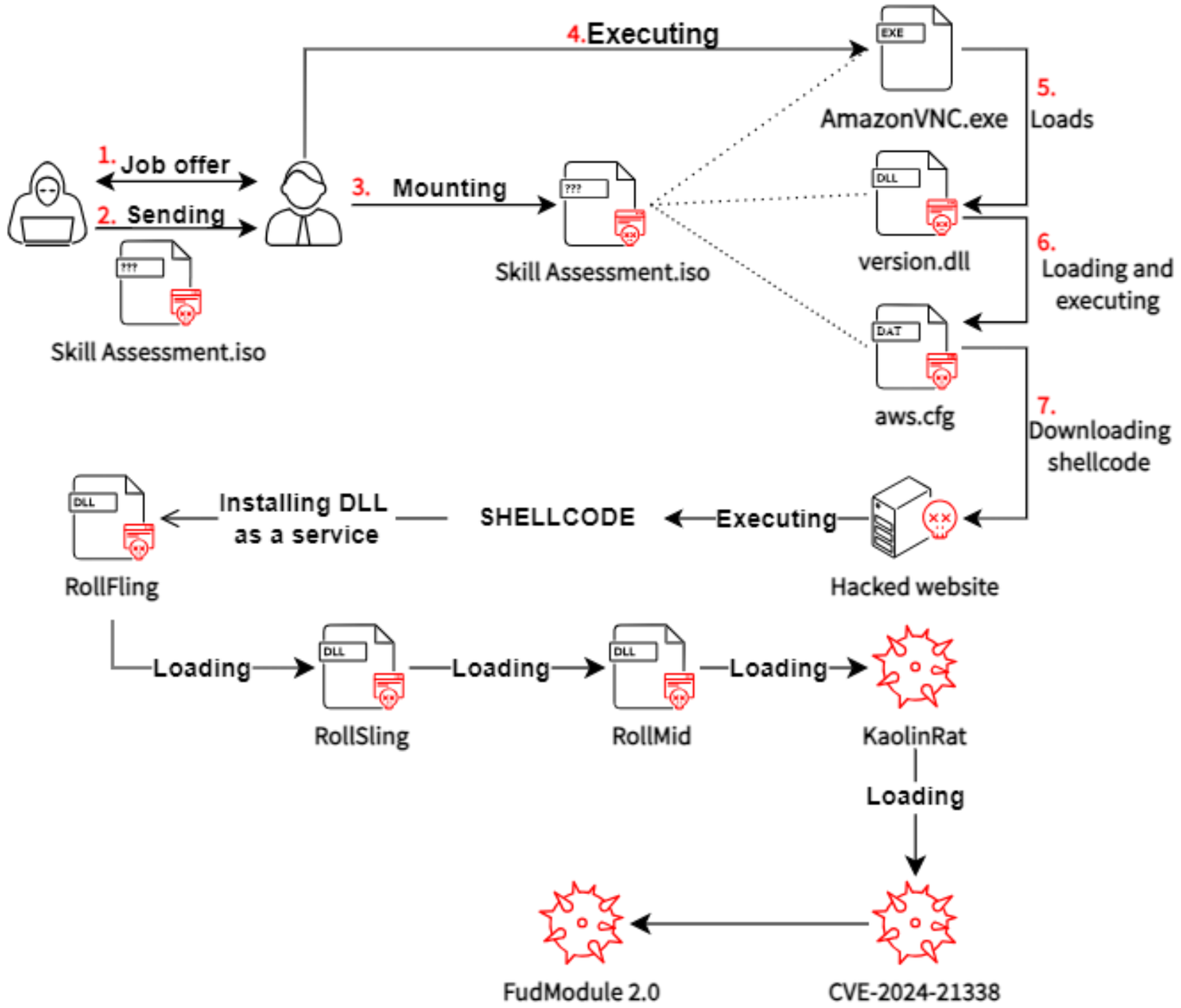
From BYOVD to a 0-day: Unveiling Advanced Exploits in Cyber Recruiting Scams

Speakers: Luigino Camastra, Igor Morgenstern

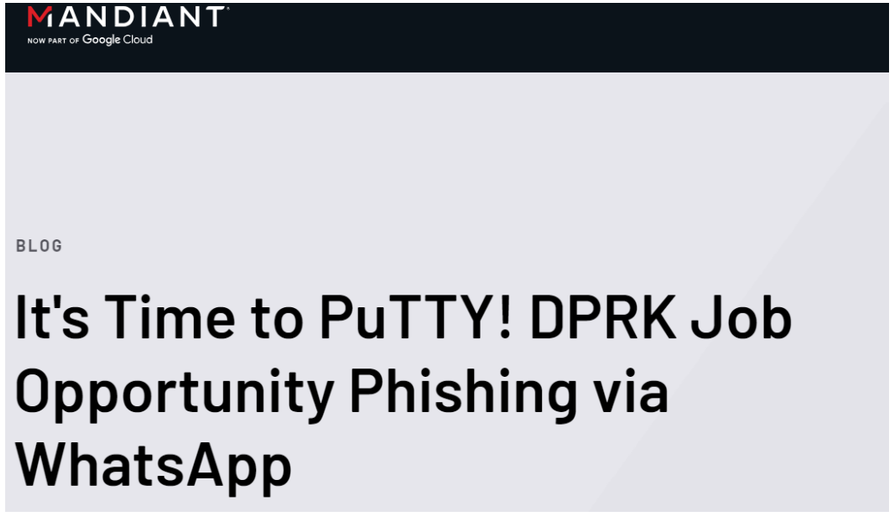
Contributor: Jan Vojtesek

Agenda

- Introduction to prior research
- Attack chain analysis
  - Initial ISO image
  - Loaders
  - RAT
- 0-day and vulnerability analysis
- Rootkit analysis



Prior research



Lazarus luring employees with trojanized coding challenges: The case of a Spanish aerospace company

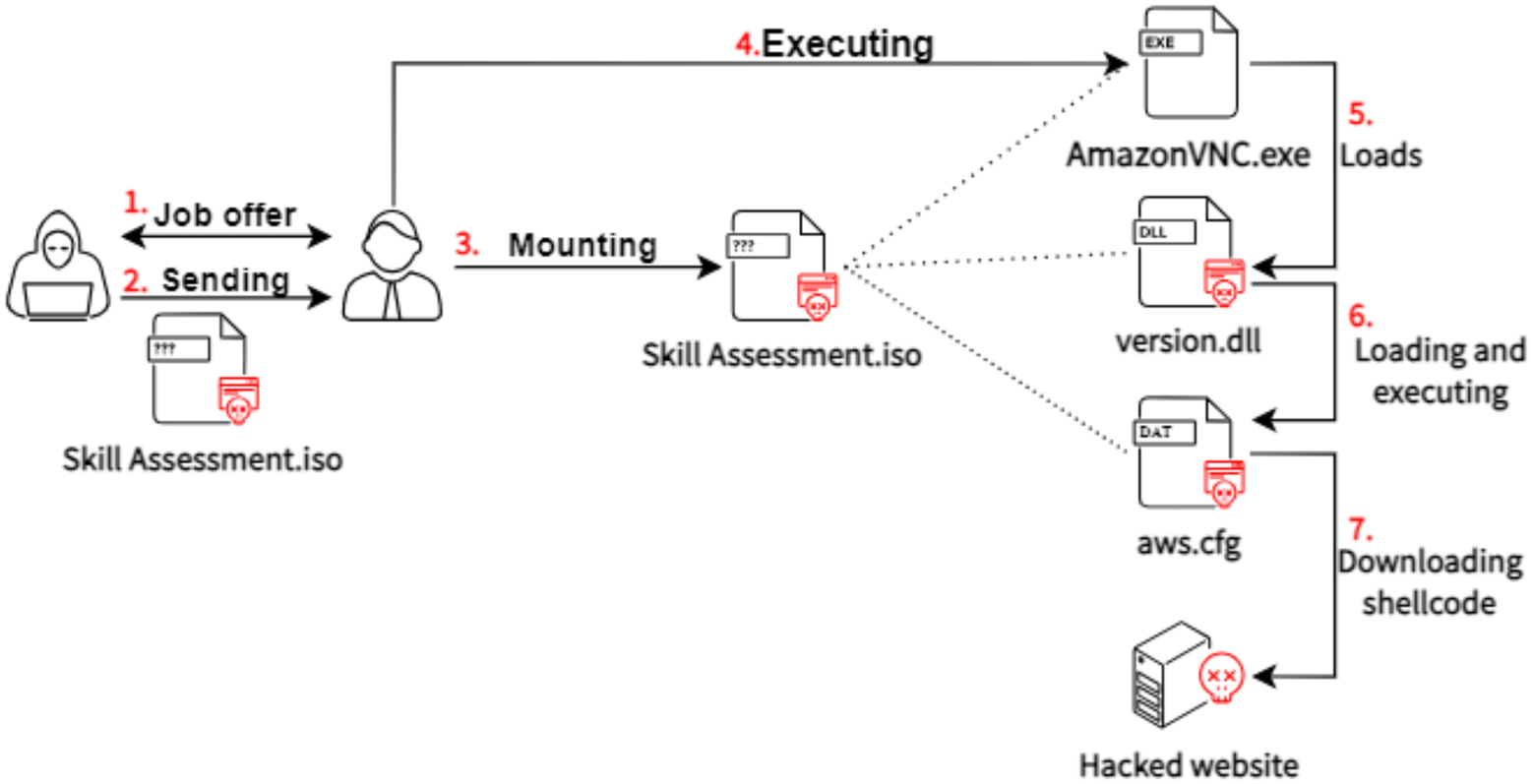
While analyzing a Lazarus attack luring employees of an aerospace company, ESET researchers discovered a publicly undocumented backdoor

Amazon-themed campaigns of Lazarus in the Netherlands and Belgium

ESET researchers have discovered Lazarus attacks against targets in the Netherlands and Belgium that use spearphishing emails connected to fake job offers

Attack chain analysis

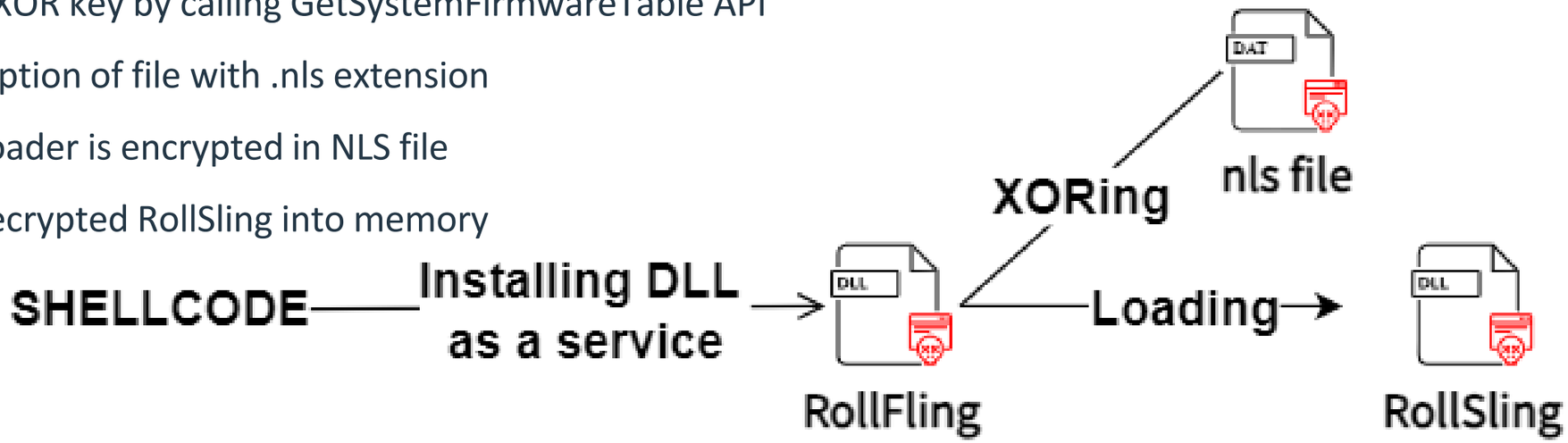
- The attack is initiated by presenting a fabricated job offer
- Contacting via LinkedIn, WhatsApp, email or other platforms



Attack chain analysis

RollFling Loader

- Shellcode executed in memory
- Discovered a new loader we called RollFling and NLS file
- Malicious DLL established as a service
- Kickstart execution chain
- Loading next stage
  - obtaining XOR key by calling GetSystemFirmwareTable API
  - XOR decryption of file with .nls extension
  - RollSling loader is encrypted in NLS file
  - Loading decrypted RollSling into memory



Attack chain analysis

- RollSling is a loader discussed in Microsoft research (Multiple North Korean threat actors exploiting the TeamCity CVE-2023-42793 vulnerability)
- Code similarities with the RollSling version discussed in the Microsoft research

```
fix_api();
pFileName = (char *)&lpFileName;
if ( lpFileName._Myres >= 0x10 )
  pFileName = lpFileName._Bx._Ptr;
FirstFile = FindFirstFileExA(pFileName, FindExInfoStandard, &FindFileData, FindExSearchNameMatch, 0LL, 0);
if ( FirstFile != (HANDLE)-1LL )
{
  while ( FindFileData.cFileName[0] == '.'
       && (!FindFileData.cFileName[1] || FindFileData.cFileName[1] == '.' && !FindFileData.cFileName[2])
       || (FindFileData.dwFileAttributes & 0x10) != 0
       || load_binary_to_memory_and_execute_StartAction_export_function(FindFileData.cFileName) )
  {
    if ( !FindNextFileA(FirstFile, &FindFileData) )
      goto looking_in_another_path;
  }
  goto exit;
}
looking_in_another_path:
if ( !load_binary_to_memory_and_execute_StartAction_export_function(0LL) )
  v1 = 0;
v3 = v1;
if ( FirstFile != (HANDLE)-1LL )
exit:
  FindClose(FirstFile);
```

Microsoft

d9add2bfdfebfa235575687de356f0cefb3e4c55964c4cb8bfdcdc58294eeaca

```
fix_api();
DLL_folder = (char *)&FullyQualifiedPath_to_folder_where_is_module;
if ( FullyQualifiedPath_to_folder_where_is_module._Myres >= 0x10 )
  DLL_folder = FullyQualifiedPath_to_folder_where_is_module._Bx._Ptr;
FirstFile = FindFirstFileExA(DLL_folder, FindExInfoStandard, &FindFileData, FindExSearchNameMatch, 0LL, 0);
if ( FirstFile == (HANDLE)-1LL )
{
looking_in_another_path:
  load_binary_to_memory_and_execute_StartAction_export_function(0LL);
  if ( FirstFile == (HANDLE)-1LL )
    goto exit;
}
else
{
  while ( FindFileData.cFileName[0] == '.'
       && (!FindFileData.cFileName[1] || FindFileData.cFileName[1] == '.' && !FindFileData.cFileName[2])
       || (FindFileData.dwFileAttributes & 0x10) != 0
       || load_binary_to_memory_and_execute_StartAction_export_function(FindFileData.cFileName) )
  {
    if ( !FindNextFileA(FirstFile, &FindFileData) )
      goto looking_in_another_path;
  }
}
FindClose(FirstFile);
```

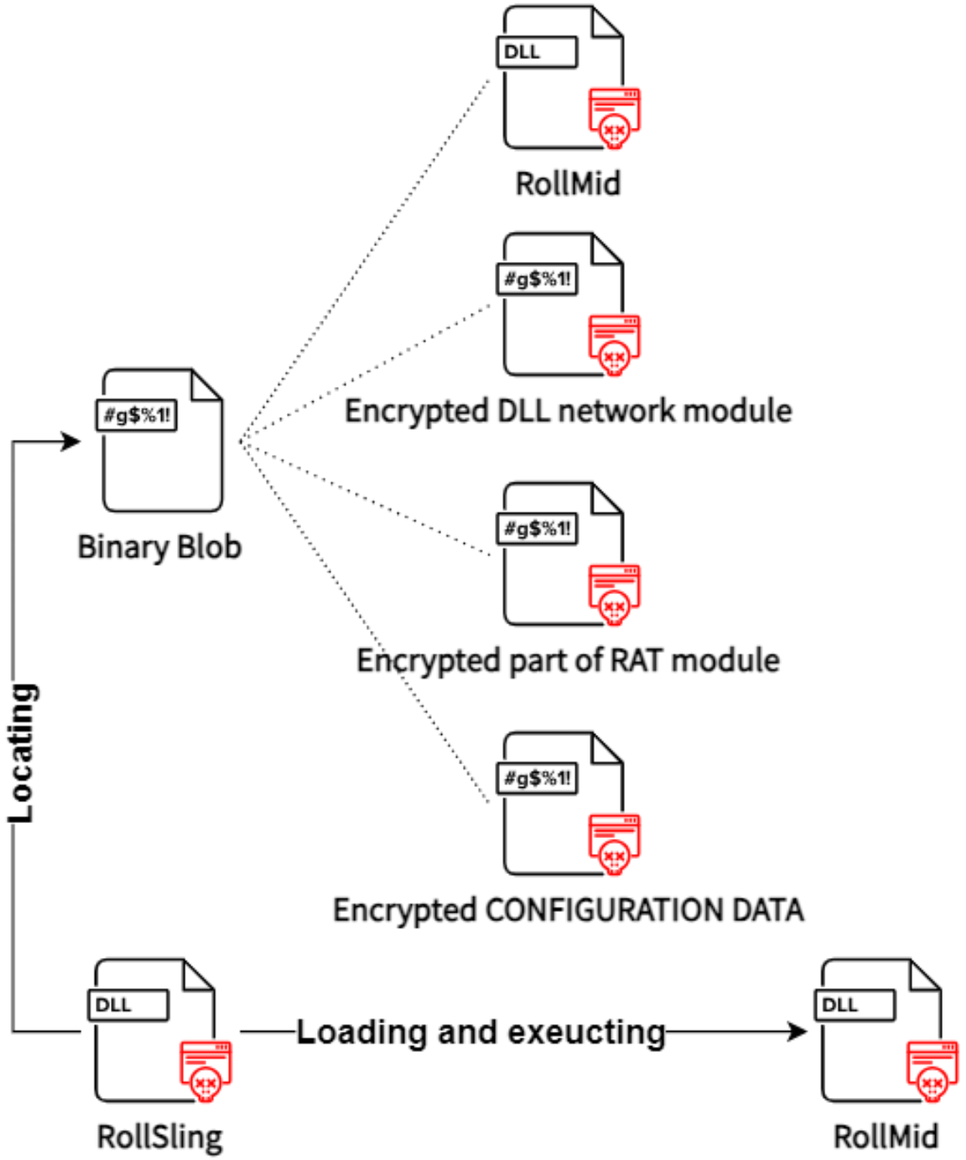
Gen Digital

e68ff1087c45a1711c3037dad427733ccb1211634d070b03cb3a3c7e836d210f

Attack chain analysis

RollSling Loader

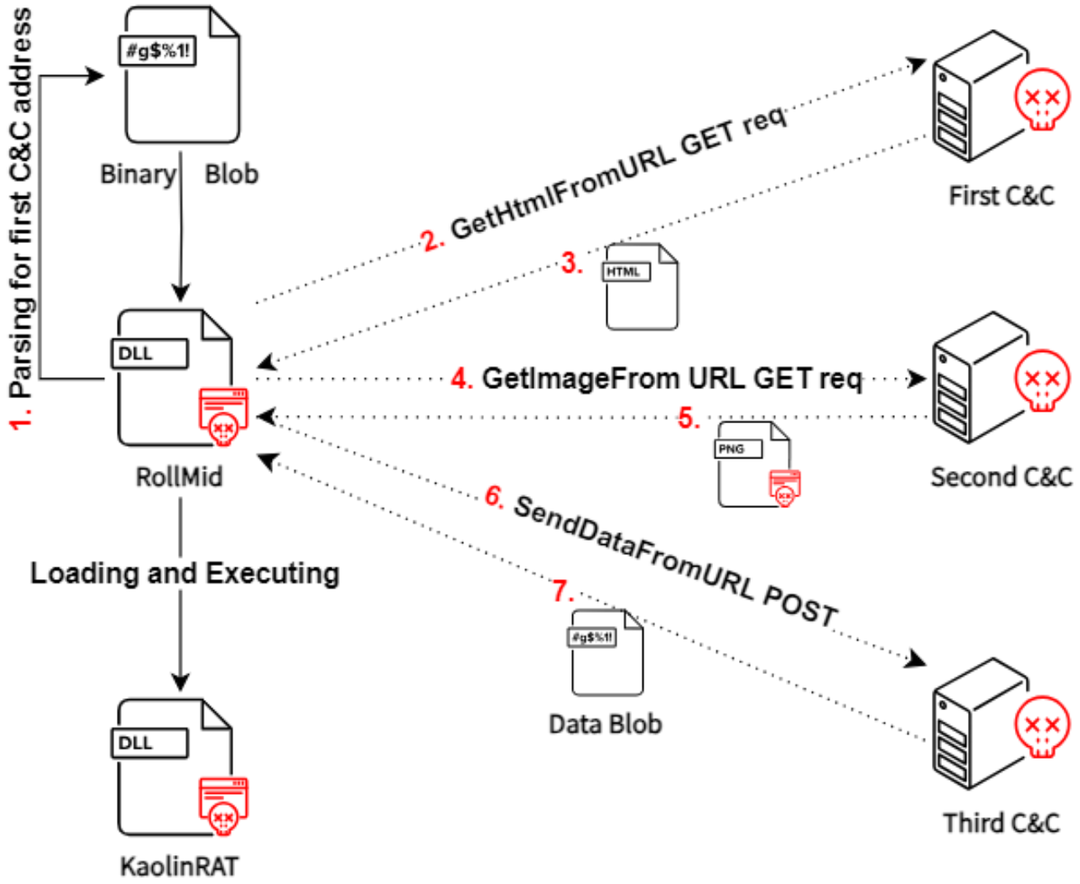
- Locate binary blob
  - Holds various stages and configuration data
    - RollMid, 2x DLL binaries and address of C&C server
  - Located without file extension
- Extracting the next stage from binary blob
  - Searching for export function "StartAction"
- Loading and executing the next stage RollMid
- (by calling "StartAction" export function)



Attack chain analysis

RollMid Loader

- Loading network module binary, parsing address of the C&C server
- Obtaining HTML file from the First C&C server
- Get PNG image from the Second C&C server
  - Steganography to extract the address of the Third C&C server
- Sending POST req to get Data Blob
  - Data blob contains configuration data for next stage
  - Appends part of Data Blob to the KaolinRAT DLL on disk as an overlay
- Loading and executing next stage, called Kaolin RAT



Attack chain analysis

Kaolin RAT

- Communication with C&C server
  - Network module DLL binary
  - Encrypted with AES
- Custom RAT
  - File compression capabilities
  - Uploading file to C&C
  - Changing file's last write timestamp
  - Downloading a DLL file from C&C server and loading it in a memory
  - Loading exploit with a FudModule rootkit

Living Off the Land: Vulnerable Drivers

Benefits

- Disrupt security software
- Hide indicators of infection
- Disable kernel-mode telemetry

Obstacles

- DSE (Driver Signature Enforcement)
- HVCI
- SMEP

Techniques

- Data-only attacks
- Signed Malicious Drivers
- Vulnerable Drivers

Living Off the Land: Vulnerable Drivers

N-Day BYOVD

- Easy to pull off
- Lazarus previously abused **`dbutil_2_3.sys`** (Dell), **`ene.sys`** (ENE Technology Inc.)
- Straightforward to detect

Zero-Day BYOVD

- Attacker needs to discover a zero-day vulnerability
- Stealthier than n-day
- **`hw.sys`** exploited by Candiru
- Generates suspicious event

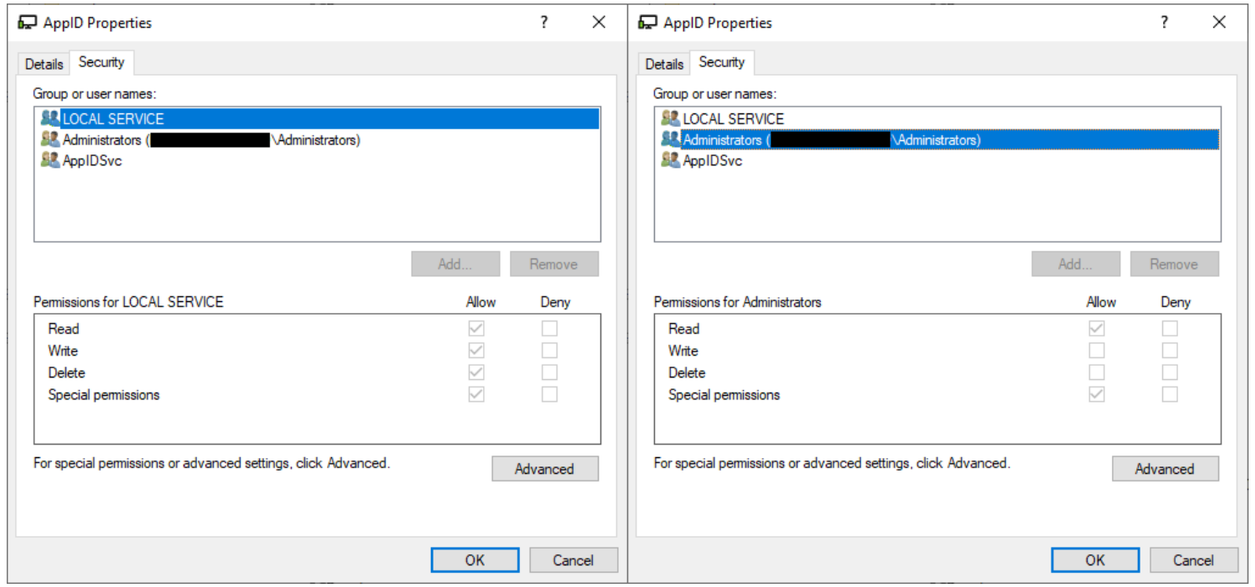
Zero-Day OS

- Abuse built-in Windows drivers
- Reduced attack surface
- Highest level of stealth

CVE-2024-21338

- Vulnerable IOCTL dispatcher in `appid.sys` (AppLocker)
- Allows calling arbitrary kernel function
- Partial control of the first argument
- **SMEP** prevents calling user-mode code
- **kCFG** requires a valid **kCFG** call targets
- IOCTL is exposed through **`\Device\AppId`**
- User should be running as **LOCAL SERVICE**

```
0: kd> p
appid!AppHashComputeImageHashInternal+0x7c:
fffff805`619fe218 ff15b2c4feff    call    qword ptr [appid!_guard_dispatch_icall_fptr (fffff805`619ea6d0)]
0: kd> r rax
rax=deadbeefdeadbeef
0: kd> dq rcx L1
ffffc38a`fba82c80  baadf00d`baadf00d
0: kd> k
 # Child-SP          RetAddr               Call Site
00 ffffd381`a623e590 fffff805`619d34af     appid!AppHashComputeImageHashInternal+0x7c
01 ffffd381`a623e690 fffff805`619f933e     appid!AppHashComputeFileHashesInternal+0x14b
02 ffffd381`a623e790 fffff805`619ee1b3     appid!AipSmartHashImageFile+0xd6
03 ffffd381`a623e860 fffff805`6068f835     appid!AipDeviceIoControlDispatch+0x123
04 ffffd381`a623e940 fffff805`60a77428     nt!IofCallDriver+0x55
05 ffffd381`a623e980 fffff805`60a77227     nt!IopSynchronousServiceTail+0x1a8
06 ffffd381`a623ea20 fffff805`60a765a6     nt!IopXxxControlFile+0xc67
07 ffffd381`a623eb60 fffff805`608092b5     nt!NtDeviceIoControlFile+0x56
08 ffffd381`a623ebd0 00000001`4000e3bd     nt!KiSystemServiceCopyEnd+0x25
09 00000000`0014f970 00000000`00000000     0x00000001`4000e3bd
```



CVE-2024-21338 - exploitation

- Load the driver by writing an event to AppLocker-related ETW provider
- Impersonates the **LOCAL SERVICE** account
- Write primitive to change **`PreviousMode`** of the current thread
- Can read and write arbitrary kernel memory with **`NtWriteVirtualMemory`**
- Fixed by introducing **`ExGetPreviousMode`** check

```
    case 0x22A018u:
      if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control && (HIDWORD(WPP_GLOBAL_Control->Timer) & 2) != 0 )
        WPP_SF_(WPP_GLOBAL_Control->AttachedDevice, 27u, &WPP_Traceguids);
      if ( Feature_2959575357__private_IsEnabled() && ExGetPreviousMode() )
        goto invalid_device_request;
      if ( CurrentStackLocation->Parameters.DeviceIoControl.InputBufferLength == 32 )
      {
        status = AipSmartHashImageFile(a2->AssociatedIrp.SystemBuffer, 0i64, 0i64, 0i64);
ret_status:
        ret_status = status;
        break;
      }
status_invalid_parameter:
      ret_status = STATUS_INVALID_PARAMETER;
      break;
```

FudModule 2.0

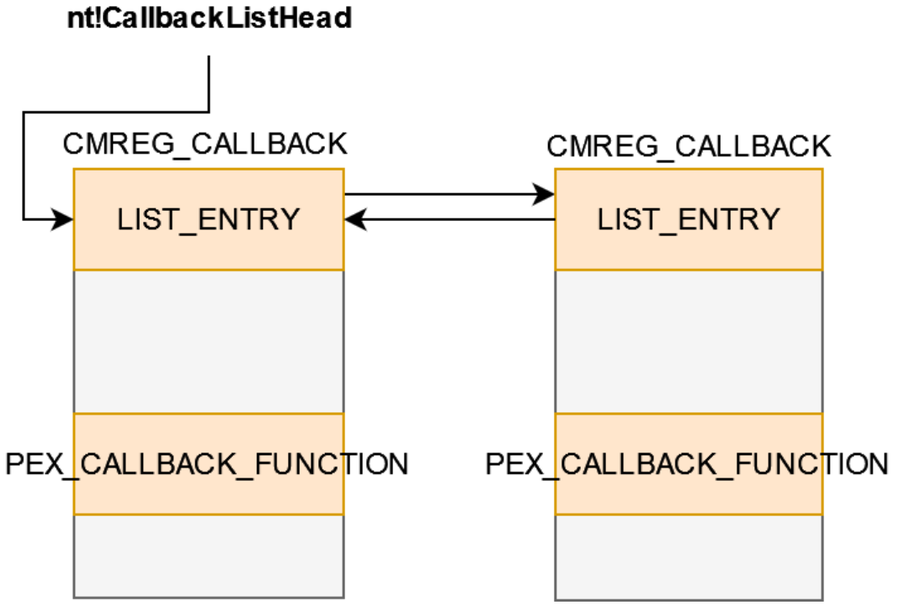
- Data-only rootkit (user space)
- DKOM Techniques
  - **0x1** - Registry Callbacks
  - **0x2** - Object Callbacks (no update)
  - **0x4** - Process, Thread, and Image Kernel Callbacks
  - **0x8** - File System MiniFilters
  - **0x10** - Windows Filtering Platform
  - **0x40** - Event Tracing for Windows: System Loggers
  - **0x80** - Event Tracing for Windows: Provider GUIDs
  - **0x100** - Image Verification Callbacks
  - **0x200** - Direct Attacks on Security Software

```
context = (__int64 *)LocalAlloc(0x40u, 0x1C0ui64);
context[51] = a1;
context[52] = a2;
result = setup(context);
if ( !(_DWORD)result )
{
  result = exploit(context);
  if ( !(_DWORD)result )
  {
    bitfield_techniques = registry_callbacks(context) != 0;
    if ( (unsigned int)object_callbacks(context) )
      bitfield_techniques |= 2u;
    if ( (unsigned int)process_image_thread_callbacks(context) )
      bitfield_techniques |= 4u;
    if ( (unsigned int)minifilters(context) )
      bitfield_techniques |= 8u;
    if ( (unsigned int)wfp_callouts(context) )
      bitfield_techniques |= 0x10u;
    if ( (unsigned int)etw_system_loggers(context) )
      bitfield_techniques |= 0x40u;
    if ( (unsigned int)etw_provider_guids(context) )
      bitfield_techniques |= 0x80u;
    if ( (unsigned int)image_verification_callbacks(context) )
      bitfield_techniques |= 0x100u;
    if ( (unsigned int)direct_attacks((__int64)context) )
      bitfield_techniques |= 0x200u;
    restore_previousmode((__int64)context);
    memset(context, 0, 0x1C0ui64);
    LocalFree(context);
```

FudModule 2.0

0x01 – Registry Callbacks

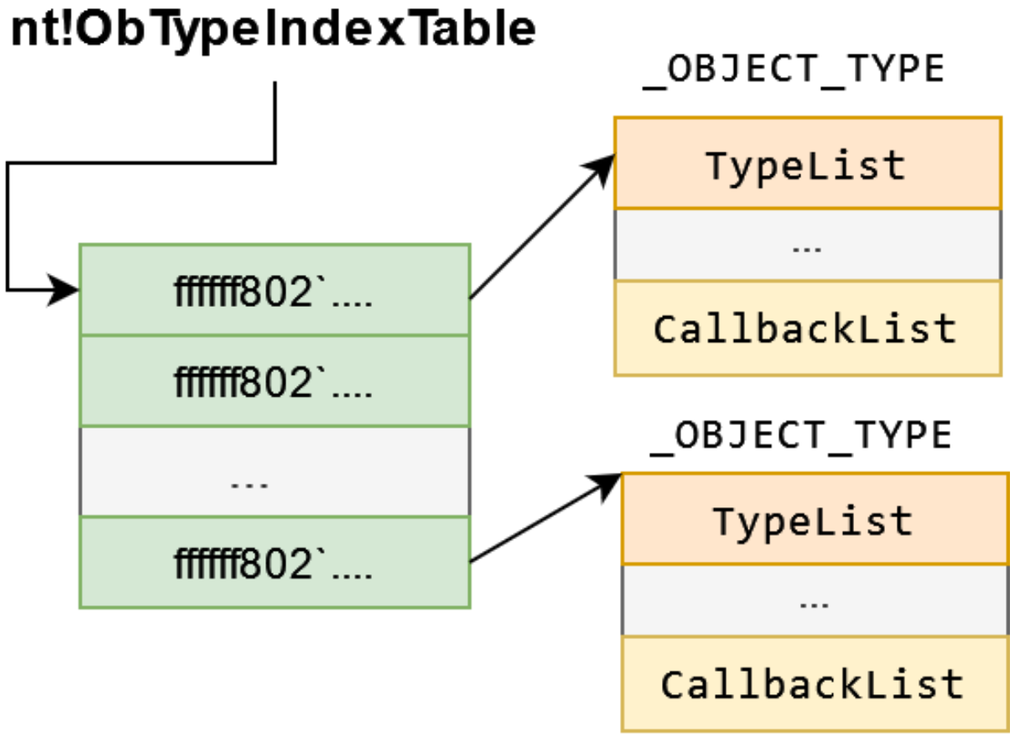
- Allow drivers to monitor and respond to changes in the registry
- Registered via **CmRegisterCallbackEx**
- DKOM
  - Resolve **CmUnRegisterCallback** (export of **ntoskrnl**)
  - Scanning function for `lea rcx,[nt!CallbackListHead]`
  - Find the address of nt!CallbackListHead
  - **New** - Skip callbacks from **ntoskrnl.exe, applockerfltr.sys, bfs.sys**
  - Replace callback with ObIsKernelHandle and unlink the callback entry



FudModule 2.0

0x02 – Object Callbacks - no update

- Monitor and respond to thread, process, and desktop handle operations
- Registered via **ObRegisterCallbacks**
- DKOM
  - Resolve **ObGetObjectType** (export of **ntoskrnl**)
  - Find **nt!ObTypeIndexTable**
  - **Nt!ObTypeIndexTable** is an array of pointers to **_OBJECT_TYPE** structures
  - Iterate over **CallbackList**
  - Make each point to itself

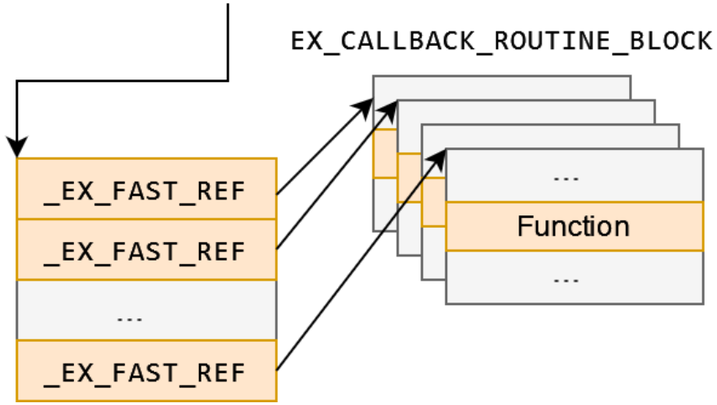


FudModule 2.0

0x04 - Process, Thread, and Image Kernel Callbacks

- Registered via
  - **PsSetCreateProcessNotifyRoutine**
  - **PsSetCreateThreadNotifyRoutine**
  - **PsSetLoadImageNotifyRoutine**
- DKOM
  - Resolve **nt!PspNotifyEnableMask, nt!Psp(LoadImage|CreateThread|CreateProcess)NotifyRoutine**
  - Clear **nt!PspNotifyEnableMask**
  - Create new arrays containing callbacks from whitelisted modules
  - Revert **nt!PspNotifyEnableMask**

nt!Psp(LoadImage|CreateThread|CreateProcess)NotifyRoutine



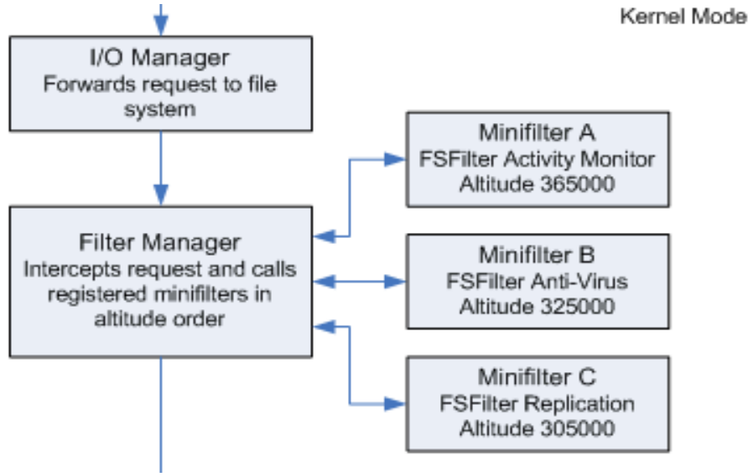
Whitelisted modules

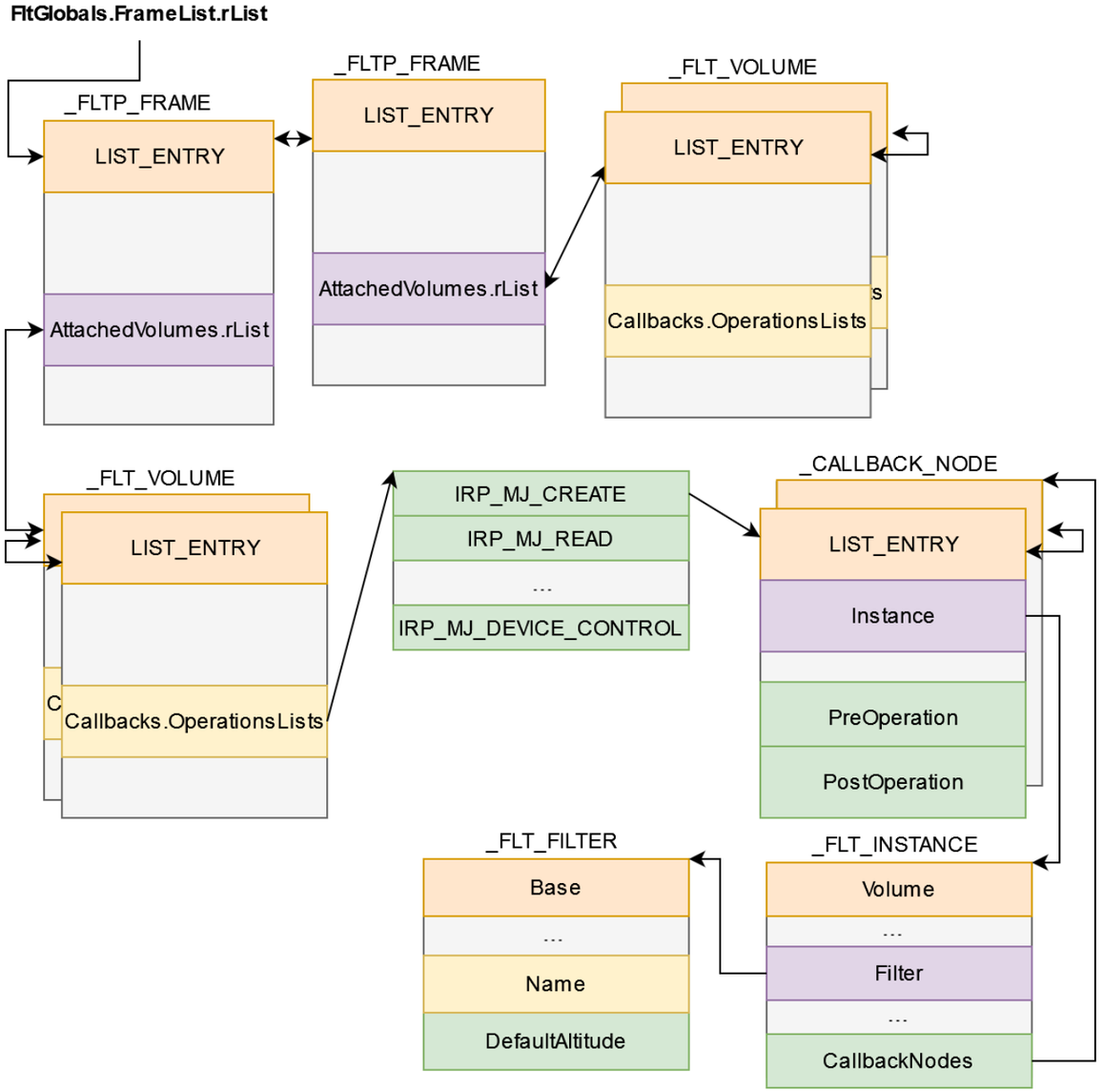
ntoskrnl.exe	ahcache.sys	mmcss.sys	cng.sys
ksecdd.sys	tcpip.sys	iorate.sys	ci.dll
dxgkrnl.sys	peauth.sys	wtd.sys	

FudModule 2.0

0x08 – Minifilter Drivers

- Mechanism for drivers to intercept file system operations
- **HVCI** prevents patching the filter function
- Iterates over **`_FLT_VOLUME.Callbacks.OperationsLists`**
- Indexed by IRP major function codes
- An array of linked lists of **`FLTMGR!_CALLBACK_NODE`**

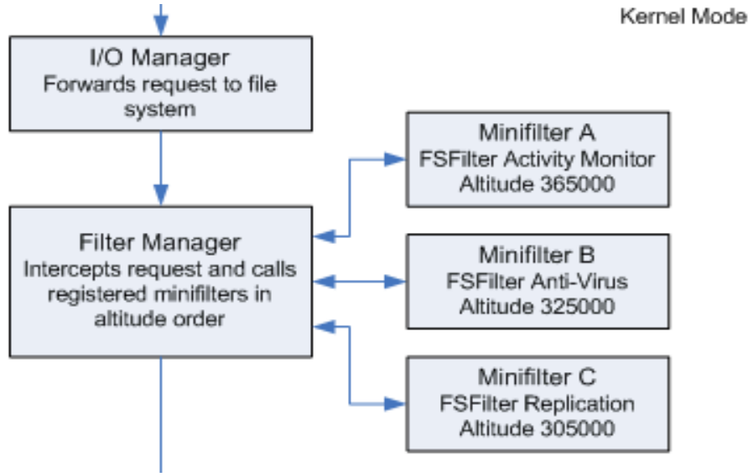


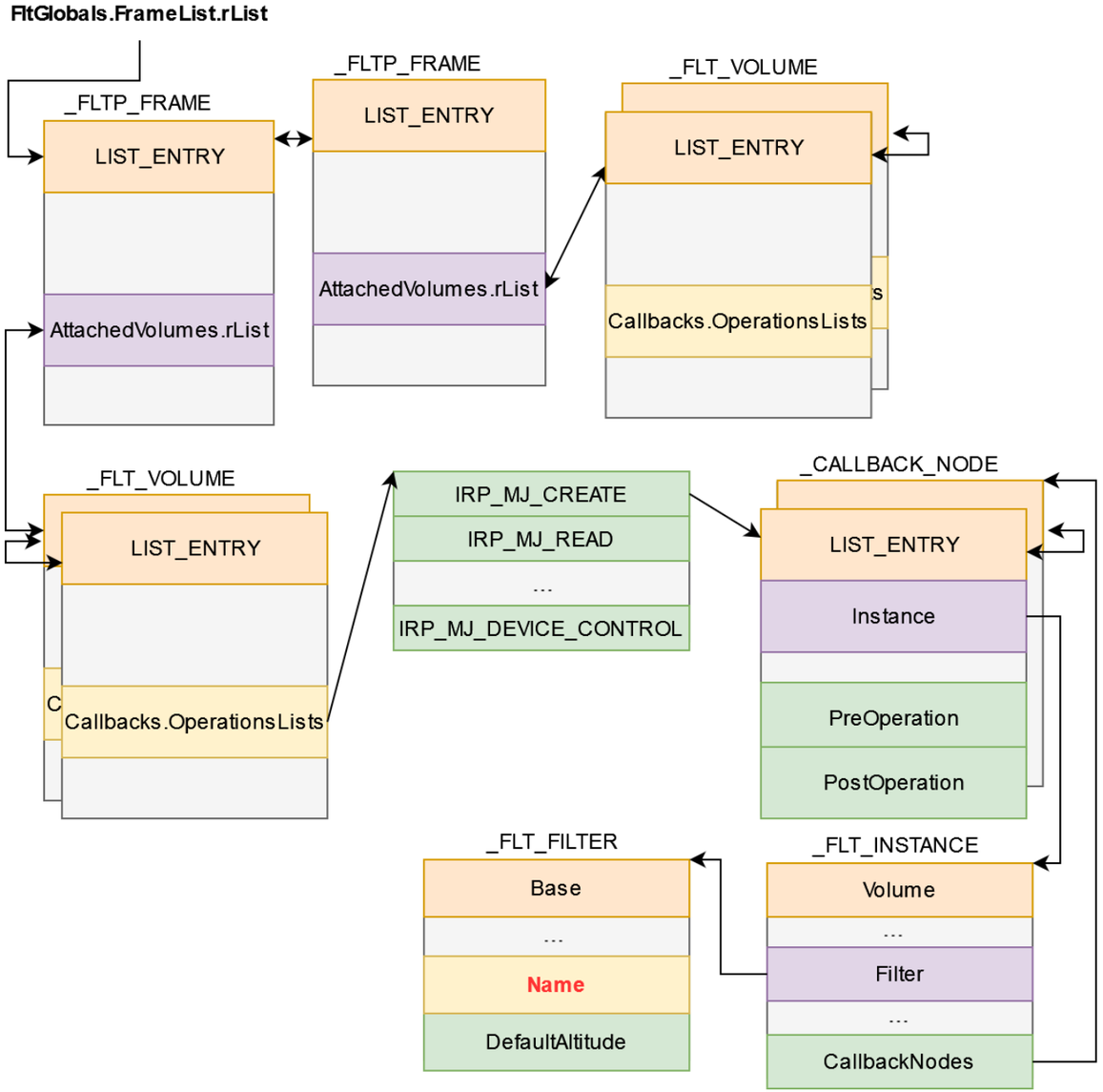


FudModule 2.0

0x08 – Minifilter Drivers

- Mechanism for drivers to intercept file system operations
- **HVCI** prevents patching the filter function
- Iterates over **`_FLT_VOLUME.Callbacks.OperationsLists`**
- Indexed by IRP major function codes
- An array of linked lists of **`FLTMGR!_CALLBACK_NODE`**

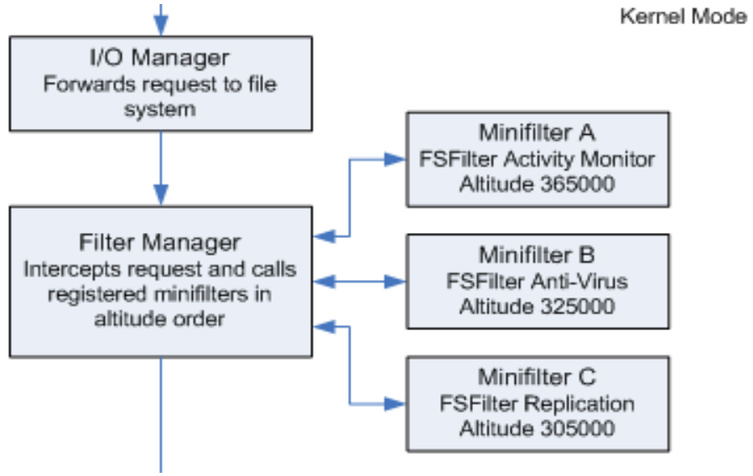


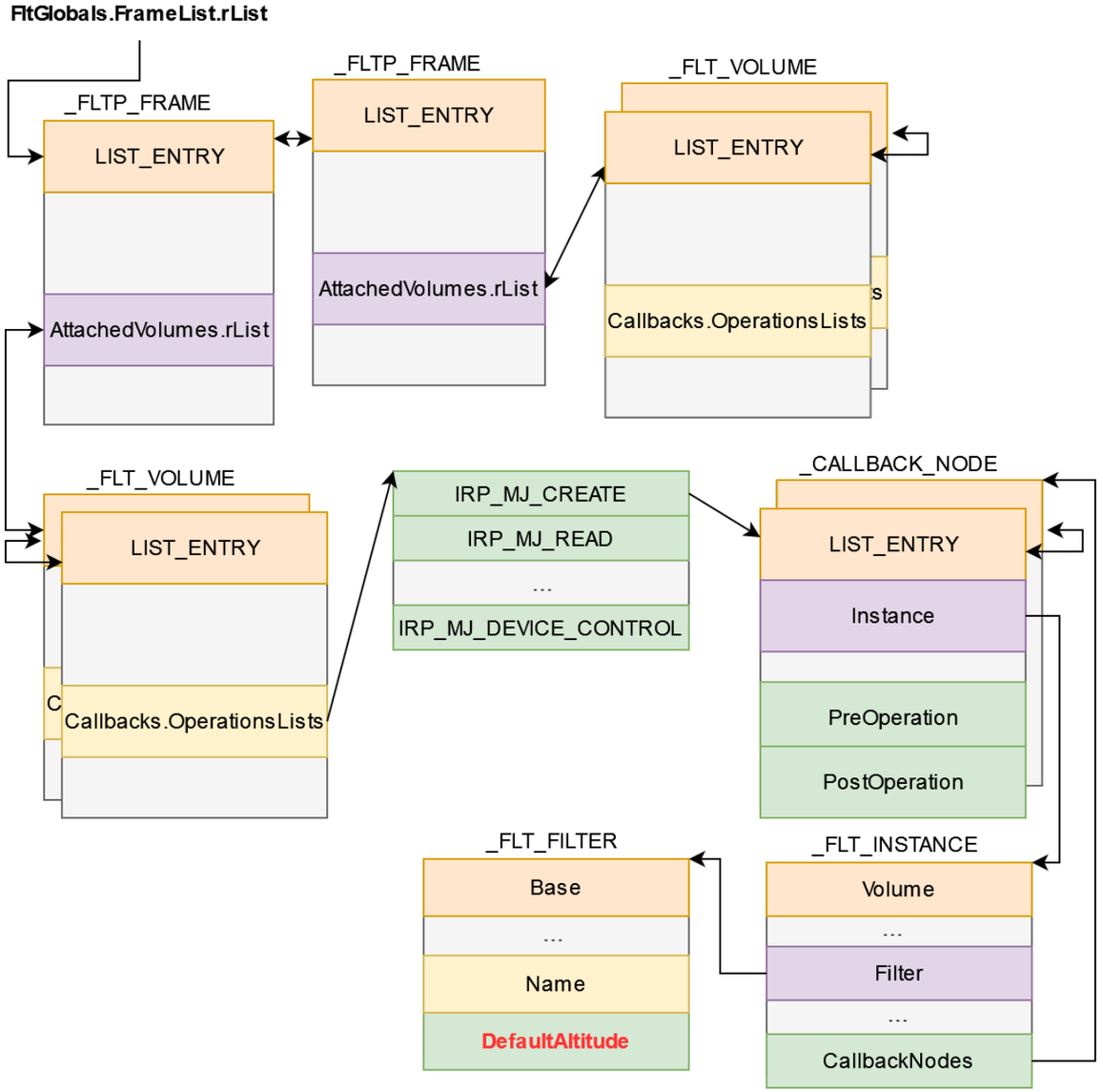


FudModule 2.0

0x08 – Minifilter Drivers

- Mechanism for drivers to intercept file system operations
- **HVCI** prevents patching the filter function
- Iterates over **`_FLT_VOLUME.Callbacks.OperationsLists`**
- Indexed by IRP major function codes
- An array of linked lists of **`FLTMGR!_CALLBACK_NODE`**

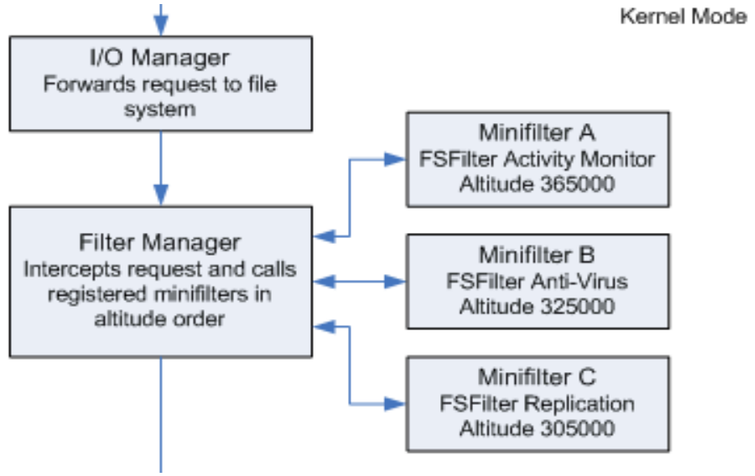


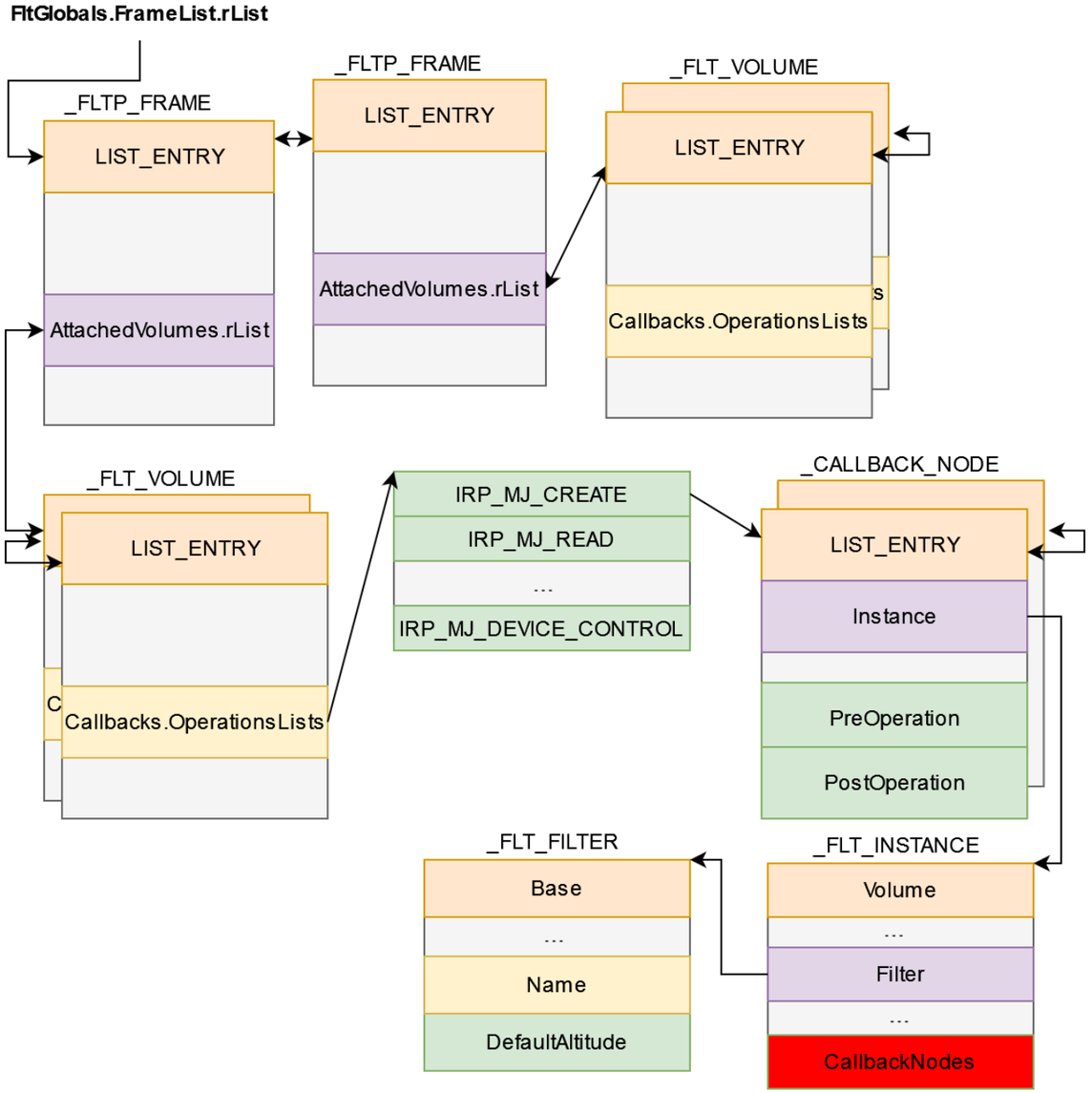


FudModule 2.0

0x08 – Minifilter Drivers

- Mechanism for drivers to intercept file system operations
- **HVCI** prevents patching the filter function
- Iterates over **`_FLT_VOLUME.Callbacks.OperationsLists`**
- Indexed by IRP major function codes
- An array of linked lists of **`FLTMGR!_CALLBACK_NODE`**

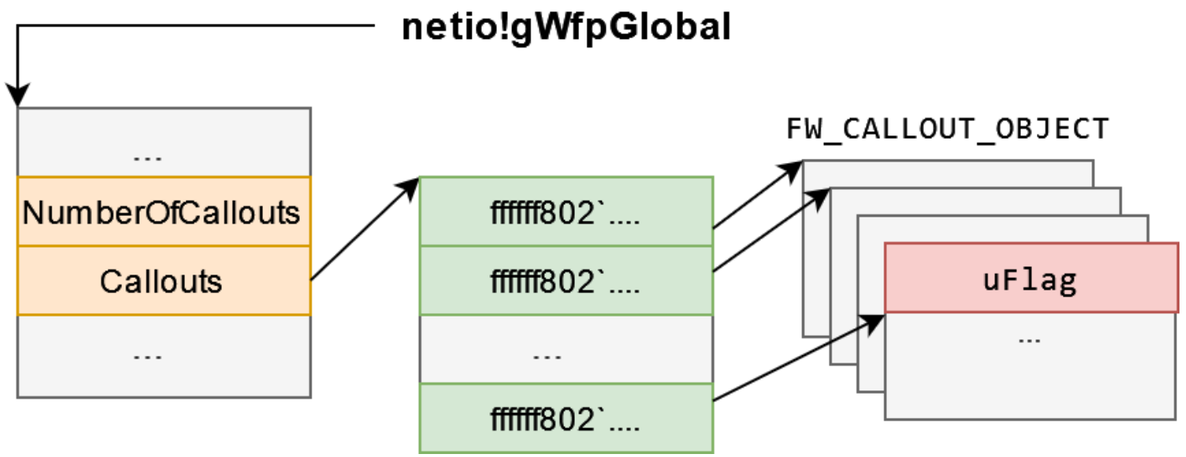




FudModule 2.0

0x10 – Windows Filtering Platform (WFP)

- Network traffic filtering
- Packet inspection



- Checks for Kaspersky drivers
- Locate `netio!gWfpGlobal`
- Iterate over the array of CALLOUT structs
- Set `FWP_CALLOUT_FLAG_CONDITIONAL_ON_FLOW`
- Call the callout function only if there is a context associated with the data flow

```
v13[0] = 0i64;
callout_structure = 0i64;
flow_context = 0i64;
FeGetRefCallout(*(v3 + 44), &callout_structure);
FlowId = GetFlowId(a2);
if ( FlowId )
{
  WfpFindAndRefFlowContext(FlowId, *(a1 + 48), *(*(a1 + 24) + 44i64), 0, &v14, v13);
  flow_context = v14;
}
v9 = callout_structure;
retval = TRUE;
callout_flags = *(callout_structure + 0x30);
if ( (callout_flags & 1) != 0 && !flow_context
  || (callout_flags & 8) == 0 && a2 && (*(a2 + 4) & 0x10) != 0
  || (callout_flags & 0x10) == 0 && a2 && (*(a2 + 4) & 0x20) != 0 )
{
  retval = FALSE;
}
if ( a3 && (callout_flags & 2) != 0 )
  retval = FALSE;
if ( FlowId && flow_context )
  WfpFindAndDeRefFlowContext(FlowId, *(a1 + 48), *(*(a1 + 24) + 44i64), 0i64);
```

FudModule 2.0

0x40 – Event Tracing for Windows System Loggers

- High-performance mechanism for tracing and logging events
- Zeroing out `EtwpActiveSystemLoggers`

```
void **__fastcall EtwpTraceKernelEventWithFilter(int a1, int a2, __int64 a3, __in
{
  void **result; // rax
  unsigned int v7; // ebx
  bool i; // zf
  void *retaddr; // [rsp+38h] [rbp+0h] BYREF

  result = &retaddr;
  v7 = a3 & EtwpActiveSystemLoggers;
  for ( i = !_BitScanForward((unsigned int *)&a3, a3 & EtwpActiveSystemLoggers);
        !i;
        i = !_BitScanForward((unsigned int *)&a3, v7) )
  {
    v7 &= v7 - 1;
    result = (void **)EtwpLogKernelEvent(a1, EtwpHostSiloState, (unsigned __int8)
  }
  return result;
}
```

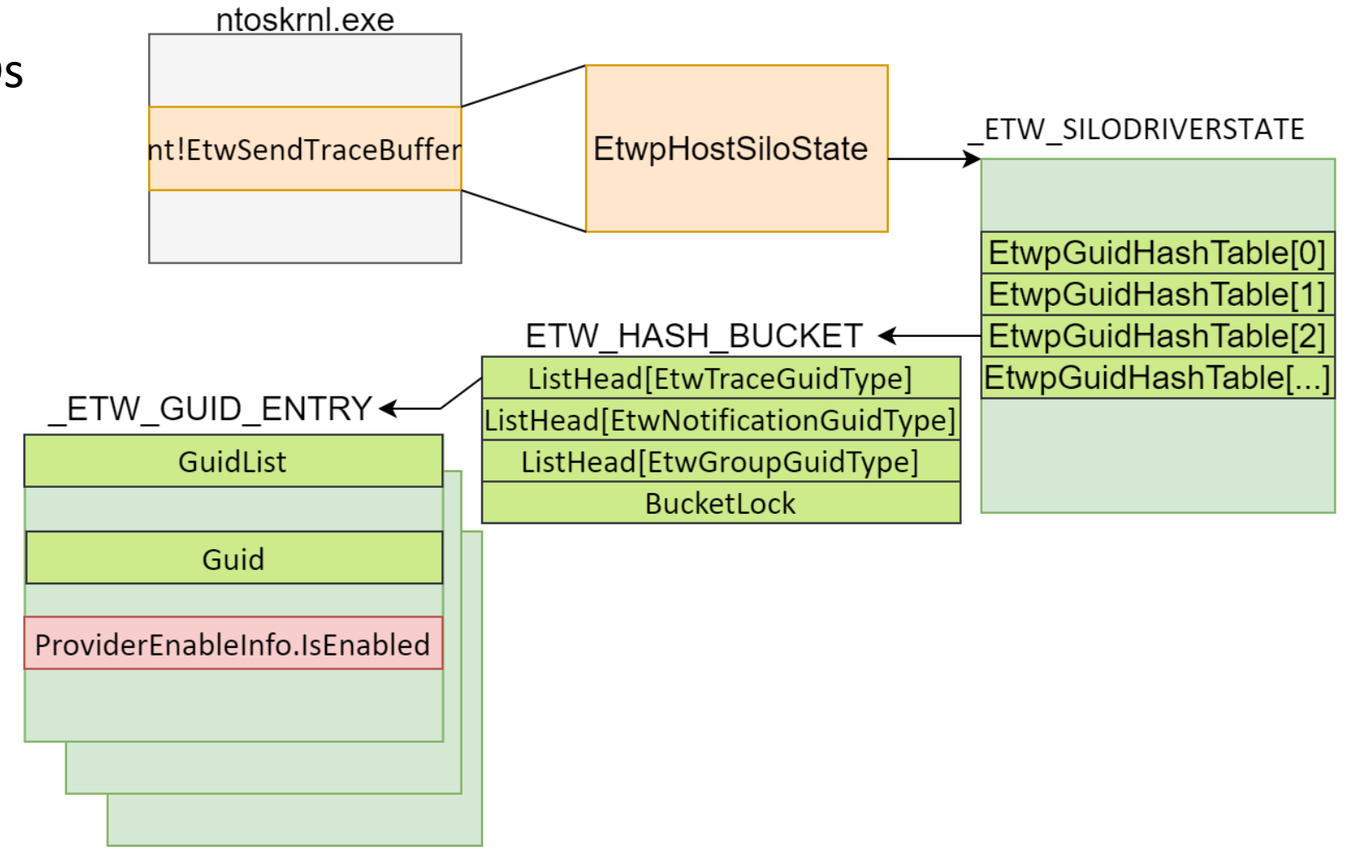
```
PS C:\WINDOWS\system32> logman query providers | Select-String threat-intel

Microsoft-Windows-Threat-Intelligence    {F4E1897C-BB5D-5668-F1D8-040F4D8DD344}


PS C:\WINDOWS\system32> logman query providers | Select-String kernel-

Microsoft-Windows-Kernel-Acpi            {C514638F-7723-485B-BCFC-96565D735D4A}
Microsoft-Windows-Kernel-AppCompat       {16A1ADC1-9B7F-4CD9-94B3-D8296AB1B130}
Microsoft-Windows-Kernel-Audit-API-Calls {E02A841C-75A3-4FA7-AFC8-AE09CF9B7F23}
Microsoft-Windows-Kernel-Boot            {15CA44FF-4D7A-4BAA-BBA5-0998955E531E}
Microsoft-Windows-Kernel-BootDiagnostics {96AC7637-5950-4A30-B8F7-E07E8E5734C1}
Microsoft-Windows-Kernel-Disk            {C7BDE69A-E1E0-4177-B6EF-283AD1525271}
```

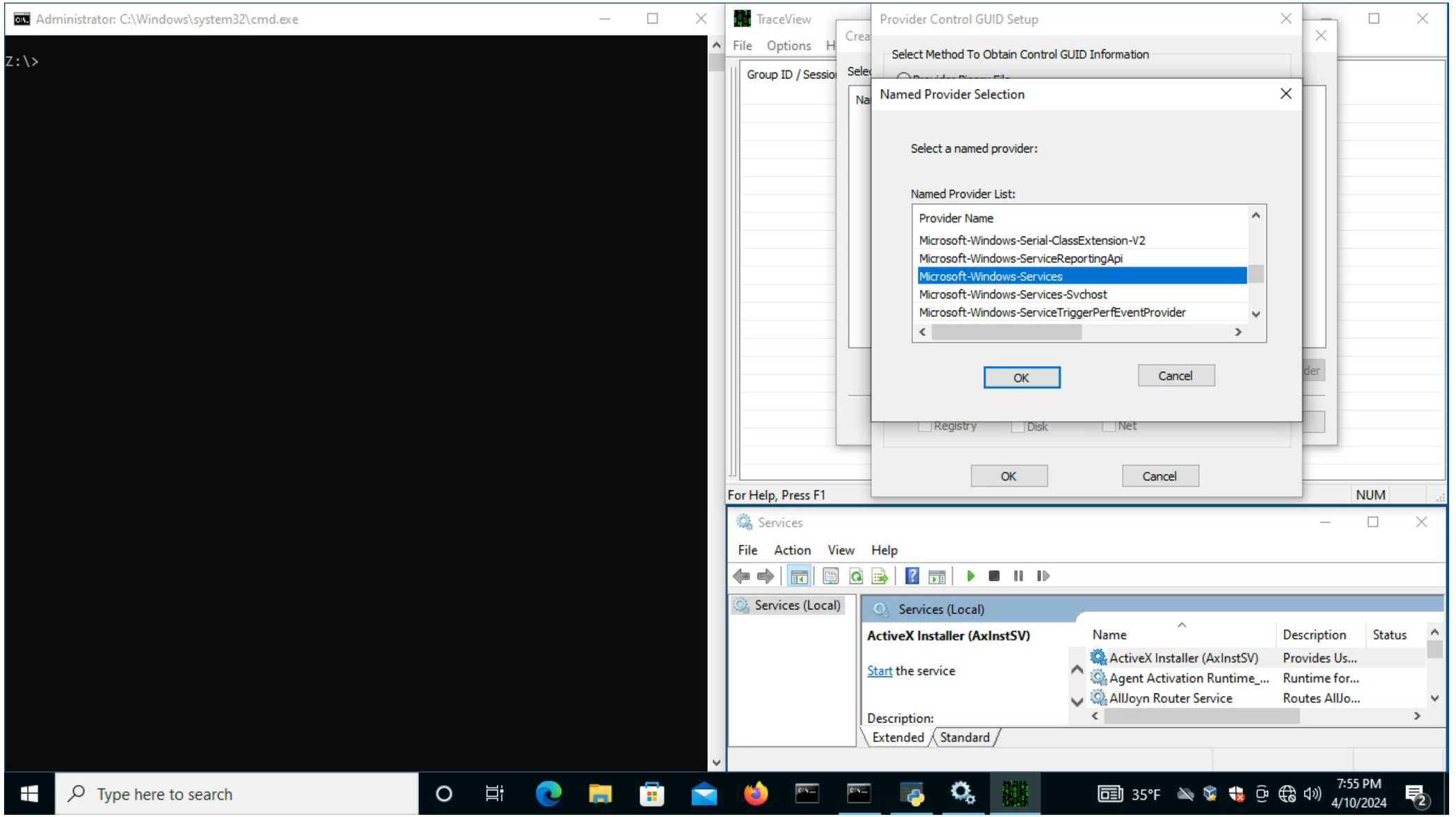
FudModule 2.0

0x80 – Event Tracing for Windows: Provider GUIDs

- Contains a hardcoded list of 95 GUIDs
- Zero out four masks, namely **EnableMask, GroupEnableMask, HostEnableMask**, and **HostGroupEnableMask**

```
Keyword = EventDescriptor->Keyword;
if ( GuidEntry->ProviderEnableInfo.IsEnabled )
{
  Level = GuidEntry->ProviderEnableInfo.Level;
  if ( (EventDescriptor->Level <= Level || !Level)
    && ((GuidEntry->ProviderEnableInfo.EnableProperty & 0x40) != 0 && !Keyword
     || (Keyword & GuidEntry->ProviderEnableInfo.MatchAnyKeyword) != 0
     && (Keyword & GuidEntry->ProviderEnableInfo.MatchAllKeyword) == GuidEntry->
  {
    return 1;
  }
}
if ( RegHandle->GroupEnableMask
  && (unsigned __int8)EtwpLevelKeywordEnabled(
                        &RegHandle->GroupEntry->ProviderEnableInfo,
                        EventDescriptor->Level,
                        EventDescriptor->Keyword) )
```

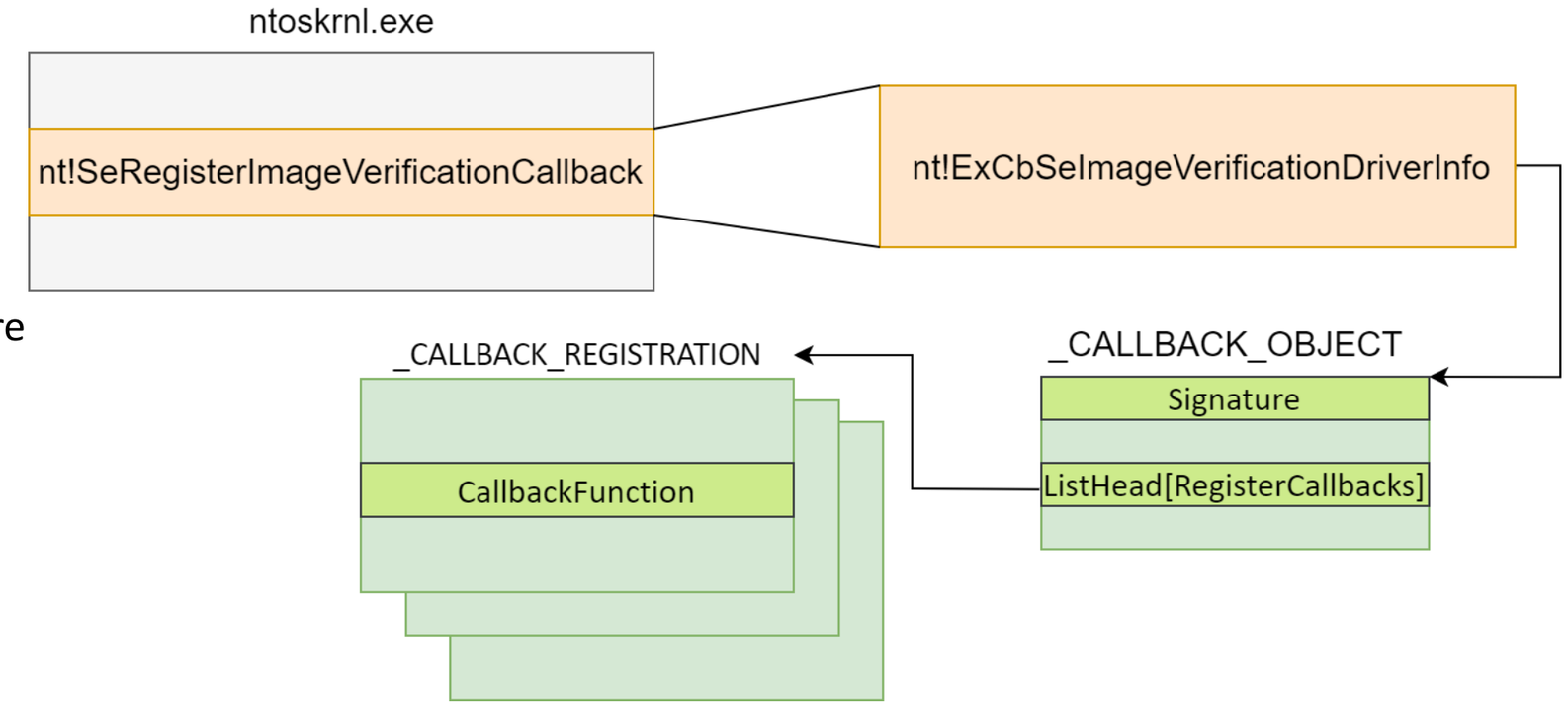



Administrator: C:\Windows\system32\cmd.exe
Z:\>
TraceView
File Options H
Group ID / Sessio
Provider Control GUID Setup
Select Method To Obtain Control GUID Information
Named Provider Selection
Select a named provider:
Named Provider List:
Provider Name
Microsoft-Windows-Serial-ClassExtension-V2
Microsoft-Windows-ServiceReportingApi
Microsoft-Windows-Services
Microsoft-Windows-Services-Svchost
Microsoft-Windows-ServiceTriggerPerfEventProvider
OK
Cancel
Registry
Disk
Net
OK
Cancel
For Help, Press F1
NUM
Services
File Action View Help
Services (Local)
Services (Local)
ActiveX Installer (AxInstSV)
Start the service
Description:
Name
Description
Status
ActiveX Installer (AxInstSV)
Provides Us...
Agent Activation Runtime_...
Runtime for...
AllJoyn Router Service
Routes AllJo...
Extended
Standard
Type here to search
35°F
7:55 PM
4/10/2024

FudModule 2.0

0x100 – Image Verification Callbacks

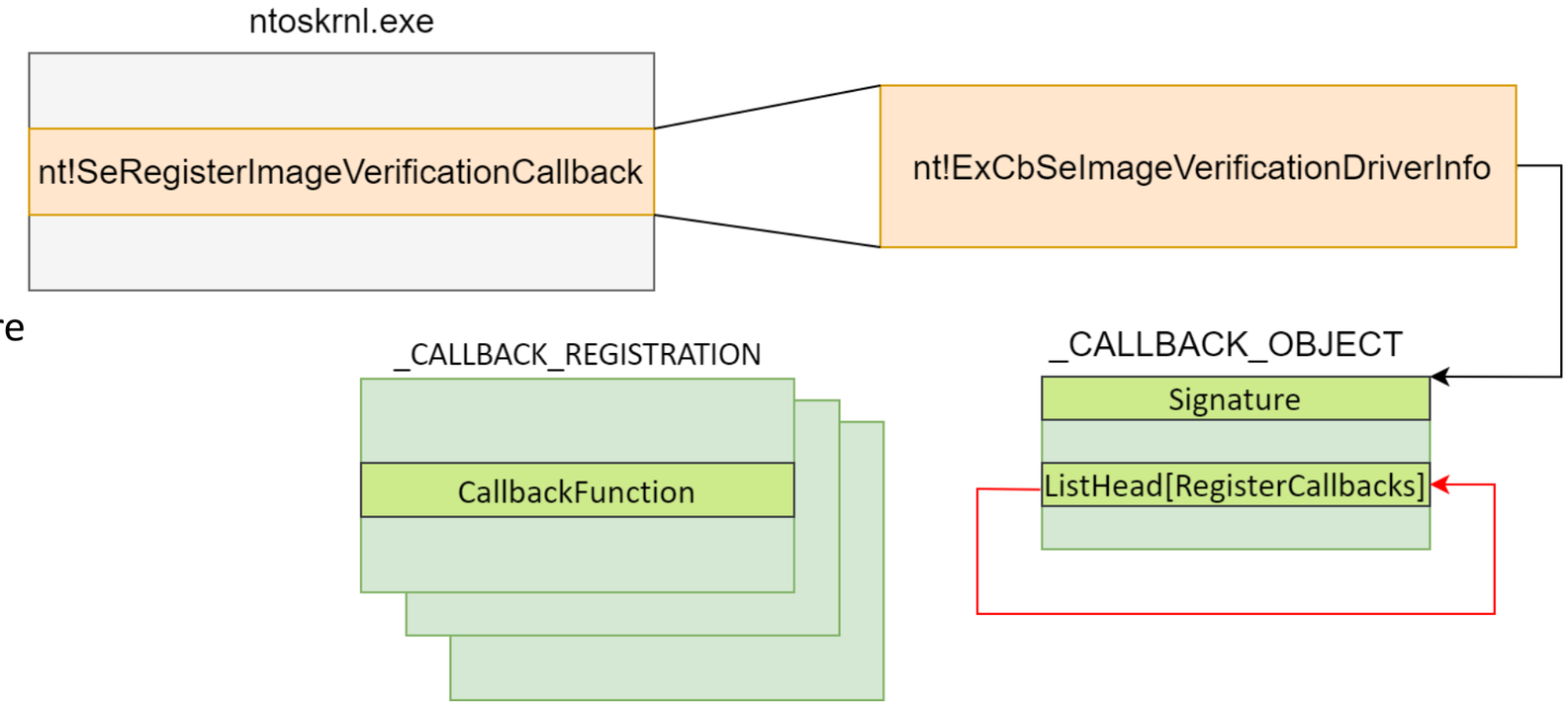
- Invoked whenever a new driver Image is loaded into a kernel memory
- Useful functionality for anti-malware software to block malicious or vulnerable drivers
- SeRegisterImageVerificationCallback (registering callback)



FudModule 2.0

0x100 – Image Verification Callbacks

- Invoked whenever a new driver Image is loaded into a kernel memory
- Useful functionality for anti-malware software to block malicious or vulnerable drivers
- SeRegisterImageVerificationCallback (registering callback)



FudModule 2.0

0x200 – Direct Attacks on Security Software

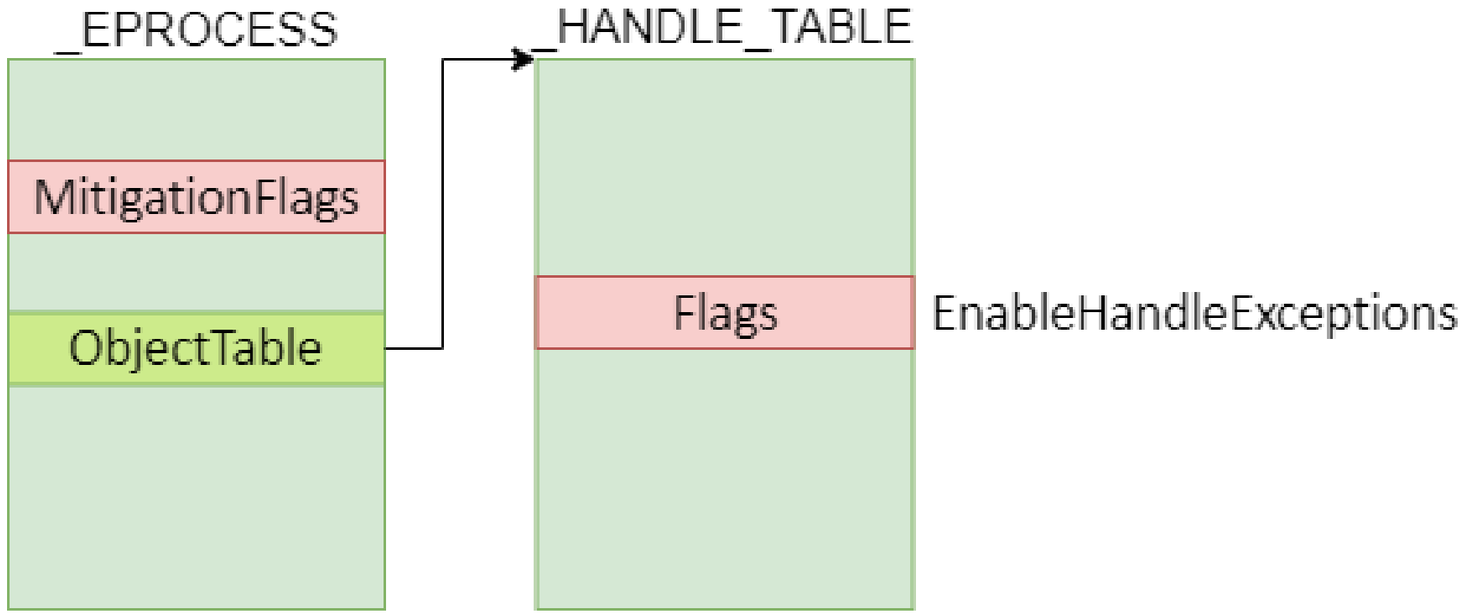
- _EPROCESS of asdsvc.exe (AhnLab Smart Defense Service)
- Targeting security solutions: AhnLab V3 Endpoint Security
- This modification makes it just a regular non-protected process
- Its opened up for further attacks from user mode
- Disrupt the link between user-mode and kernel-mode components

```
struct _EPROCESS
{
   struct _KPROCESS Pcb;
   struct _EX_PUSH_LOCK ProcessLock;
   VOID* UniqueProcessId;
   struct _LIST_ENTRY ActiveProcessLinks;
   struct _EX_RUNDOWN_REF RundownProtect;
   ...
   struct _EJOB* ServerSilo;
   UCHAR SignatureLevel;
   UCHAR SectionSignatureLevel;
   struct _PS_PROTECTION Protection;
   UCHAR HangCount:3;
   UCHAR GhostCount:3;
   UCHAR PrefilterException:1;

   ...
}
```

FudModule 2.0

0x200 – Direct Attacks on Security Software

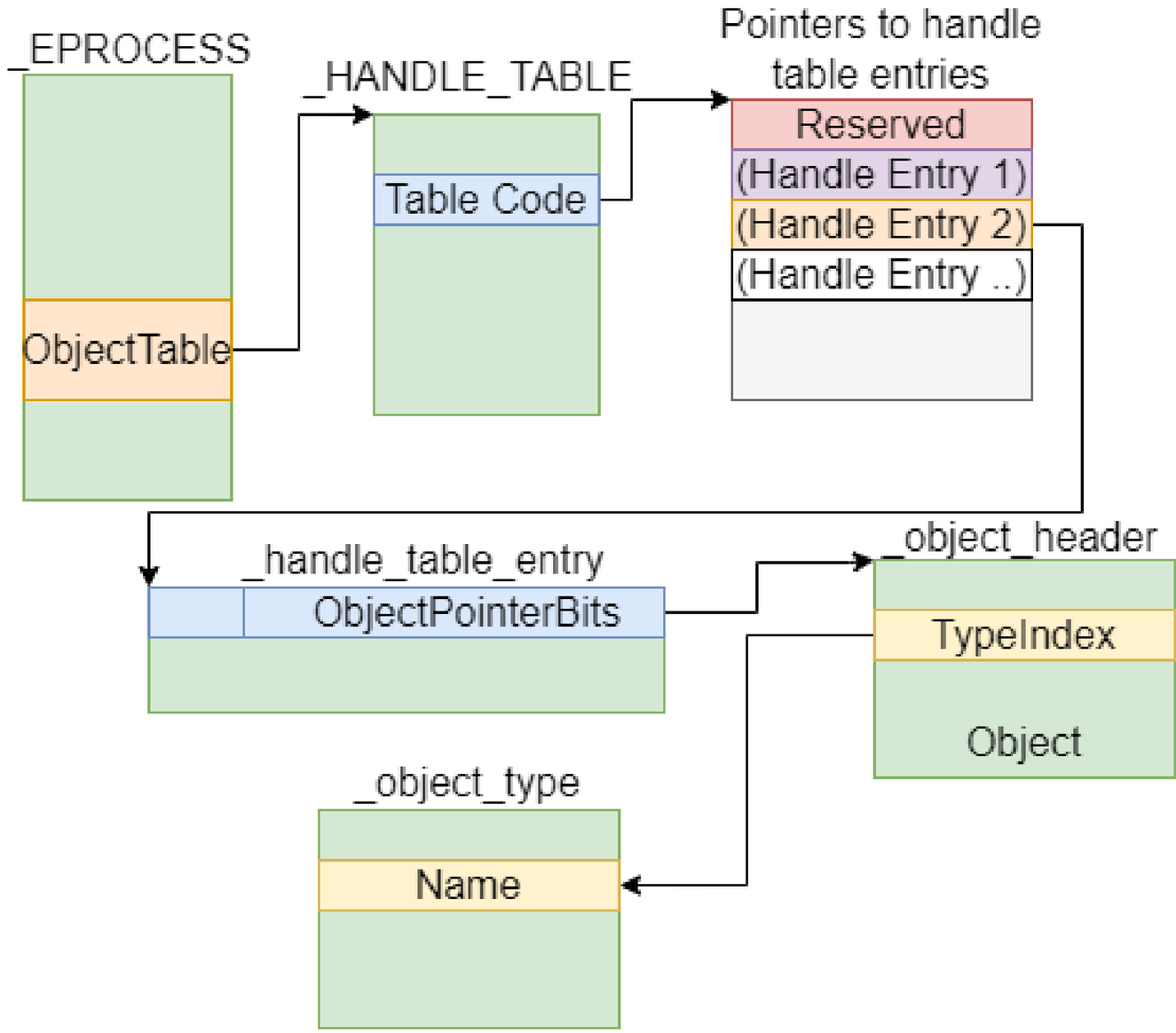
- This is used to increase stability
- Leaks its own _EPROCESS structure
- Zeroes out MitigationFlags
- Clears “EnableHandleExceptions” flag from “_EPROCESS.ObjectTable.Flags”



FudModule 2.0

0x200 – Direct Attacks on Security Software

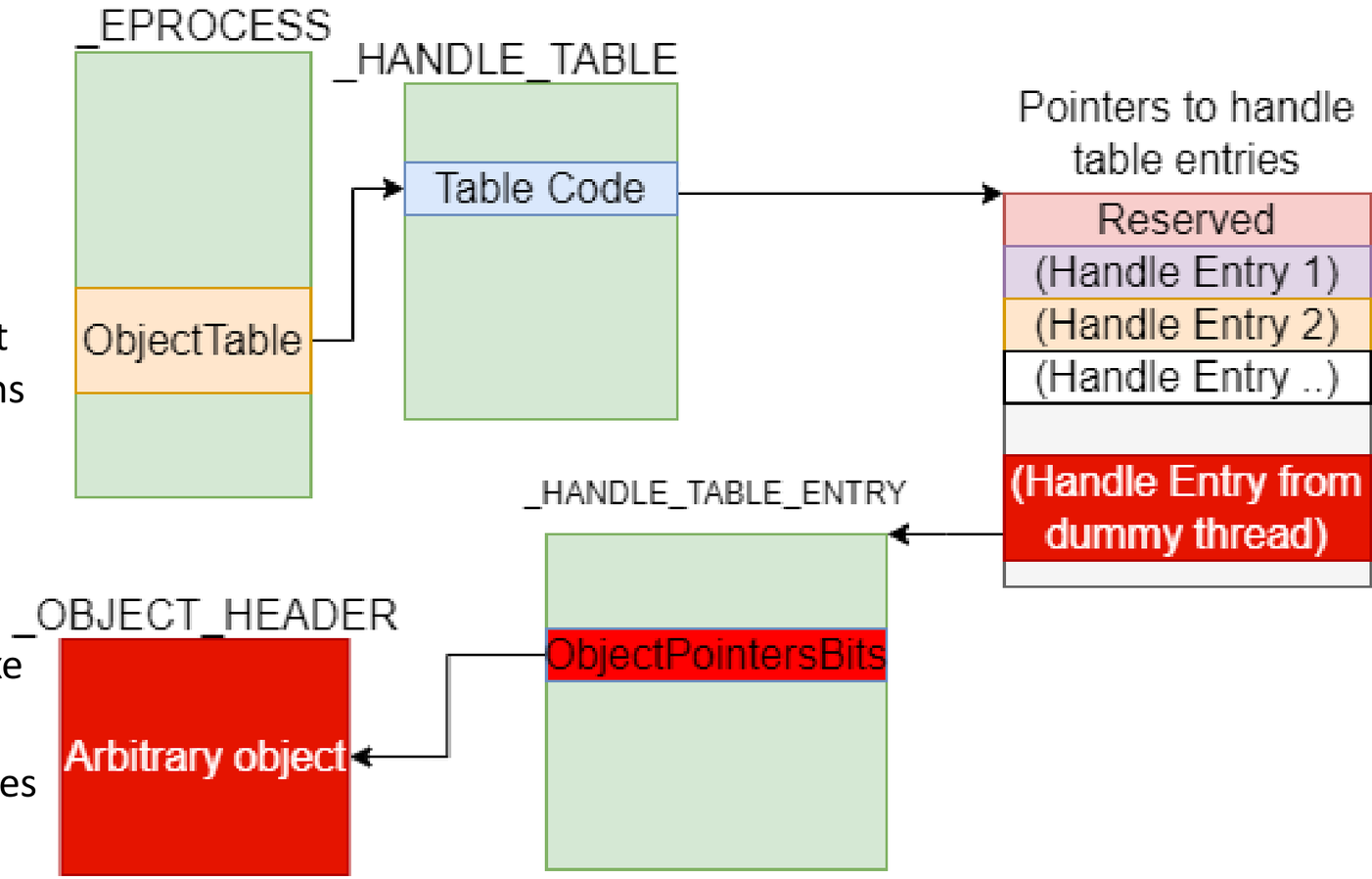
- What is the Handle Table used for?
- Kernel must be able to translate the handle to the corresponding object



FudModule 2.0

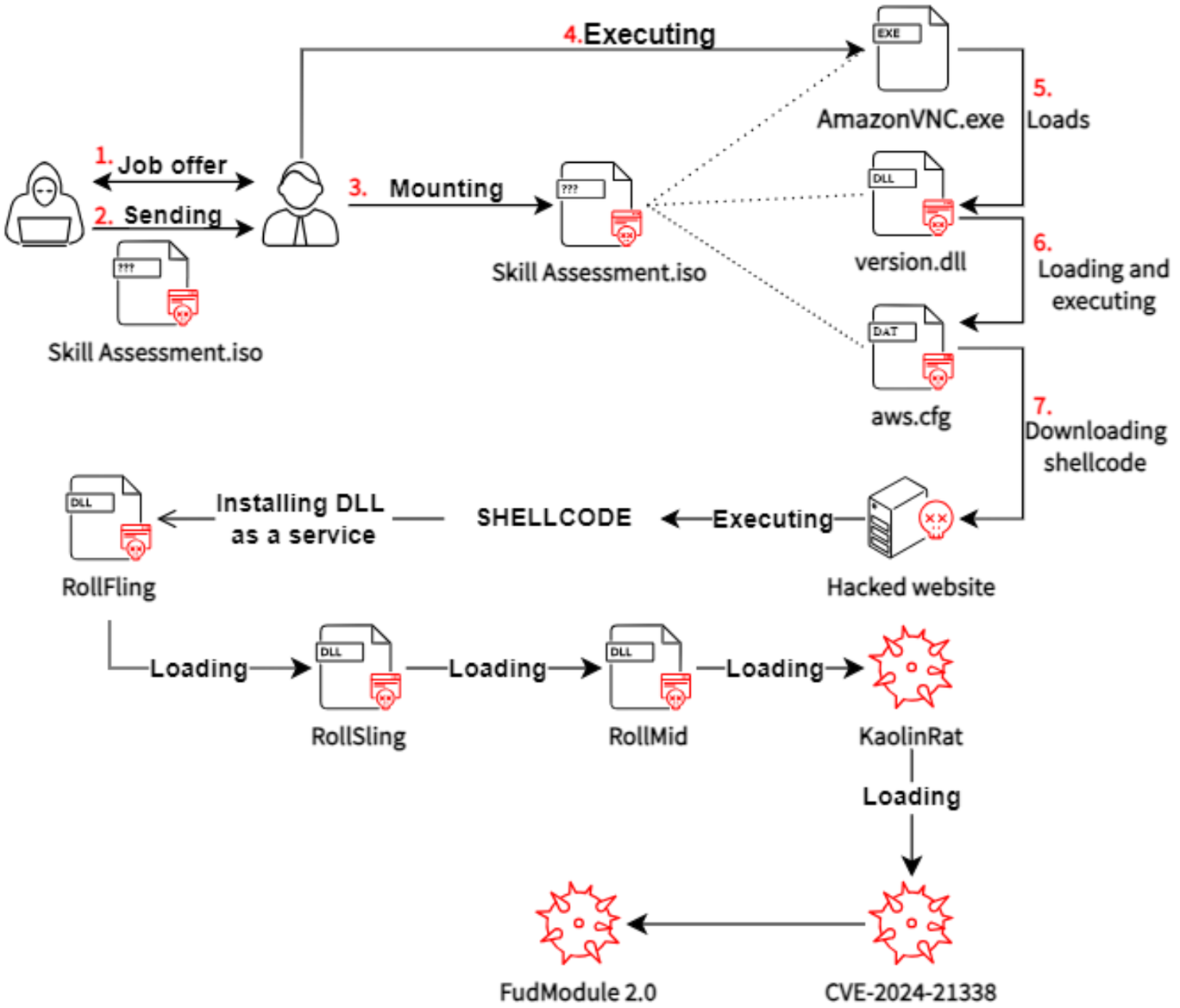
0x200 – Direct Attacks on Security Software

- Create a dummy thread with “THREAD_ALL_ACCESS” rights
- Modifying ObjectPointerBits
- This will make the handle reference that arbitrary object and enable the rootkit to perform a privileged operations on it
- Target _EPROCESS structure on one of the targeted processes MsSense.exe(Windows Defender), MsMpEng.exe (Malware Protection Engine),CSFalconService.exe (CrowdStrike), Hmpalert.exe (HitmanPro)
- Suspending process and all threads for targeted processes



Conclusion

- Lazarus group is investing significant resources
- Despite various mitigations, the kernel-based security solutions remain vulnerable
- Lazarus despite sophisticated attacks is still using phishing as an infection vector




black hat
ASIA 2024
Thank you
Gen
#BHASIA @BlackHatEvents