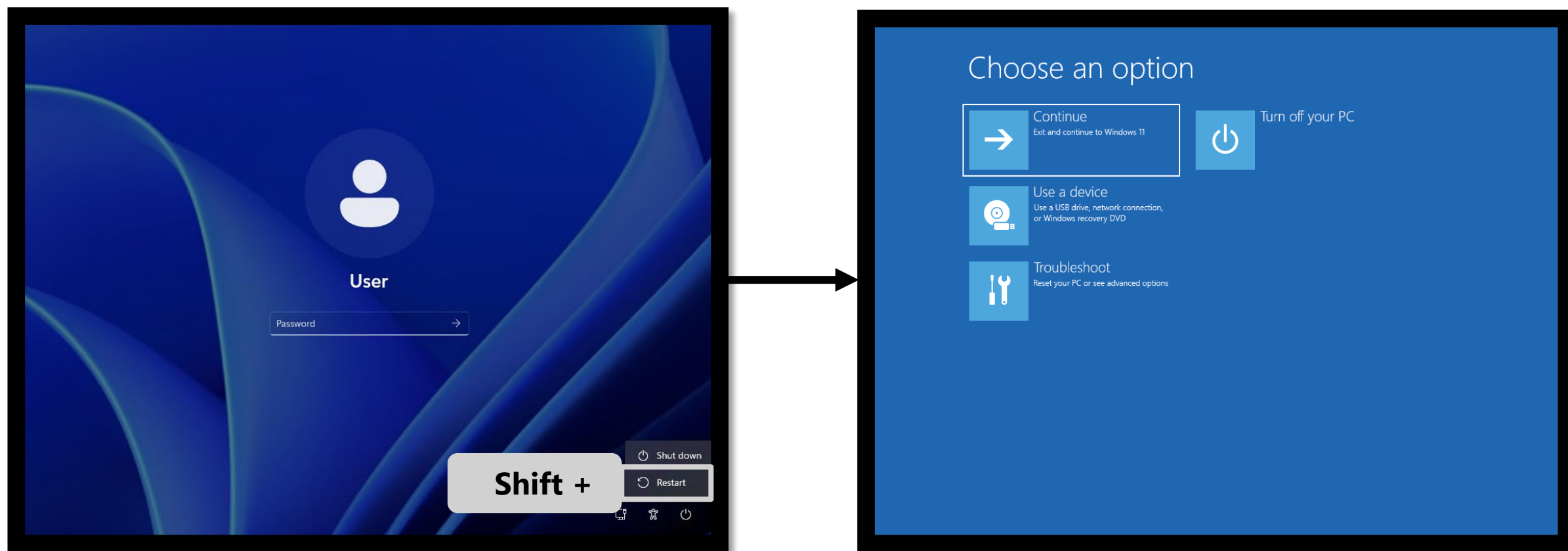
Full physical access



No login credentials

The Hidden Attack Surface - The Windows Recovery Environment (WinRE)

Physical attackers without logon credentials can directly boot into WinRE



Targeting the Windows Recovery Environment (WinRE)

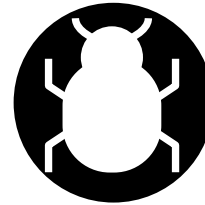
We performed a security review of WinRE focused on –



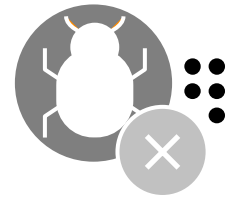
Finding new
vulnerabilities



Exploiting them

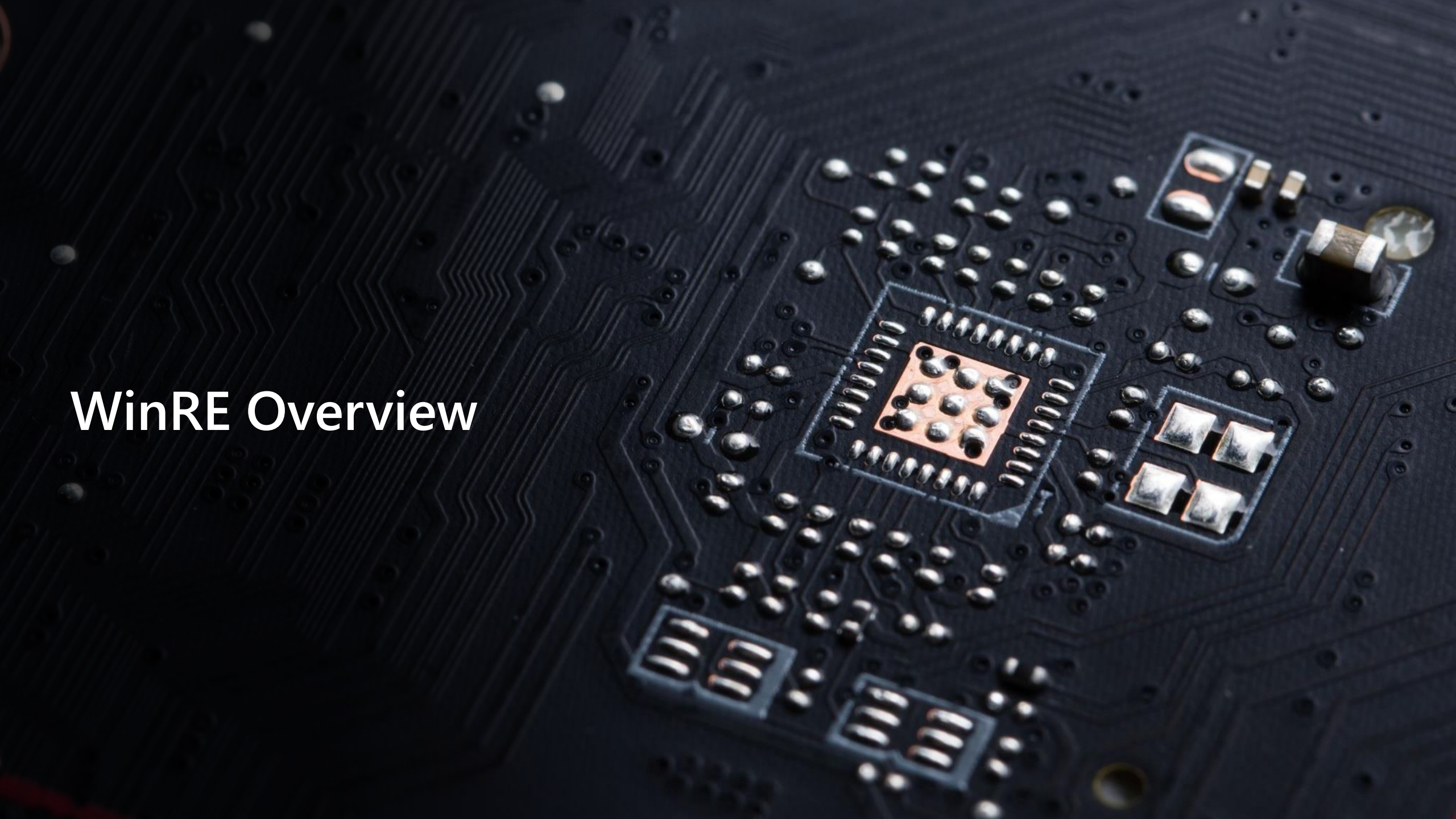


Fixing them



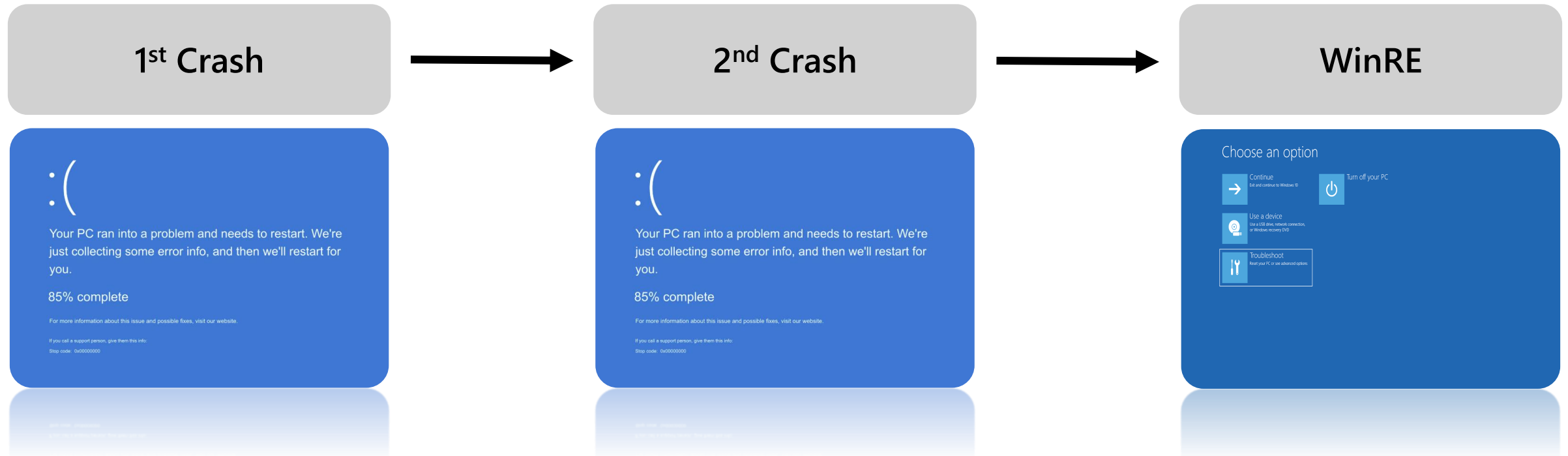
Hardening WinRE

WinRE Overview



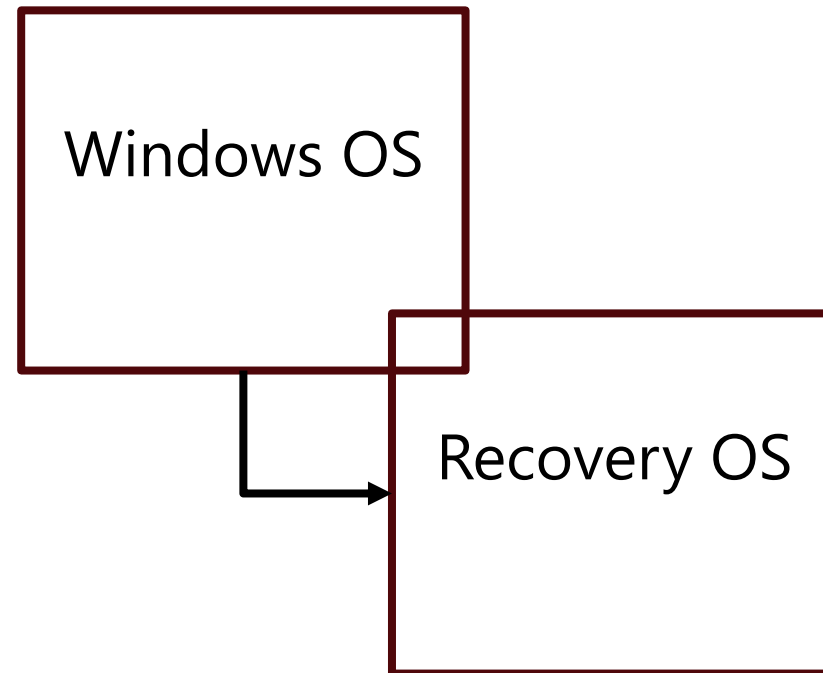
WinRE Overview

- WinRE is the recovery platform of Windows
- WinRE is designed to recover from critical system issues



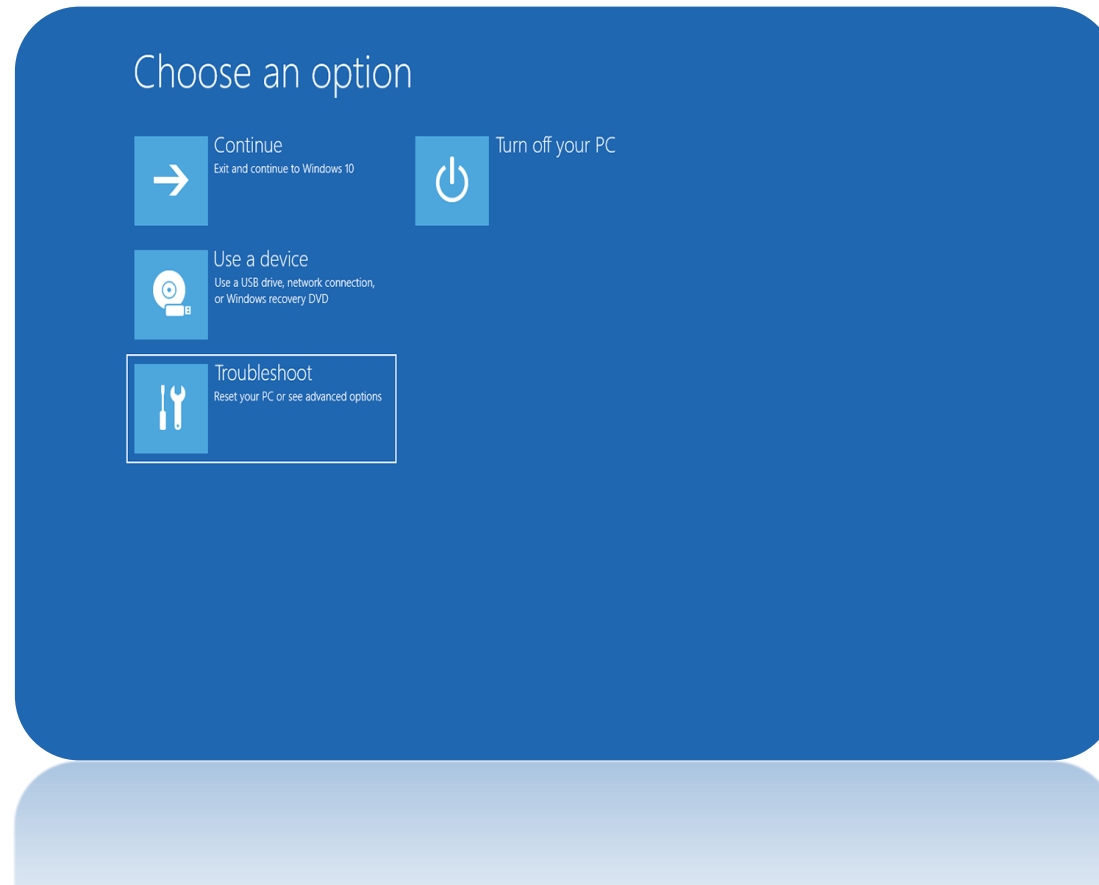
WinRE Architecture – Recovery OS

- WinRE is a lean Windows OS with recovery customizations (aka. Recovery OS)



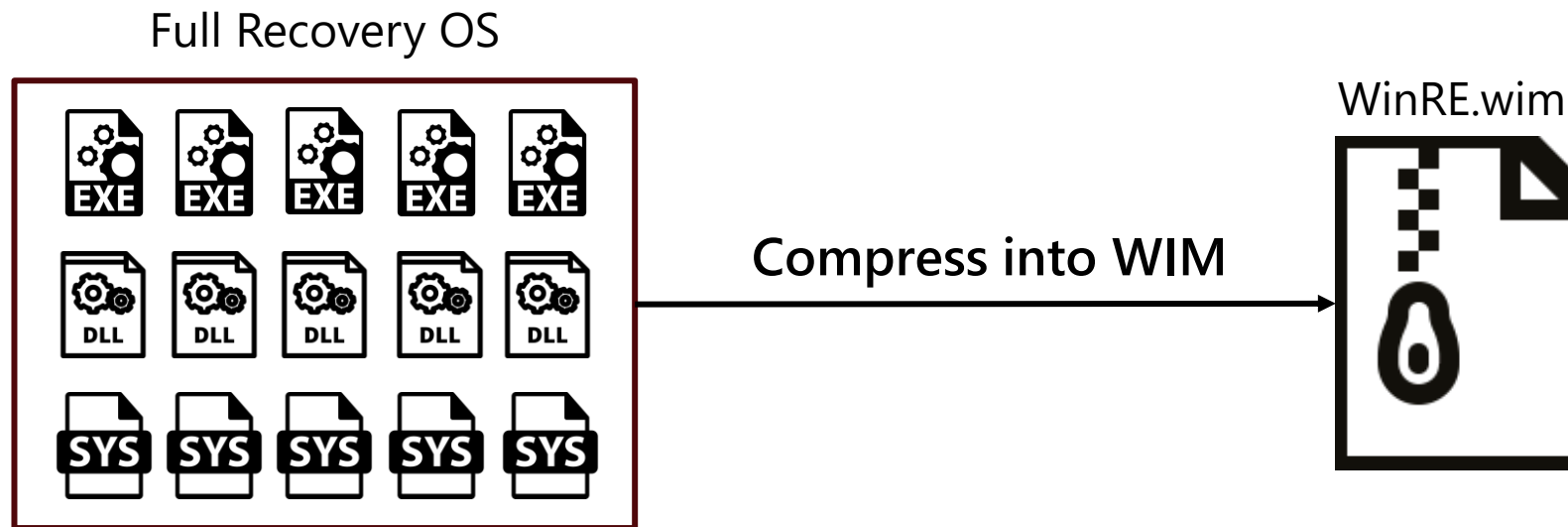
WinRE Architecture – Recovery OS Customizations

- The customizations include Startup Repair, System Reset, System Restore etc.



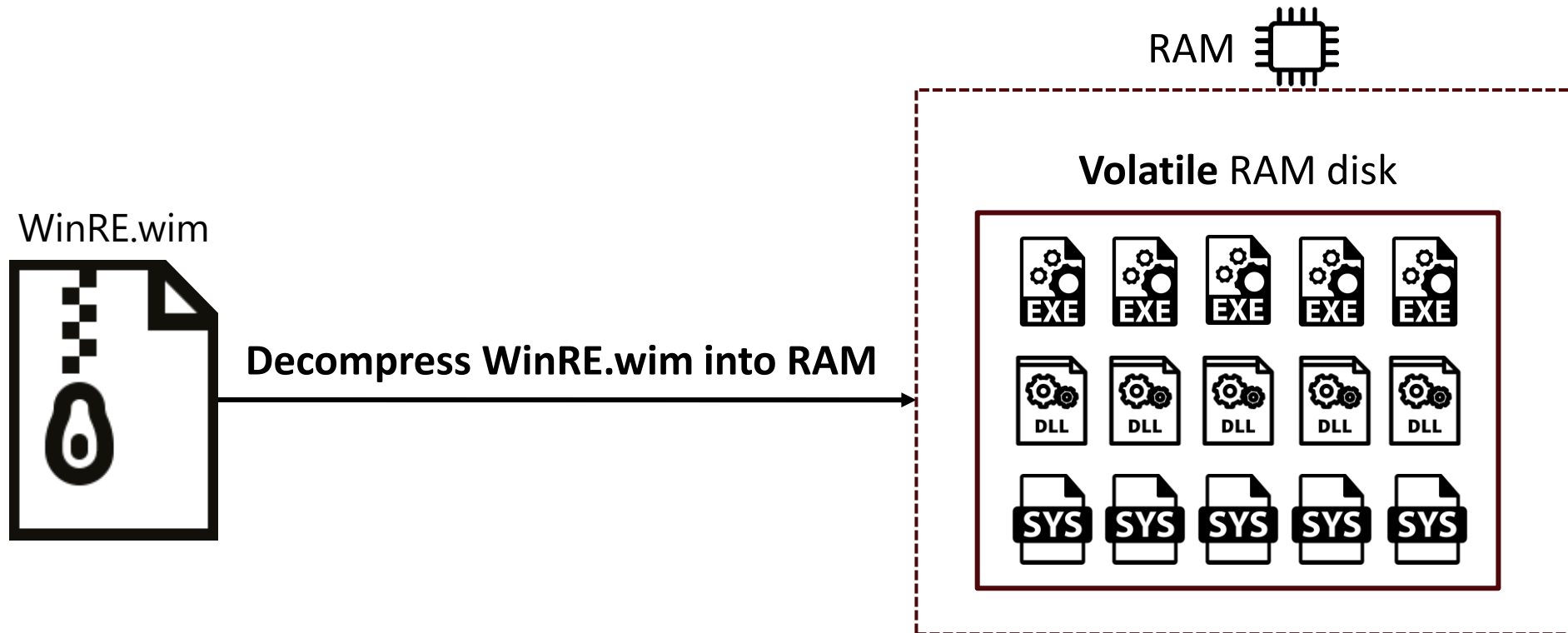
WinRE Architecture – WinRE.wim

- The recovery OS is compressed into a single WIM file – WinRE.wim

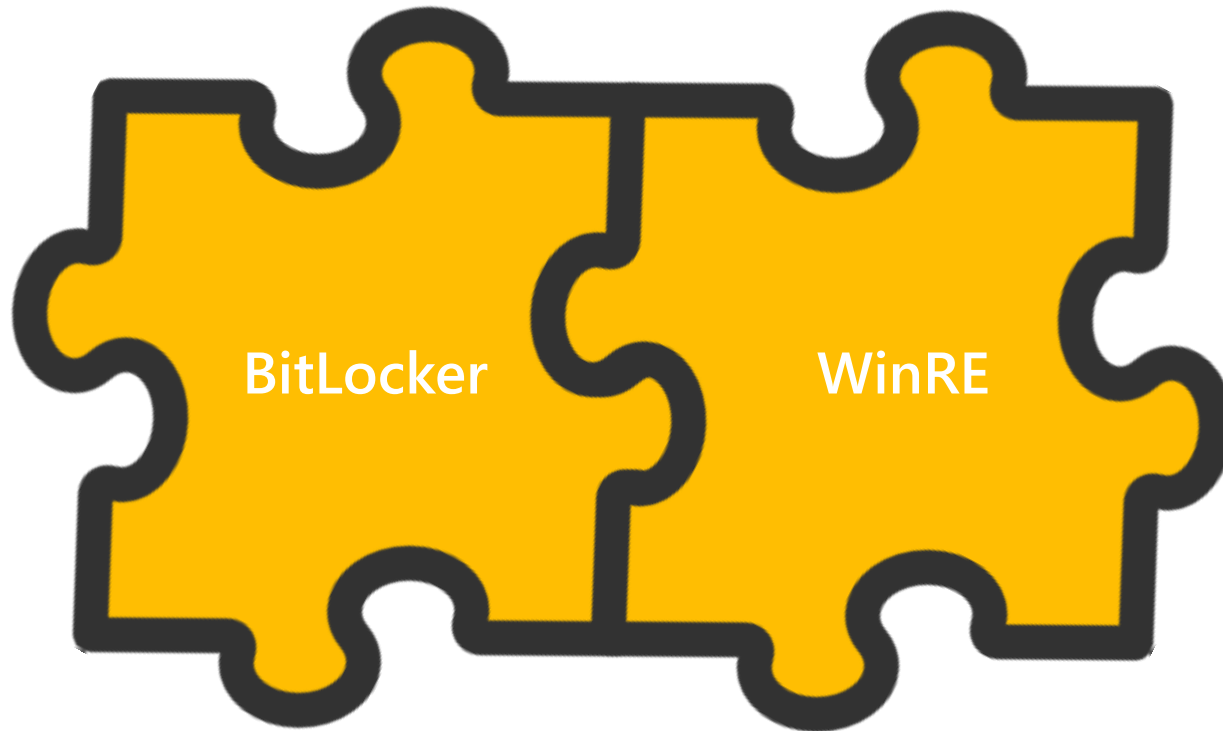


WinRE Architecture – RAM disk boot

- WinRE.wim is booted from RAM disk
- Changes to RAM disk are never committed back to WinRE.wim

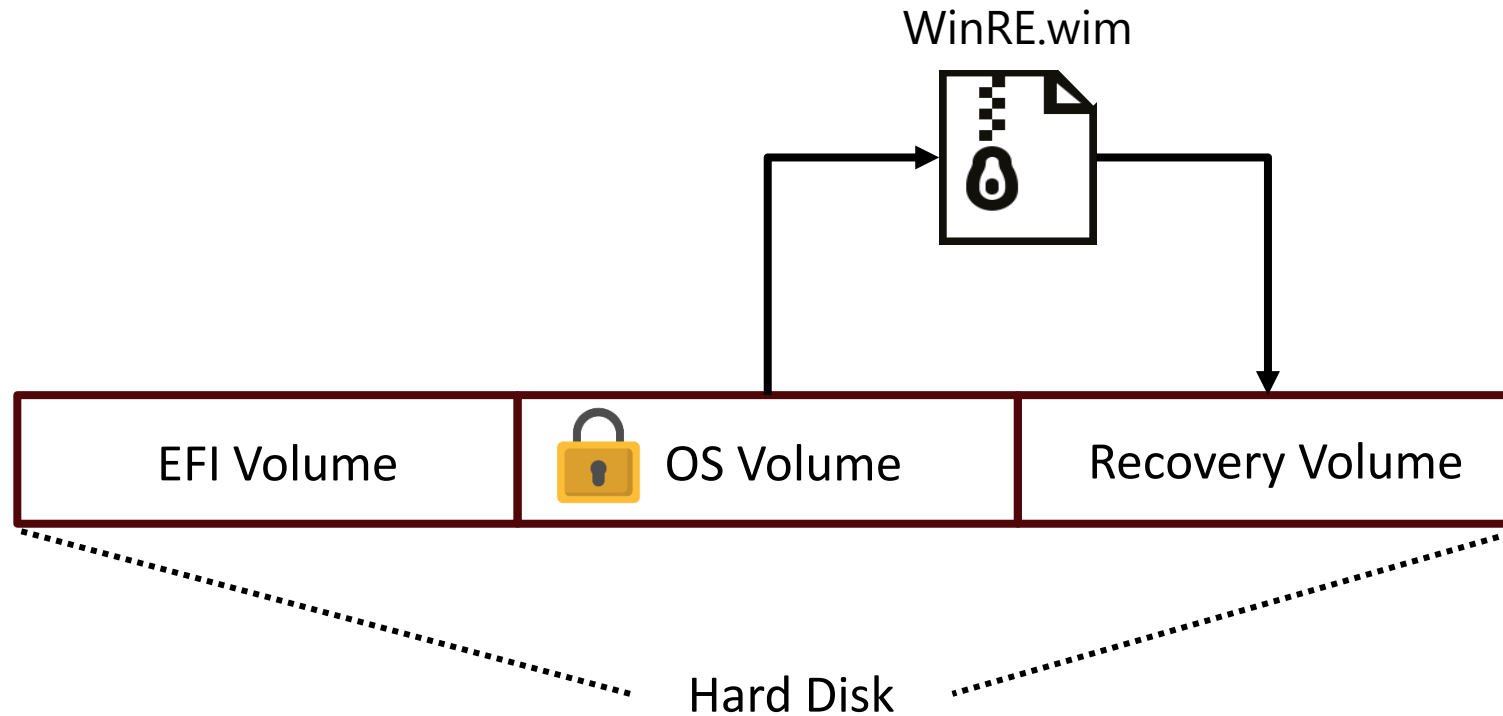


WinRE Changes As Part Of BitLocker's Introduction



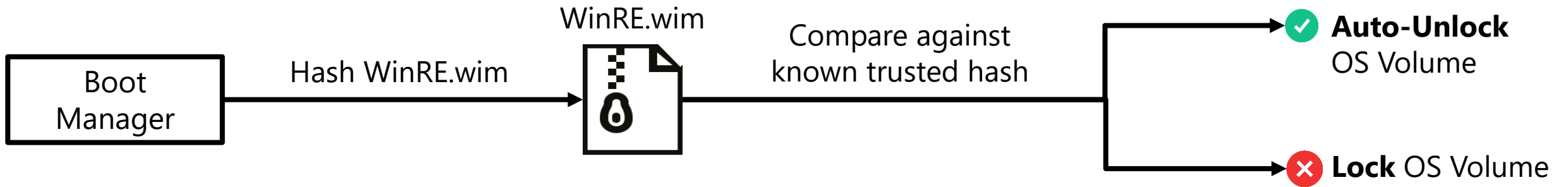
WinRE Design Change #1 – WinRE.wim Relocation

WinRE.wim was relocated from the BitLocker protected OS volume to the unprotected recovery volume



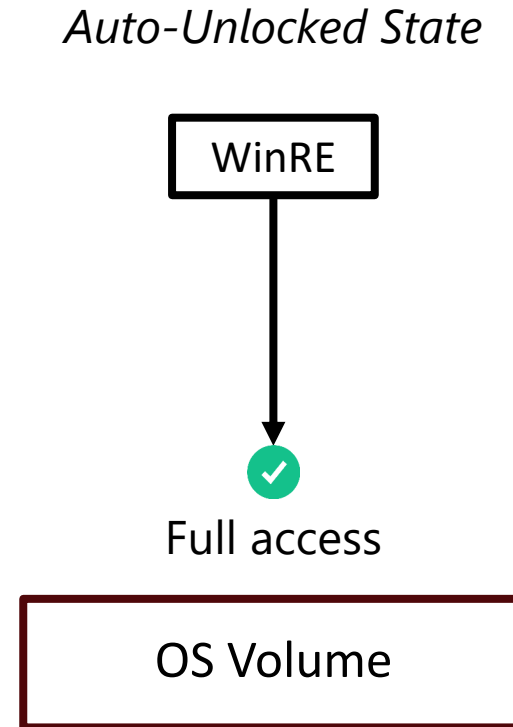
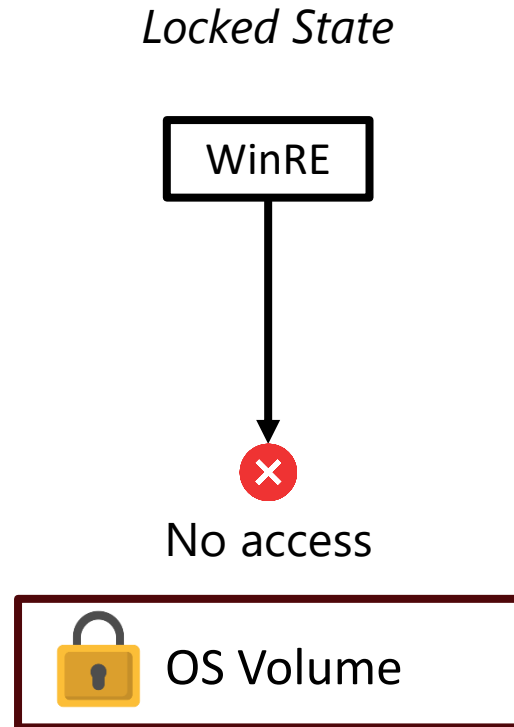
WinRE Design Change #2 – Trusted WIM Boot

Trusted WIM boot was developed to establish trust between the Recovery OS and the Main OS



WinRE Design Change #2 – Trusted WIM Boot

- In Auto-Unlock state – WinRE can fully access the OS volume
- In Locked state – WinRE cannot access the OS volume

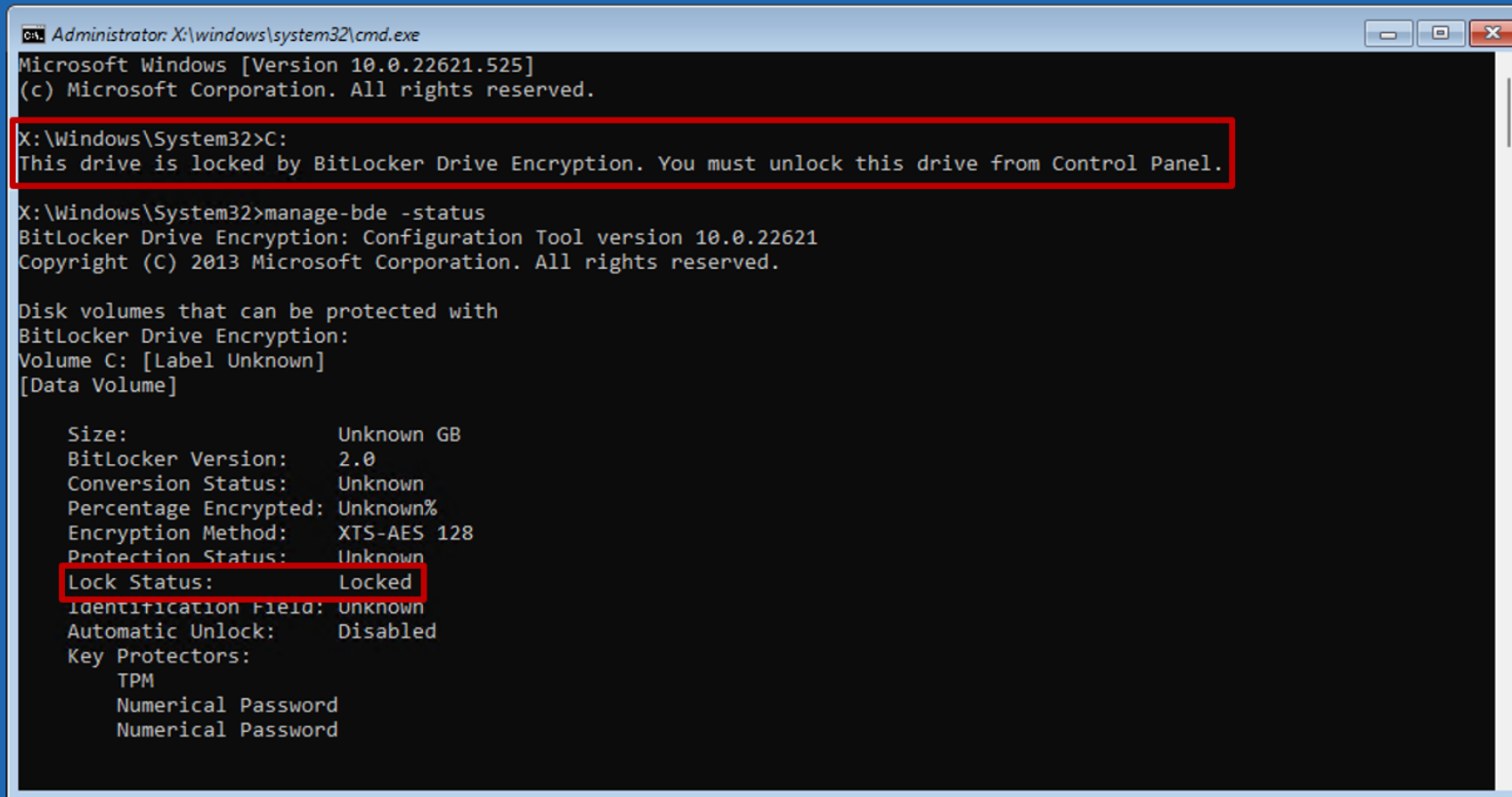


WinRE Design Change #3 – Volume Re-Lock

Main OS volume re-lock functionality was developed to safeguard BitLocker from by-design harmful recovery operations (e.g., Command Prompt)



WinRE Design Change #3 – Volume Re-Lock



A screenshot of a Windows command prompt window titled "Administrator: X:\windows\system32\cmd.exe". The window shows the output of the "manage-bde -status" command. A red rectangular box highlights the message: "X:\Windows\System32>C: This drive is locked by BitLocker Drive Encryption. You must unlock this drive from Control Panel." Another red rectangular box highlights the "Lock Status: Locked" line in the command's output.

```
Administrator: X:\windows\system32\cmd.exe
Microsoft Windows [Version 10.0.22621.525]
(c) Microsoft Corporation. All rights reserved.

X:\Windows\System32>C:
This drive is locked by BitLocker Drive Encryption. You must unlock this drive from Control Panel.

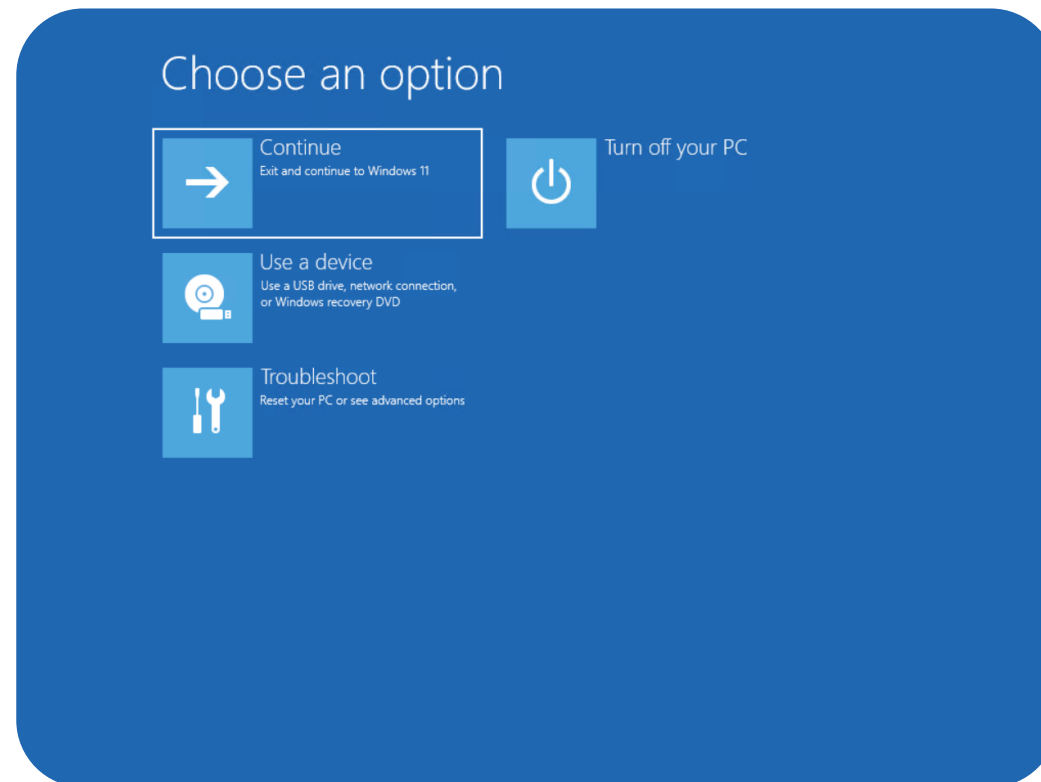
X:\Windows\System32>manage-bde -status
BitLocker Drive Encryption: Configuration Tool version 10.0.22621
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

Disk volumes that can be protected with
BitLocker Drive Encryption:
Volume C: [Label Unknown]
[Data Volume]

Size:                Unknown GB
BitLocker Version:   2.0
Conversion Status:   Unknown
Percentage Encrypted: Unknown%
Encryption Method:   XTS-AES 128
Protection Status:   Unknown
Lock Status:         Locked
Identification Field: Unknown
Automatic Unlock:    Disabled
Key Protectors:
    TPM
    Numerical Password
    Numerical Password
```

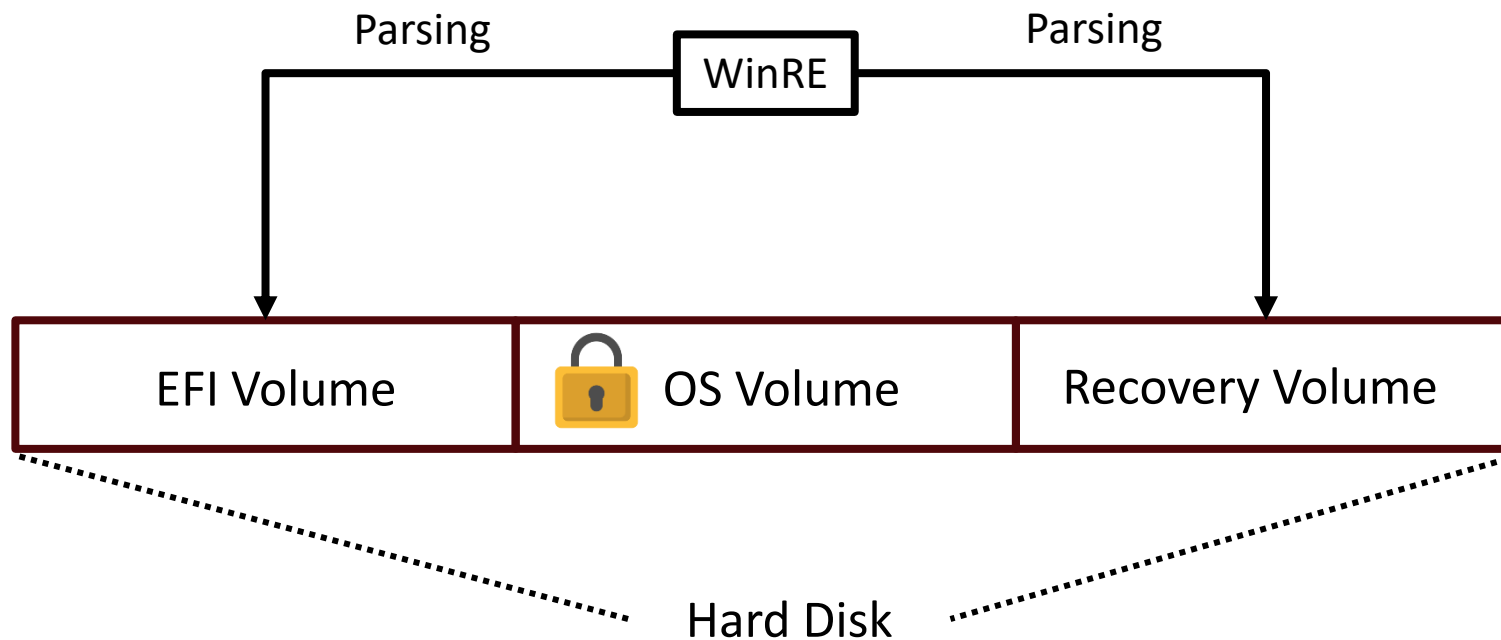
WinRE Design Changes Summary

As long as WinRE.wim hash is trusted, and no harmful recovery operations are triggered – the main OS volume is unlocked!

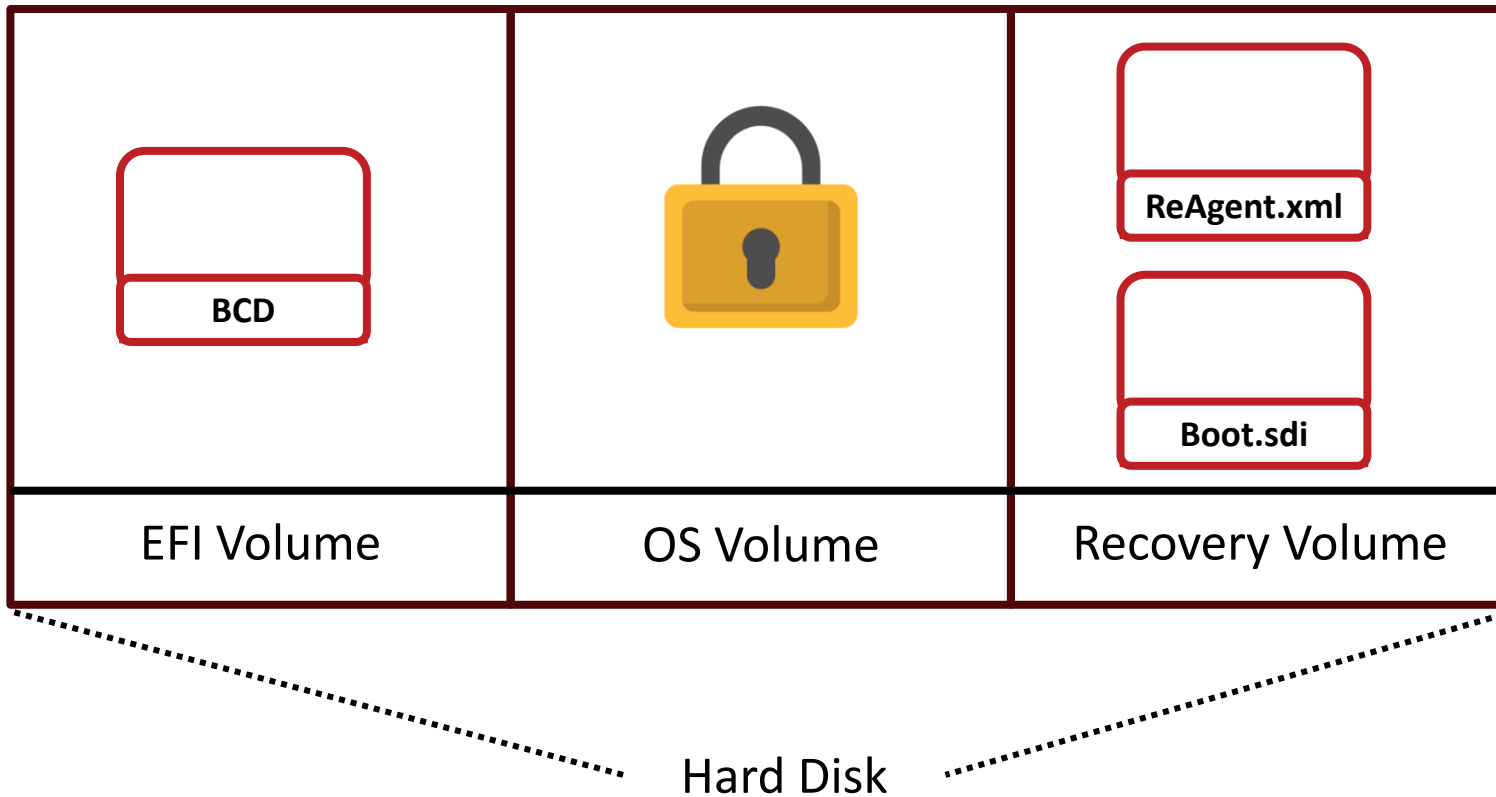


Any Attack Surfaces Exposed?

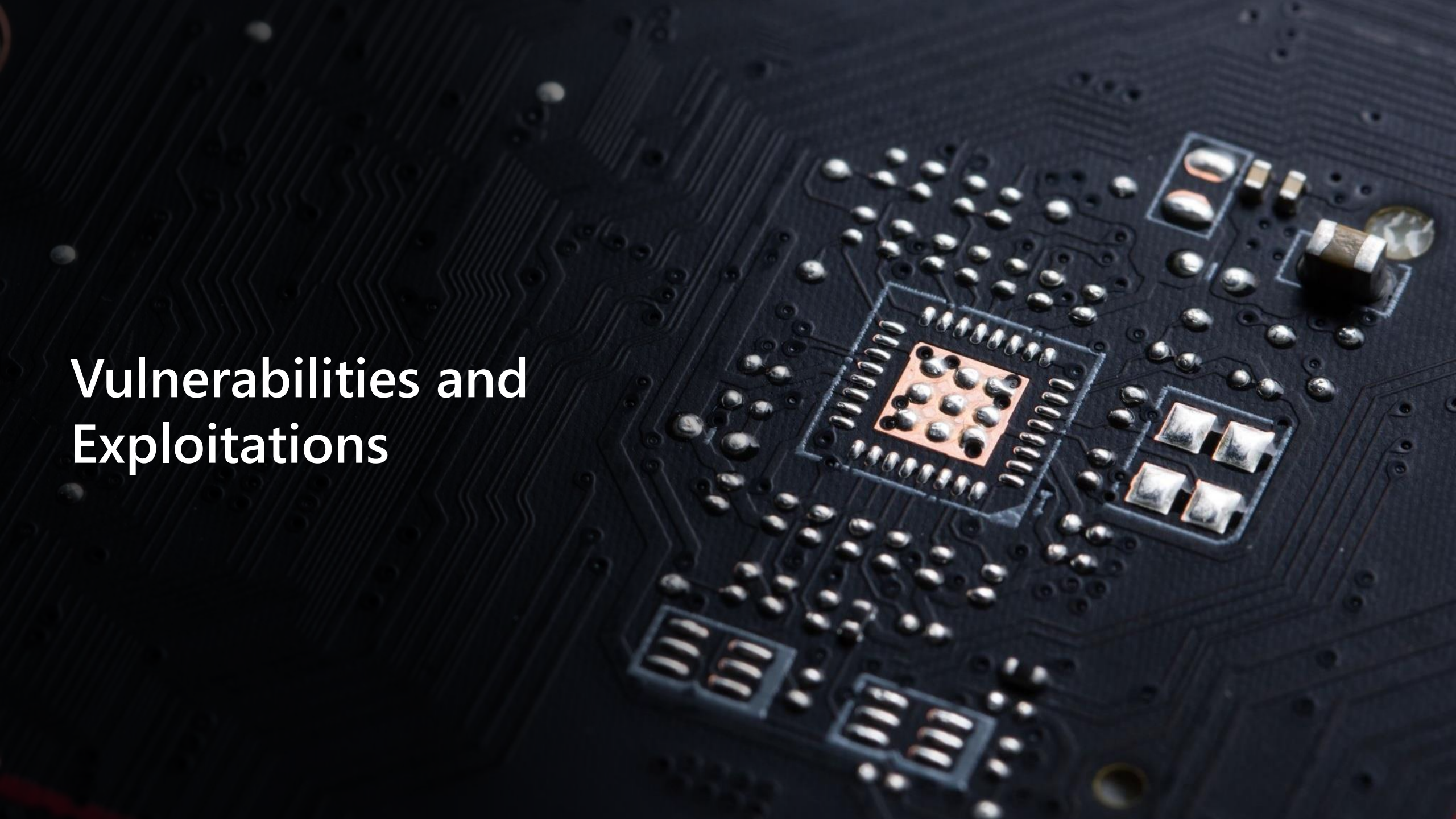
Attacking parsers of external files residing in non-protected volumes



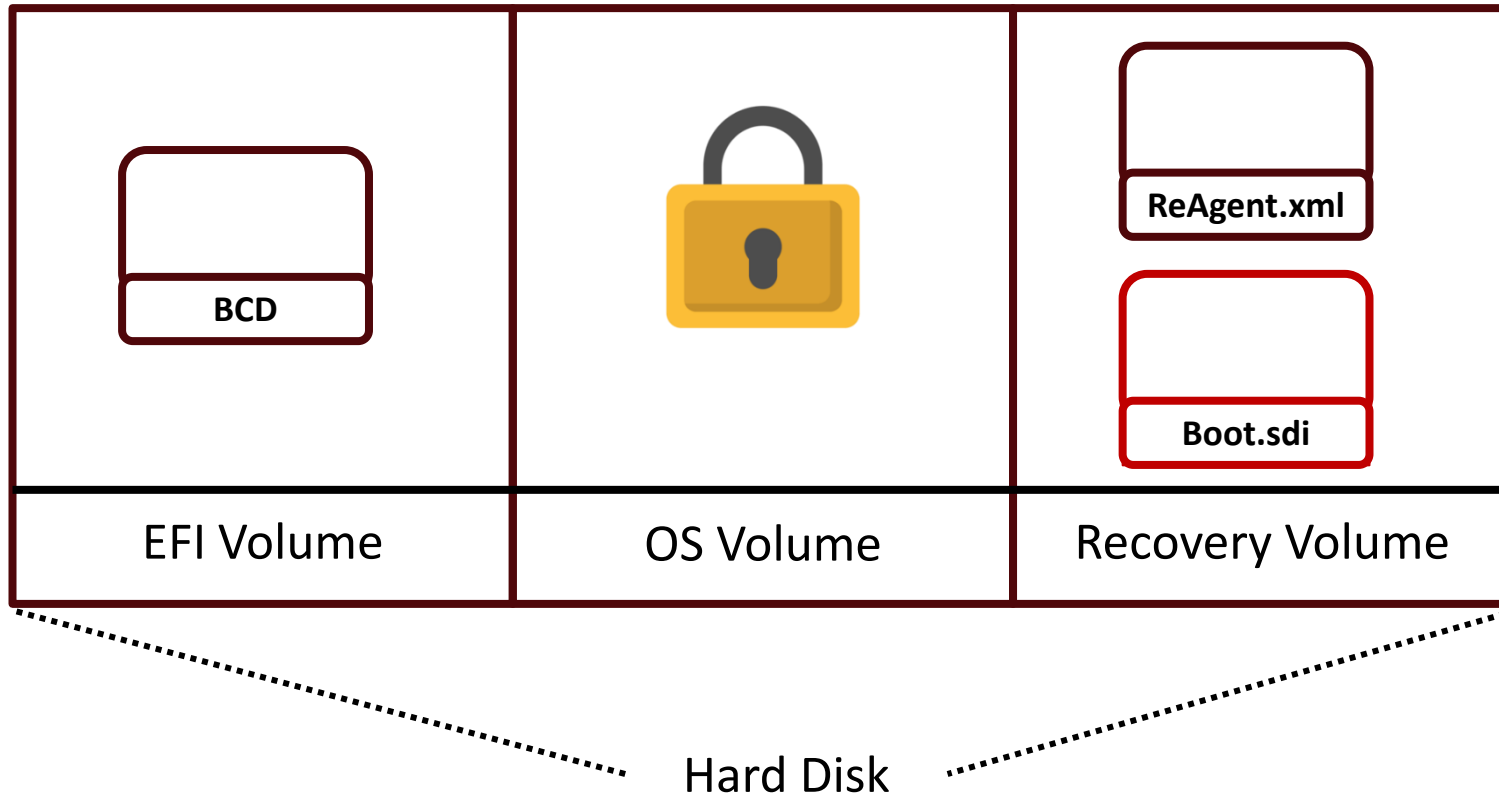
Our Focus Today



Vulnerabilities and Exploitations

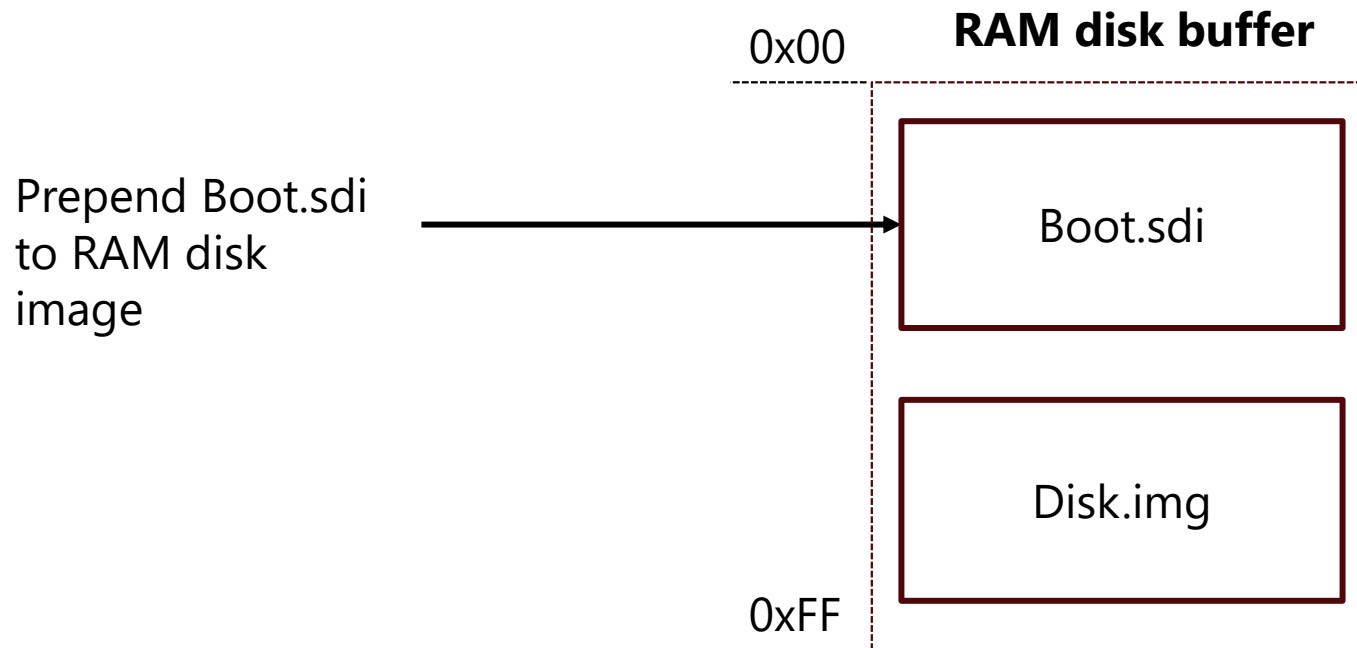


Attacking Boot.sdi Parsing



Boot.sdi purpose

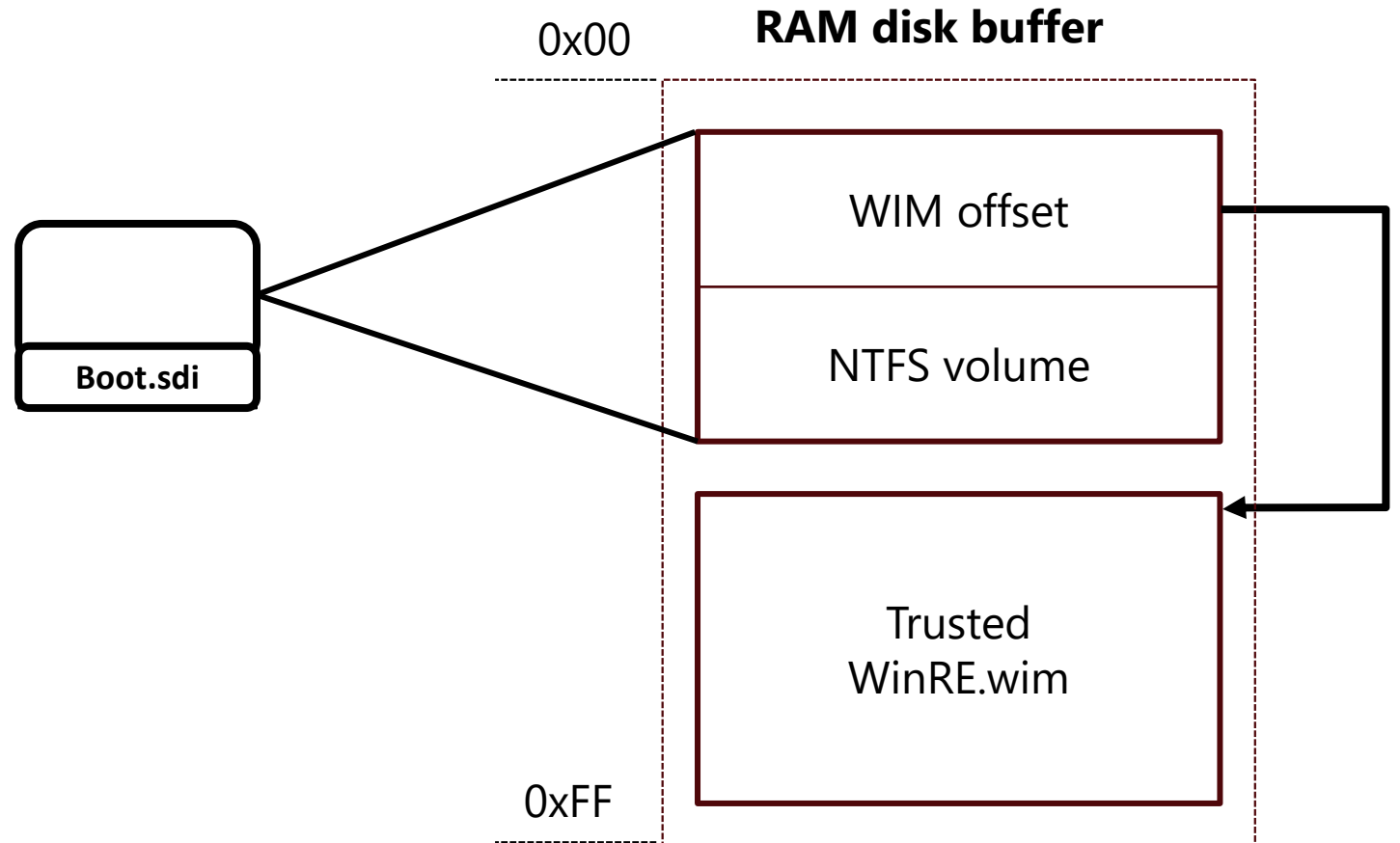
- Boot.sdi is an optional component in the RAM disk boot procedure
- It contains metadata used for the RAM disk creation
- If specified, it is **prepended to the RAM disk image**



Boot.sdi Usage In WIM boot

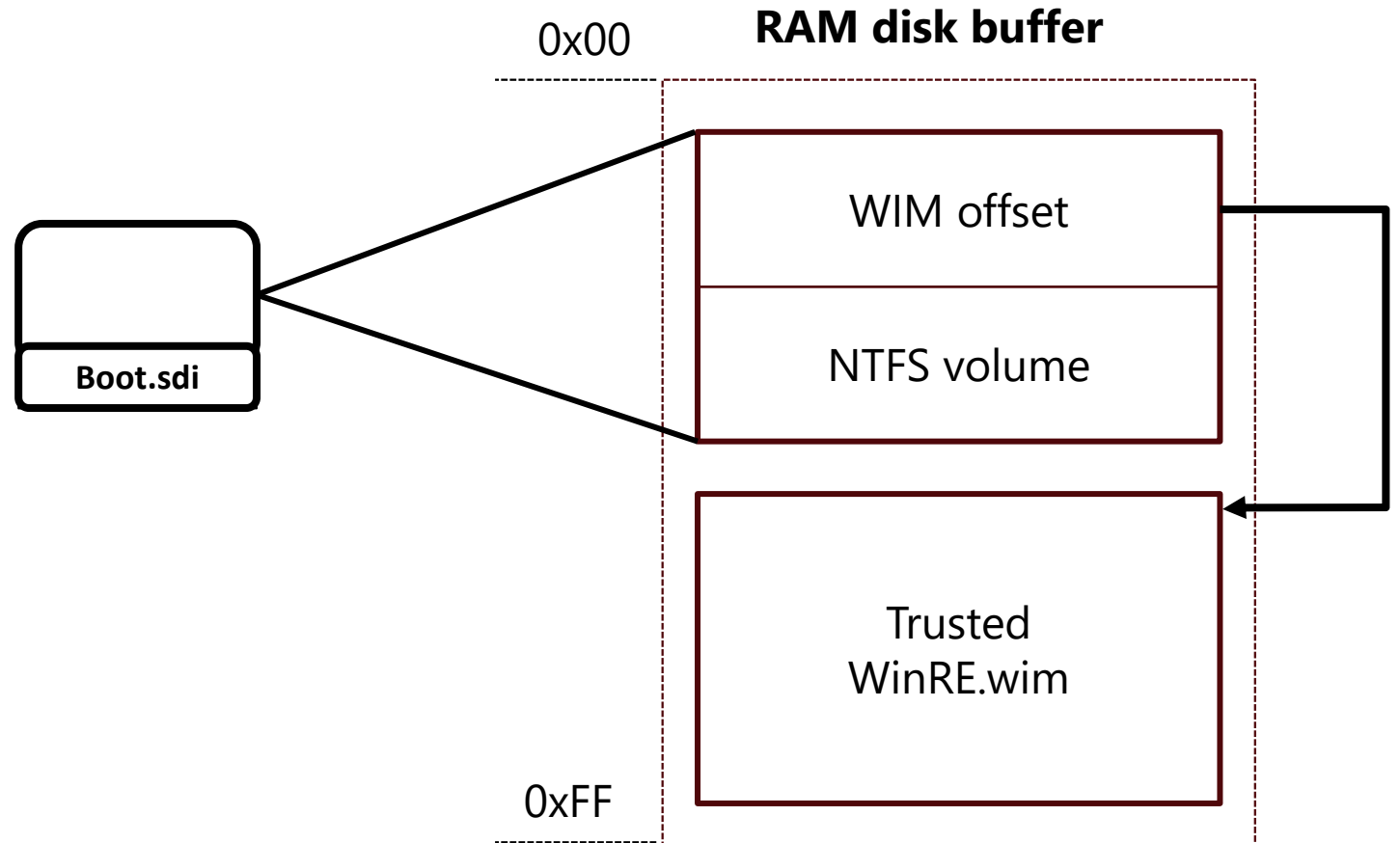
In the context of WIM boot, Boot.sdi contains –

- Relative offset to WinRE.wim
- Empty NTFS volume



Why is this Setup Required?

For compatibility



RAM Disk Load – Pseudo Code Analysis

1) Allocate the SDI +
WIM buffer

```
// Allocate memory for both SDI and WIM
RamDiskImageBuffer = AllocateMemory(SdiSize + WimSize);

// Load SDI to buffer and DO NOT calculate SDI hash
LoadFileToBuffer(SdiPath,
                RamDiskImageBuffer,
                SdiSize,
                NULL);

// Load WIM to buffer right after SDI and calculate WIM hash
LoadFileToBuffer(WimPath,
                Add2Ptr(RamDiskImageBuffer, SdiSize),
                WimSize,
                WimHash);

// The WIM booted from is the one pointed by the SDI!
WimAddress = RamDiskImageBuffer + SdiTrust->WimOffset;
```

RAM Disk Load – Pseudo Code Analysis

2) Load the SDI into the allocated buffer

```
// Allocate memory for both SDI and WIM
RamDiskImageBuffer = AllocateMemory(SdiSize + WimSize);

// Load SDI to buffer and DO NOT calculate SDI hash
LoadFileToBuffer(SdiPath,
                 RamDiskImageBuffer,
                 SdiSize,
                 NULL);

// Load WIM to buffer right after SDI and calculate WIM hash
LoadFileToBuffer(WimPath,
                 Add2Ptr(RamDiskImageBuffer, SdiSize),
                 WimSize,
                 WimHash);

// The WIM booted from is the one pointed by the SDI!
WimAddress = RamDiskImageBuffer + SdiTrust->WimOffset;
```

RAM Disk Load – Pseudo Code Analysis

3) Load the WIM into the allocated buffer, following the SDI. **The load API also calculates the WIM hash used for WIM trust validation!**

```
// Allocate memory for both SDI and WIM
RamDiskImageBuffer = AllocateMemory(SdiSize + WimSize);

// Load SDI to buffer and DO NOT calculate SDI hash
LoadFileToBuffer(SdiPath,
                 RamDiskImageBuffer,
                 SdiSize,
                 NULL);

// Load WIM to buffer right after SDI and calculate WIM hash
LoadFileToBuffer(WimPath,
                 Add2Ptr(RamDiskImageBuffer, SdiSize),
                 WimSize,
                 WimHash);

// The WIM booted from is the one pointed by the SDI!
WimAddress = RamDiskImageBuffer + SdiTrust->WimOffset;
```

RAM Disk Load – Pseudo Code Analysis

4) The WIM that is booted is the one pointed by the SDI. **There is no correlation between the used WIM and the hashed WIM!**

```
// Allocate memory for both SDI and WIM
RamDiskImageBuffer = AllocateMemory(SdiSize + WimSize);

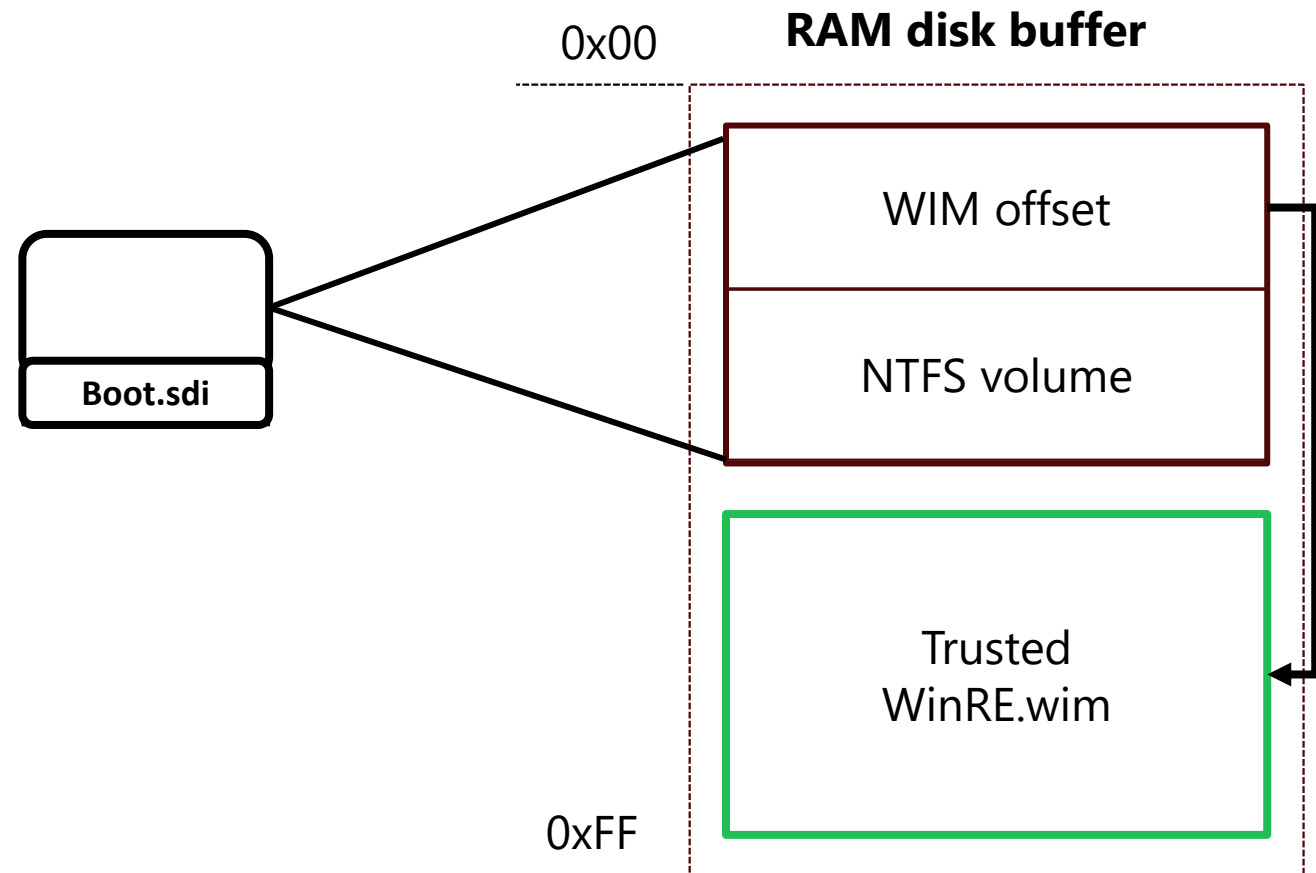
// Load SDI to buffer and DO NOT calculate SDI hash
LoadFileToBuffer(SdiPath,
                 RamDiskImageBuffer,
                 SdiSize,
                 NULL);

// Load WIM to buffer right after SDI and calculate WIM hash
LoadFileToBuffer(WimPath,
                 Add2Ptr(RamDiskImageBuffer, SdiSize),
                 WimSize,
                 WimHash);

// The WIM booted from is the one pointed by the SDI!
WimAddress = RamDiskImageBuffer + SdiTrust->WimOffset;
```

Vulnerability #1 – Bypassing WIM Validation Through WIM offset

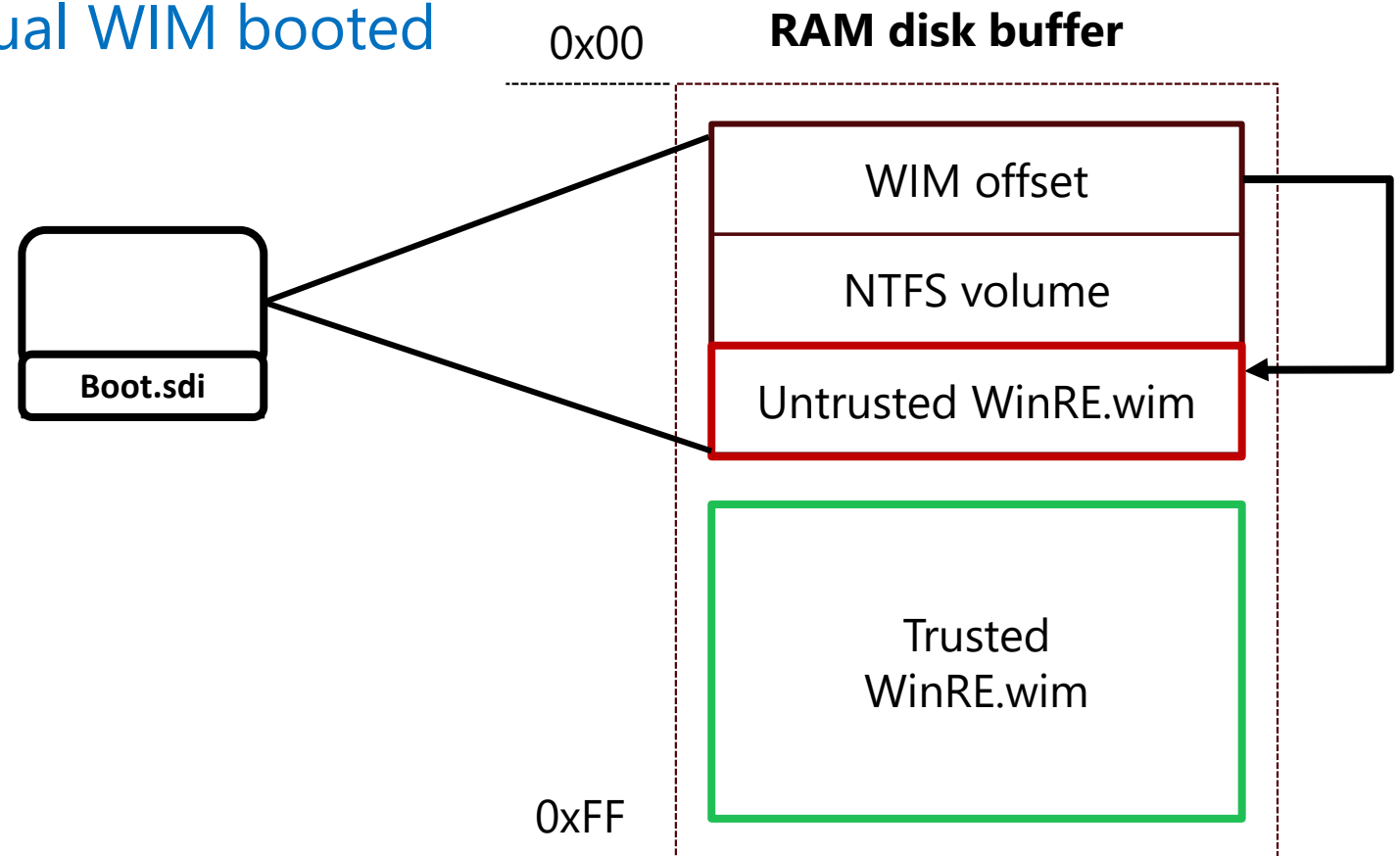
Trusted WinRE.wim is used for WIM validation



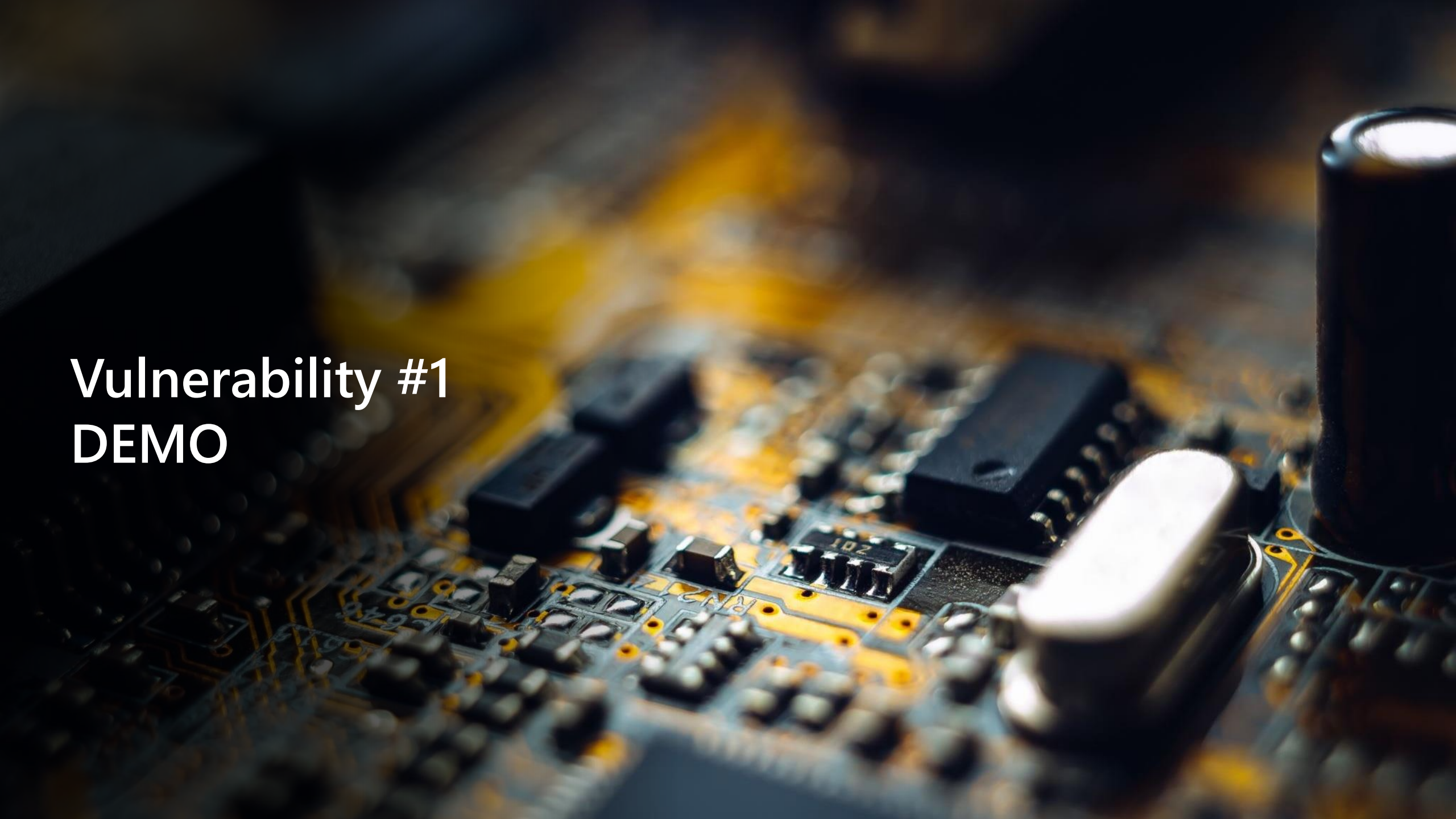
Vulnerability #1 – Bypassing WIM Validation Through WIM offset

Trusted WinRE.wim is used for WIM validation

Untrusted WinRE.wim is the actual WIM booted



Vulnerability #1 DEMO

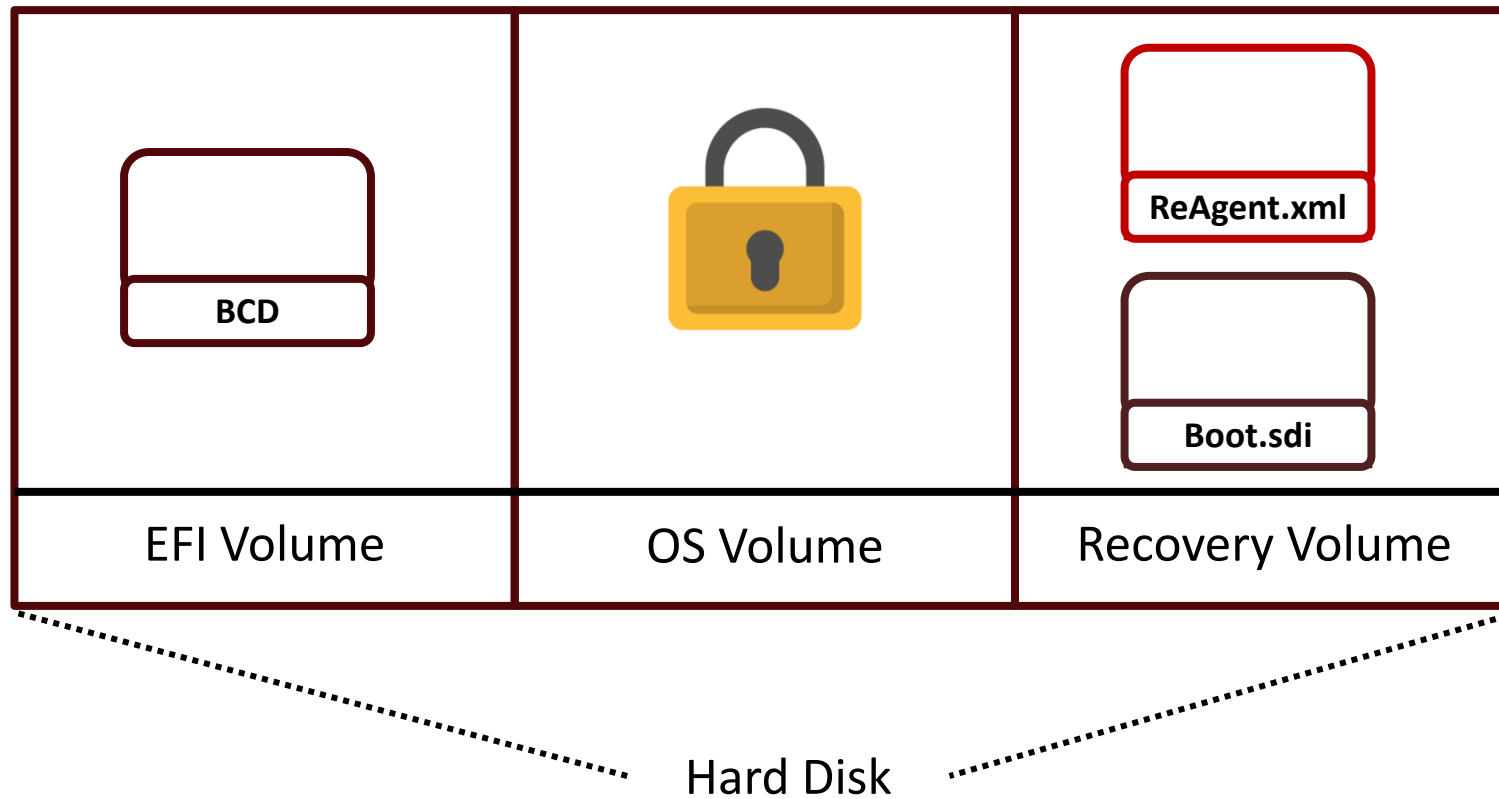




User

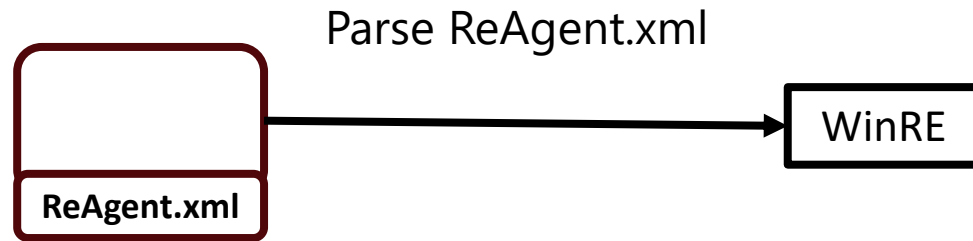
 →

Attacking ReAgent.xml Parsing



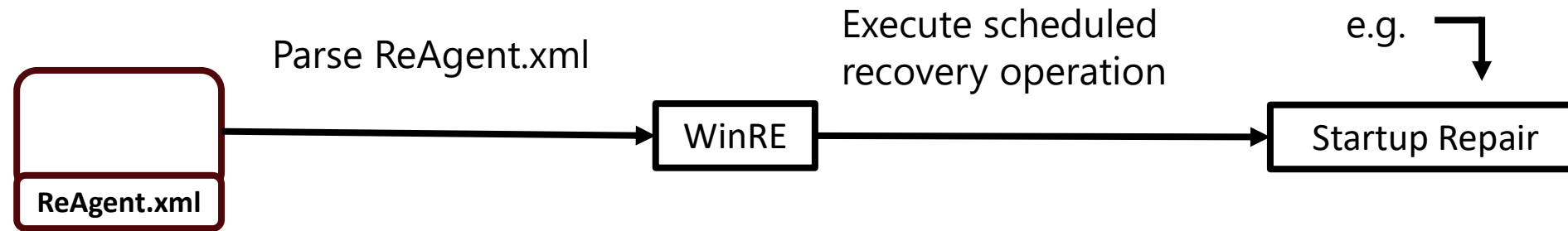
ReAgent.xml Purpose

ReAgent.xml represents the state and configuration of WinRE runtime



ReAgent.xml Scheduled Operations

ReAgent.xml controls which recovery operation will be executed in WinRE



Focused Scheduled Operations



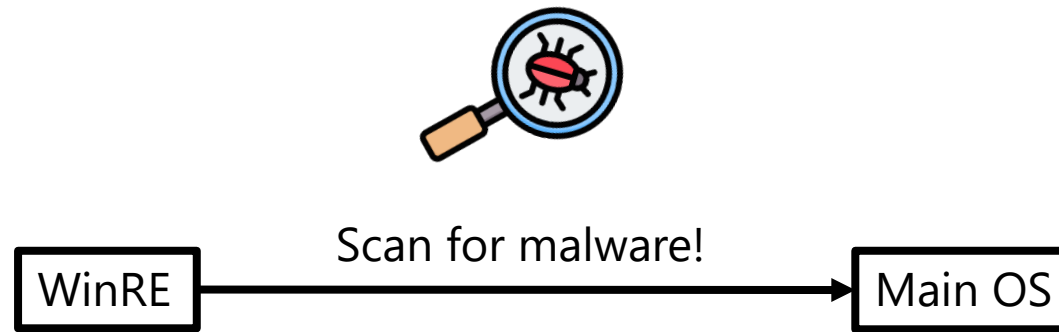
Offline Scanning



WinRE Apps

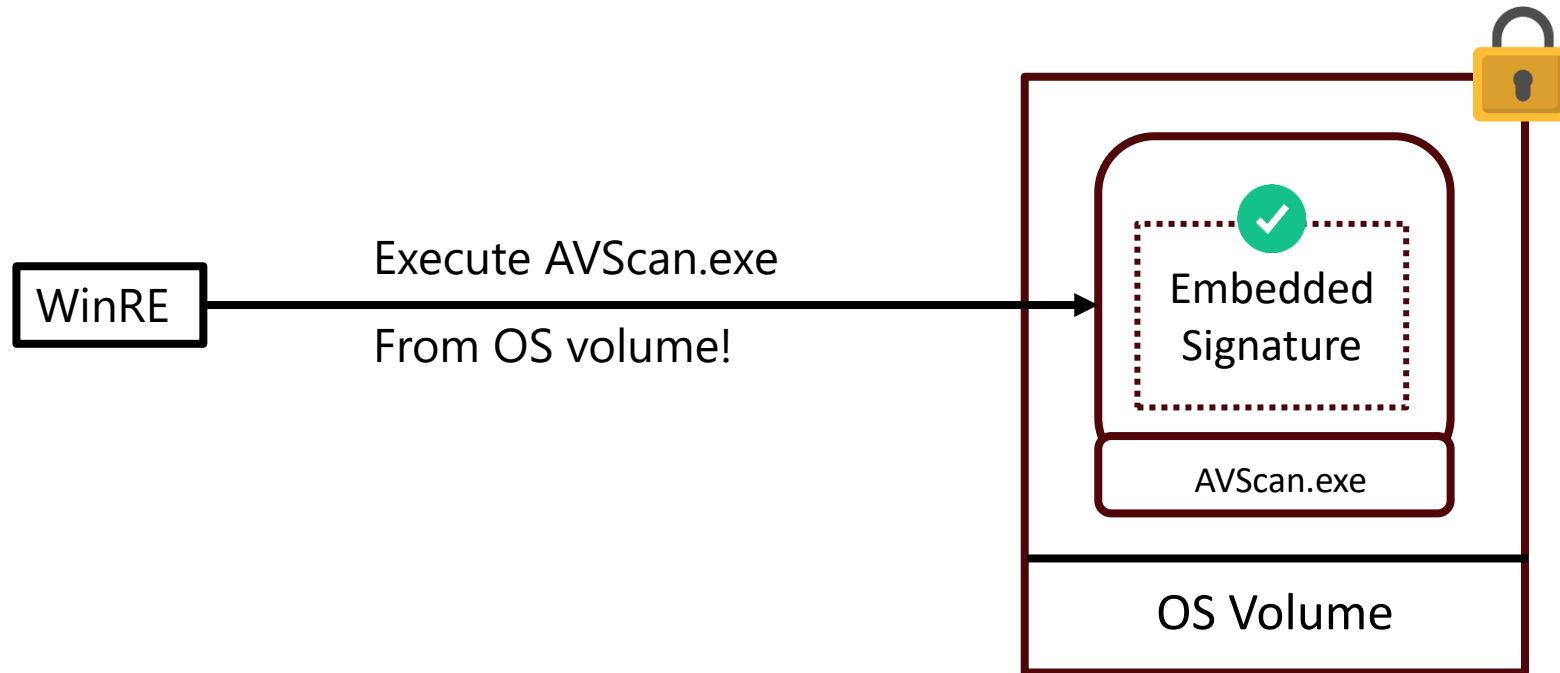
Offline Scanning Scheduled Operation

- Offline scanning allows launching Anti-Virus scan from WinRE against the main OS
- This is valuable against malwares that do not persist in WinRE runtime



Offline Scanning Scheduled Operation Limitations

- 1) Offline scanning apps are always executed from the Main OS
- 2) Offline scanning apps must be signed by Microsoft or WHQL
- 3) Signatures must be embedded



How Many Apps Fit With The Limitations?

~ 30 apps that fit with the limitations

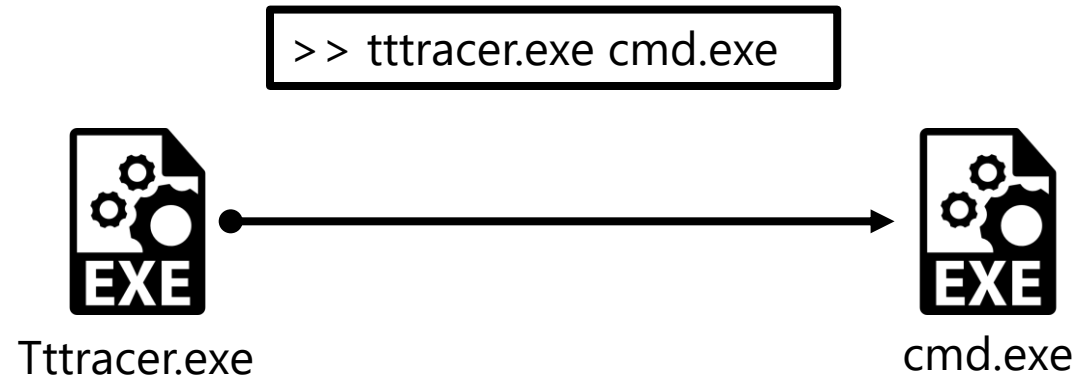


Not all apps are compatible running in WinRE



Any Apps That Can Be Abused?

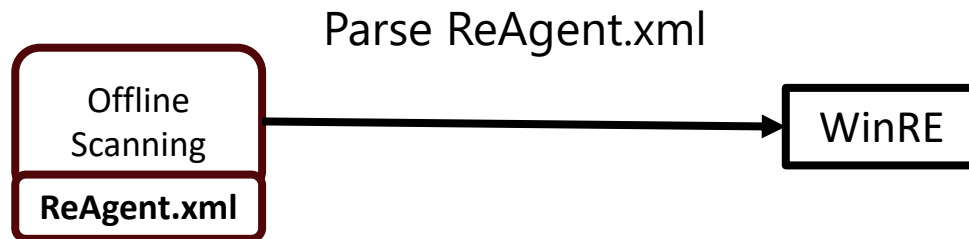
- Among the allowed apps is tttracer.exe, a Time Travel Debugging utility
- Tttracer.exe allows tracing an arbitrary executable



Vulnerability #2 – Exploitation Flow

1

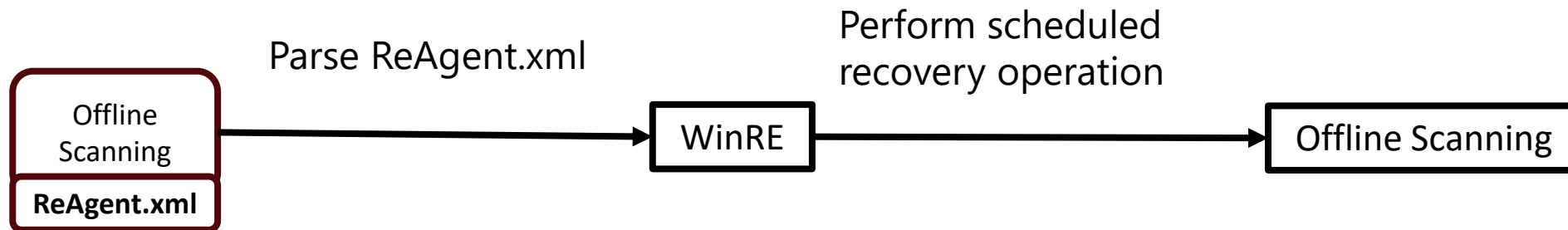
Schedule offline scanning in ReAgent.xml



Vulnerability #2 – Exploitation Flow

2

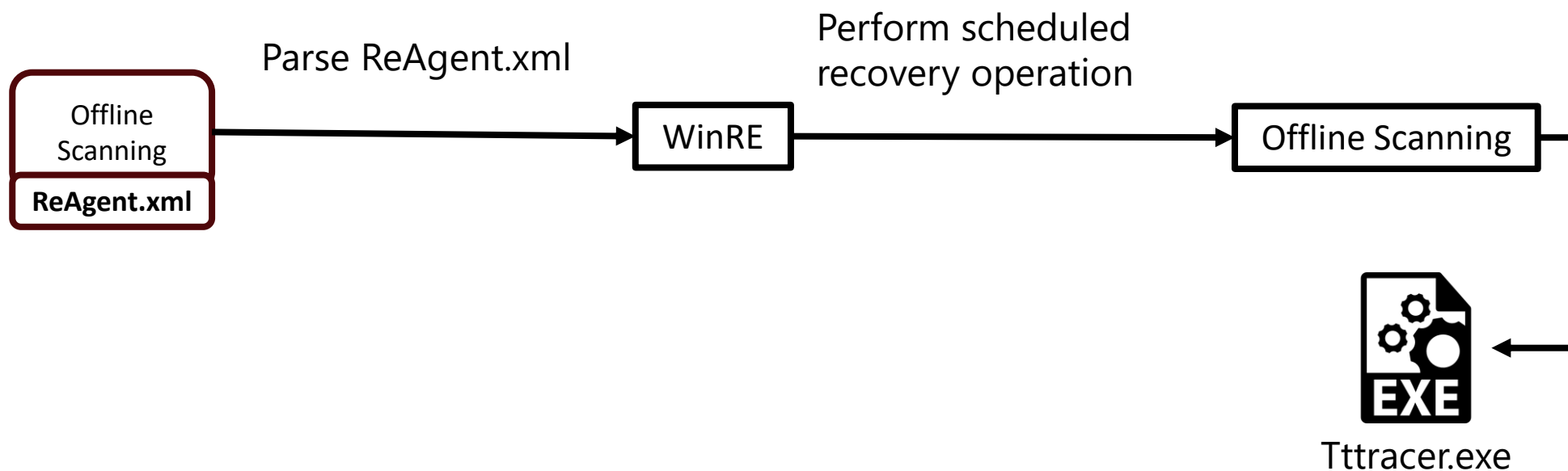
WinRE performs offline scanning operation



Vulnerability #2 – Exploitation Flow

3

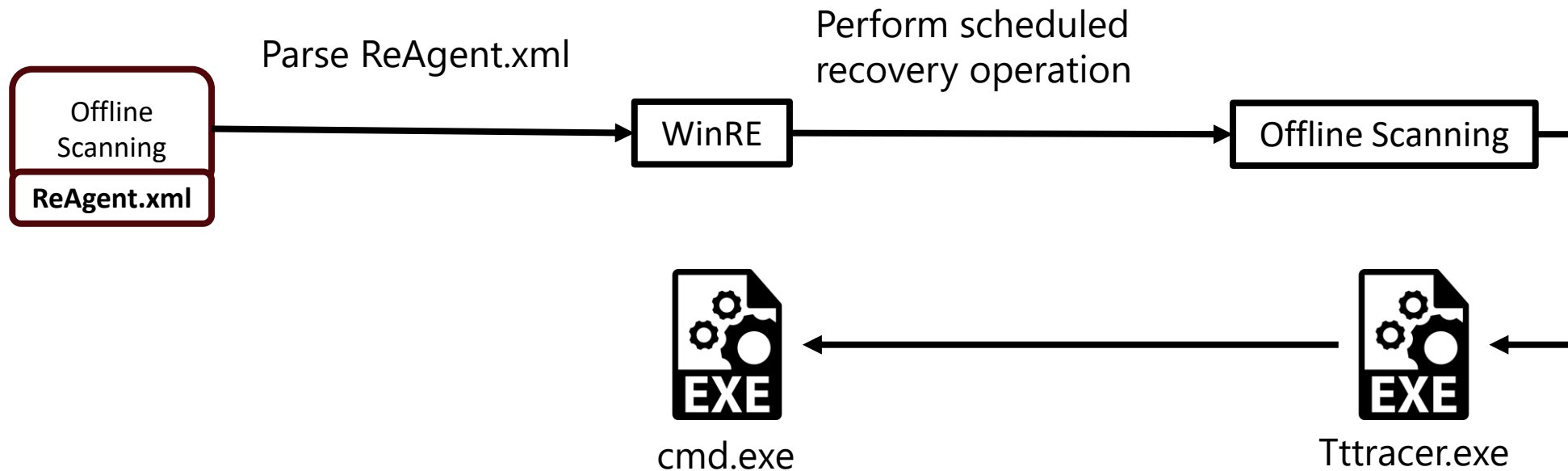
The offline scanning app tttracer.exe is executed



Vulnerability #2 – Exploitation Flow

4

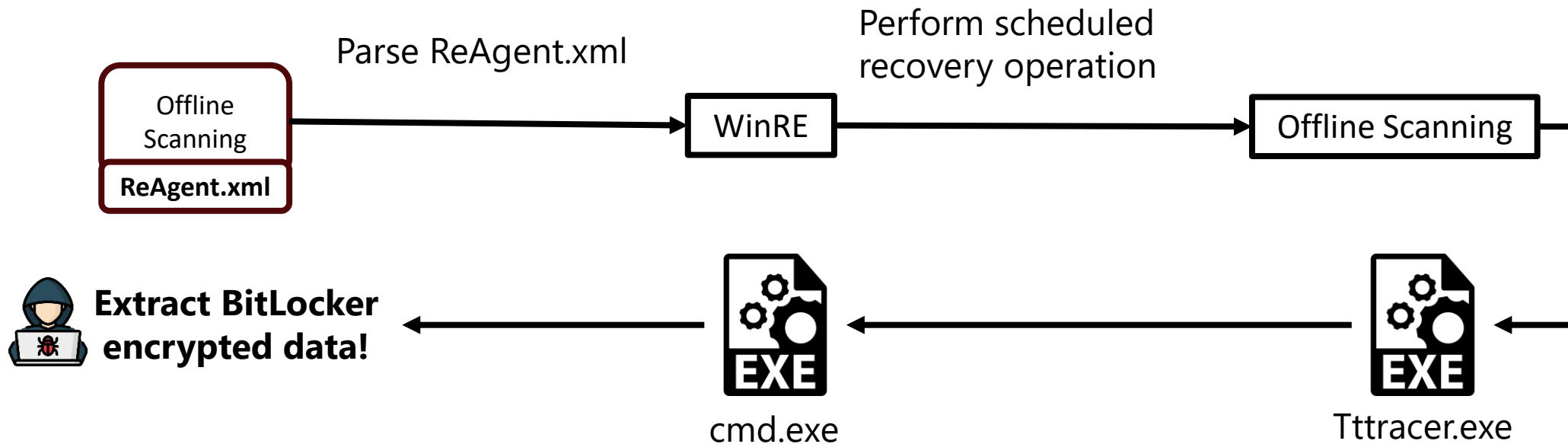
Tttracer.exe executes cmd.exe



Vulnerability #2 – Exploitation Flow

5

The executed cmd.exe now has full access to the BitLocker encrypted data!



Vulnerability #2 DEMO



10:44

Monday, July 21



Today's Focused Scheduled Operations

In today's presentation we will focus on analyzing two operations –



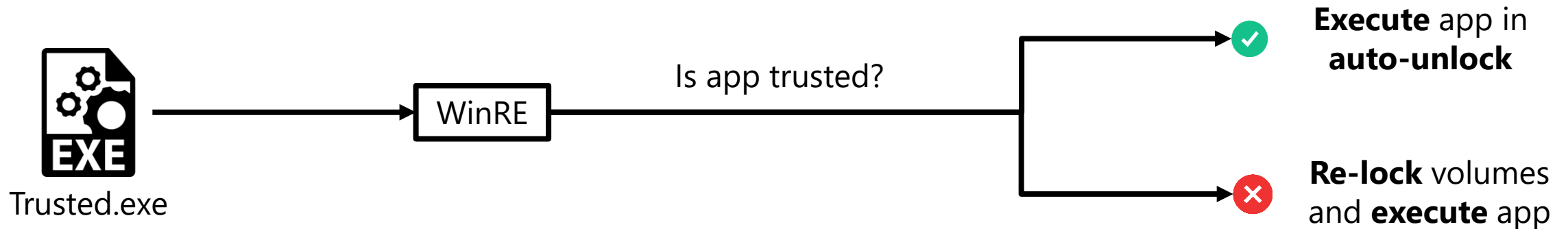
Offline Scanning



WinRE Apps

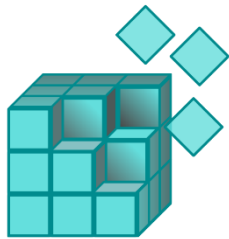
WinRE Apps Scheduled Operation

- WinRE apps allows the execution of apps in WinRE runtime
- If the app is trusted, it is executed in auto-unlock state
- If the app is not trusted, it is executed after volumes re-locked



WinRE Apps Trust Validation

- Trusted WinRE apps are registered by name and hash in WinRE's registry
- WinRE's registry lives in WinRE.wim and is **not accessible to an attacker!**



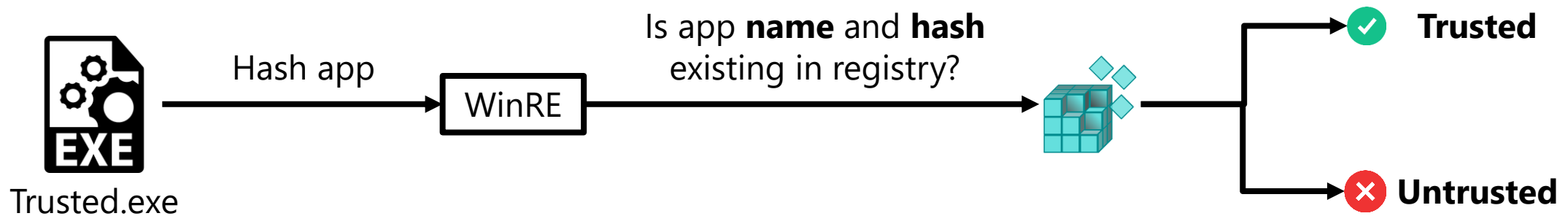
Trusted.exe

:

83e83...e1919

WinRE Apps Trust Validation

- If the hash of the app **matches a registry entry**, the app is **trusted**
- If the hash of the app **does not match a registry entry**, the app is **untrusted**



Is the Trust Validation Secure?

WinRE apps trust validation
is SOLID!

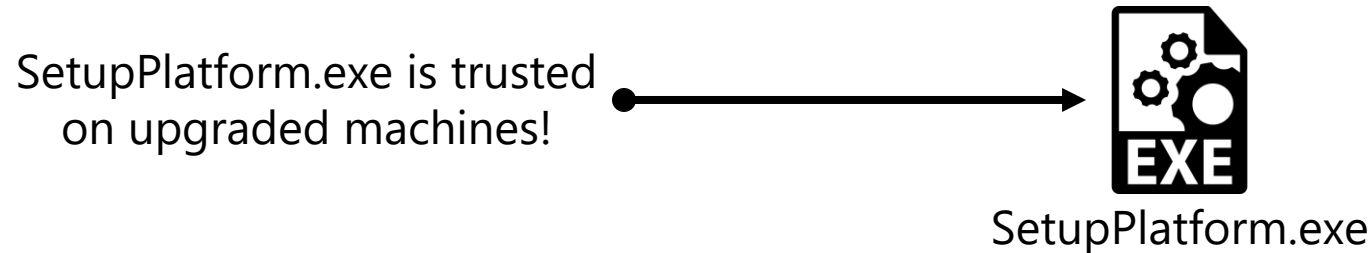


But already registered apps
also expose an attack surface!

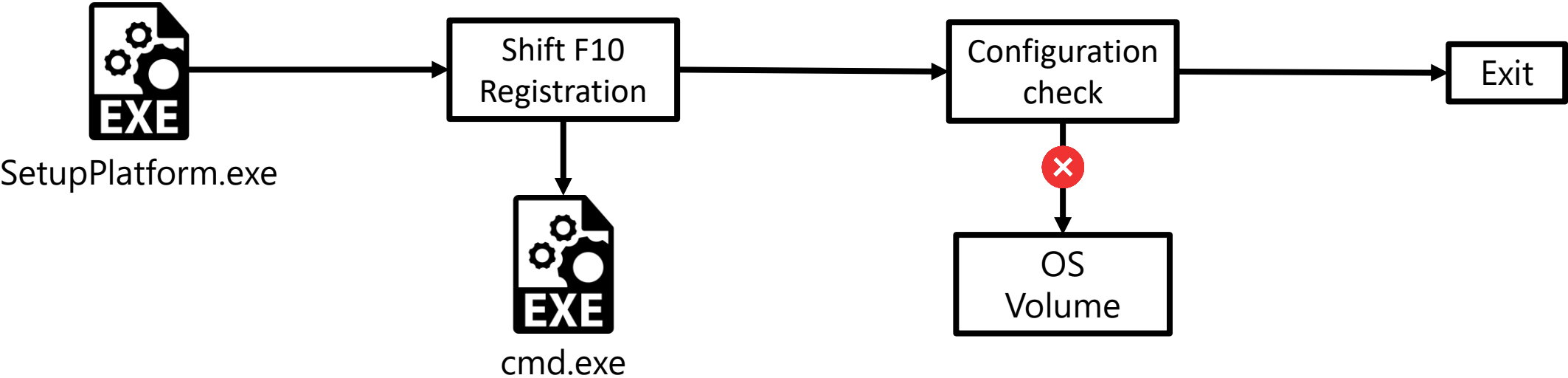


The SetupPlatform.exe Trusted App

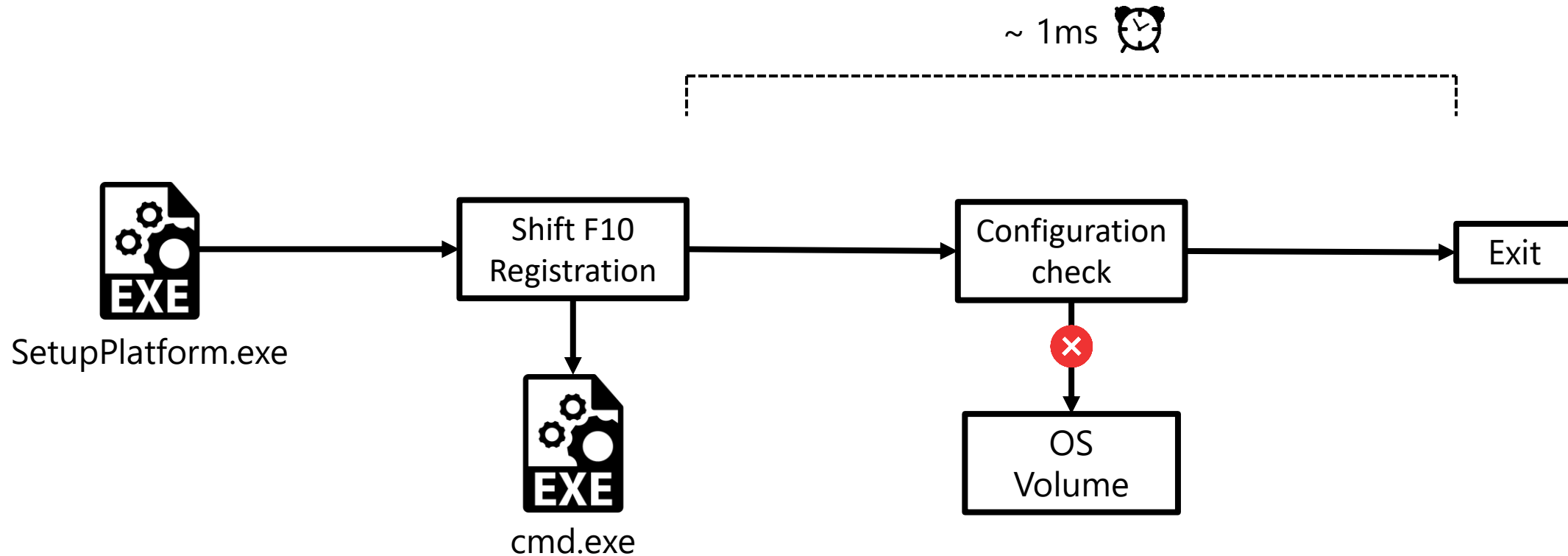
- During Windows Upgrade, the **SetupPlatform.exe** app is registered as trusted
- After the upgrade completes, the trusted app entry is **not removed!**



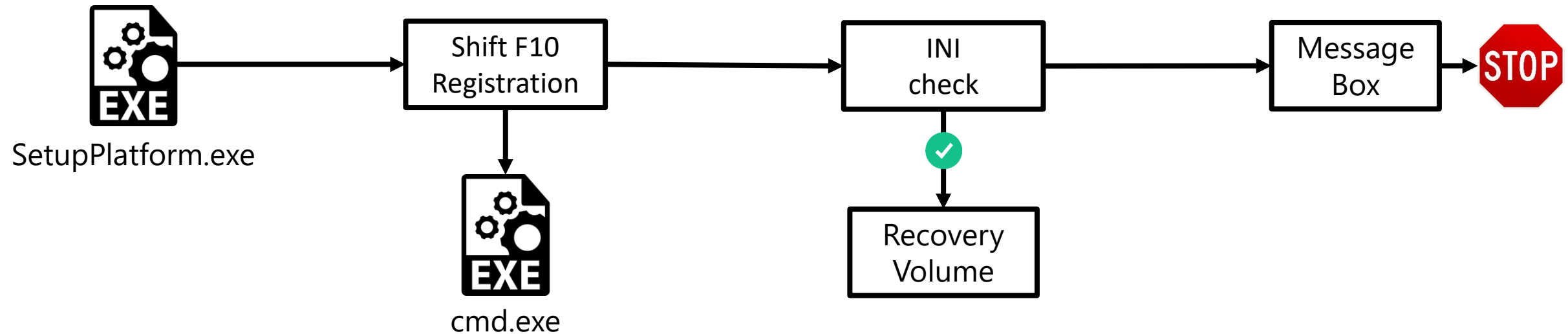
Peeking Inside SetupPlatform.exe



Time Window – Impossible to Trigger



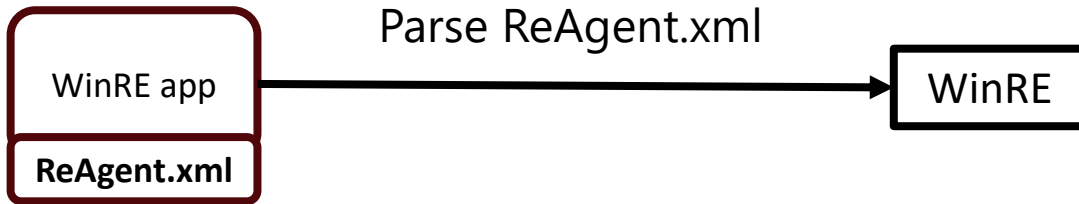
Creating Infinite Time Window



Vulnerability #3 – Exploitation Flow

1

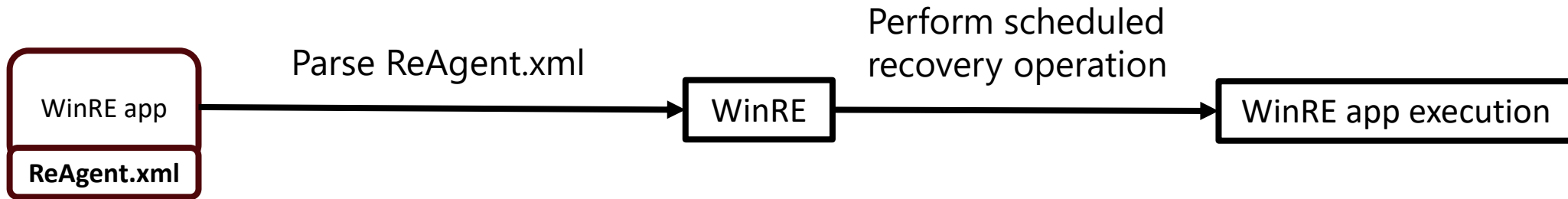
Schedule SetupPlatform WinRE app in ReAgent.xml



Vulnerability #3 – Exploitation Flow

2

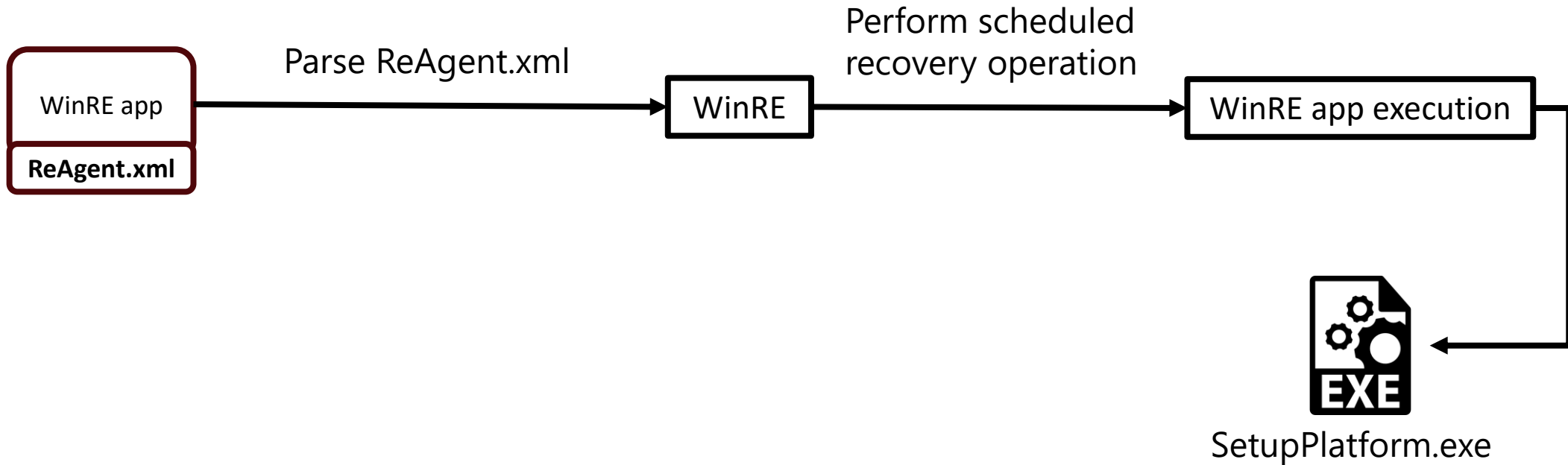
WinRE validates the app to be executed



Vulnerability #3 – Exploitation Flow

3

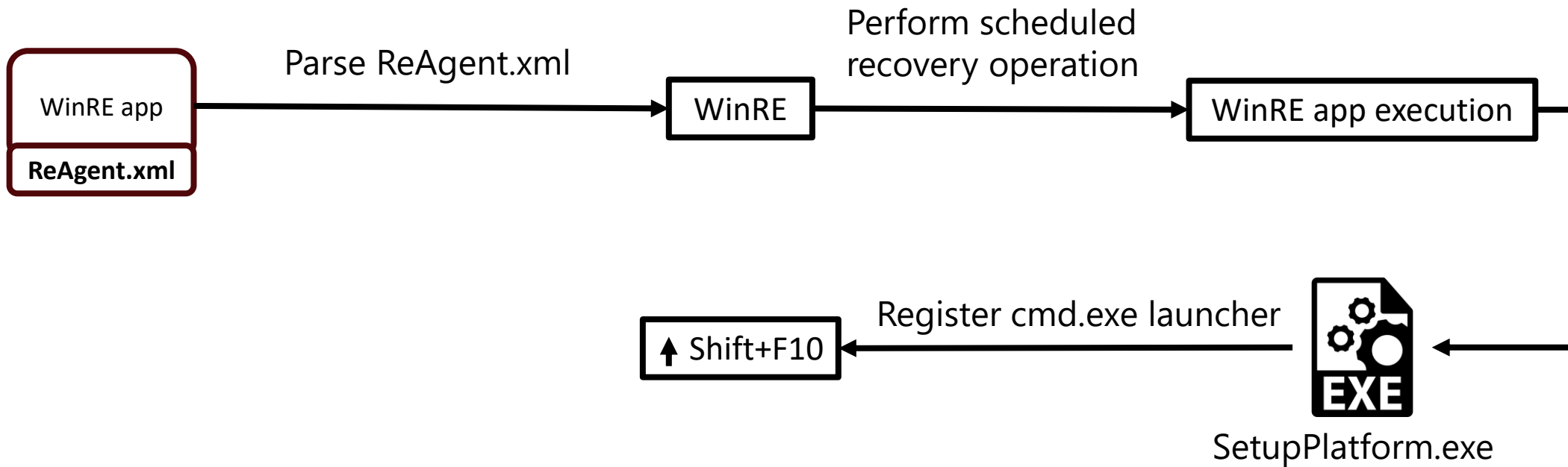
The WinRE app SetupPlatform.exe is executed



Vulnerability #3 – Exploitation Flow

4

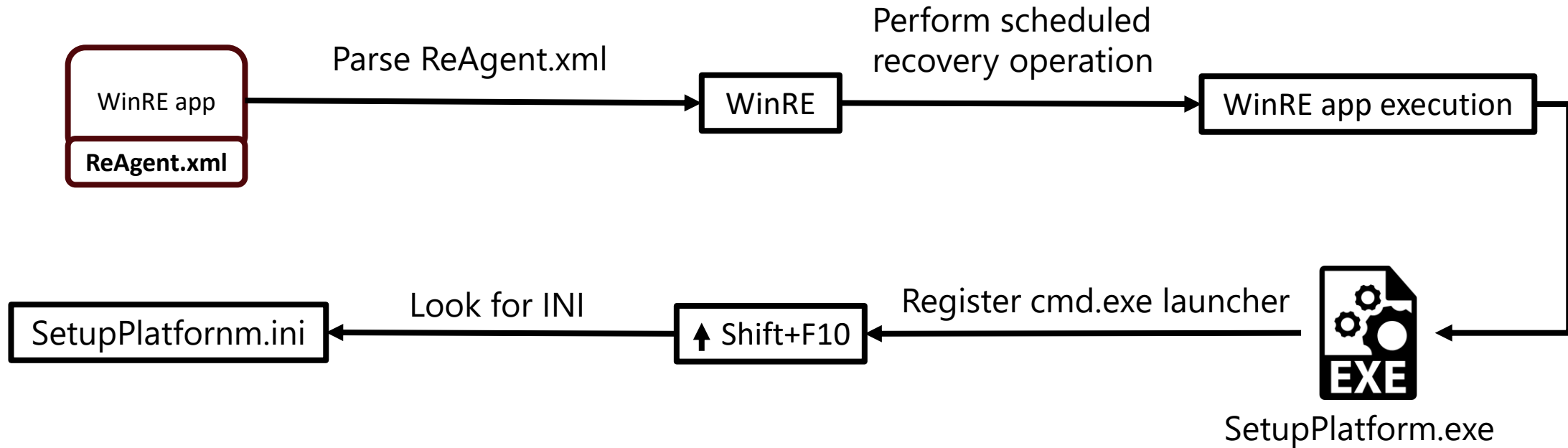
SetupPlatform.exe registers Shift+F10 hotkey as cmd.exe launcher



Vulnerability #3 – Exploitation Flow

5

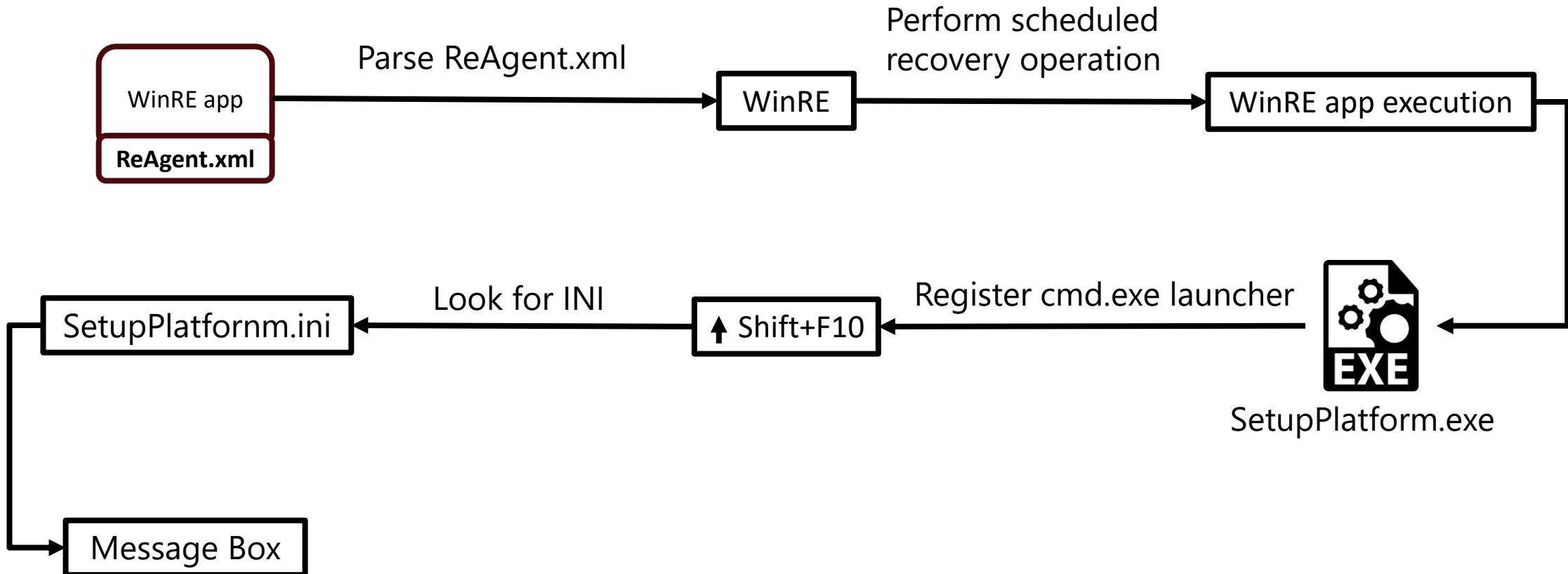
SetupPlatform.exe locates and parses SetupPlatform.ini



Vulnerability #3 – Exploitation Flow

6

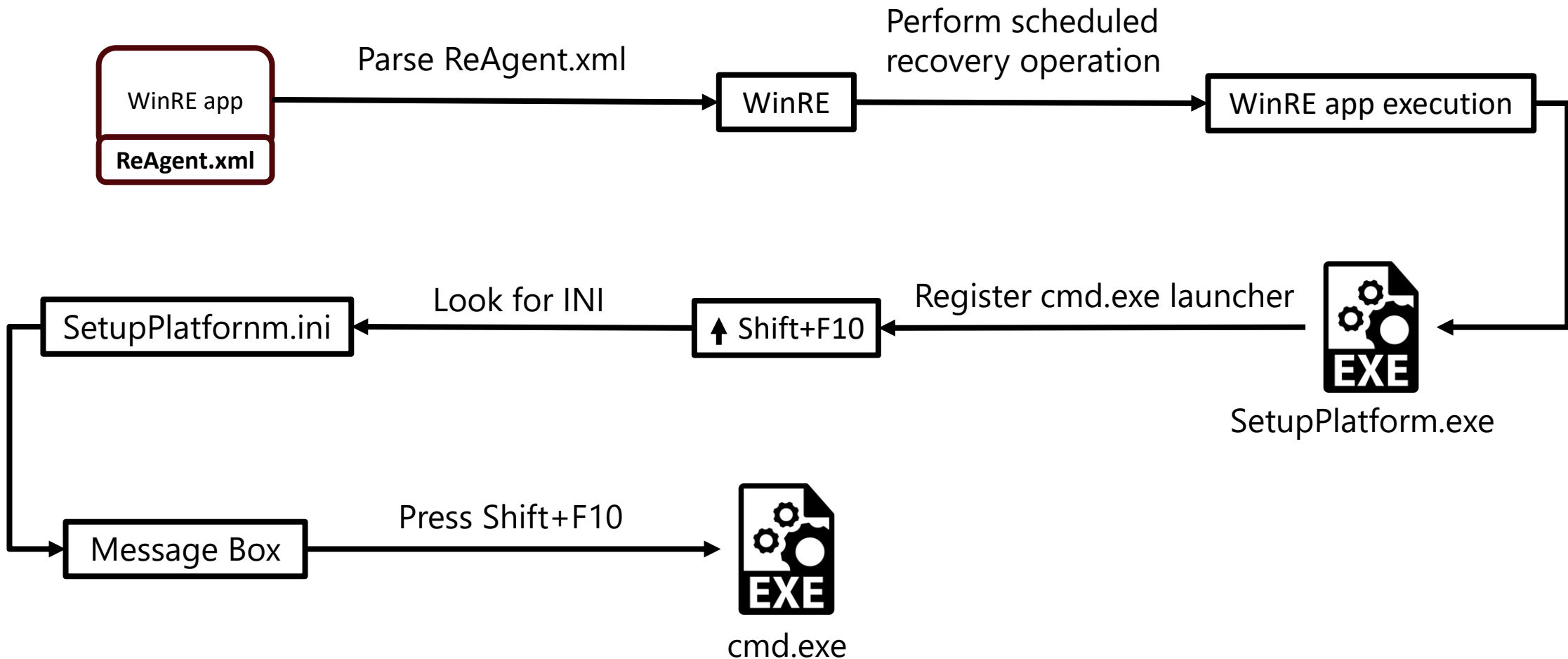
SetupPlatform.exe launches message box and blocks execution until message box is closed



Vulnerability #3 – Exploitation Flow

7

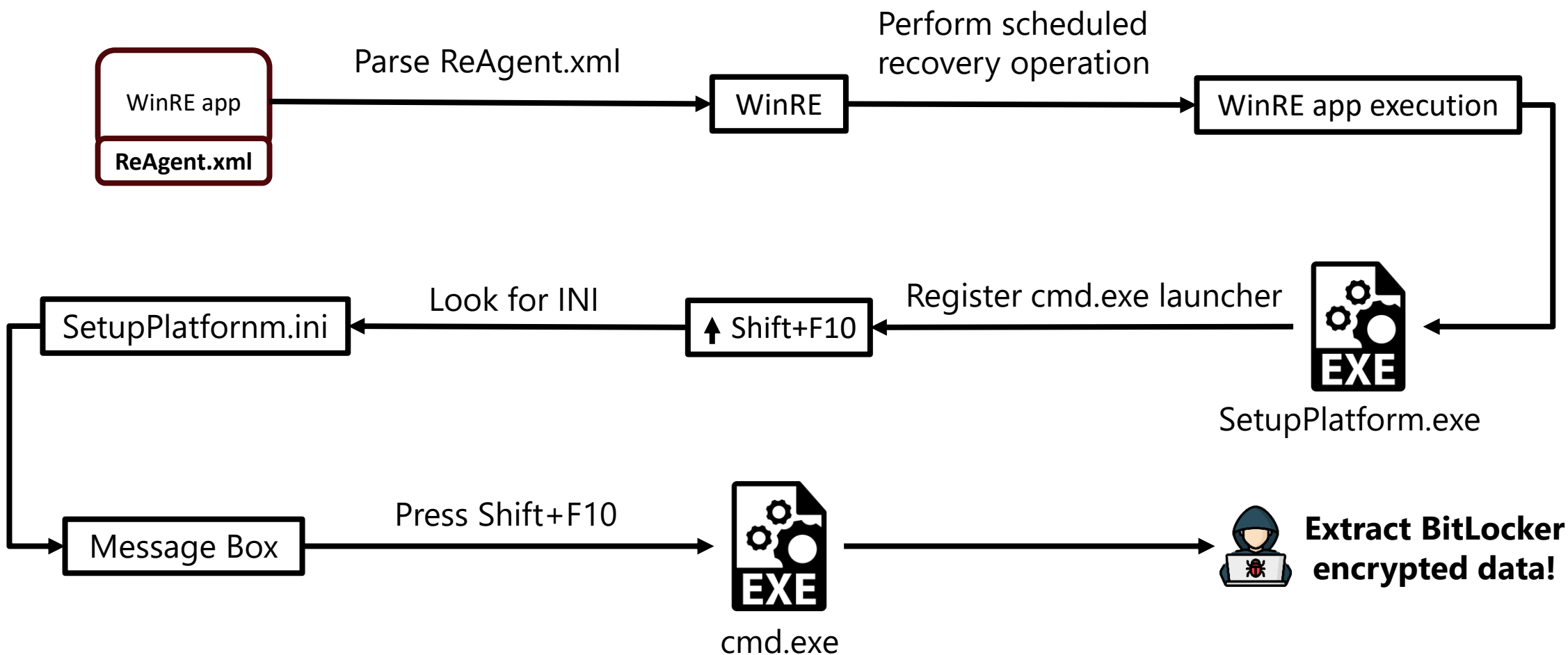
Shift+F10 is pressed to launch cmd.exe



Vulnerability #3 – Exploitation Flow

8

The executed cmd.exe now has full access to the BitLocker encrypted data!



Vulnerability #3 DEMO

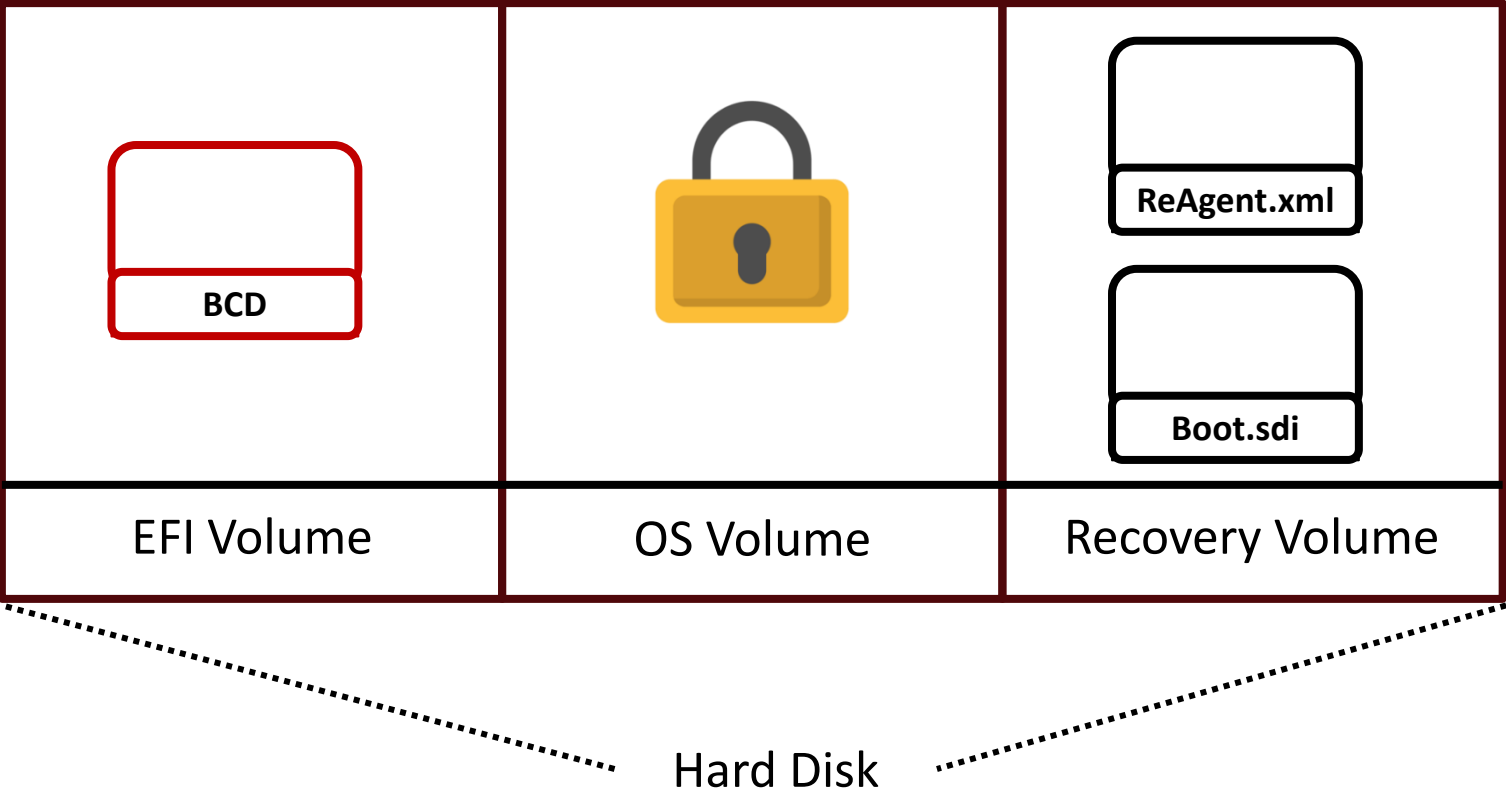




User

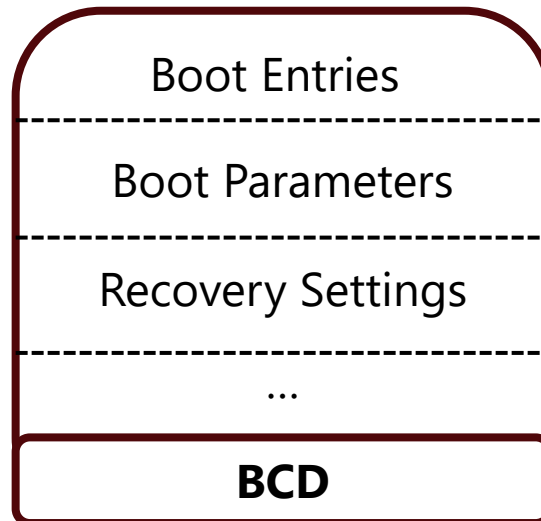
A dark gray rectangular input field with the text "Password" in white. A small white arrow points to the right at the end of the field.

Attacking BCD Parsing

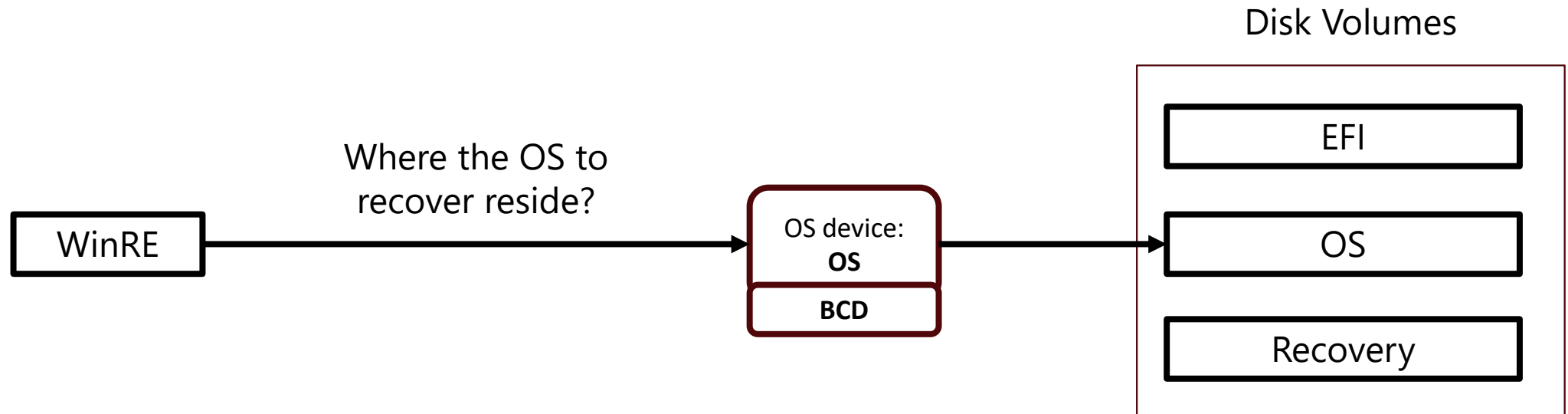


BCD Purpose

- BCD – Boot Configuration Data
- BCD represents Windows Boot configuration
- BCD controls boot entries, boot parameters, recovery settings and more



WinRE BCD Usage – Locate Target OS Volume

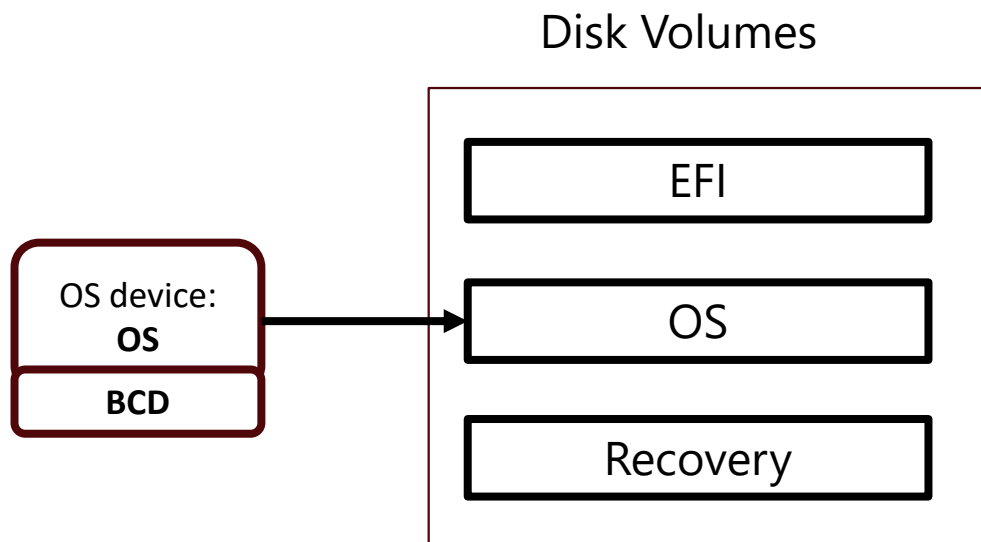


WinRE Blindly Trusts Target OS Volume

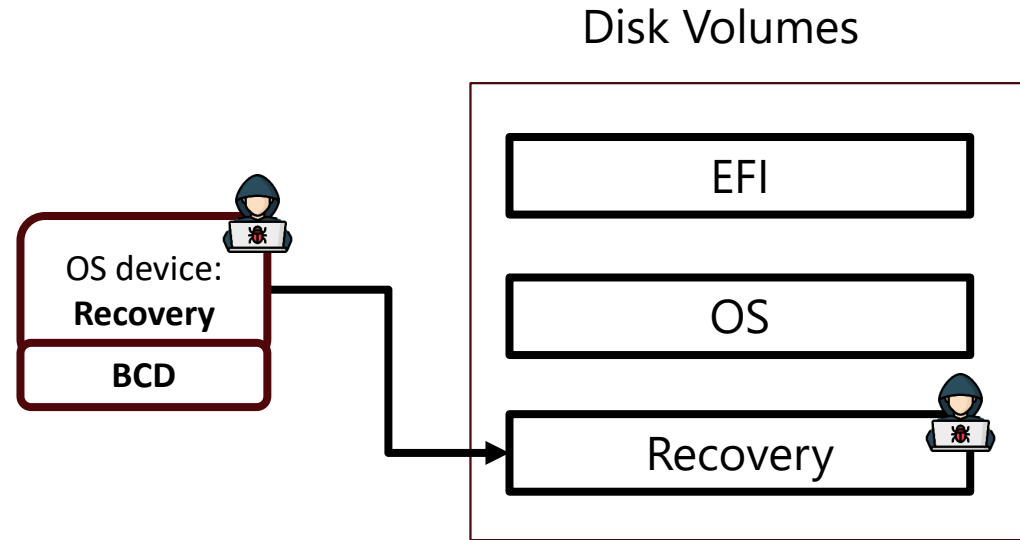


Desired Primitive – Target OS Location Impersonation

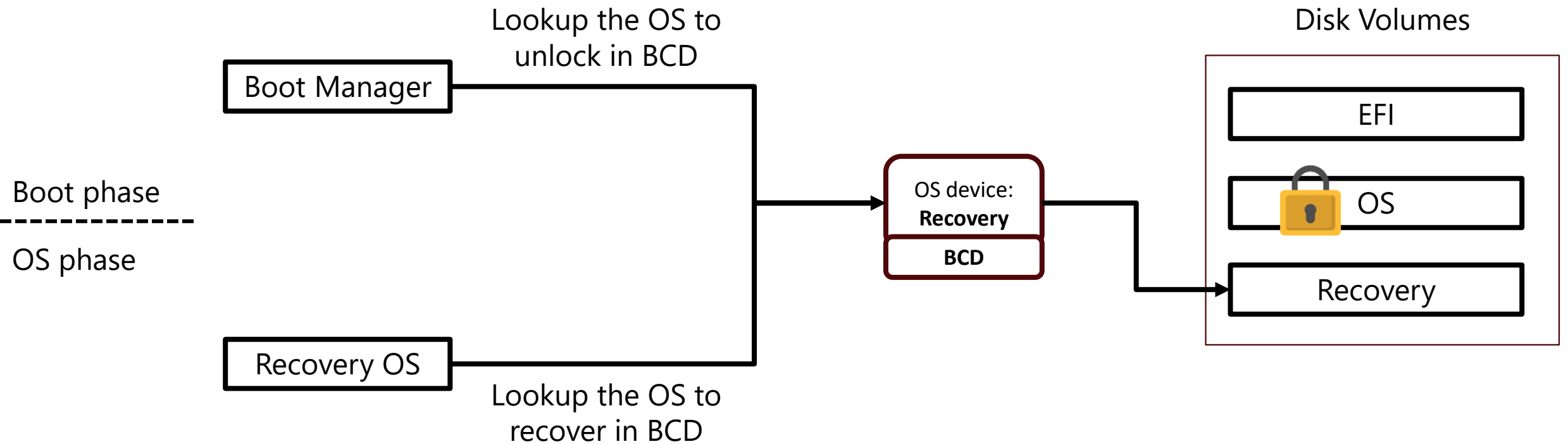
Current State



Desired State

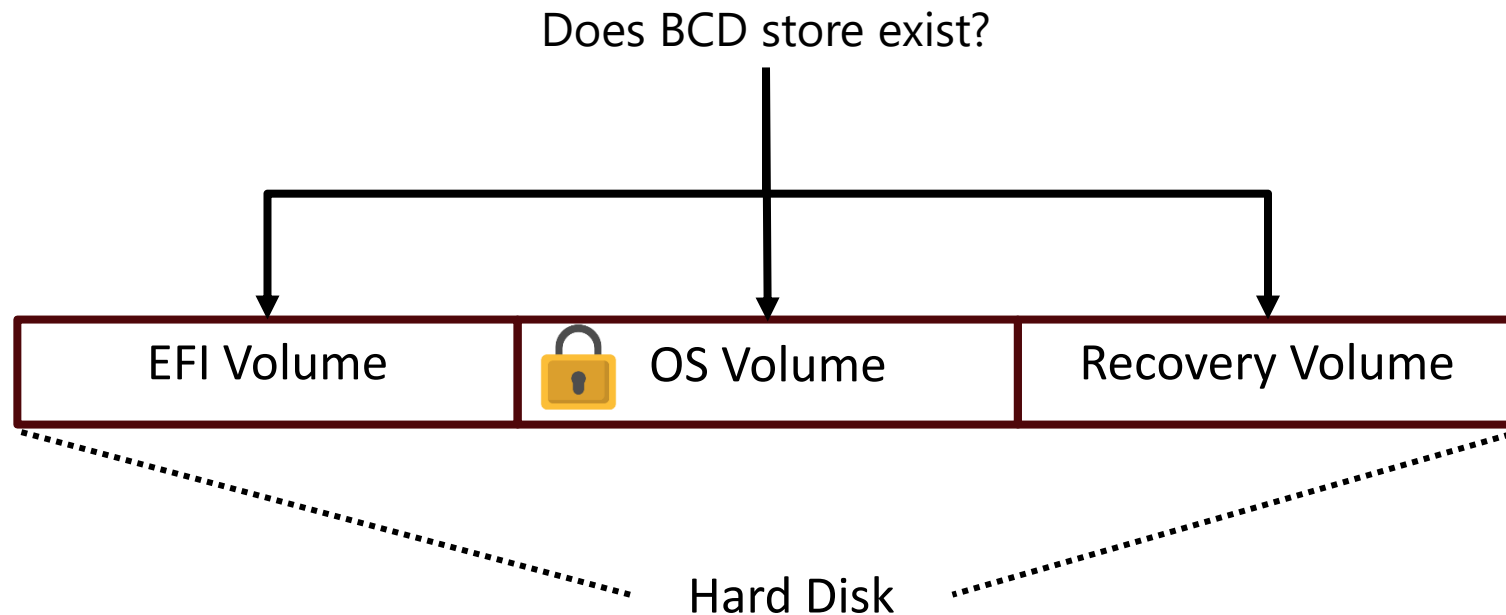


Directly Controlling Target OS Location – Not Valuable



WinRE BCD Store Search Logic

WinRE iterates over disk volumes and searches each one for the BCD store

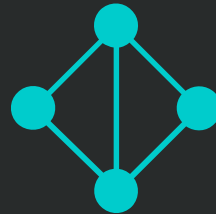


Volume Iteration Functions – Find[First/Next]Volume



FindFirstVolume and FindNextVolume Remarks Section from MSDN

"You should not assume any correlation between the order of the volumes that are returned by these functions and the order of the volumes that are on the computer [...]"

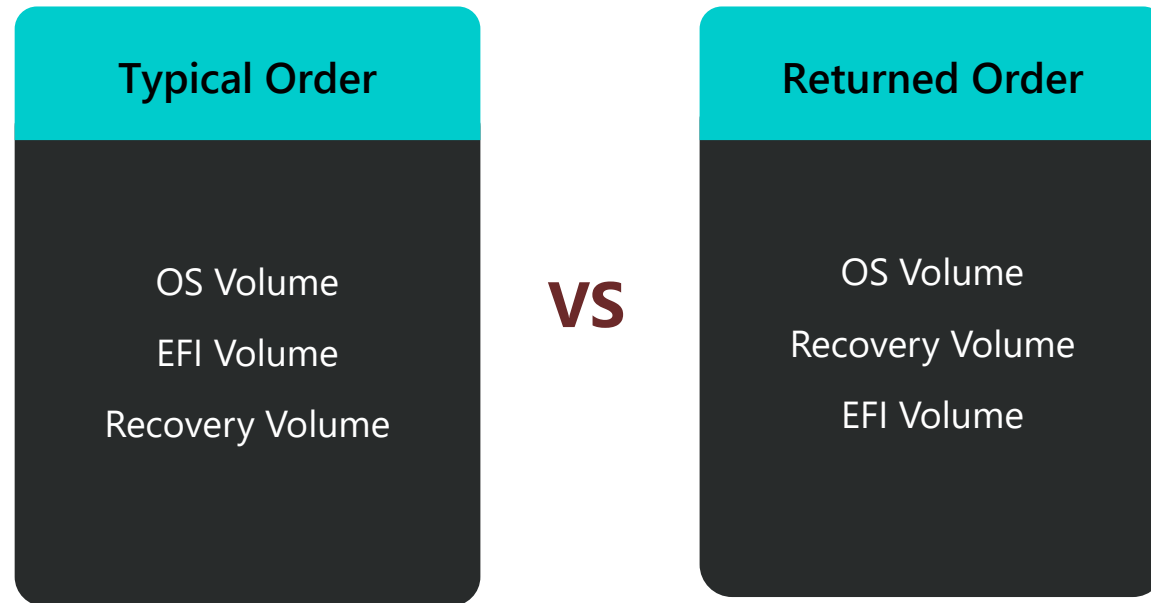


Typical Volume Order

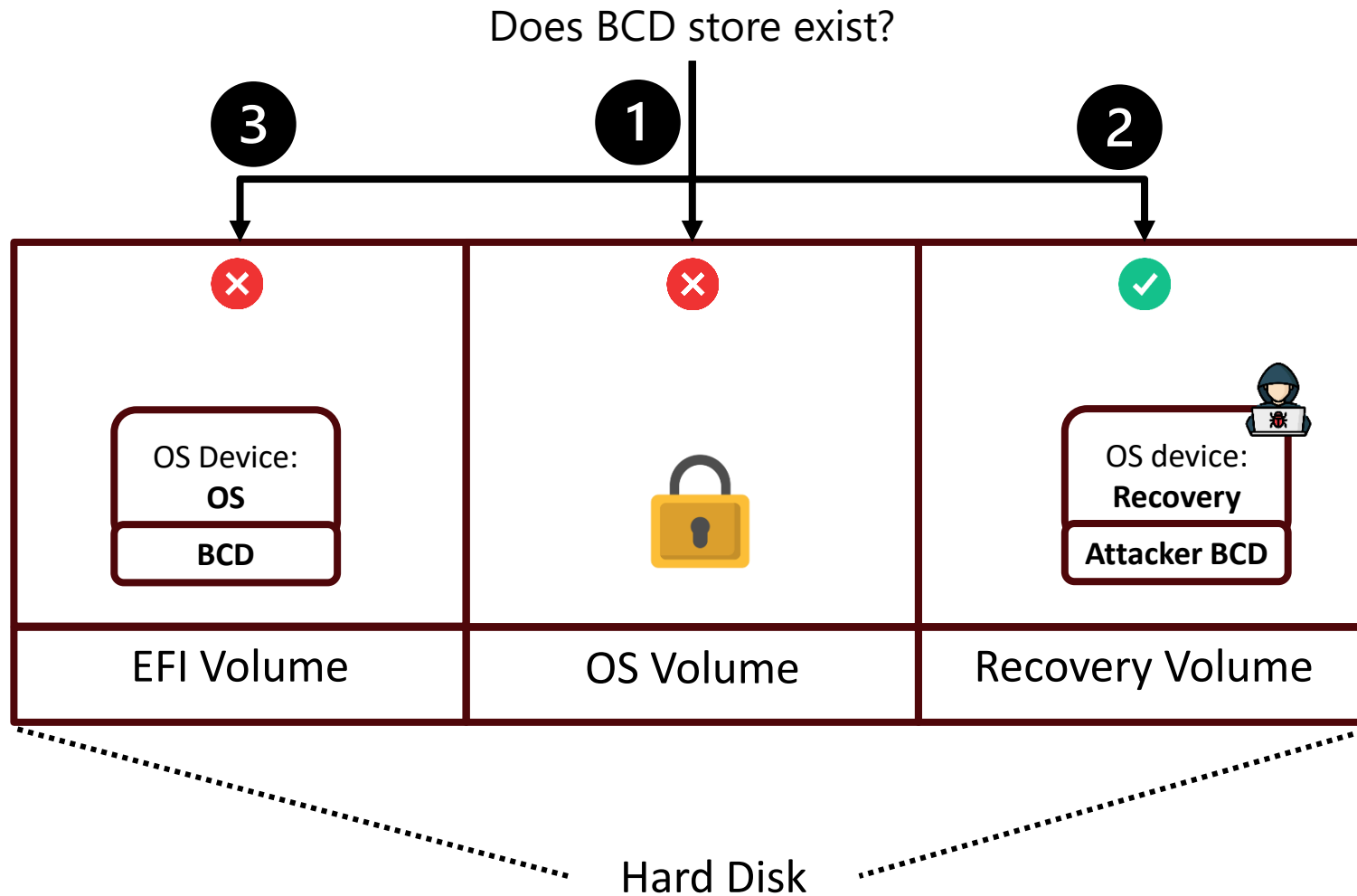
```
DISKPART> list vol
```

Volume ###	Ltr	Label	Fs	Type	Size	Status	Info
-----	---	-----	-----	-----	-----	-----	-----
Volume 0	C	Windows	NTFS	Partition	951 GB	Healthy	Boot
Volume 1		SYSTEM	FAT32	Partition	260 MB	Healthy	System
Volume 2		WinRE_DRV	NTFS	Partition	2000 MB	Healthy	Hidden

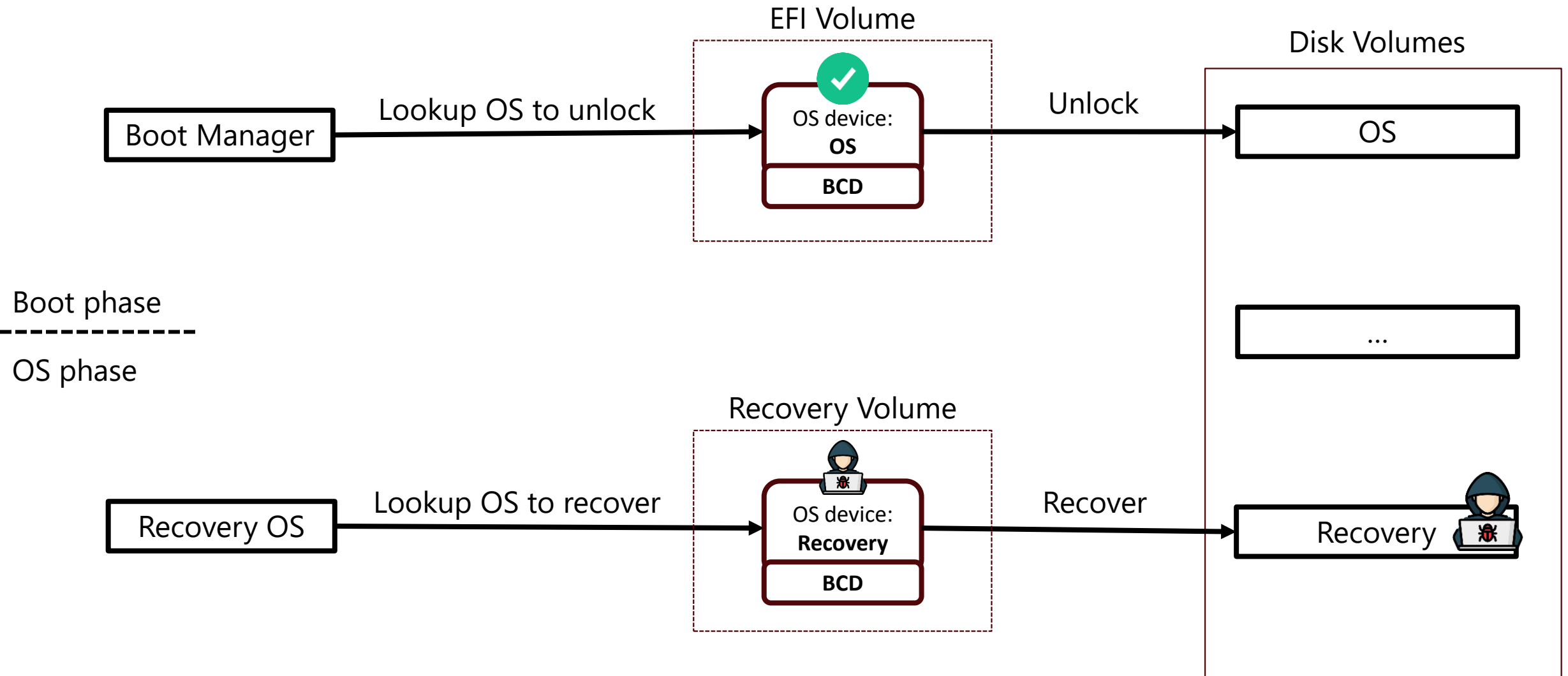
WinRE BCD Store Search Logic



Gaining The Desired Primitive



The Gained Primitive



WinRE Blindly Trusts The Recovery Volume



Exploitation Requirements

Find a WinRE flow that:



Can be executed from
WinRE UI or ReAgent.xml



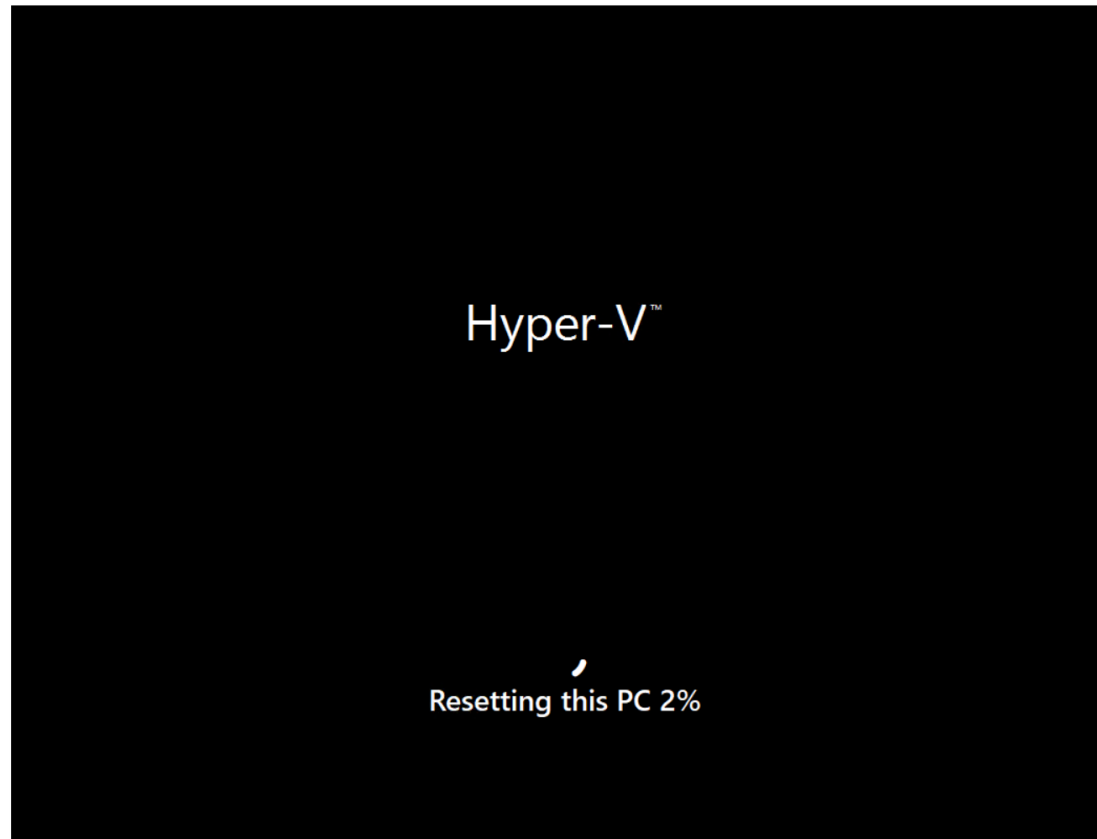
Does not trigger the relock
functionality



Queries configuration from
the target OS to perform
sensitive operations

Potential Candidate – Push Button Reset (PBR)

Push Button Reset (PBR) - Windows's System Reset tool



Online PBR Exploitation Applicability

Online PBR mode was found applicable for exploitation



Can be executed from WinRE
UI or ReAgent.xml



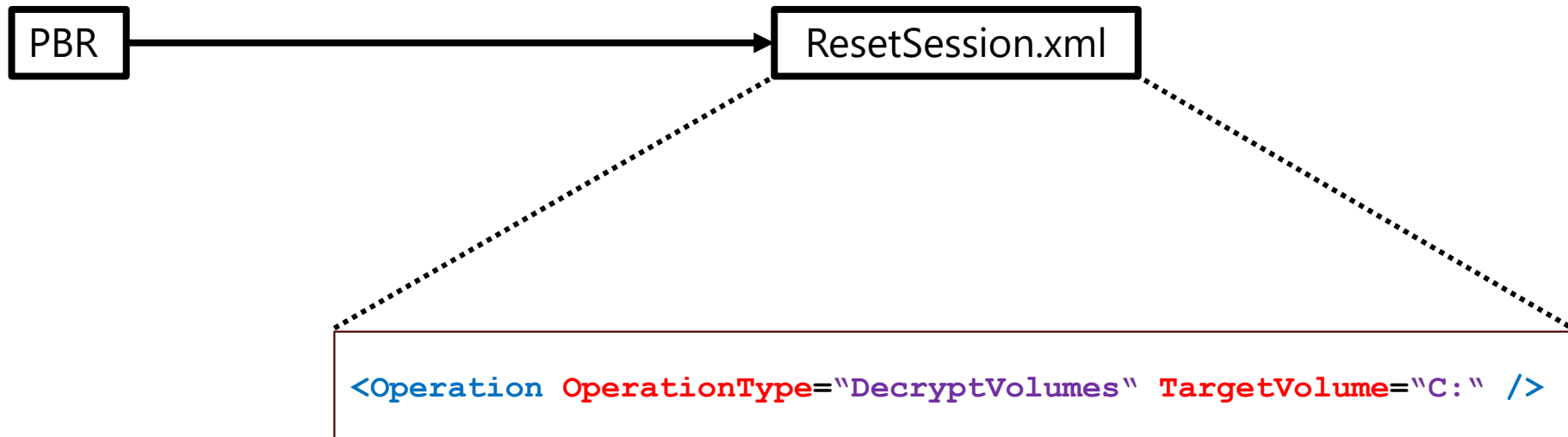
Does not trigger the relock
functionality



Queries configuration from
the target OS to perform
sensitive operations

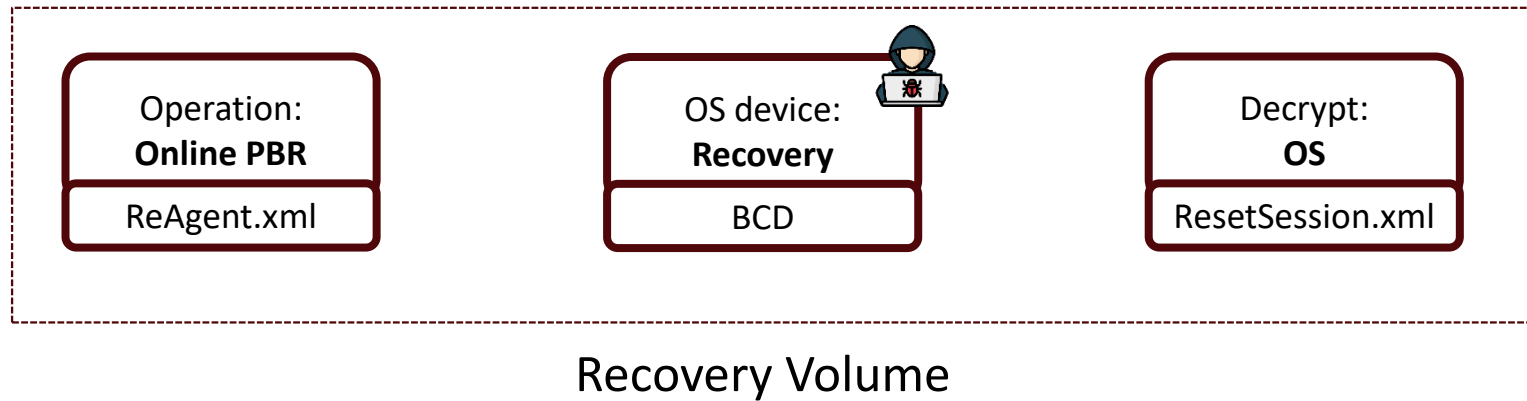
The Sensitive Configuration - PBR ResetSession.xml

PBR can decrypt BitLocker volumes if stated in ResetSession.xml

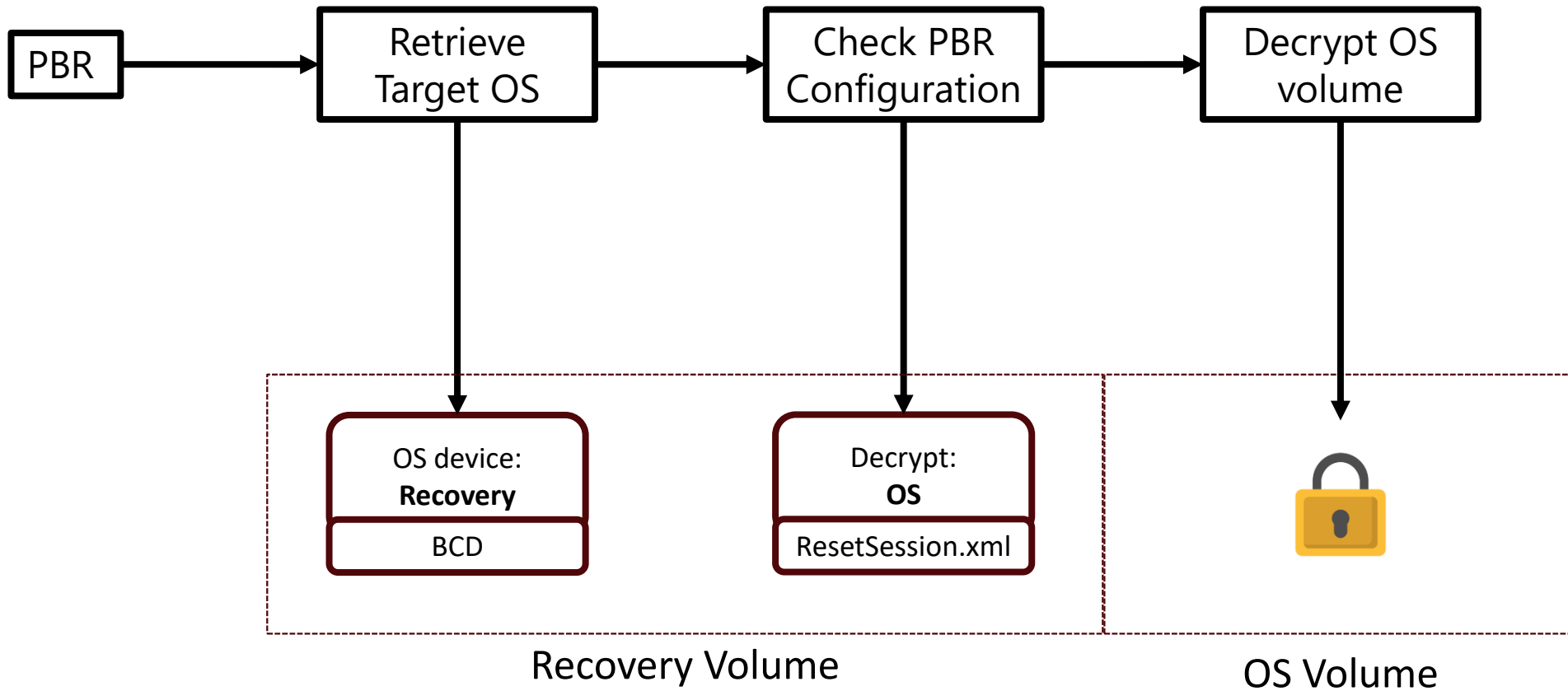



Exploitation Setup

The exploit requires creating few files on the Recovery volume:



Exploitation Flow





Vulnerability #4 DEMO



Copilot and AI agents are helping
people do more each day at work in
just a fraction of the time



Like the image that you see?

7:32

Wednesday, July 9

Washington



81°F



Areal
flood
watch

[See full forecast](#)

Daily Wonder



Sri Lanka's Cloud-Kissed
Nuwara Eliya: A Drone...

[Watch more videos like this](#)

Singapore quiz



Singapore is located on
which continent?

[Take the quiz](#)



A photograph of a server room with multiple racks of servers. The servers are dark-colored with perforated front panels. The background is filled with a bokeh effect of yellow and blue lights, suggesting a large, brightly lit data center. The text "Closing Remarks" is overlaid on the left side of the image.

Closing Remarks

Vulnerability Fixes

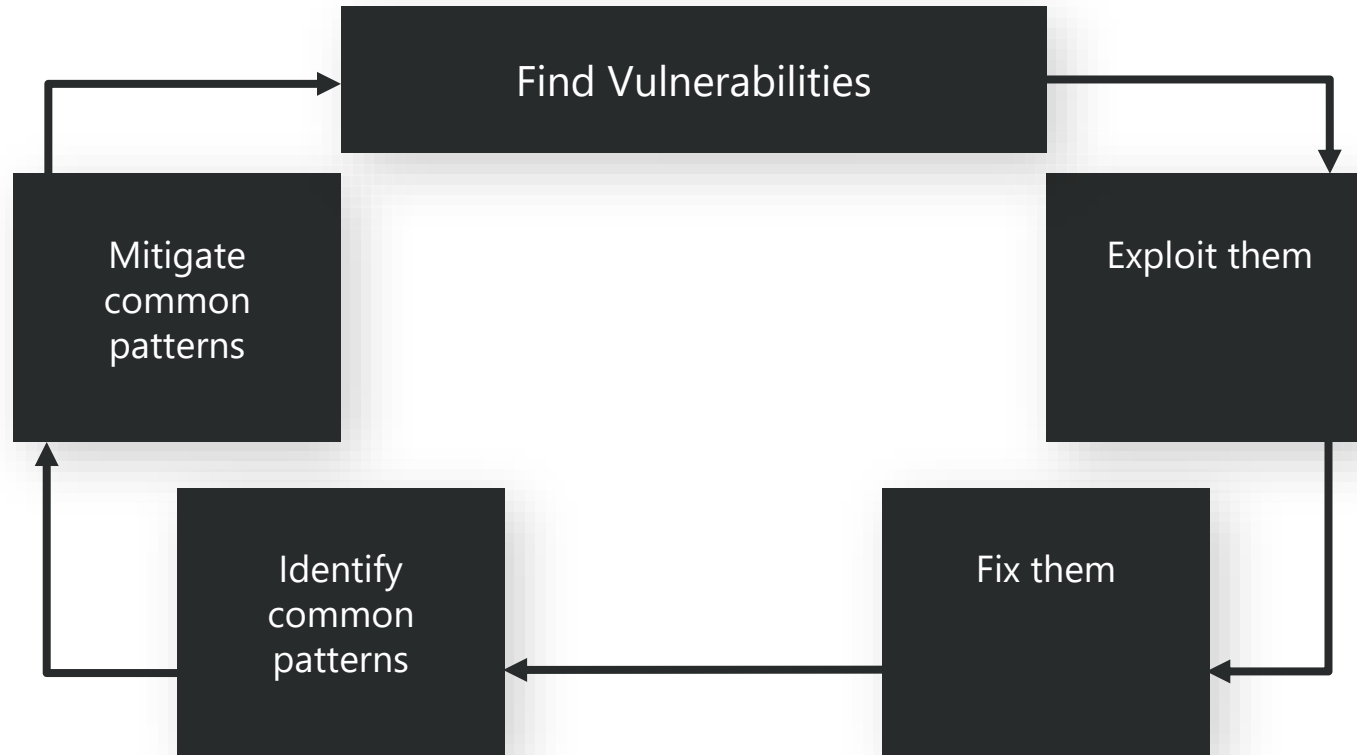
- The CVEs for the vulnerabilities presented today are –
 - CVE-2025-48800
 - CVE-2025-48003
 - CVE-2025-48804
 - CVE-2025-48818
- Fixes were shipped in July's Patch Tuesday

BitLocker Countermeasures

- Enable TPM+PIN for pre-boot authentication
- Enable the [REVISE mitigation](#) for anti-rollback protection



We're Never Done!



Thank You!

Security **T**esting & **O**ffensive **R**esearch at **M**icrosoft (STORM)

Alon Leviev (@alon_leviev)
Security Researcher @ Microsoft

Netanel Ben Simon (@NetanelBenSimon)
Senior Security Researcher @ Microsoft