CONFUSED RECOVERY

A New Attack Class on Windows Recovery

Who Am I?

Alon Leviev

Senior Security Researcher
@ Microsoft (STORM)

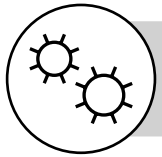
Microsoft

STORM

Agenda

Research Background

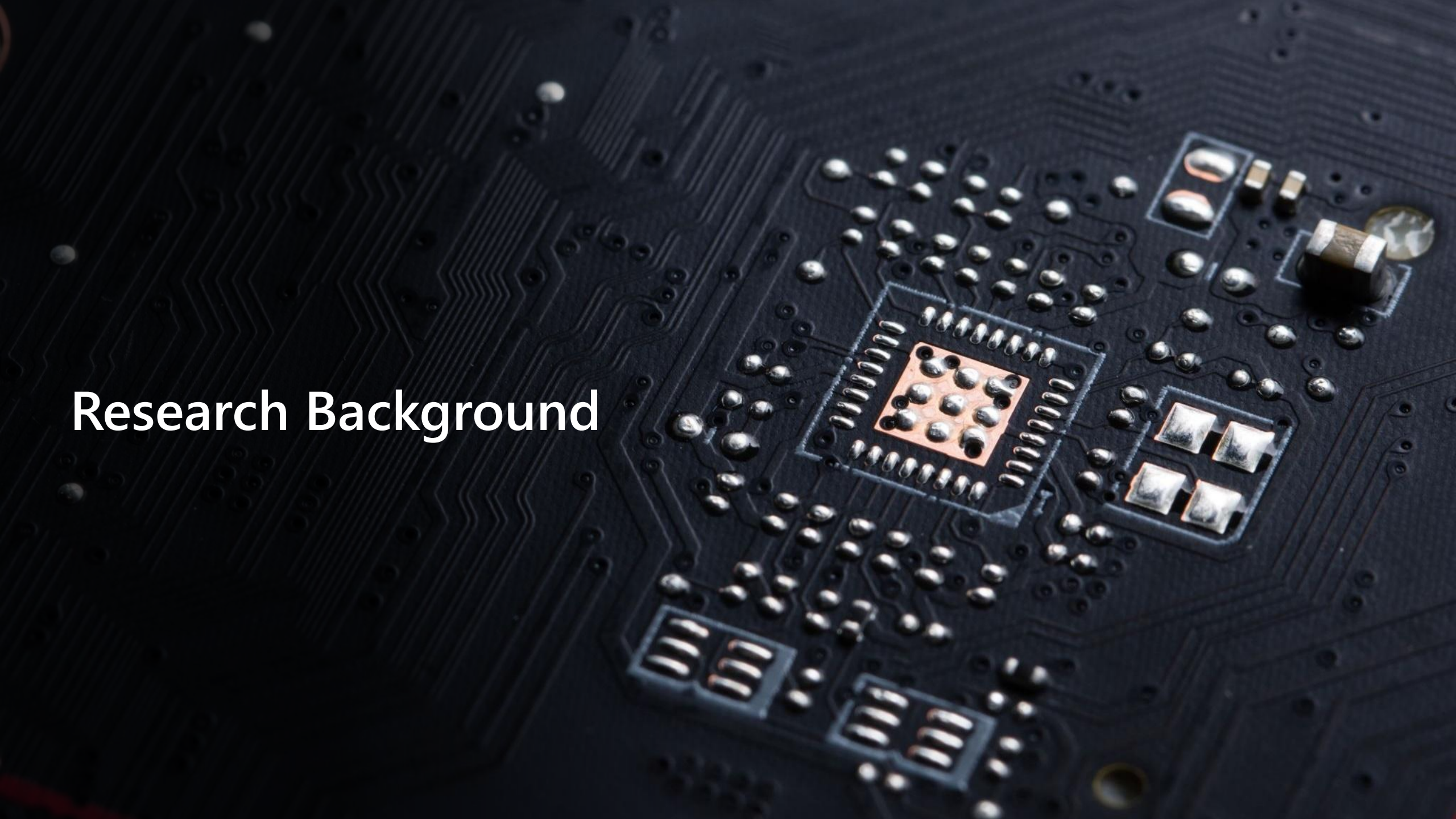
The Attack Class



The Mitigation

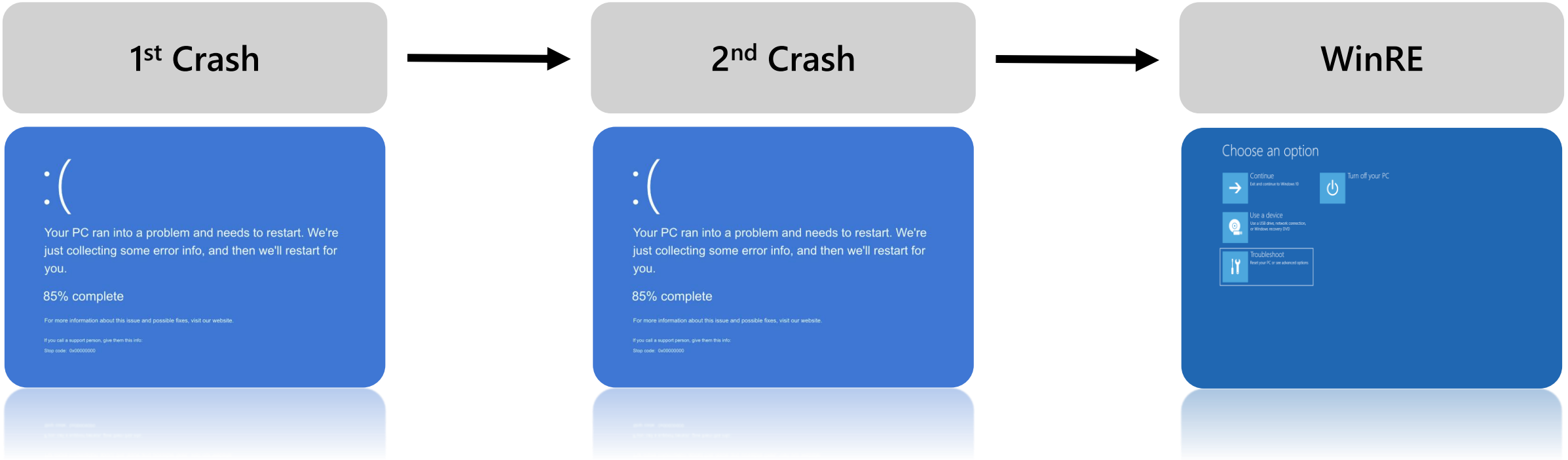
Closing Remarks

Research Background



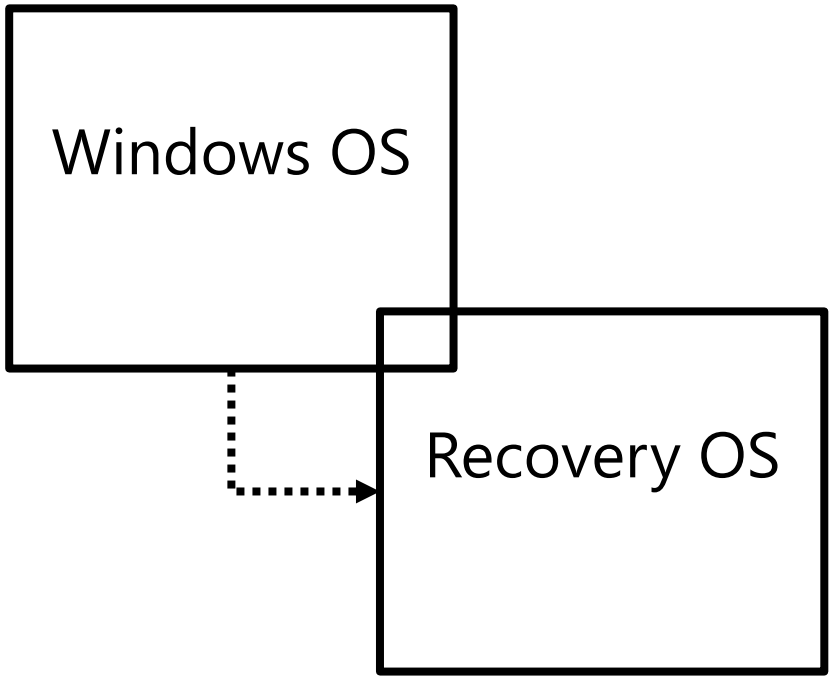
WinRE Overview

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



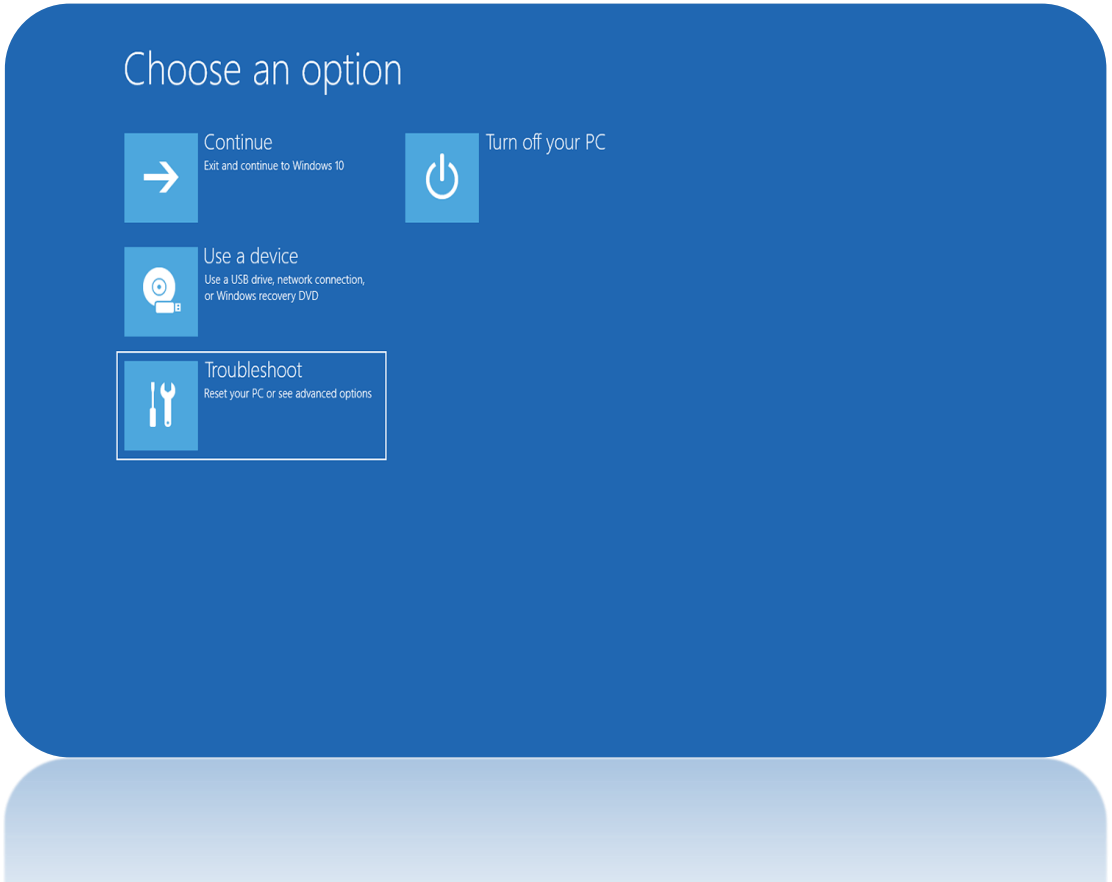
WinRE Architecture – Recovery OS

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



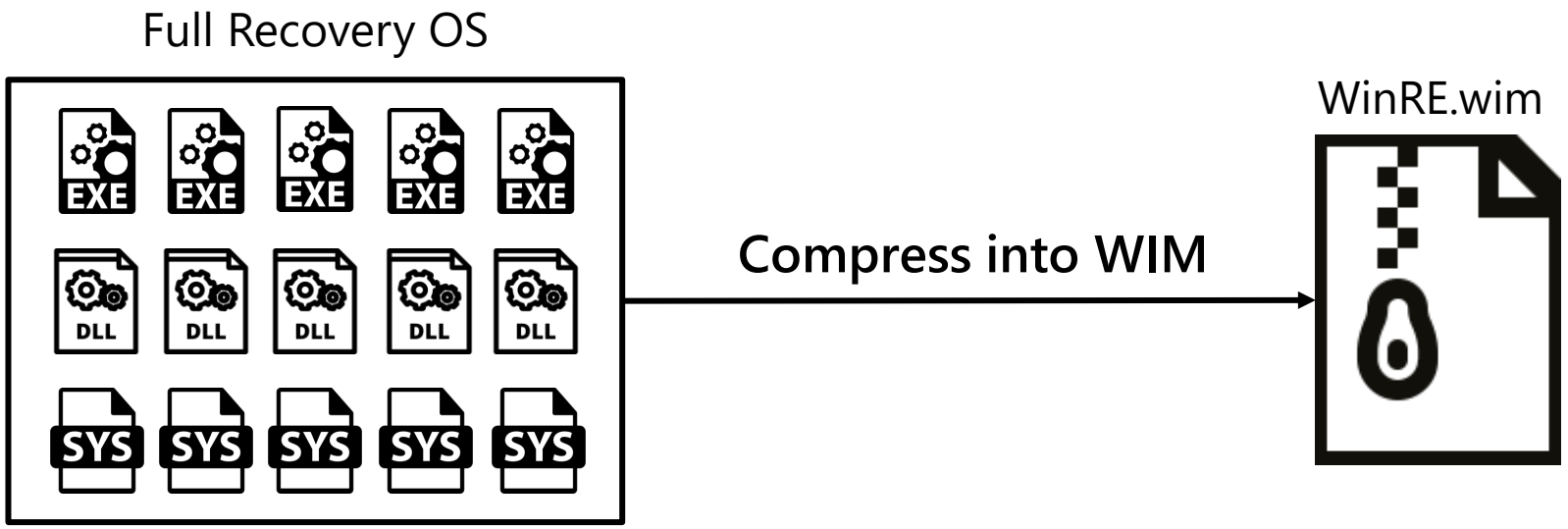
WinRE Architecture – Recovery OS Customizations

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



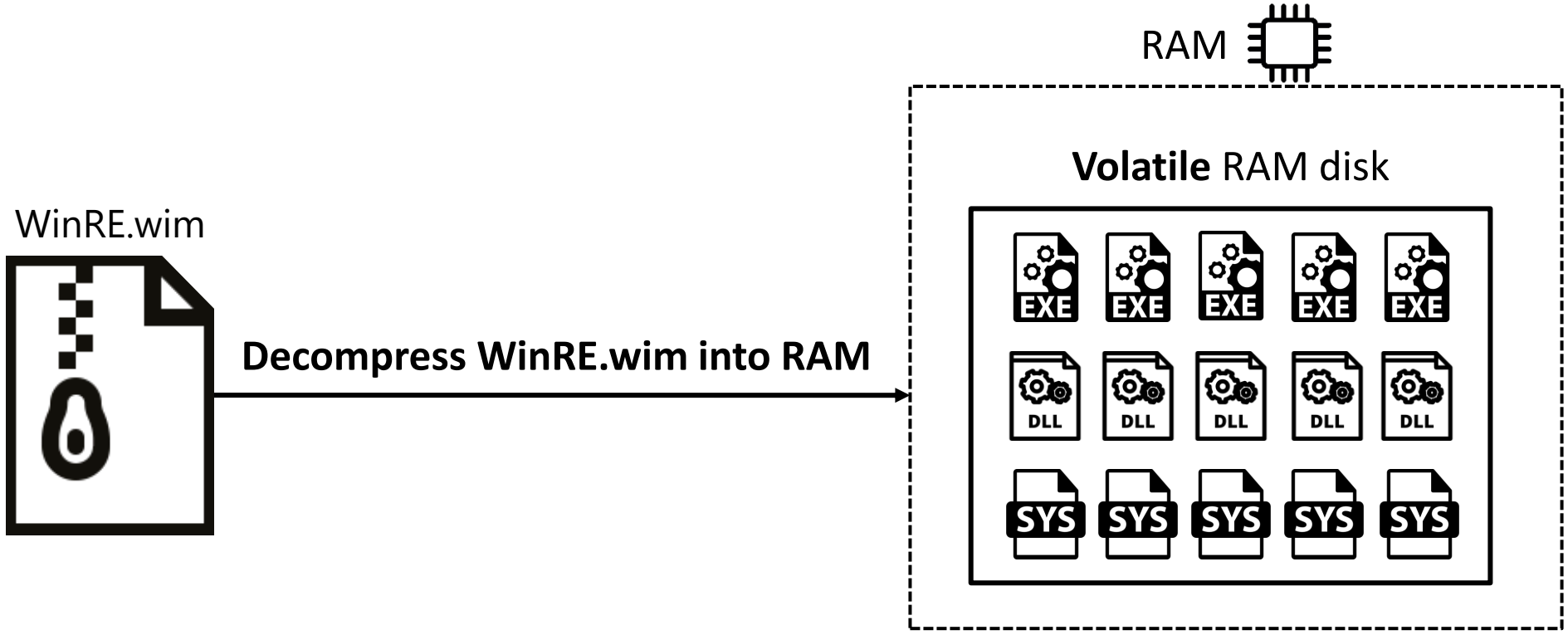
WinRE Architecture – WinRE.wim

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



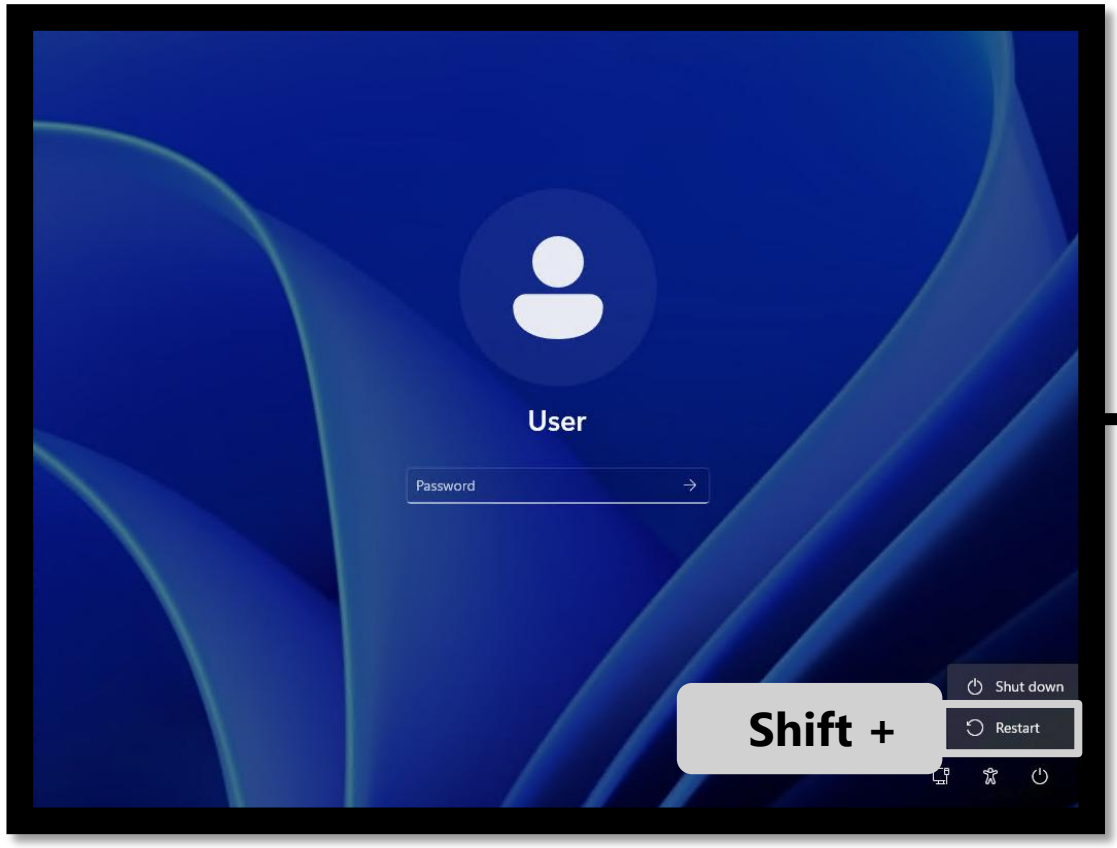
WinRE Architecture – RAM disk boot

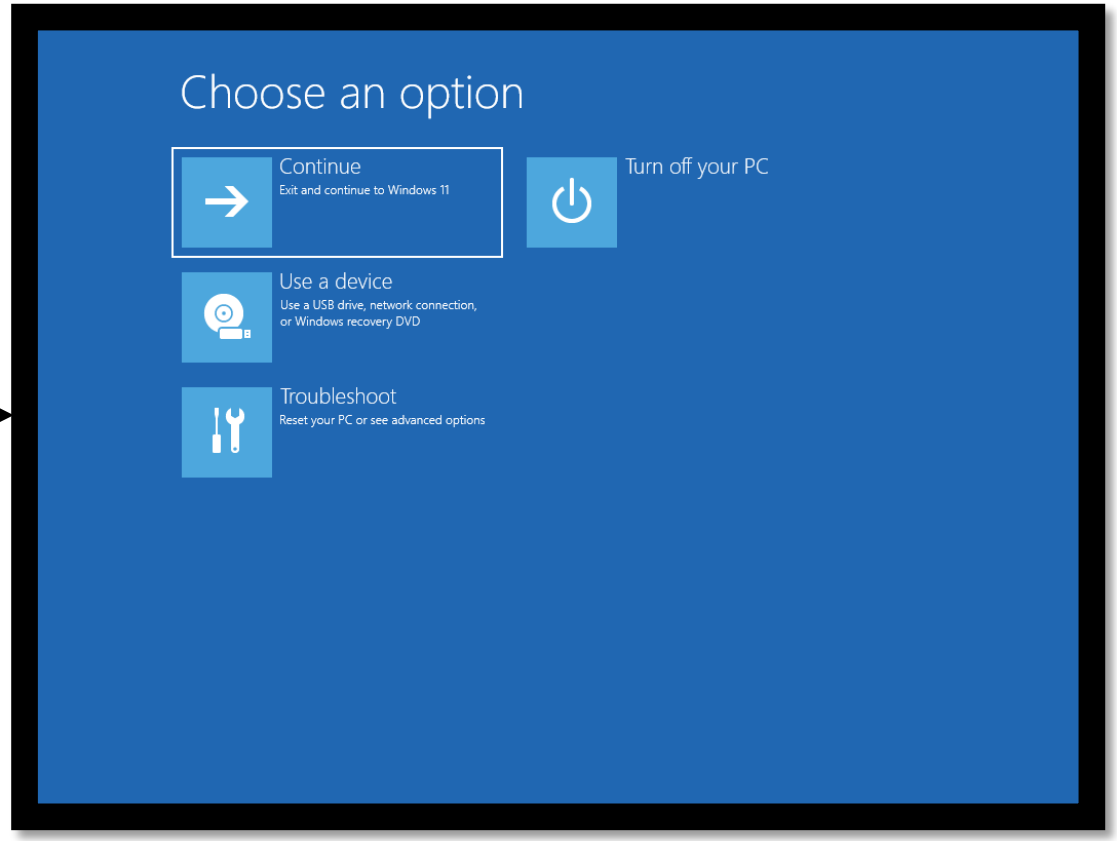
- WinRE.wim is booted from RAM disk
- Changes to RAM disk are never committed back to WinRE.wim



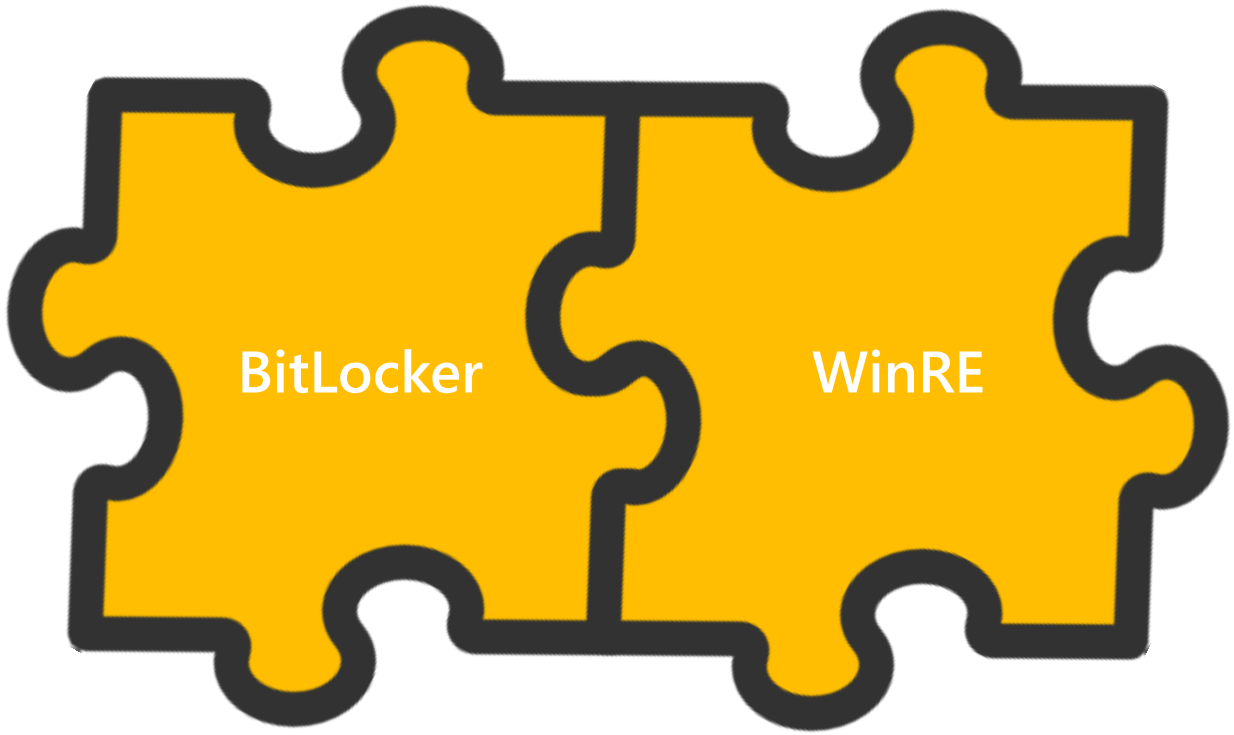
WinRE Interface

- A physical attacker can boot into WinRE by pressing
  - Shift + Restart from the logon screen



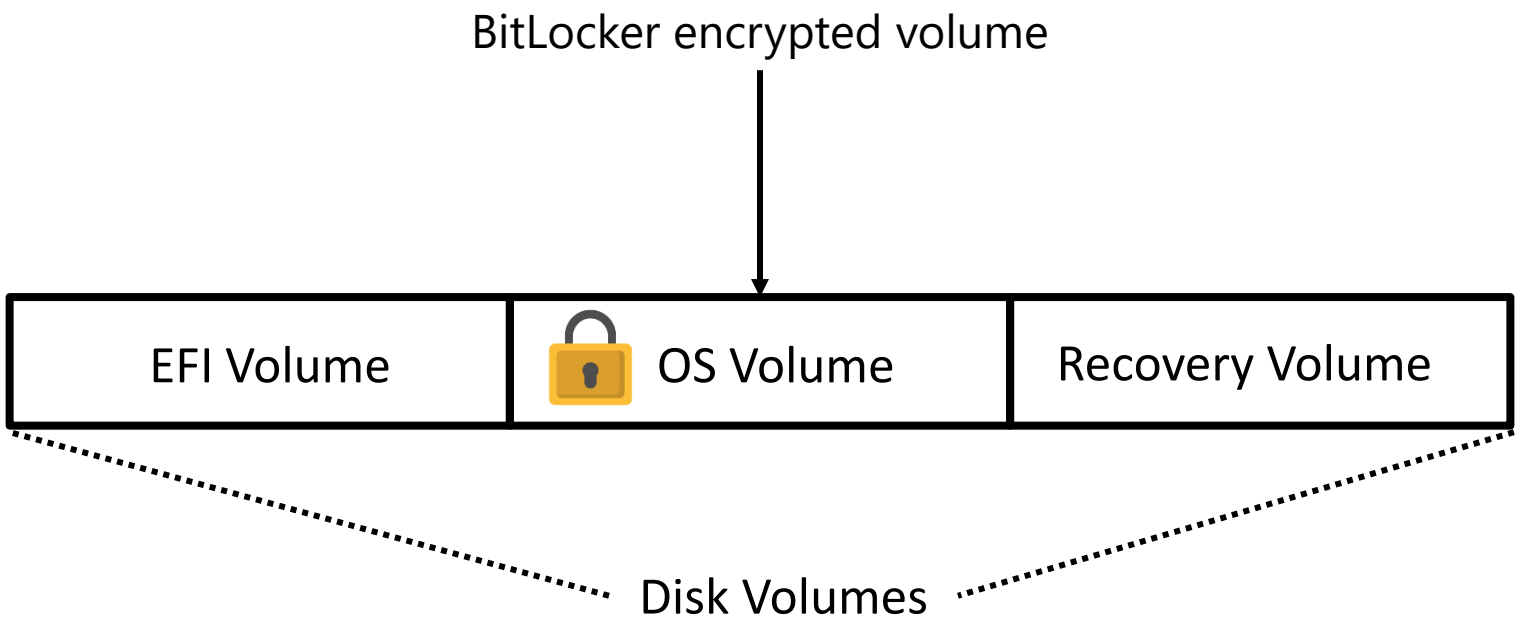


WinRE and BitLocker Intersection



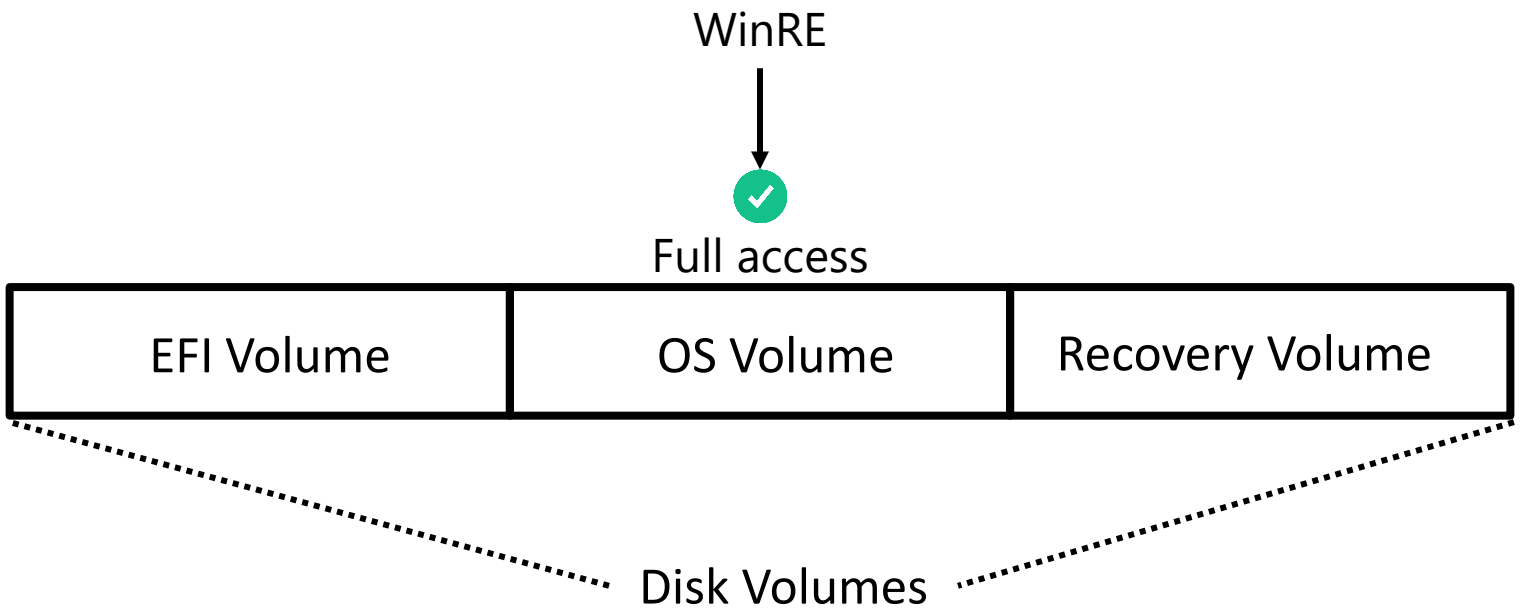
WinRE and BitLocker Intersection

- BitLocker is a Full Volume Encryption (FVE) technology



WinRE and BitLocker Intersection

- If certain conditions are met – WinRE has full access to the OS volume

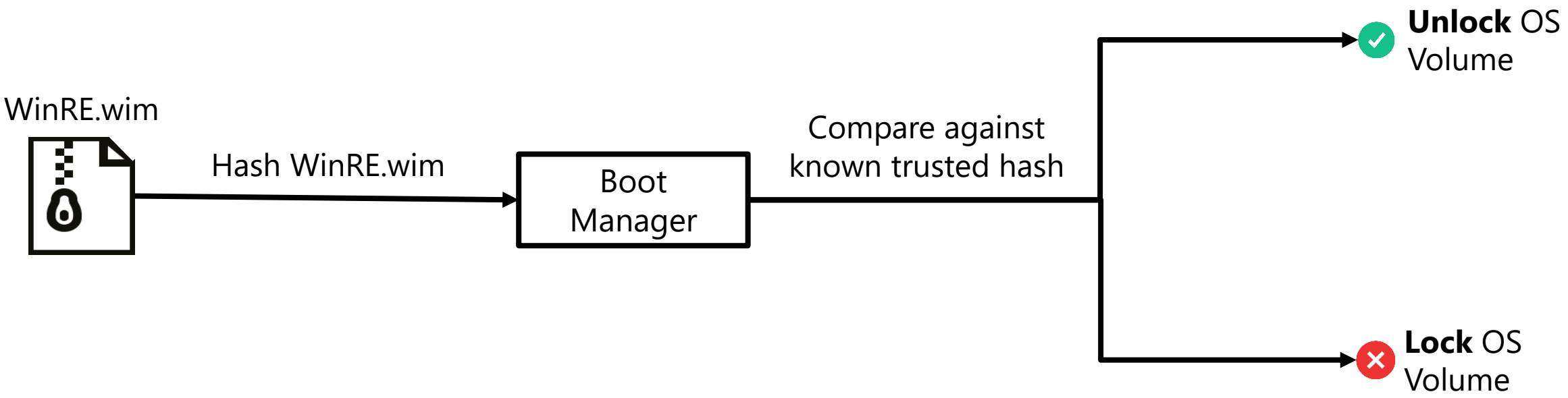


WinRE OS Volume Access Conditions

1. WinRE.wim it trusted
2. Overly permissive recovery tools aren’t selected

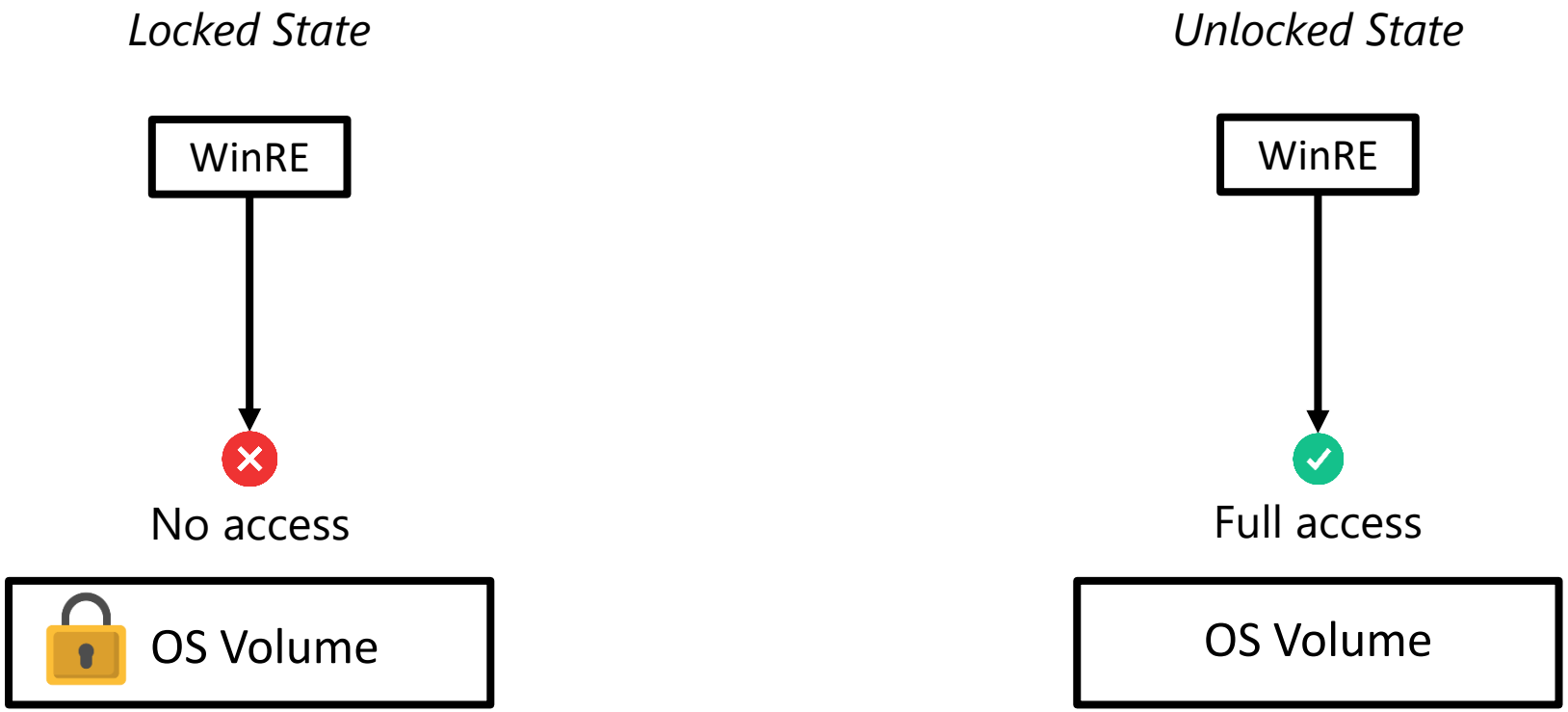
Condition #1 – WinRE.wim is Trusted

- Trusted WIM boot compares the WinRE.wim hash against a known trusted hash



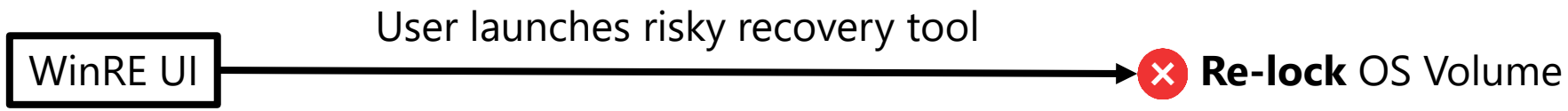
Condition #1 – WinRE.wim is Trusted

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



Condition #2 – Risky Recovery Tools

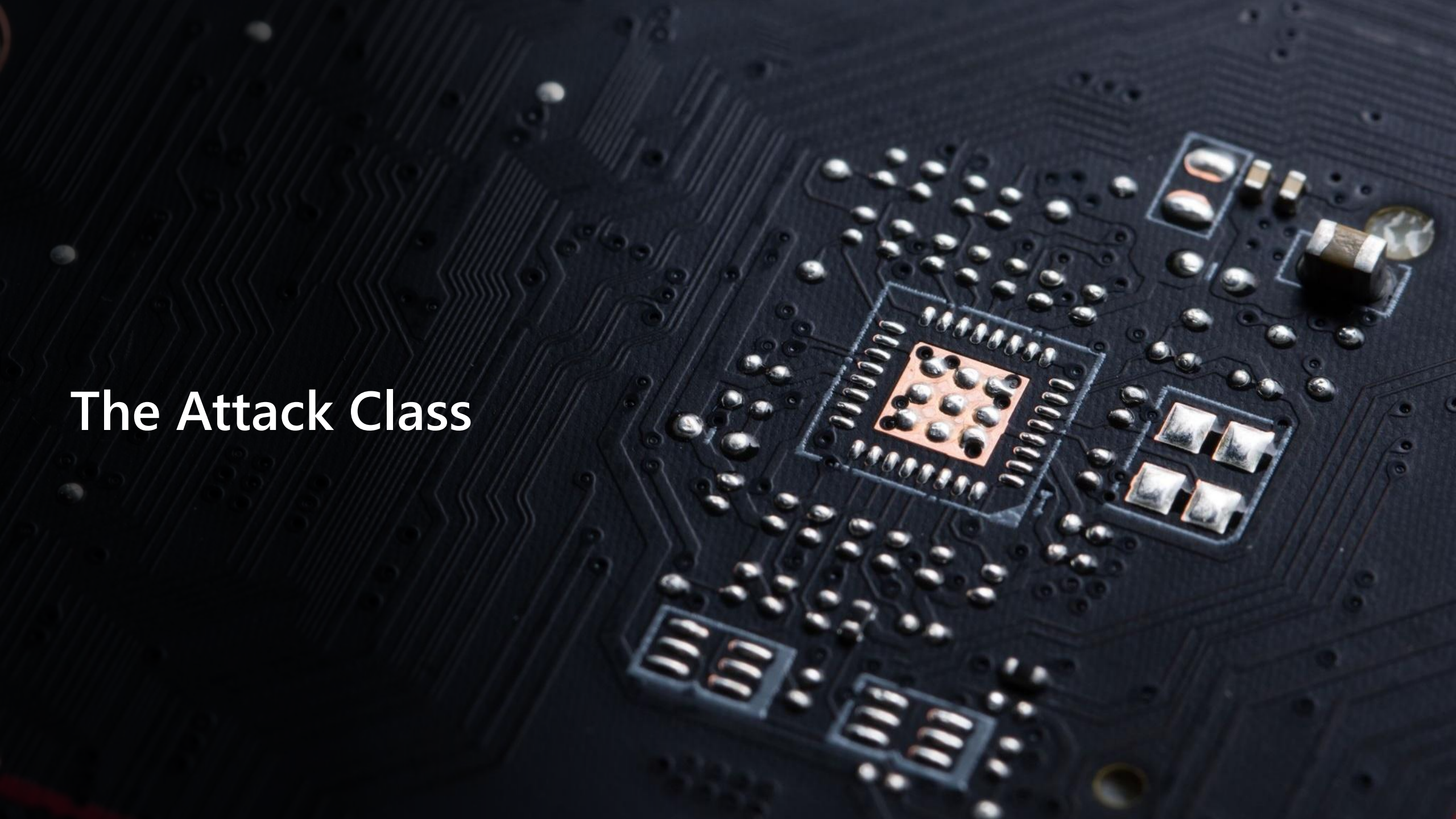
- If a risky recovery tool is selected from the recovery UI – the OS volume is re-locked
  - For Example: Command Prompt Tool



WinRE Research Guidelines

- Static WinRE.wim image cannot be modified – due to trusted WIM boot
- Recovery tools that trigger re-lock are out-of-scope

The Attack Class

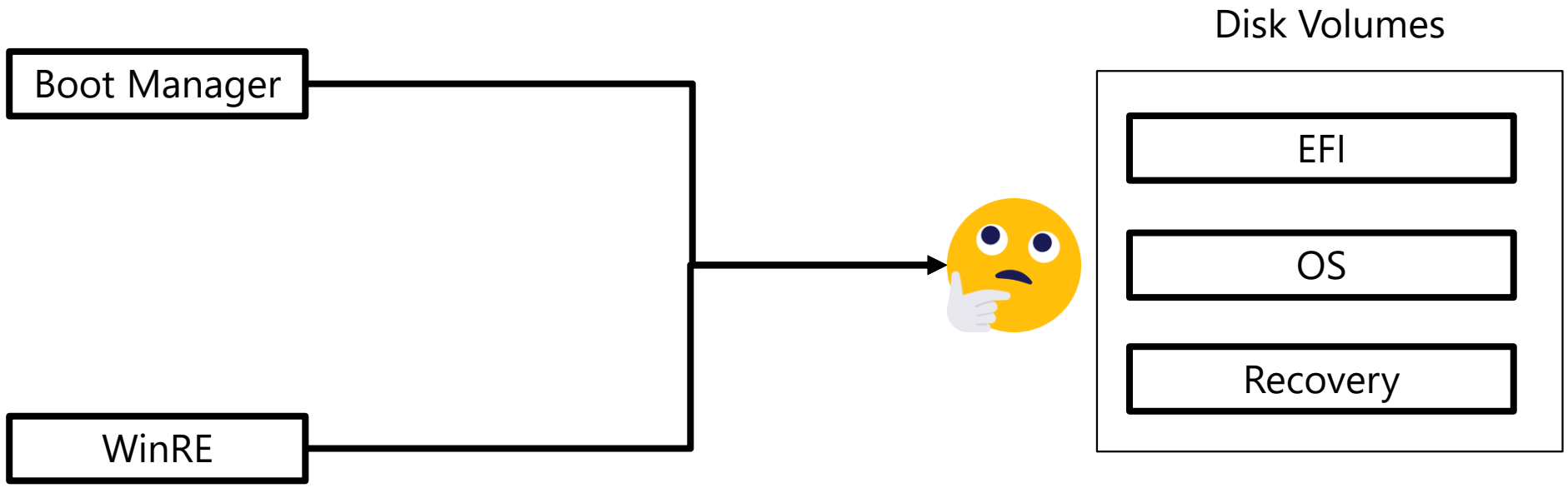


The Attack Class – Background

- WinRE places full trust in the target OS volume it is meant to recover

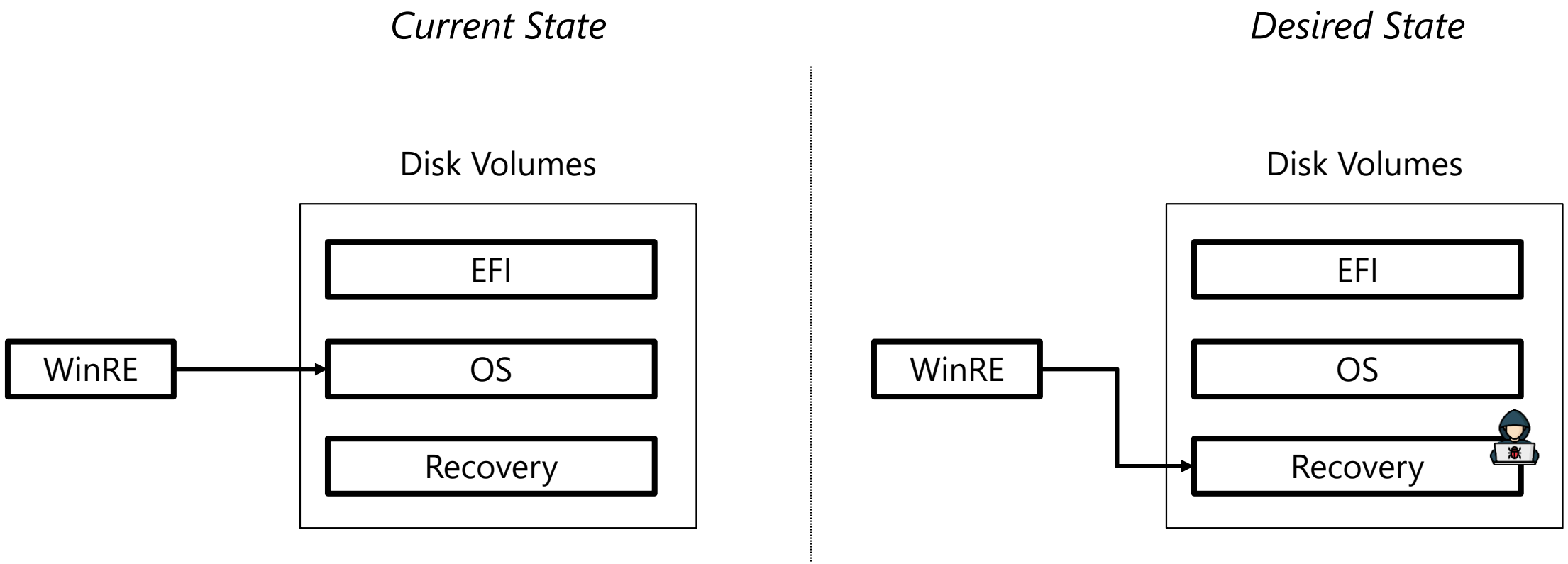
The Attack Class – Lookups

- Boot phase – looks for the volume to unlock
- WinRE phase – looks for the volume to recover
- The lookup logics aren't shared



The Attack Class – Goal

- Confuse WinRE to recover an attacker-controlled volume



The Attack Class – Constraints

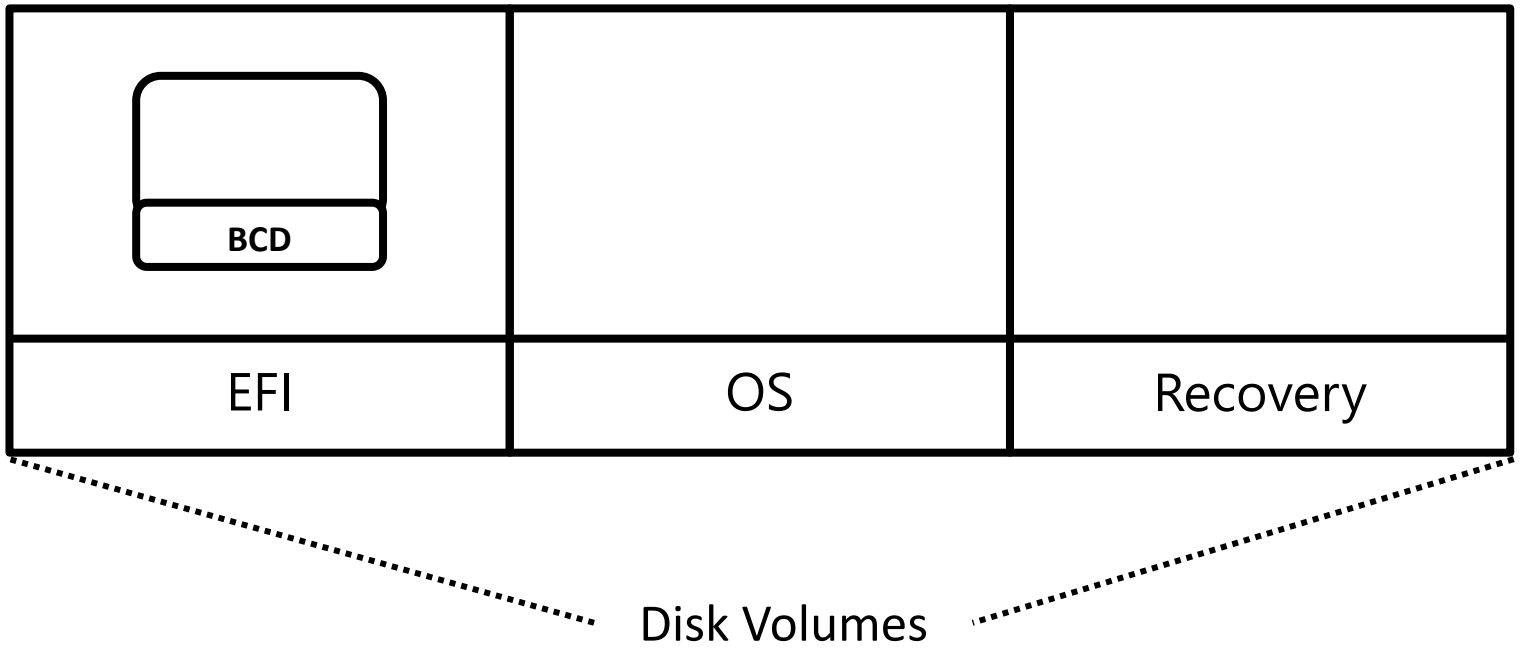
1. Boot phase must always unlock the BitLocker-encrypted volume
2. WinRE phase must recover an attacker-controlled volume

The Attack Class – Steps

1. Find suitable confusion primitives
2. Find a way to further exploit these primitives to bypass BitLocker

Boot Phase OS Lookup Implementation – BCD-based

- Lookups are based on Boot Configuration Data (BCD) queries
- BCD represents Windows Boot configuration
- BCD is stored on the EFI volume



BCD Overview

- The BCD contains a list of objects identified by GUID
- Each object contains elements as key-value pairs
- Element values format can be:
  - Device
  - String
  - GUID
  - GUID list
  - Integer
  - Integer list
  - Boolean

BCD Overview

- BCDEdit.exe can be used to query and set BCD

```
C:\Windows\System32>bcdedit /enum {f227cc65-aa4d-11f0-806c-28a06b34d7de} /v

Windows Boot Loader
-------------------
identifier              {f227cc65-aa4d-11f0-806c-28a06b34d7de}
device                  partition=C:
path                    \WINDOWS\system32\winload.efi
description             Windows 11
locale                  en-US
inherit                 {6efb52bf-1766-41db-a6b3-0ee5eff72bd7}
recoverysequence        {f227cc67-aa4d-11f0-806c-28a06b34d7de}
displaymessageoverride  Recovery
recoveryenabled         Yes
isolatedcontext         Yes
allowedinmemorysettings 0x15000075
osdevice                partition=C:
systemroot              \WINDOWS
resumeobject            {f227cc64-aa4d-11f0-806c-28a06b34d7de}
nx                      OptIn
bootmenupolicy          Standard
hypervisorlaunchtype    Auto
claimedtpmcounter       0x10002
```

Boot Phase OS Lookup Implementation

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	{main_os}

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	partition=C:
Recovery Sequence	{winre_os}

Booted entry

identifier	{winre_os}
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
0x11000083	N/A

1. bootmgfw.efi unlocks the device pointed by the trustedosdevice (custom:0x11000083) element if it exists

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	{main_os}

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	partition=C:
Recovery Sequence	{winre_os}

Booted entry

identifier	{winre_os}
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	**N/A**

2. bootmgfw.efi iterates over {bootmgr} object displayorder element entries

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	**{main_os}**

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	partition=C:
Recovery Sequence	{winre_os}

Booted entry

identifier	{winre_os}
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	N/A

3. bootmgfw.efi verifies that the iterated entry has a recoverysequence element that matches the {winre_os}

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	{main_os}

Iterated entry

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	partition=C:
Recovery Sequence	**{winre_os}**

Booted entry

identifier	**{winre_os}**
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	N/A

4. bootmgfw.efi verifies that the iterated entry has a device, path and description element

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	{main_os}

Iterated entry

identifier	{main_os}
device	**partition=C:**
path	**\winload.efi**
description	**Windows 11**
osdevice	partition=C:
Recovery Sequence	{winre_os}

Booted entry

identifier	{winre_os}
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	N/A

5. bootmgfw.efi unlocks the iterated entry’s osdevice if it exists

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	{main_os}

Iterated entry

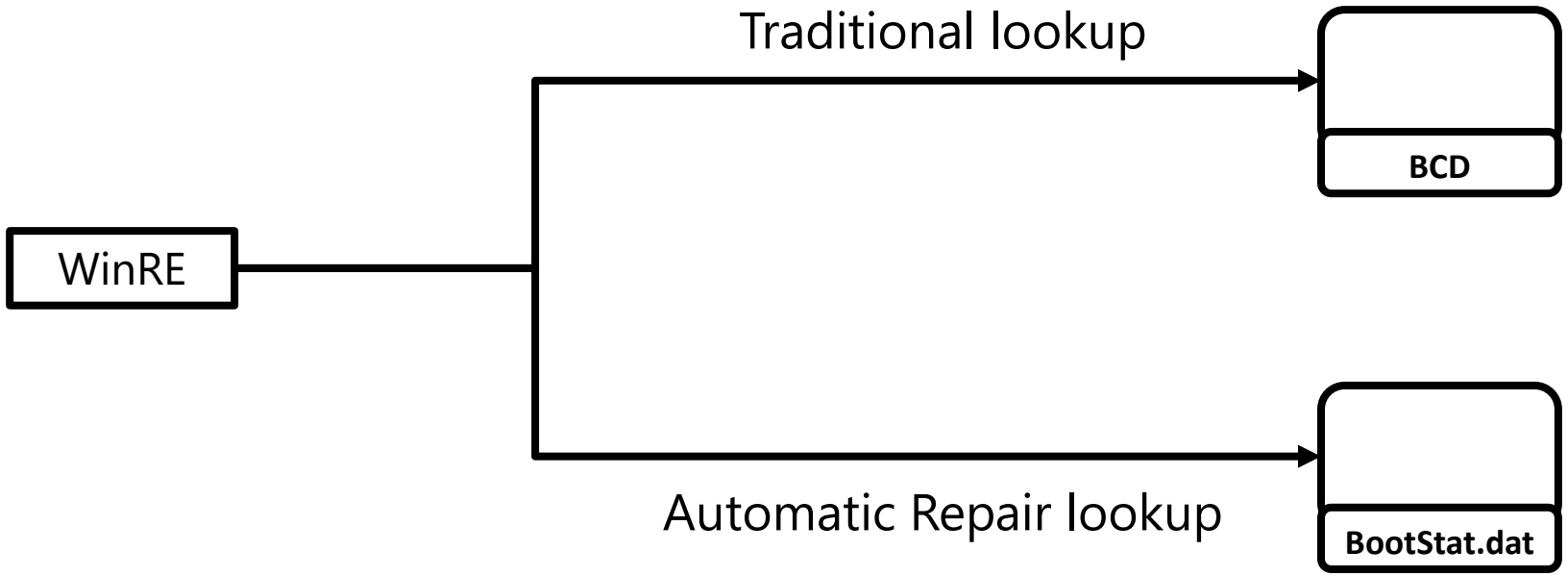
identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	**partition=C:**
Recovery Sequence	{winre_os}

Booted entry

identifier	{winre_os}
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	N/A

WinRE Phase OS Lookup Implementation

- WinRE lookups differ based on the WinRE state -
  - Traditional – BCD based
  - Automatic Repair – BootStat.dat based



WinRE Phase OS Lookup Implementation – Traditional

- WinRE Iterate over all volumes and searches each one for the BCD store
- FindFirstVolume and FindNextVolume are used for volume iteration
- Further BCD lookups are performed on the read store

```
hVolume = FindFirstVolumeW(VolumePath, MAX_PATH);

do {

    AppendPath(VolumePath, "\\EFI\\Microsoft\\Boot\\BCD", &StorePath);

    if (PathExist(StorePath)) {

        // [FURTHER LOOKUPS ON STORE]

    }

} while (FindNextVolume(hVolume, VolumePath, MAX_PATH));
```

1. WinRE iterates over {bootmgr} object displayorder element entries

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	**{main_os}**

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	partition=C:
Recovery Sequence	{winre_os}

Booted entry

identifier	{winre_os}
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	N/A

2. WinRE verifies that the iterated entry has a recoverysequence element that matches the {winre_os}

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	{main_os}

Iterated entry

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	partition=C:
Recovery Sequence	**{winre_os}**

Booted entry

identifier	**{winre_os}**
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	N/A

3. WinRE considers the iterated entry's osdevice as the target volume to recover

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	{main_os}

Iterated entry

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	**partition=C:**
Recovery Sequence	{winre_os}

Booted entry

identifier	{winre_os}
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	N/A

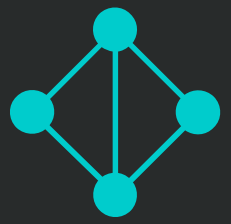
Boot vs WinRE BCD Lookups Key Differences

1. WinRE doesn’t directly use the BCD store from the EFI volume
2. WinRE doesn’t account for the trustedosdevice element
3. WinRE doesn’t filter objects without device, path or description elements

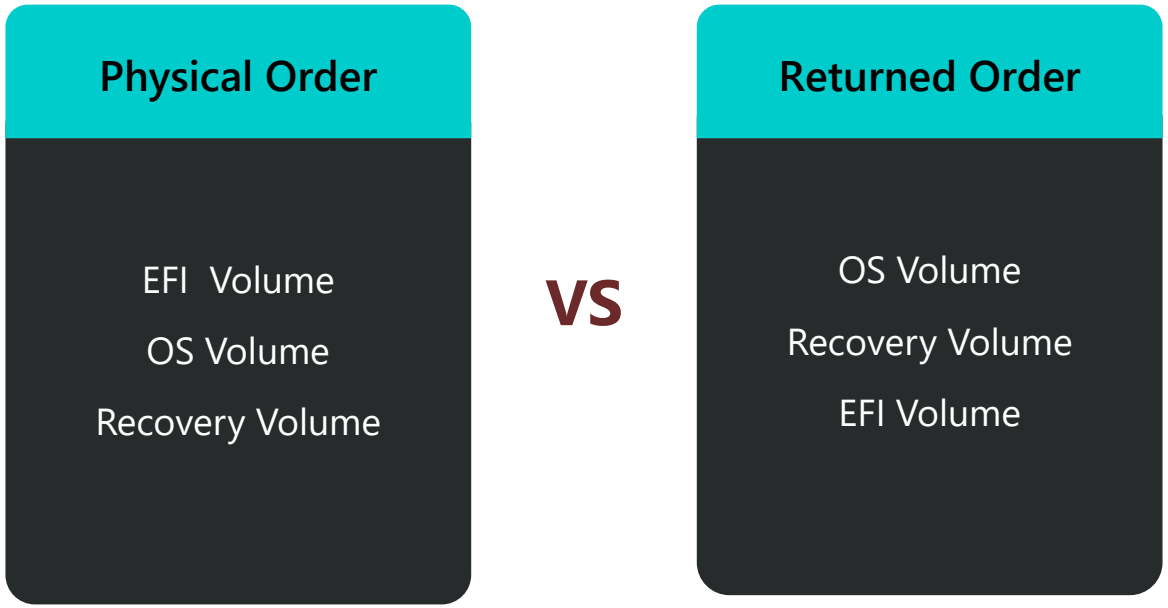
Confusion Primitive – First Instance

FindFirstVolume and FindNextVolume Remarks Section from MSDN

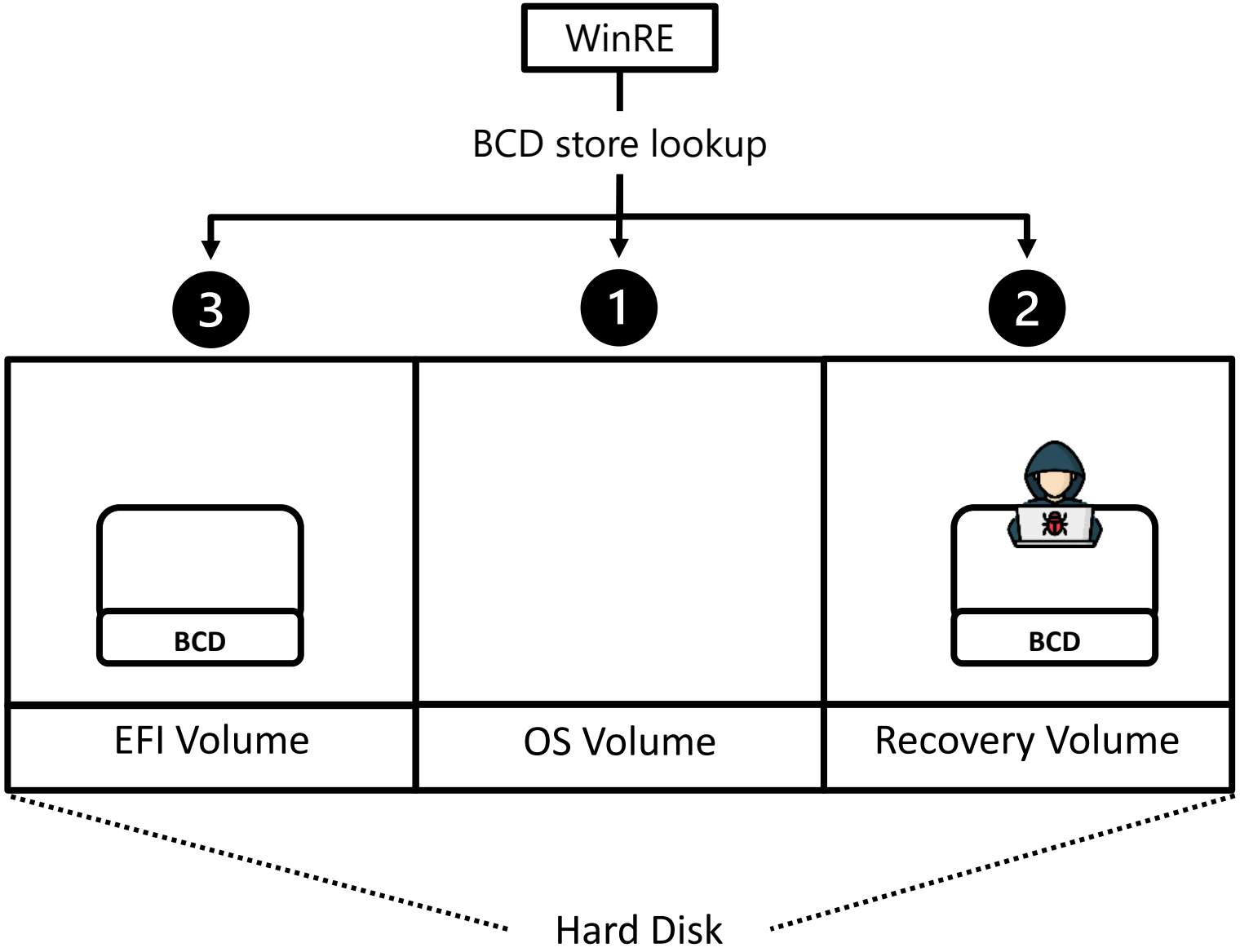
"You should not assume any correlation between the order of the volumes that are returned by these functions and the order of the volumes that are on the computer [...]"



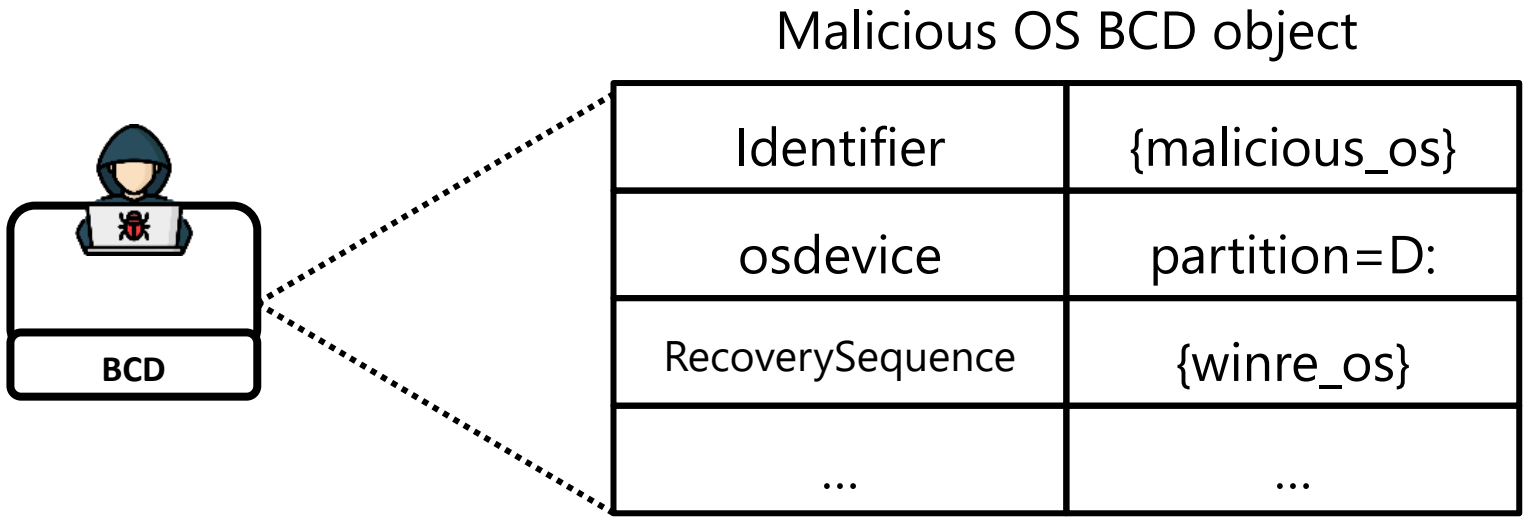
Confusion Primitive – First Instance



Confusion Primitive – First Instance



Confusion Primitive – First Instance



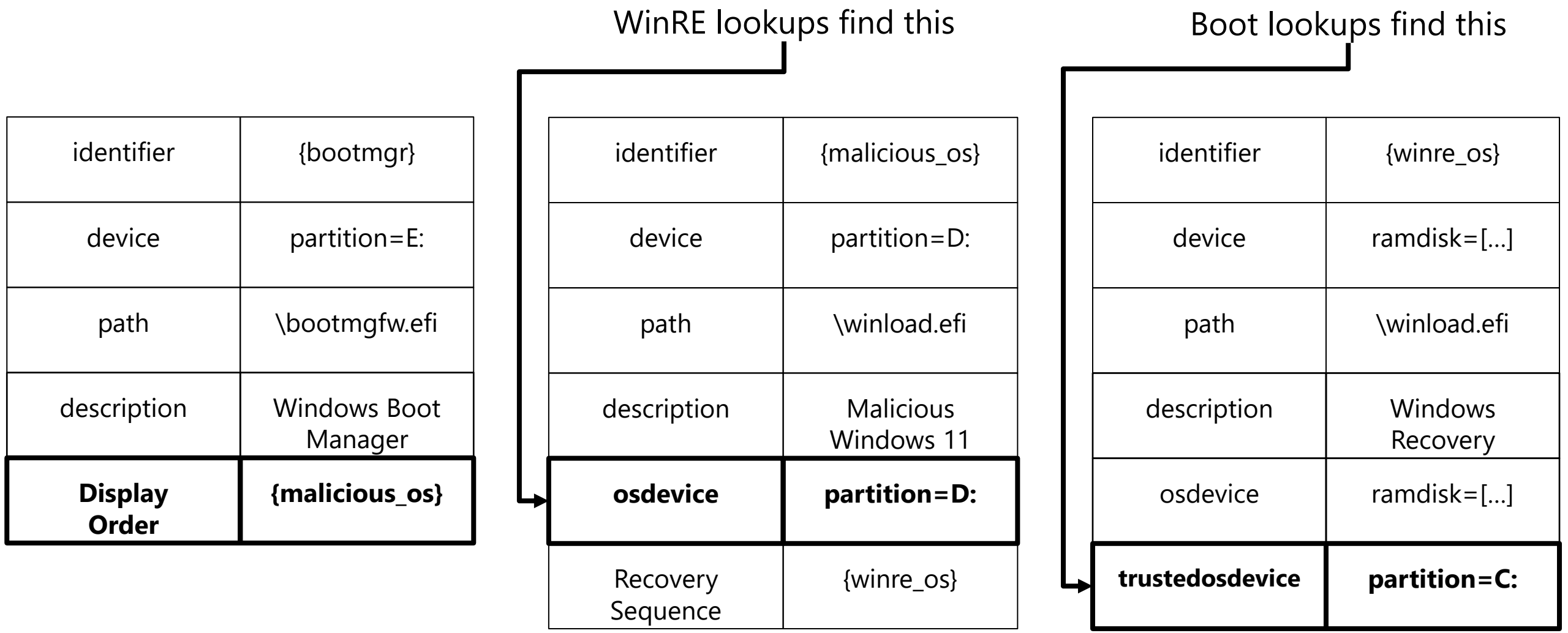
Confusion Primitive – Second Instance

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	**{malicious_os}**

identifier	{malicious_os}
device	partition=D:
path	\winload.efi
description	Malicious Windows 11
osdevice	**partition=D:**
Recovery Sequence	{winre_os}

identifier	{winre_os}
device	ramdisk=[…]
path	\winload.efi
description	Windows Recovery
osdevice	ramdisk=[…]
trustedosdevice	**partition=C:**

Confusion Primitive – Second Instance



Confusion Primitive – Third Instance

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	**{malicious_os}, {main_os}**

identifier	{malicious_os}
device	partition=D:
path	\winload.efi
~~description~~	~~Malicious Windows 11~~
osdevice	**partition=D:**
Recovery Sequence	{winre_os}

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	partition=C:
Recovery Sequence	{winre_os}

Confusion Primitive – Third Instance

identifier	{bootmgr}
device	partition=E:
path	\bootmgfw.efi
description	Windows Boot Manager
Display Order	**{malicious_os}, {main_os}**

WinRE lookups find this

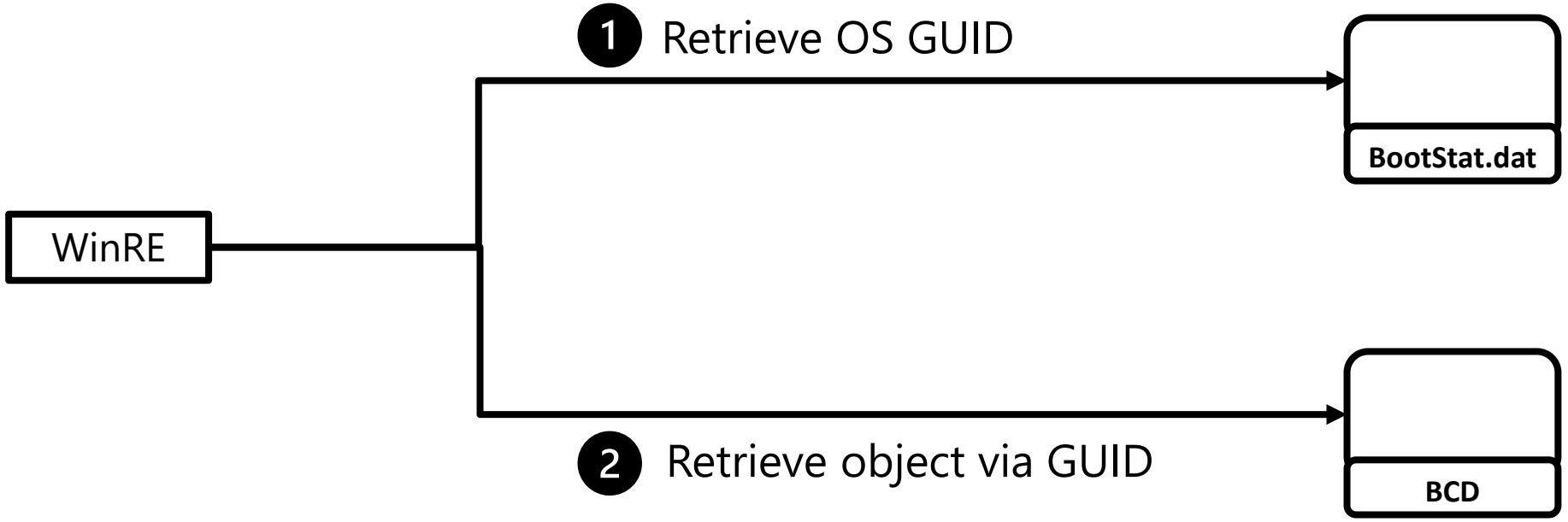
identifier	{malicious_os}
device	partition=D:
path	\winload.efi
~~description~~	~~Malicious Windows 11~~
osdevice	**partition=D:**
Recovery Sequence	{winre_os}

Boot lookups find this

identifier	{main_os}
device	partition=C:
path	\winload.efi
description	Windows 11
osdevice	partition=C:
Recovery Sequence	{winre_os}

WinRE Phase OS Lookup Implementation – Repair

- In WinRE Automatic Repair state – OS volume lookups are based on BootStat.dat

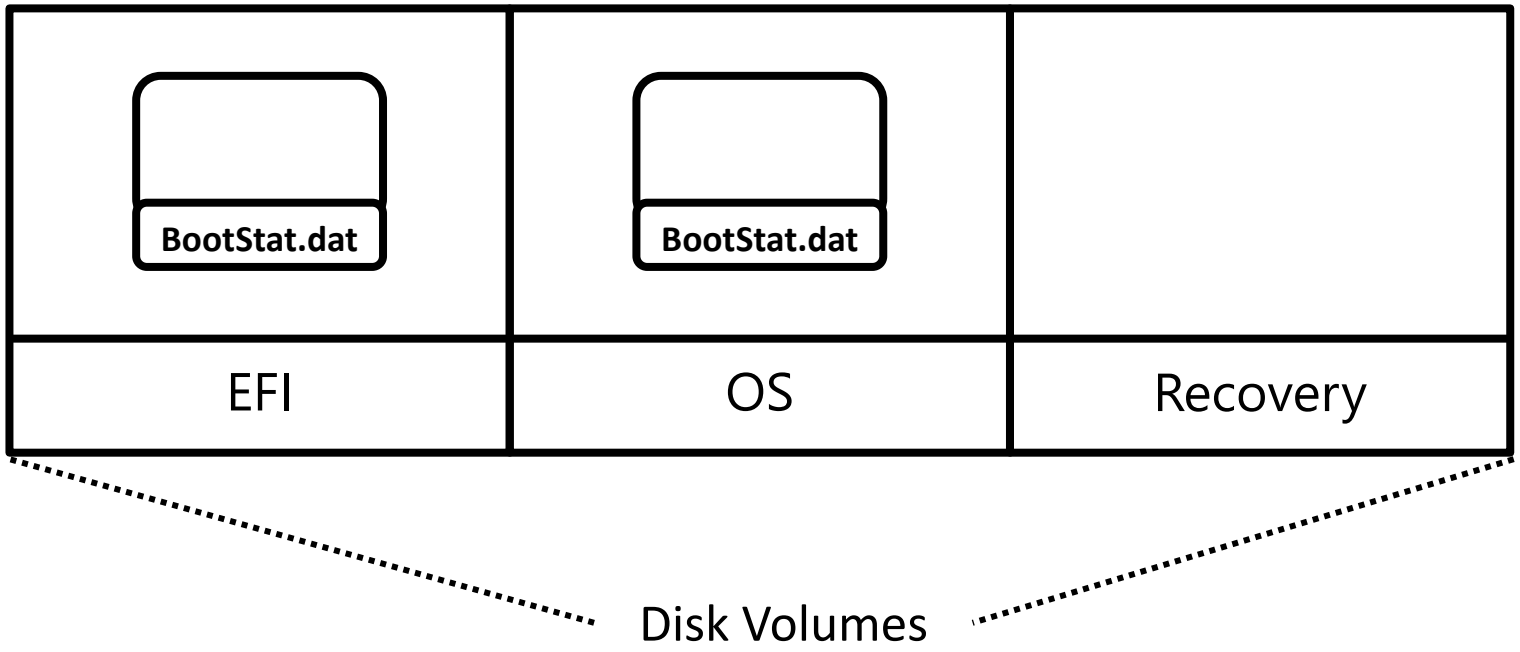


BootStat.dat Overview

- BootStat.dat is Windows's Boot Status Data log file
- Boot components insert events into the log file
- Runtime components can analyze this log file to learn about the boot events

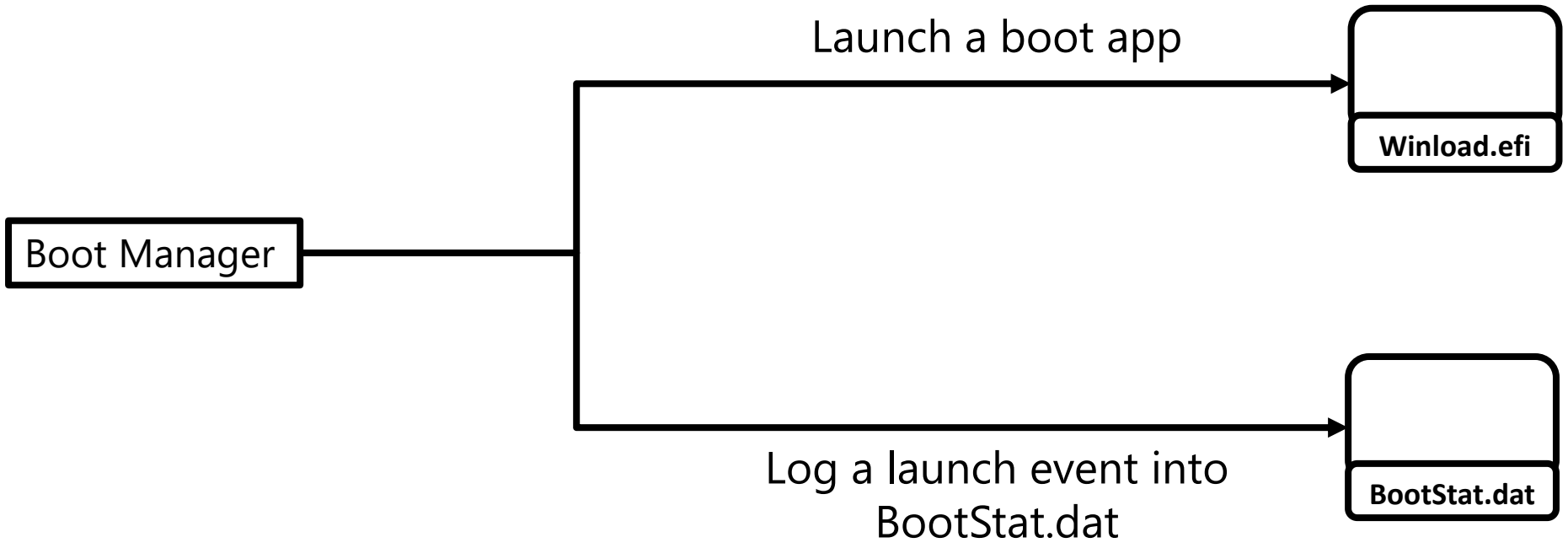
BootStat.dat Instances

- There are two BootStat.dat instances:
  - Boot Manager instance – located in the EFI volume
  - OS Loader instance – located in the OS volume, but configurable
- The lookups are based on the Boot Manager instance



BootStat.dat Boot App Launch Events

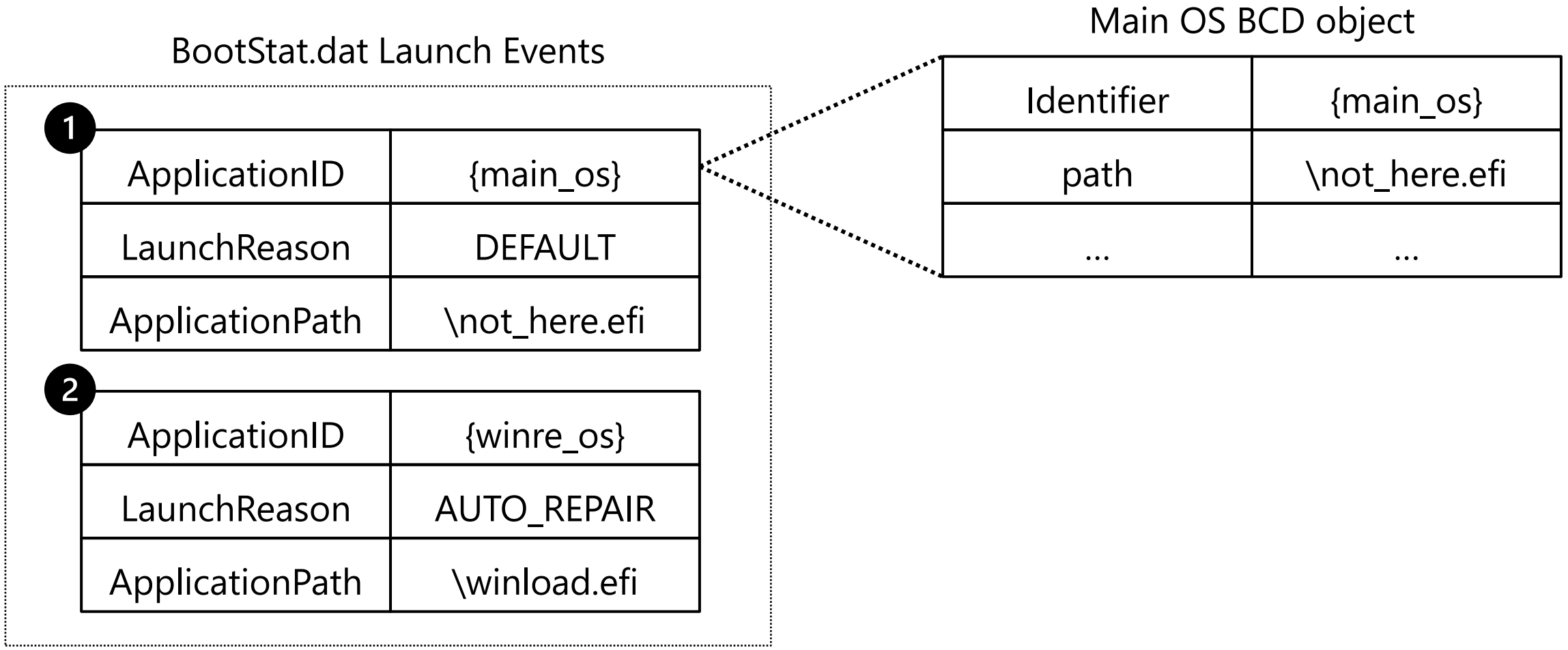
- Bootmgfw.efi logs any boot app launch into BootStat.dat as launch events



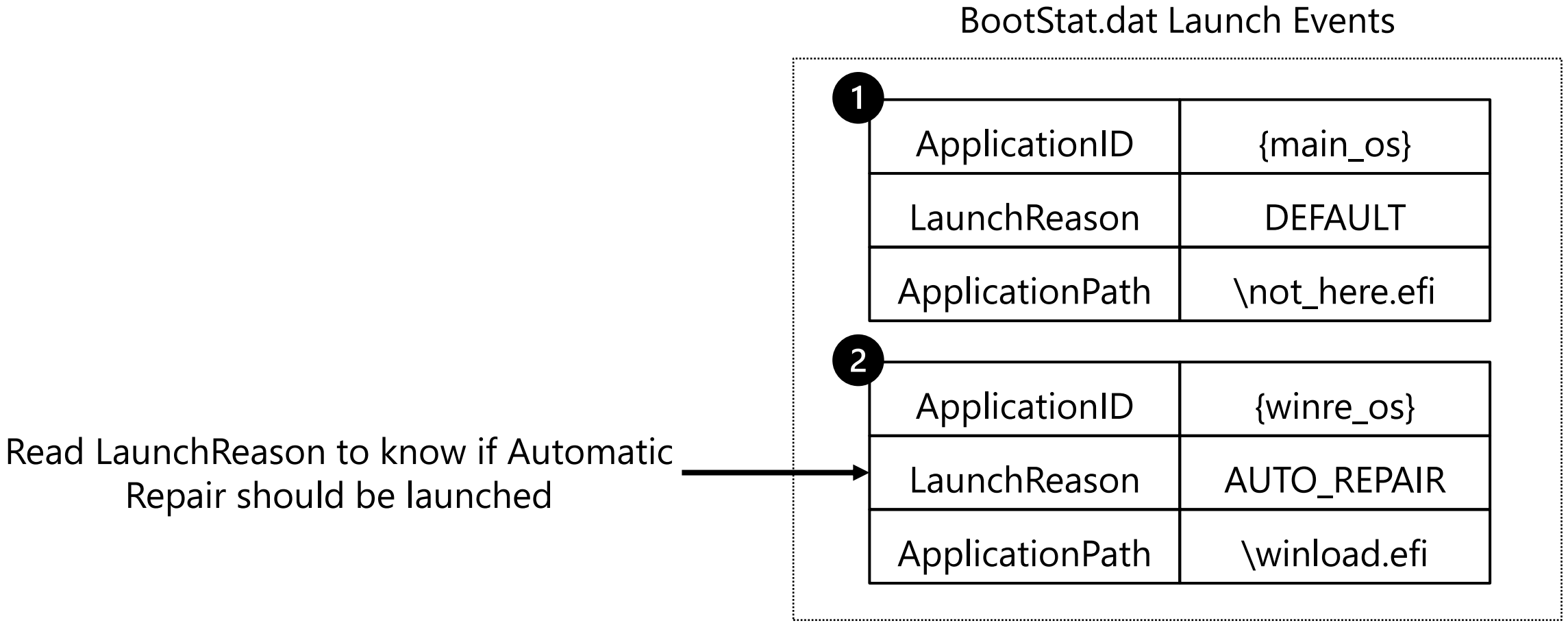
BootStat.dat Boot App Launch Events Format

```
typedef struct _BOOTSTAT_BOOT_APP_LAUNCH_EVENT {

    GUID ApplicationId;                         // offset: 0x00
    ULONG LaunchReason;                        // offset: 0x10
    WCHAR ApplicationPath[ANYSIZE_ARRAY];    // offset: 0x14

} BOOTSTAT_BOOT_APP_LAUNCH_EVENT, *PBOOTSTAT_BOOT_APP_LAUNCH_EVENT;
```

Auto-Repair Launch Events

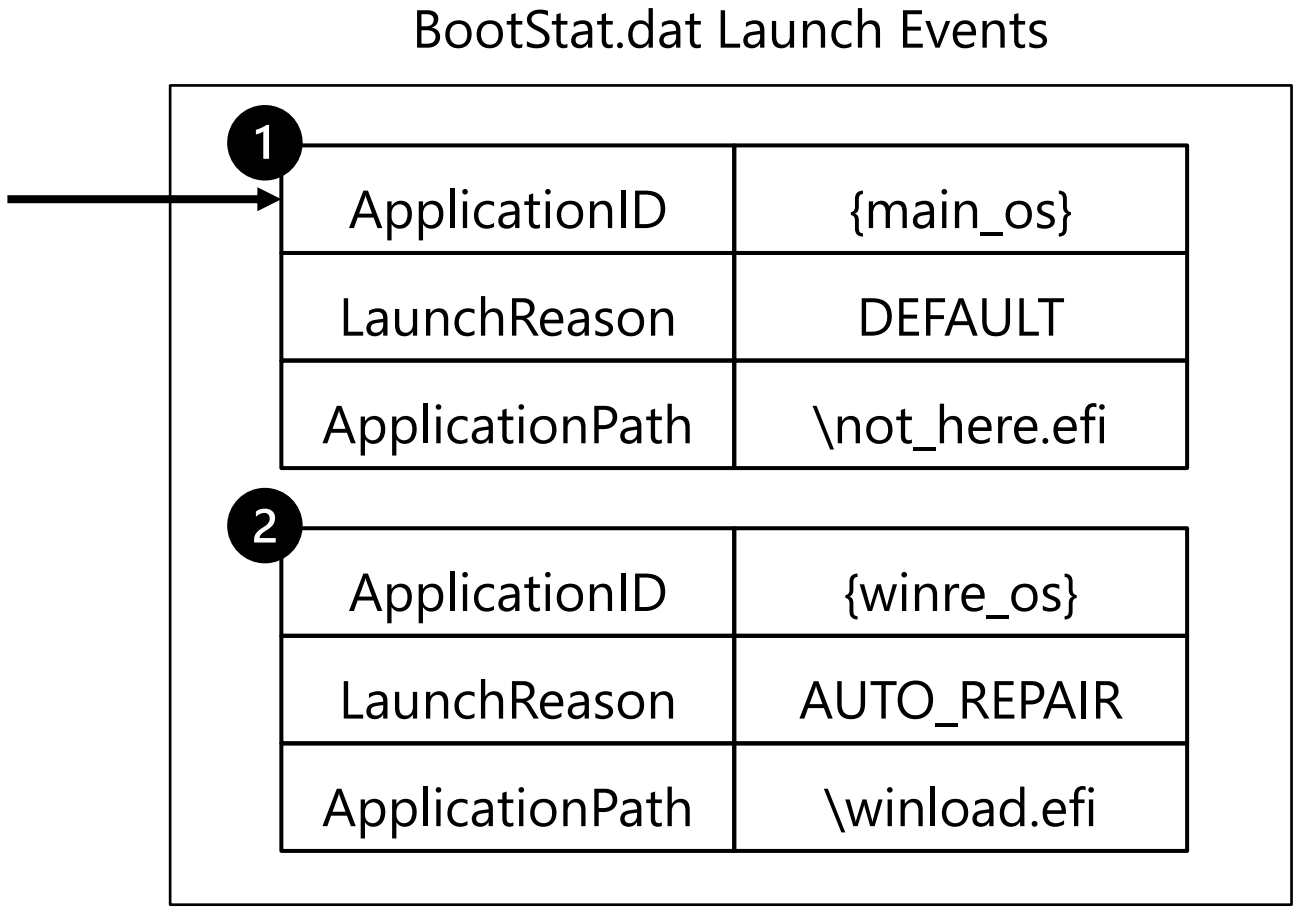


What WinRE Reads From BootStat.dat



What WinRE Reads From BootStat.dat

Read ApplicationID to know which BCD object is the one that needs a repair



Controlling Launch Events - The Challenge

- Bootmgfw.efi writes new launch events to BootStat.dat every boot
- WinRE only uses the two most recent ones

Controlling BootStat.dat – Solution Ideas

1. Find a way to prevent bootmgfw.efi from writing new events every boot
2. Find a boot sequence that once executed by bootmgfw.efi shapes the BootStat.dat file as desired

Preventing Bootmgfw.efi From Writing New Events

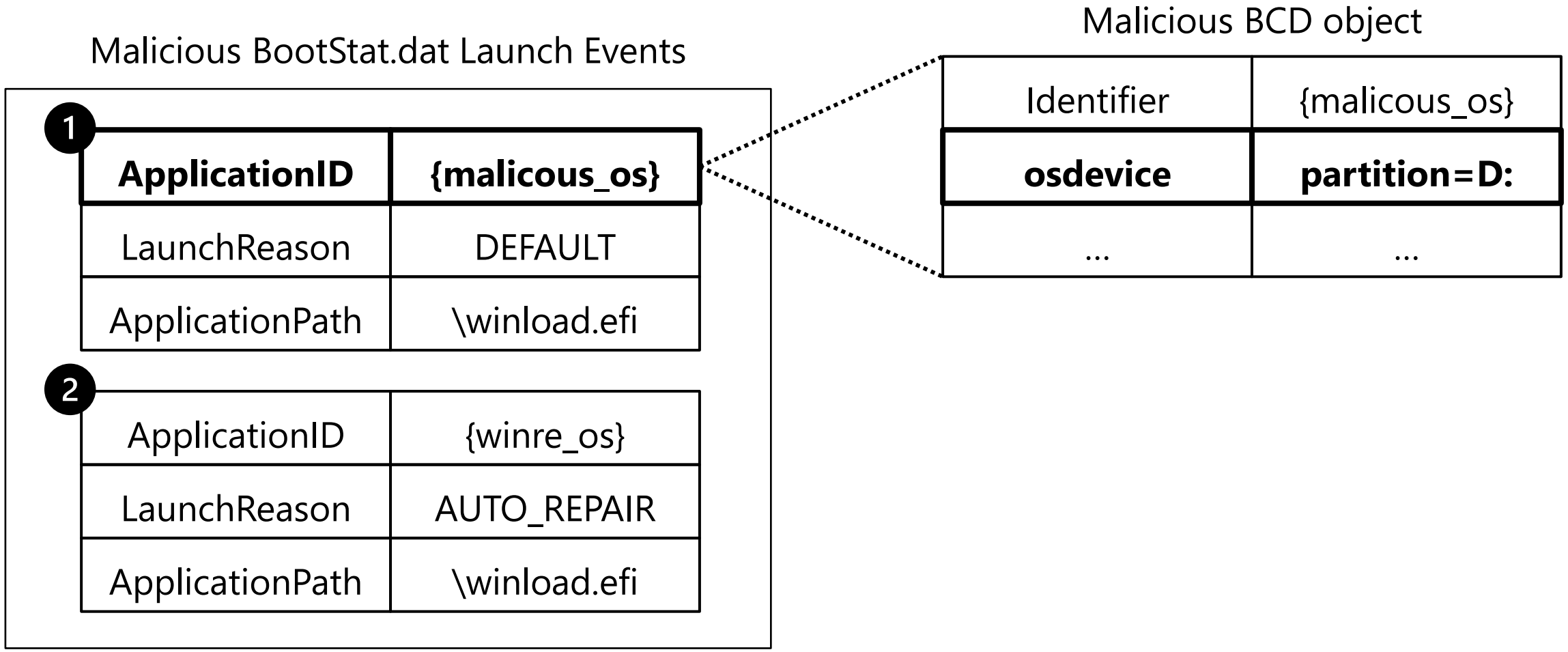
- bootmgfw.efi won't log events to BootStat.dat if its size isn't 65536
- WinRE runtime doesn't implement this size check – and considers all sizes valid

```
if (FileSize != 65536) {
    Status = STATUS_INVALID_PARAMETER;
    goto InitializeLogEnd;
}
```

Controlling BootStat.dat

- Craft attacker-controlled BootStat.dat
- Ensure its size isn't 65536
- The result –
  - Bootmgfw.efi doesn't log any events to BootStat.dat
  - WinRE consumes the attacker-controlled BootStat.dat as is

Confusion Primitive – Fourth Instance



Exploit Flow Requirement

Find a WinRE flow that:

Can be triggered by an attacker

Does not trigger the relock functionality

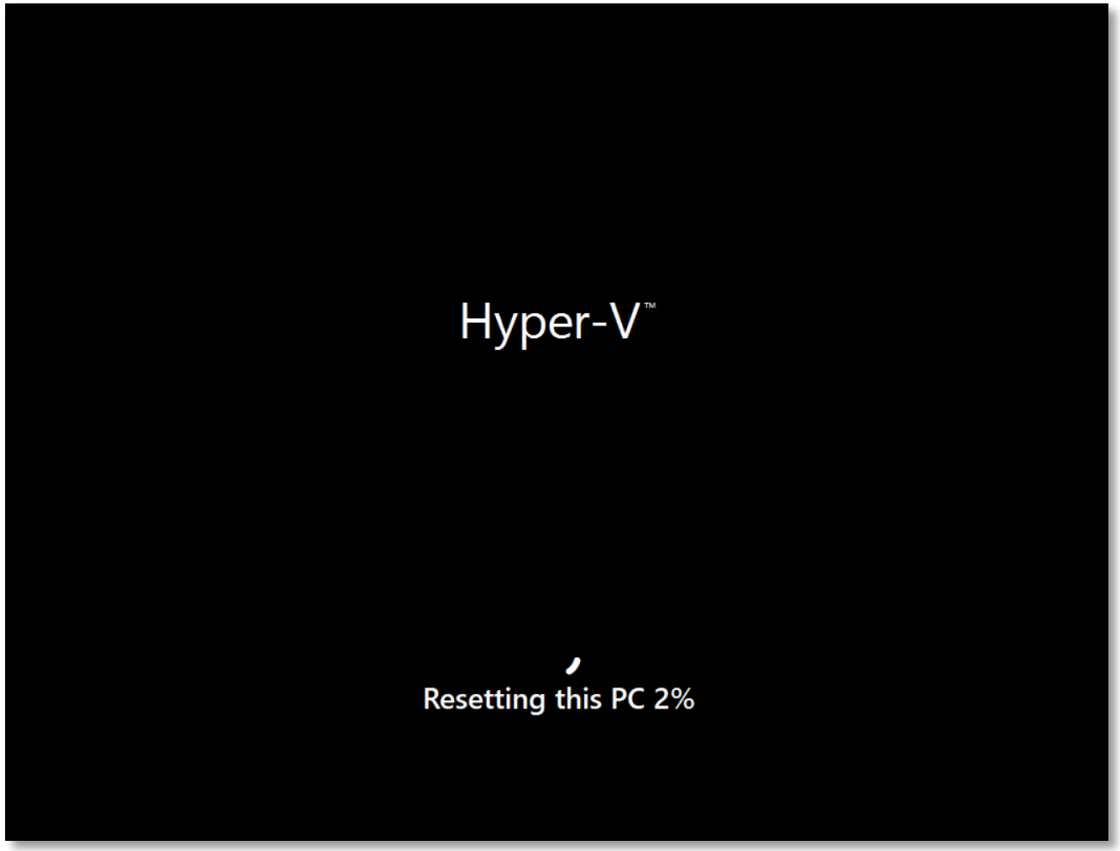
Queries configuration from the target OS to perform sensitive operations

Potential Candidates

1. Push Button Reset (PBR)
2. Startup Repair Tool (SRT)

Push Button Reset (PBR)

- Push Button Reset – Windows's System Reset Tool



Online PBR Is Applicable

- Online PBR meets all exploit requirements

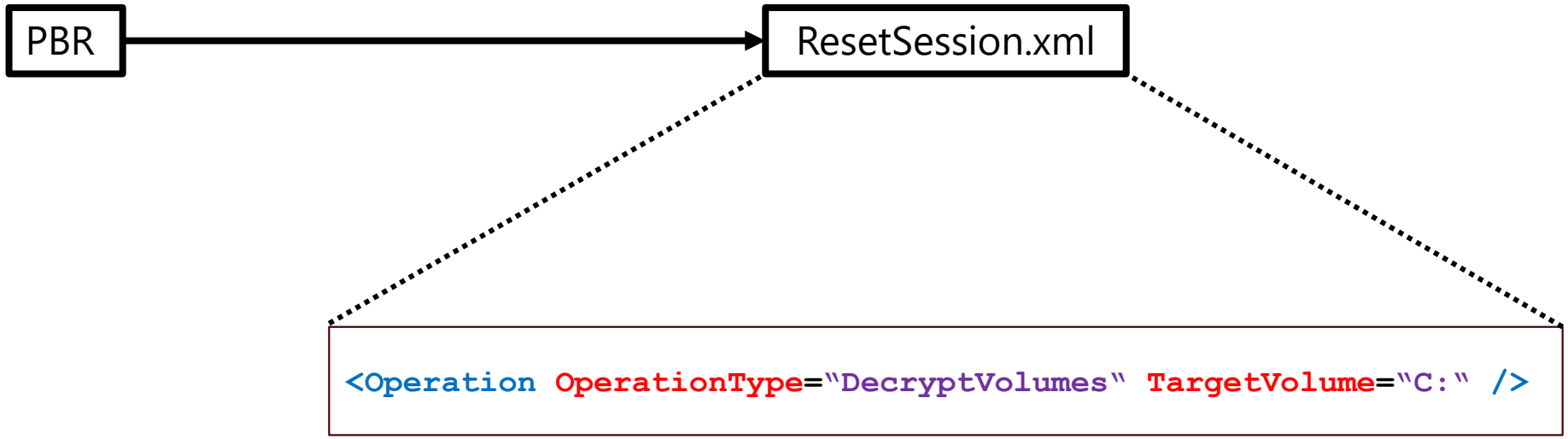
Can be triggered by an attacker

Does not trigger the relock functionality

Queries configuration from the target OS to perform sensitive operations

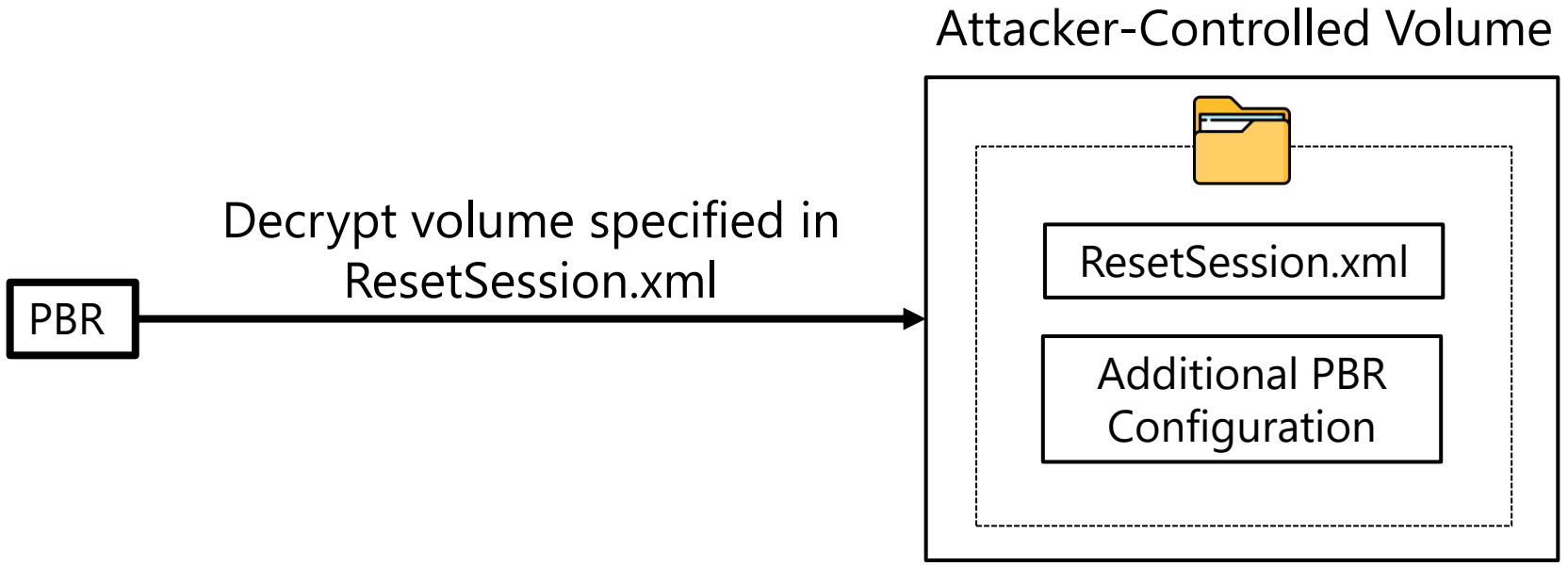
The Sensitive Configuration - PBR ResetSession.xml

- PBR can decrypt BitLocker volumes if stated in ResetSession.xml

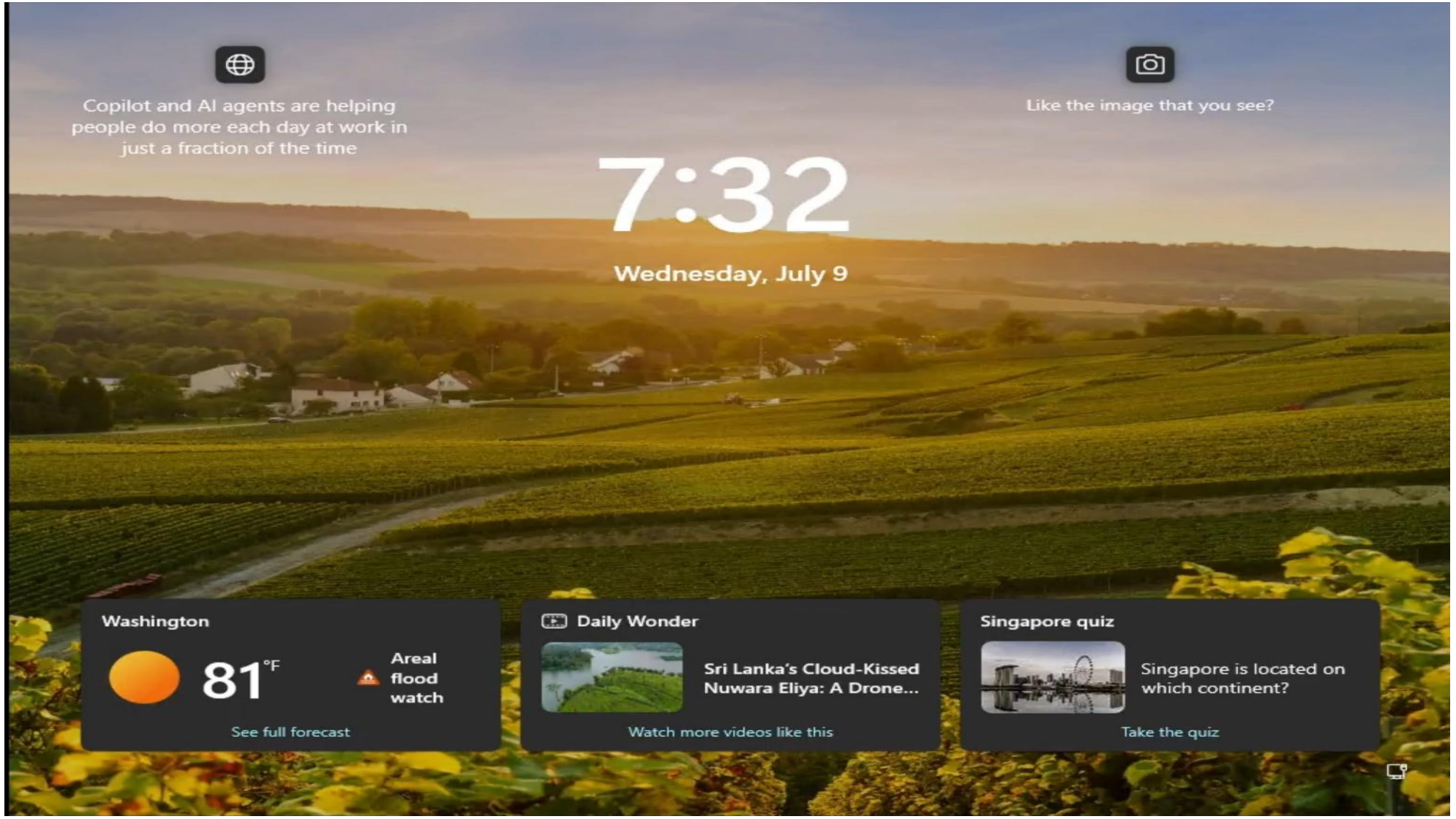


Chaining It All Together

- BitLocker bypassing flow:
  - Place PBR configuration on the attacker-controlled volume under \$SysReset
  - In ResetSession.xml – add the DecryptVolume operation targeting the BitLocker-encrypted volume
  - Use one of the confusion primitives to force WinRE into resetting the attacker-controlled volume
  - Profit

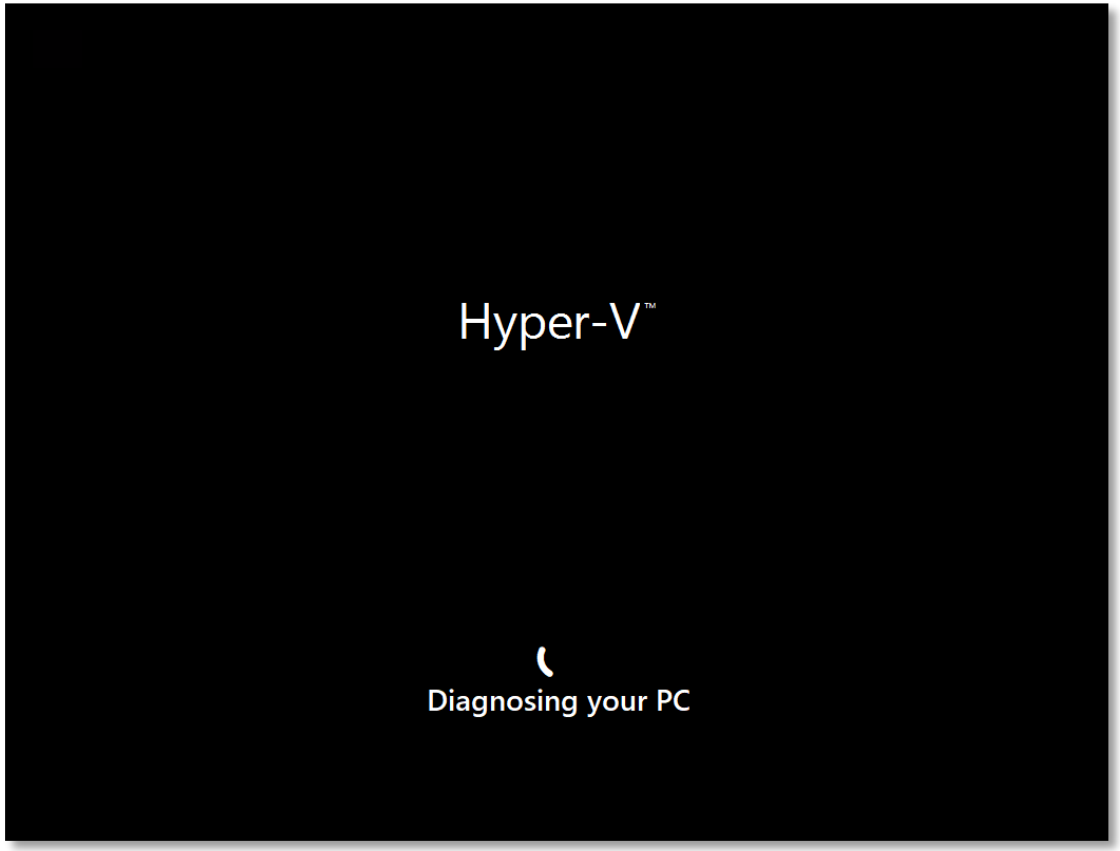


Demo


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

Startup Repair Tool (SRT)

- Startup Repair Tool (SRT) - Windows's Repair Tool



SRT Is Applicable

- SRT meets all exploit requirements

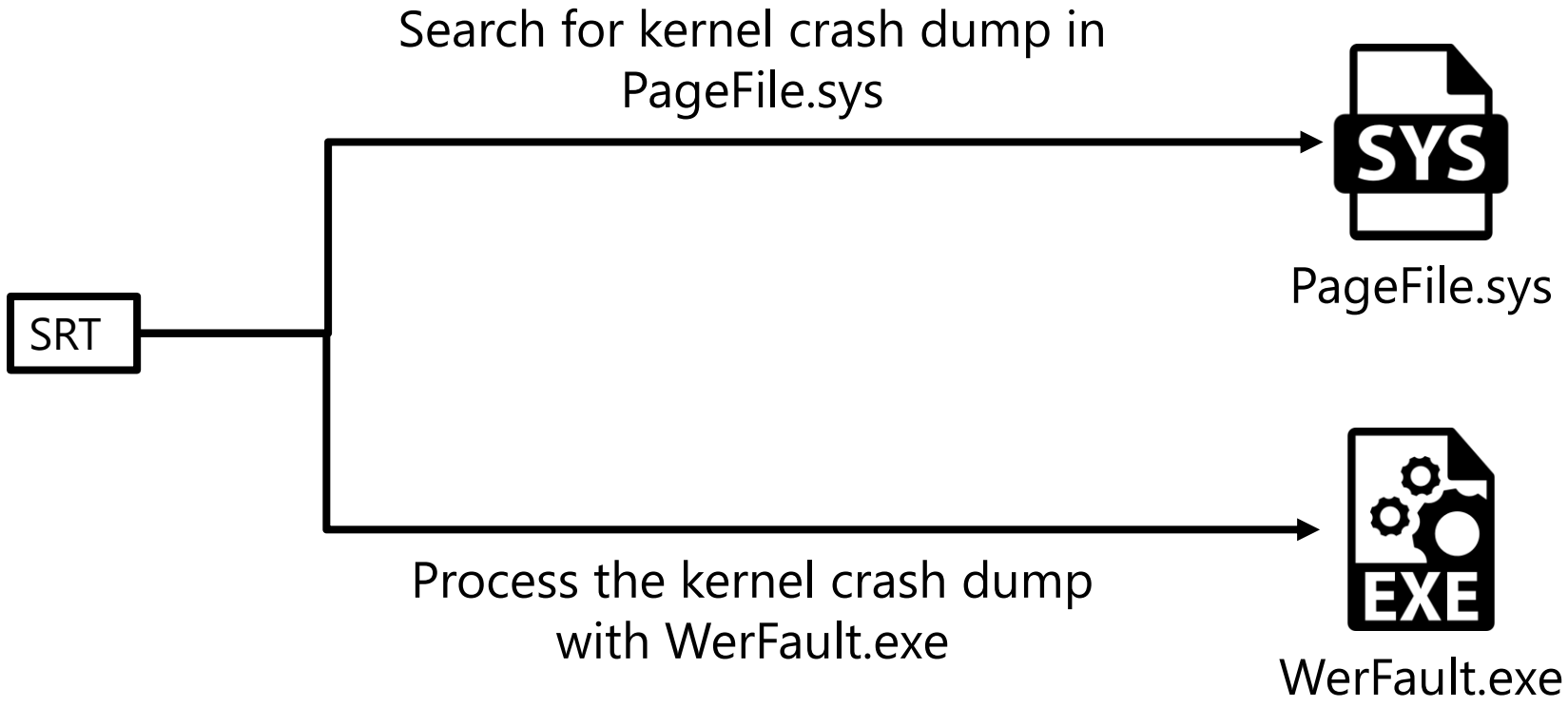
Can be triggered by an attacker

Does not trigger the relock functionality

Queries configuration from the target OS to perform sensitive operations

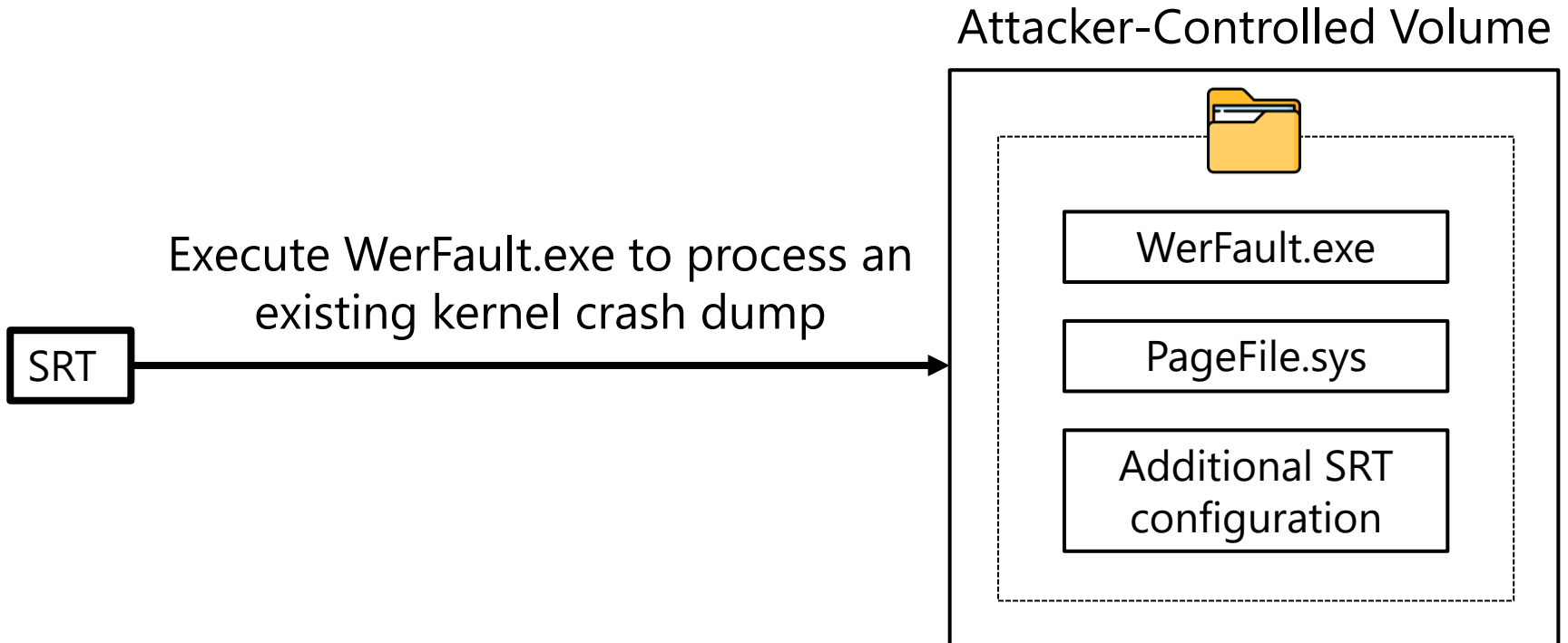
The Vulnerable Operation – SRT Crash Dump Upload

- SRT searches for kernel crash dump in PageFile.sys
- If a crash dump is detected, it is processed via WerFault.exe
- No signature checks are performed

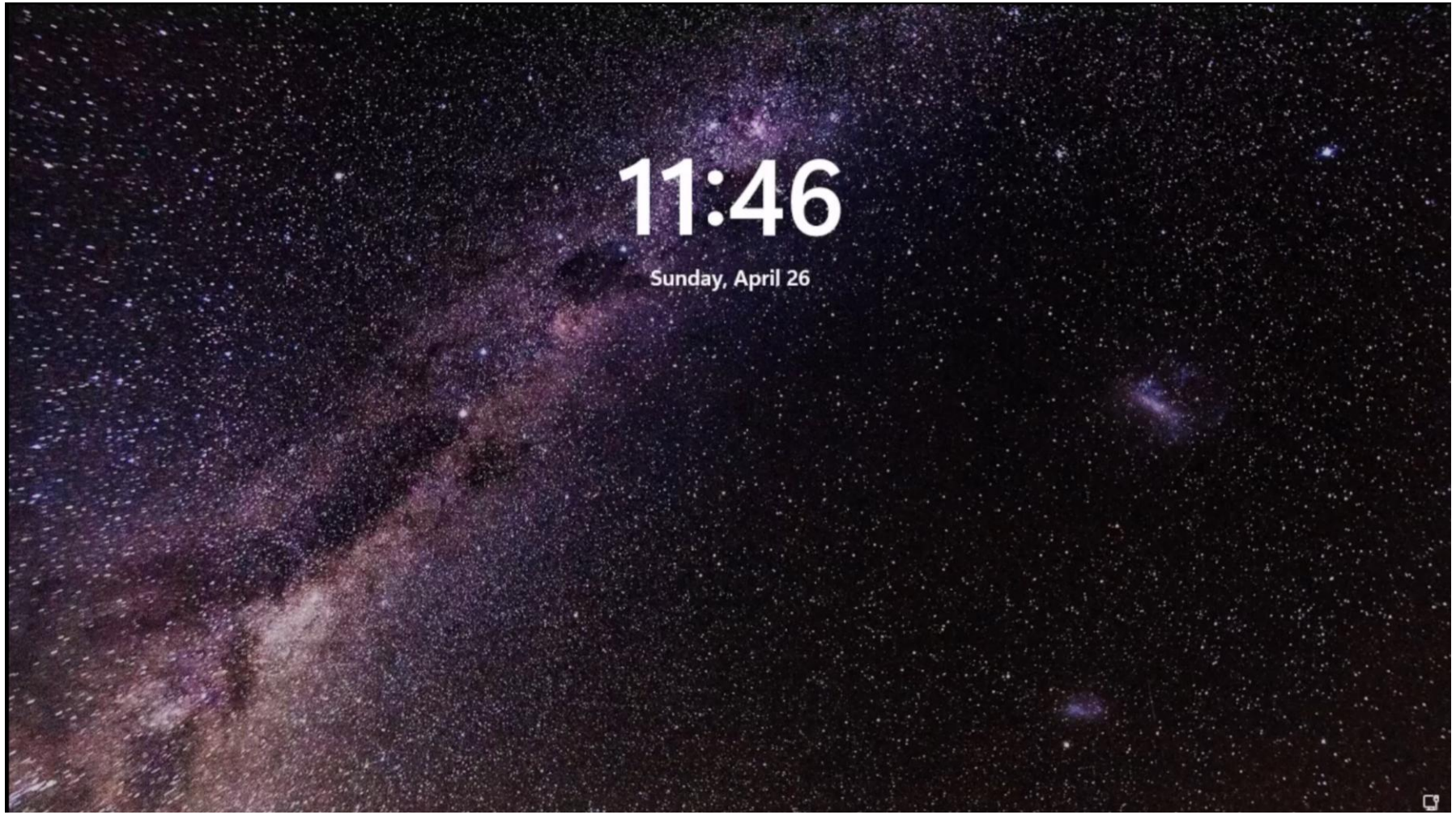


Chaining It All Together

- BitLocker bypassing flow:
  - Place SRT configuration and files on the attacker-controlled volume
    - Among the files - an arbitrary executable under \Windows\System32\Werfault.exe
    - Among the files - A Page File with a crash dump under \PageFile.sys
  - Use the BootStat.dat confusion primitive to force WinRE into repairing the attacker-controlled volume
  - Profit



Demo

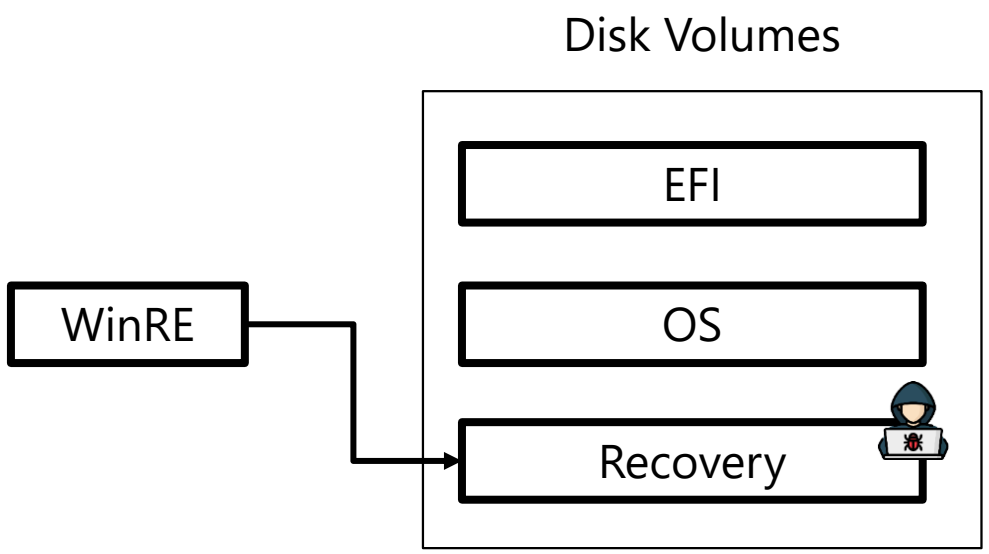


The Mitigation

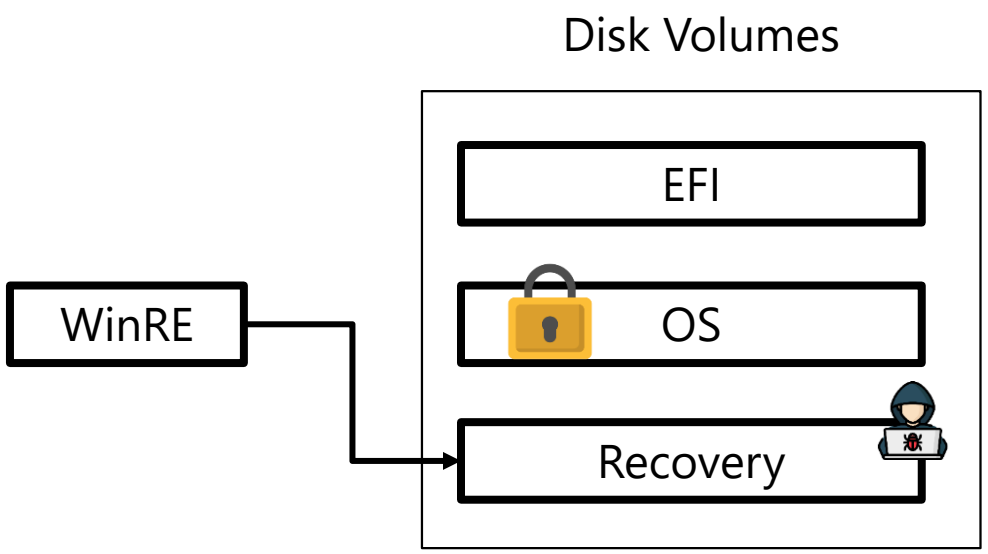
Mitigating The Attack Class

- WinRE now re-locks all volumes other than the target volume
- Any confusion primitive becomes ineffective – BitLocker-encrypted data can't be accessed

Before Mitigation



After Mitigation



Closing Remarks

Research Results

- The attack class vulnerabilities CVEs:
  - CVE-2025-48818
  - CVE-2025-55333
  - CVE-2025-55337
  - CVE-2025-55682
- Fixes were shipped in July (2025) and October (2025) Patch Tuesday’s
- This research was part of a broader security review
  - Additional findings were presented in our
    - “BitUnlocker” talk (Black Hat USA 2025, DEFCON 33, CCC 2025)
    - “Recovery Reimagined” talk (BlueHat IL 2026)

Key Takeaway

Always challenge security-critical trust assumptions

Thank You!

Alon Leviev

Senior Security Researcher
@ Microsoft (STORM)

Microsoft

STORM