Running Rootkits Like A Nation-State Hacker

Omri Misgav
CTO, Security Research Group @ Fortinet IL
Introduction

What Is Driver Signing Enforcement (DSE)

• Code Integrity was first introduced over 15 years ago
• Kernel drivers must be digitally signed on x64-based Windows
  • Code signing certificate from cross-certificate supporting CA
  • From Win10 Redstone (August 2016) a 2nd signature from Microsoft is required
• Drivers are checked each time they are loaded into memory to be executed
• Allows to improve the security of the OS
  • No drivers from unknown/untrusted origin
  • No drivers that were modified post build, possibly by privileged malicious actor
Introduction

Code Integrity Internals

* Query info wrapper does not verify g_CiEnabled

nt!Phase1Initialization
nt!SeInitSystem
nt!SepInitializeCodeIntegrity

nt!g_CiEnabled = TRUE

CI!CiInitialize(&nt!g_CiCallbacks)

CI!g_CiOptions |= DSE_ON

Callbacks = { CI!CiValidateImageData, CI!CiValidateImageHeader, CI!CiQueryInformation }

nt!MmLoadSystemImage
nt!...

nt!MmCreateSection

nt!MiValidateImageHeader
nt!SeValidateImageHeader
nt!SeValidateImageData
Introduction

Code Integrity Internals Changes

- From Win8 nt!g_CiEnabled variable was removed
- The callbacks structure also changed
  - CI.dll provides more callbacks
  - Same important callbacks remain but their offsets changed
- Only validate image header is invoked when loading a driver
- From Win8.1 callbacks structure symbol changed to nt!SeCiCallbacks
Introduction

DSE Tampering in the Wild

• Change C!g_CiOptions or nt!g_CiEnabled in memory ("Flag Flipping")

• Symbols are located by simple pattern matching
  • Minimal changes from between all Windows versions

• Overwrite and quickly proceed to load unsigned driver

• Once finished restore the flags back to their original state to avoid PatchGuard

• Bring Your Own Driver (BYOD) to gain kernel write primitive
  • With vulnerable or poorly written code that breaks security boundaries between user and kernel-mode
  • Pros: portable between OS versions, reusable after patching
  • Cons: Require administrative privileges, artifacts on disk
Protections

• Kernel Patch Protection (KPP, PatchGuard)
  • Callbacks structure in ntoskrl.exe from Win8
  • CI\!g\_CiOptions from Win8.1

• Driver Blocklists
  • Enforced via Windows Defender Application Control (WDAC) or Secure Kernel (VTL1)
  • Reduce the attack vector – make it harder to gain a write primitive
  • The latest list can be found [here](#)

• Kernel Data Protection (KDP) introduced in Win10 20H1
  • Protect drivers in the Windows kernel against data-driven attacks
  • Reduce the attack surface – prevent changes to CI\!g\_CiOptions
  • Memory is marked by Secure Kernel (VTL1) as protected using the SLAT PTEs
  • Does not enforce how the GVA range mapping of a protected region is translated
    • Verifies only on a periodic basis that it translates to the appropriate GPAs
Protections

Cl.dll With KDP
New Techniques
DSE Tampering Procedure

1. Locate
2. Overwrite
3. Load
4. Revert
New Techniques

Page Swapping

• Swap GPA (PFN in PTE) to a GPA we control
  • KDP doesn’t enforce PFN, only permissions

![Diagram showing Page Swapping](image)
New Techniques

Page Swapping

• Swap GPA (PFN in PTE) to a GPA we control
  • KDP doesn’t enforce PFN, only permissions
New Techniques
Page Swapping

- Allocate a new, writable page
- Find C!g_CiOptions (same as in the wild)
- Read PTE base
  - Randomized by KASLR from Win10 Redstone
  - [Turning (Page) Tables](#), BSidesLV 2019 (slide 18)
New Techniques

Page Swapping

- Allocate a new, writable page
- Find Cl!g_CiOptions (same as in the wild)
- Read PTE base
- Read PFN from PTE for Cl!g_CiOptions’ page
- Read PFN from PTE for our page
- Copy the entire page
- Modify our g_CiOptions
- Set our page’s PFN in PTE for Cl!g_CiOptions’ page
- Load driver
- Restore original PFN
New Techniques
Page Swapping

- Allocate a new, writable user-space page
- Find Clg_CiOptions
- Read PTE base
- Read PFN from PTE for Clg_CiOptions’ page
- Read PFN from PTE for our page
- Initialize our page with g_CiOptions as disabled
- Set our page’s PFN in PTE for Clg_CiOptions’ page
- Load driver
- Restore the original PFN
Demo
New Techniques

Callback Swapping

• Replace CI.dll functions
  • Finding CI!g_CiOptions is not necessary

```c
int nt!SeValidateImageHeader(pe)
{
    int status = 0;

    if (nt!g_CiEnabled)
    {
        if (nt!g_CiCallbacks.CiValidateImageHeader == NULL)
            status = STATUS_INVALID_SIGNATURE;
        else
            status = nt!g_CiCallbacks.CiValidateImageHeader(pe);
    }

    return (status);
}
```

```c
int CI!CiValidateImageHeader(pe)
{
    int error;
    bool is_valid = validate_signature(pe, &error);

    if (is_valid)
        return (0);
    else
        return (ci_status_to_ret_value(error));
}
```
New Techniques

Callback Swapping

- Replace CI.dll functions
  - Finding CI!g_CiOptions is not necessary

```c
int ntSeValidateImageHeader(pe)
{
    int status = 0;
    if (nt!g_CiEnabled)
    {
        if (nt!g_CiCallbacks.CiValidateImageHeader == NULL)
            status = STATUS_INVALID_SIGNATURE;
        else
            status = nt!g_CiCallbacks.CiValidateImageHeader(pe);
    }
    return (status);
}
```

```c
int CI!CiValidateImageHeader(pe)
{
    int error;
    bool is_valid = validate_signature(pe, &error);
    if (is_valid)
        return (0);
    else
        return (ci_status_to_ret_value(error));
}
```
New Techniques

Callback Swapping

- Find the callbacks structure in ntoskrnl.exe
  - Find the call to CI!CiInitialize by reference to the Import Lookup Table (ILT) entry
  - Walk back to register assignment of a parameter pointing to uninitialized memory “.data" section
New Techniques

Callback Swapping

• Find the callbacks structure in ntoskrnl.exe
  • Find the call to CI!CiInitialize by reference to the Import Lookup Table (ILT) entry
  • Walk back to register assignment of a parameter pointing to uninitialized memory “.data" section
New Techniques

Callback Swapping

• Find the callbacks structure in ntoskrnl.exe
• Find a new callback
  • Exported functions in ntoskrnl.exe
  • Exported functions or gadgets in CI.dll
New Techniques

Callback Swapping

• Find the callbacks structure in ntoskrnl.exe
• Find a new callback
• Find the original callbacks
  • Search for the instructions setting the callbacks in the structure in CI!CipInitialize
  • Verify the addresses by traversing unwind information
New Techniques

Callback Swapping

- Find the callbacks structure in ntoskrnl.exe
- Find a new callback
- Find the original callbacks
- Set the callback\’s to the new function
- Load driver
- Restore the original callback\’s

Locate

Overwrite
## New Techniques
### Comparison

<table>
<thead>
<tr>
<th>Flag Flipping</th>
<th>Page Swapping</th>
<th>Callback Swapping</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Naïve</td>
<td>User Page</td>
</tr>
<tr>
<td>Read</td>
<td>0</td>
<td>Page + 3</td>
</tr>
<tr>
<td>Alloc</td>
<td>0</td>
<td>Page</td>
</tr>
<tr>
<td>Write</td>
<td>1</td>
<td>Page + 1</td>
</tr>
</tbody>
</table>

* Assuming default values
Mitigation

- Confirm the state of DSE during driver loading
  - Why not find the internals variables the same way done for tampering?

- Capture the state
  - Copy the callbacks structure
  - Copy the flags

- Driver loading can be intercepted by
  - Hooking NtLoadDriver
  - Registry callbacks accessing the driver’s key
  - File system minifilter callback for IRP_MJ_ACQUIRE_FOR_SECTION_SYNCHRONIZATION

- Prevent by blocking the I/O request in callback or restore the variables
Summary

• Data-oriented mitigations are intricate to implement
  • “Robustly preventing this is non-trivial…” – Matt Miller, MSRC [BlueHatIL 2019]

• Defenders should adopt suggested mitigation as defense in-depth

• HVCI is the real solution for this case, enable it!
  • Introduced 7 years ago, with VBS, as Device Guard
  • Runs independently from KDP
  • Eliminate the attack surface – prevent any code from running in the kernel without being validated first in the Secure World (VTL1) by SKCI.DLL (Secure Kernel Code Integrity)

• Won’t be completely obsolete due to misconfigured and legacy systems

• https://www.fortinet.com/blog/threat-research/driver-signature-enforcement-tampering.html
Questions?

in/omri-misgav