POPKORN: Popping Windows Kernel Drivers At Scale

Rajat Gupta, **Lukas Dresel**, Noah Spahn,
Giovanni Vigna, Christopher Kruegel, Taesoo Kim

lukas.dresel@cs.ucsb.edu
@dreselli

Dec. 9th, 2022, Austin, TX, USA

Summary

- Vulnerable Windows Kernel drivers are crucial for modern malware

- Straight-forward analysis techniques can effectively find high-impact vulnerabilities

- Our Prototype POPKORN found 27 vulnerabilities
  - 2 CVEs and 6 acknowledgements from driver vendors

Robinhood Ransomware Borrow Vulnerable Driver To Kill Antivirus and Encrypt Windows System Files

By BALAJI N - February 10, 2020

Ransomware

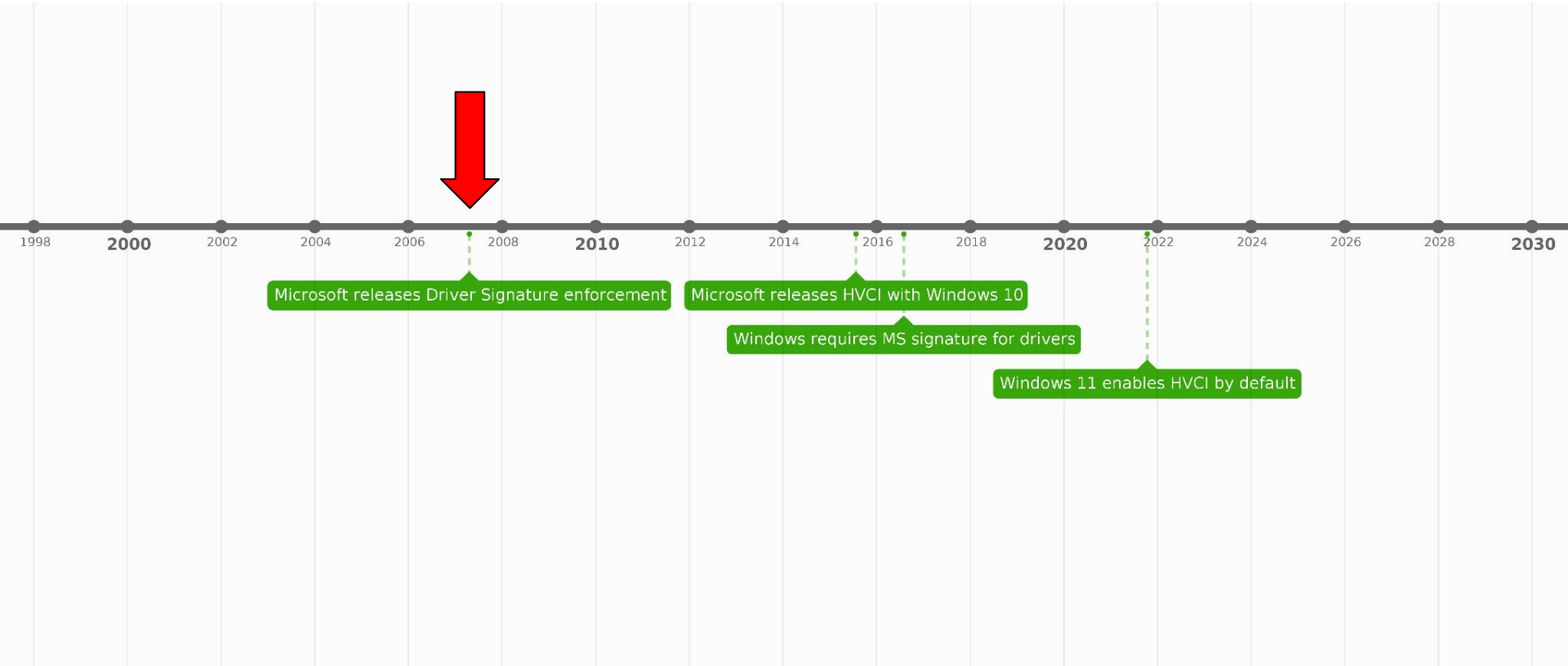
AvosLocker Ransomware Variant Abuses Driver File to Disable Antivirus, Scans for Log4shell

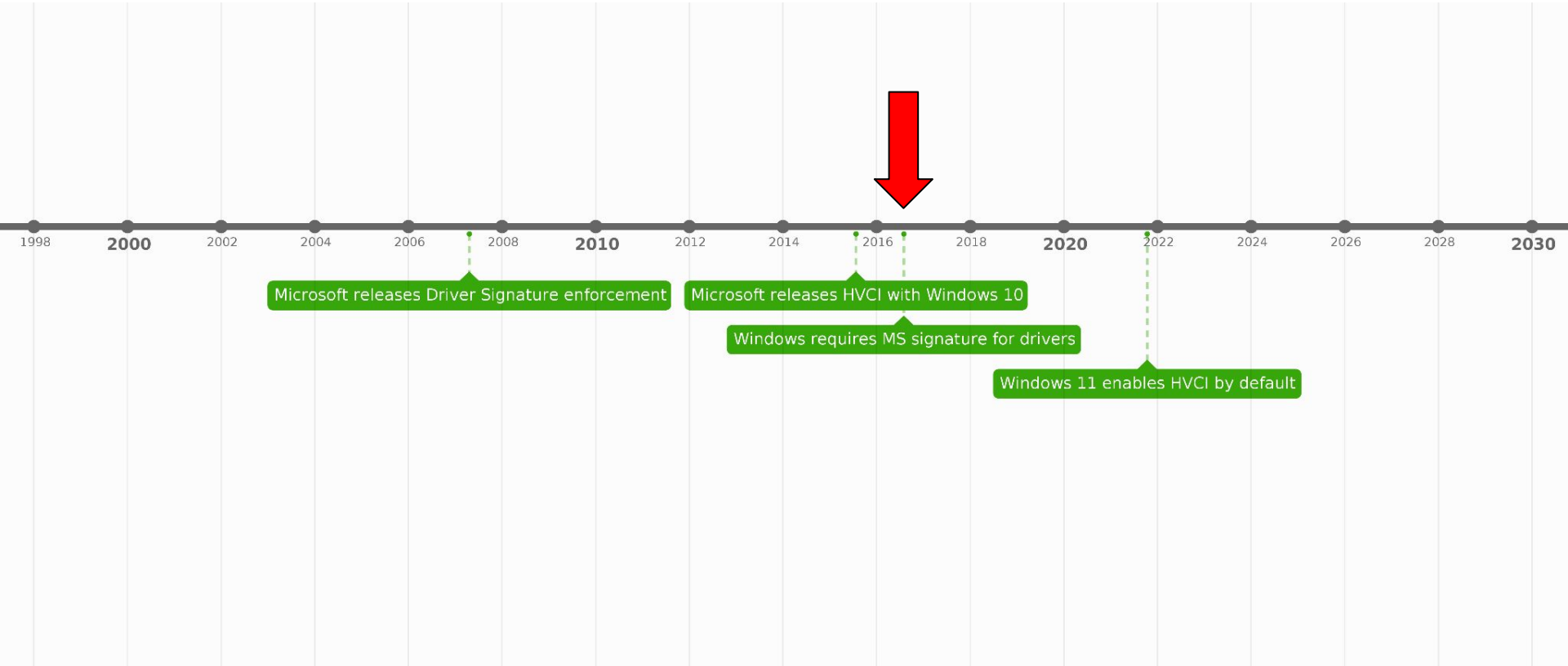
We found an AvosLocker ransomware variant using a legitimate antivirus component to disable detection and blocking solutions.

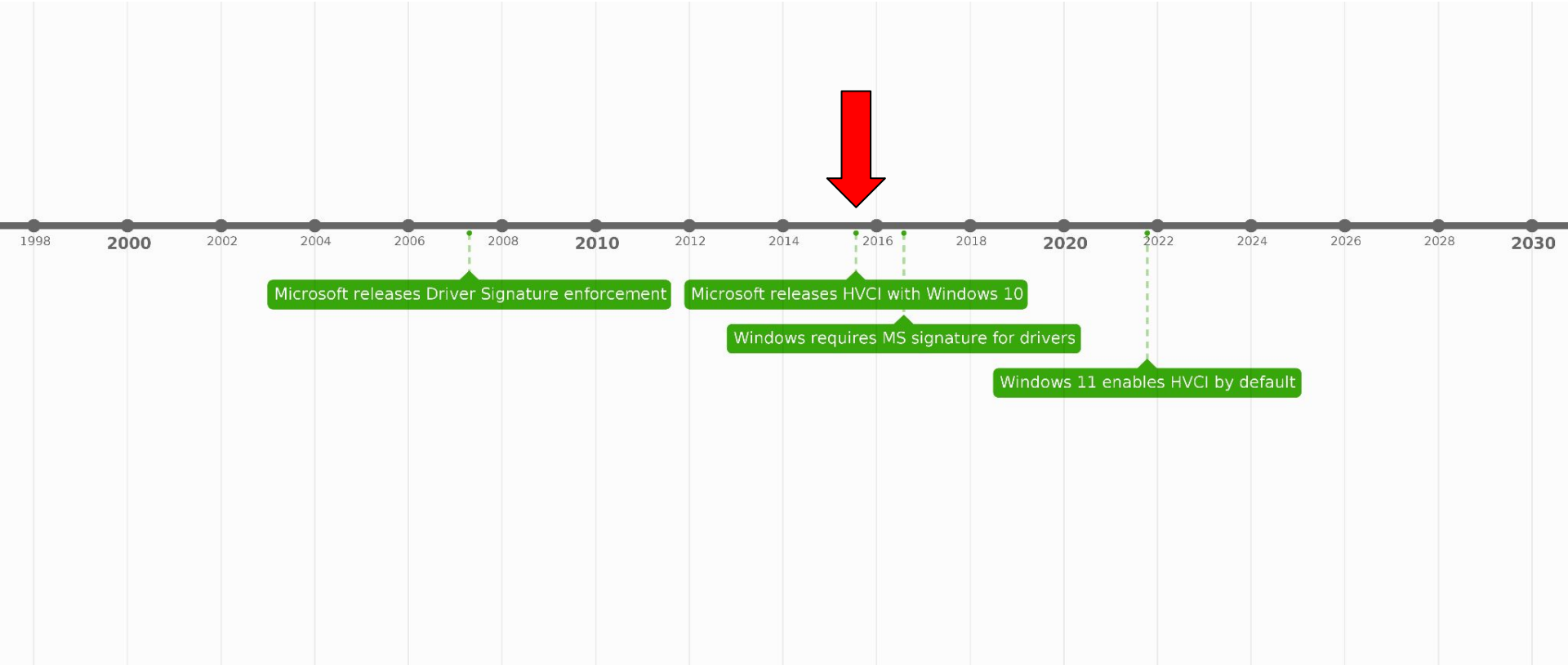
By: Christoper Ordonez, Alvin Nieto
May 02, 2022
Read time: 7 min (1825 words)

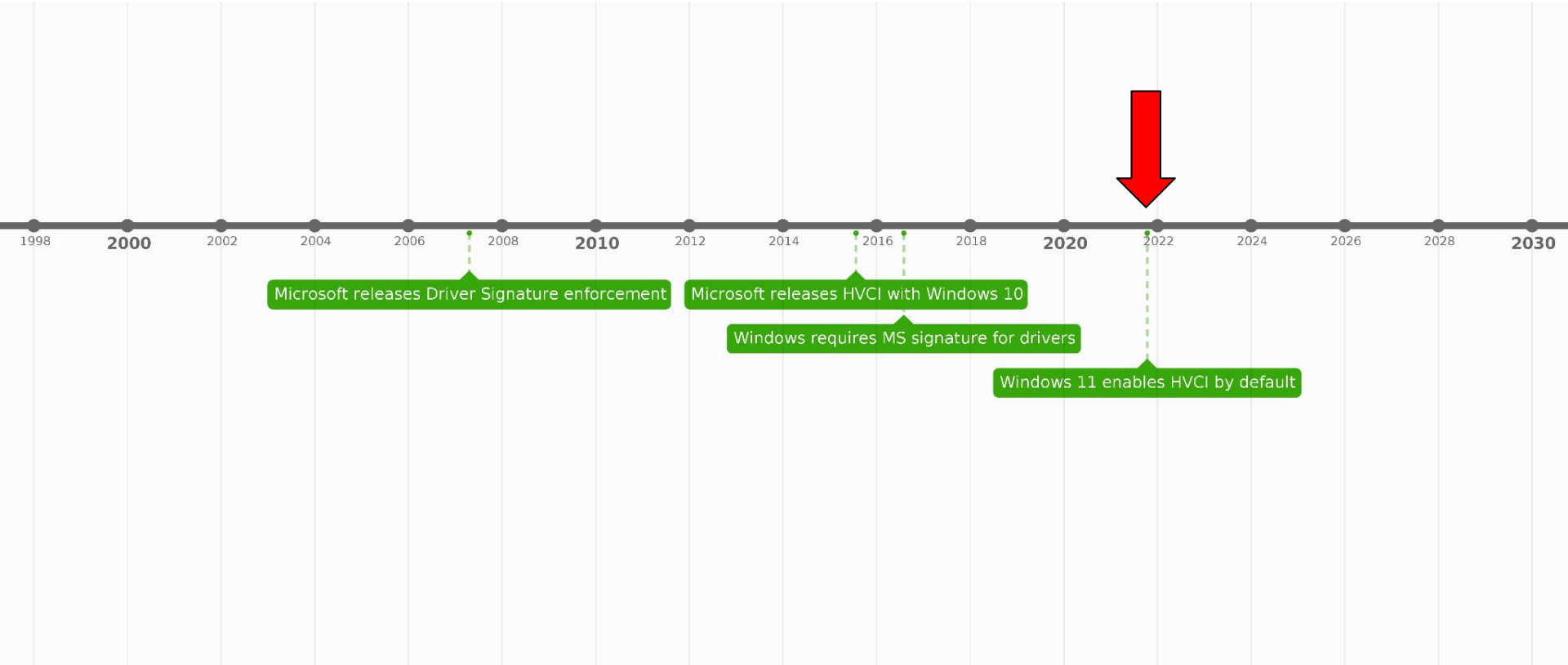

Robinhood Ransomware Borrow Vulnerable
Driver To Kill Antivirus
Systen
By BALAJI N
Ransomware
AvosLocker Ransomware Variant Abuses Driver File to Disable Antivirus, Scans for Log4shell
gitimate antivirus component to disable detection and blocking solutions.
Home | Security
Lazarus APT Abuses Vulnerable Dell Drivers to Bypass Windows Security
Rabia Noureen | OCT 7, 2022

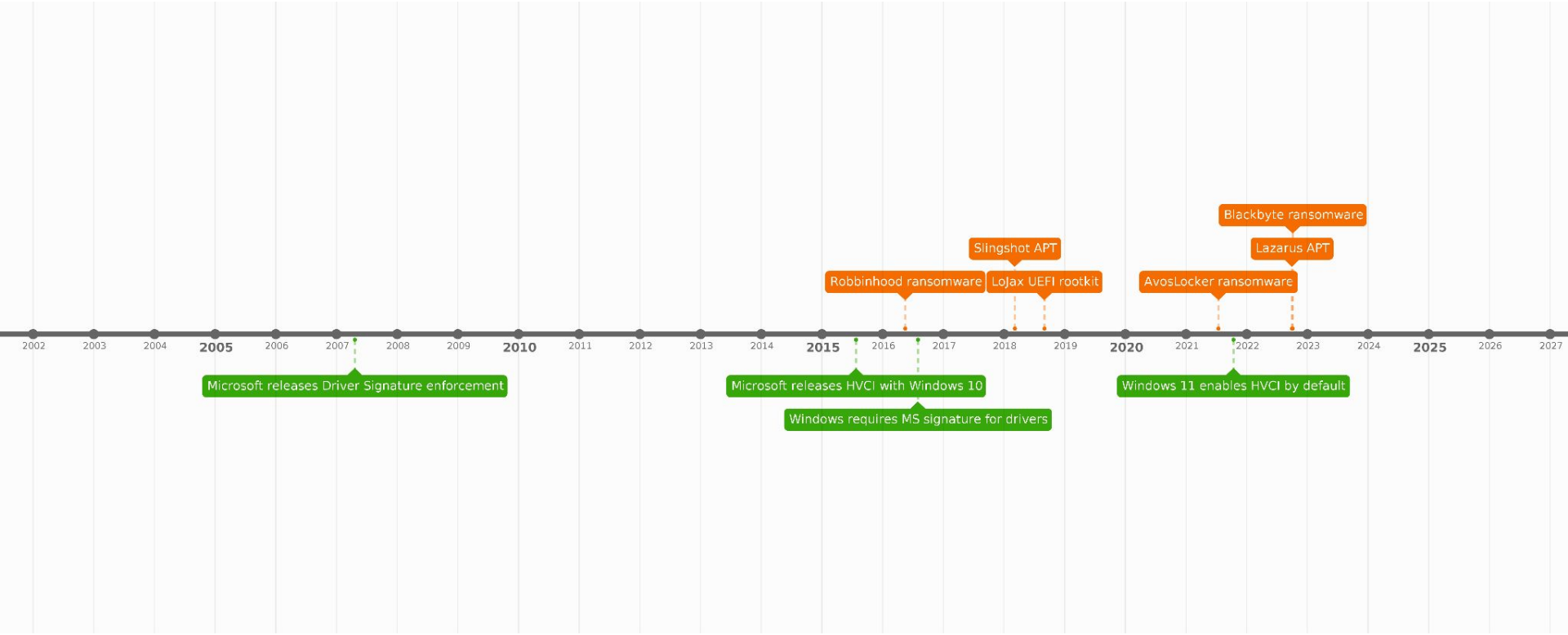

Robinhood Ransomware Borrow Vulnerable
Driver To Kill Antivirus
Syste
By BALAJI N
Ransomware
AvosLocker Ransomware Variant Abuses Driver File to Disable Antivirus, Scans for Log4shell
gitimate antivirus component to disable detection and blocking solutions.
Home | Security
Lazarus APT Abuses Vulnerable Dell Drivers to Bypass Windows Security
Rabia Noureen | OCT 7, 2022
BlackByte Ransomware Abuses Vulnerable Windows Driver to Disable Security Solutions
Oct 07, 2022
Ravie Lakshmanan


1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
2026
2028
2030
Microsoft releases Driver Signature enforcement
Microsoft releases HVCI with Windows 10
Windows requires MS signature for drivers
Windows 11 enables HVCI by default


1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
2026
2028
2030
Microsoft releases Driver Signature enforcement
Microsoft releases HVCI with Windows 10
Windows requires MS signature for drivers
Windows 11 enables HVCI by default


1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
2026
2028
2030
Microsoft releases Driver Signature enforcement
Microsoft releases HVCI with Windows 10
Windows requires MS signature for drivers
Windows 11 enables HVCI by default


1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
2026
2028
2030
Microsoft releases Driver Signature enforcement
Microsoft releases HVCI with Windows 10
Windows requires MS signature for drivers
Windows 11 enables HVCI by default


Blackbyte ransomware
Lazarus APT
Slingshot APT
Robbinhood ransomware
LoJax UEFI rootkit
AvosLocker ransomware
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
Microsoft releases Driver Signature enforcement
Microsoft releases HVCI with Windows 10
Windows requires MS signature for drivers
Windows 11 enables HVCI by default

POPKORN

```
1  void* SectionHandle = ZwOpenSection(
2    GENERIC_WRITE,
3    "\\Device\\PhysicalMemory",
4  );
5
6  void* user_input = IRP->SystemBuffer;
7
8  (SectionOffset, BaseAddress) = ZwMapViewOfSection(
9    SectionHandle,
10   ZwCurrentProcess(),
11   *(size_t*)user_input, // offset
12   *(size_t)user_input+8, // size
13 );
14
15 *(size_t*)user_input = SectionOffset;
16 *(void**)user_input + 8 = BaseAddress;
```

POPKORN

```
1  void* SectionHandle = ZwOpenSection(
2    GENERIC_WRITE,
3    "\\Device\\PhysicalMemory",
4  );
5
6  void* user_input = IRP->SystemBuffer;
7
8  (SectionOffset, BaseAddress) = ZwMapViewOfSection(
9    SectionHandle,
10   ZwCurrentProcess(),
11   *(size_t*)user_input, // offset
12   *(size_t)user_input+8, // size
13 );
14
15 *(size_t*)user_input = SectionOffset;
16 *(void**)user_input + 8 = BaseAddress;
```

POPKORN

```
1  void* SectionHandle = ZwOpenSection(
2    GENERIC_WRITE,
3    "\\Device\\PhysicalMemory",
4  );
5
6  void* user_input = IRP->SystemBuffer;
7
8  (SectionOffset, BaseAddress) = ZwMapViewOfSection(
9    SectionHandle,
10   ZwCurrentProcess(),
11   *(size_t*)user_input, // offset
12   *(size_t)user_input+8, // size
13 );
14
15 *(size_t*)user_input = SectionOffset;
16 *(void**)user_input + 8 = BaseAddress;
```

POPKORN

```
1  void* SectionHandle = ZwOpenSection(
2    GENERIC_WRITE,
3    "\\Device\\PhysicalMemory",
4  );
5
6  void* user_input = IRP->SystemBuffer;
7
8  (SectionOffset, BaseAddress) = ZwMapViewOfSection(
9    SectionHandle,
10   ZwCurrentProcess(),
11   *(size_t*)user_input, // offset
12   *(size_t)user_input+8, // size
13 );
14
15 *(size_t*)user_input = SectionOffset;
16 *(void**)user_input + 8 = BaseAddress;
```

POPKORN - Goals

- Lightweight analysis that can scale to large datasets of drivers

POPKORN - Goals

- Lightweight analysis that can scale to large datasets of drivers

- Exploitable & high-impact bugs
  - Arbitrary physical memory access
  - Arbitrary process access

POPKORN - Goals

- Lightweight analysis that can scale to large datasets of drivers

- Exploitable & high-impact bugs
  - Arbitrary physical memory access
  - Arbitrary process access

- Easily verifiable bug reports with high-confidence

POPKORN

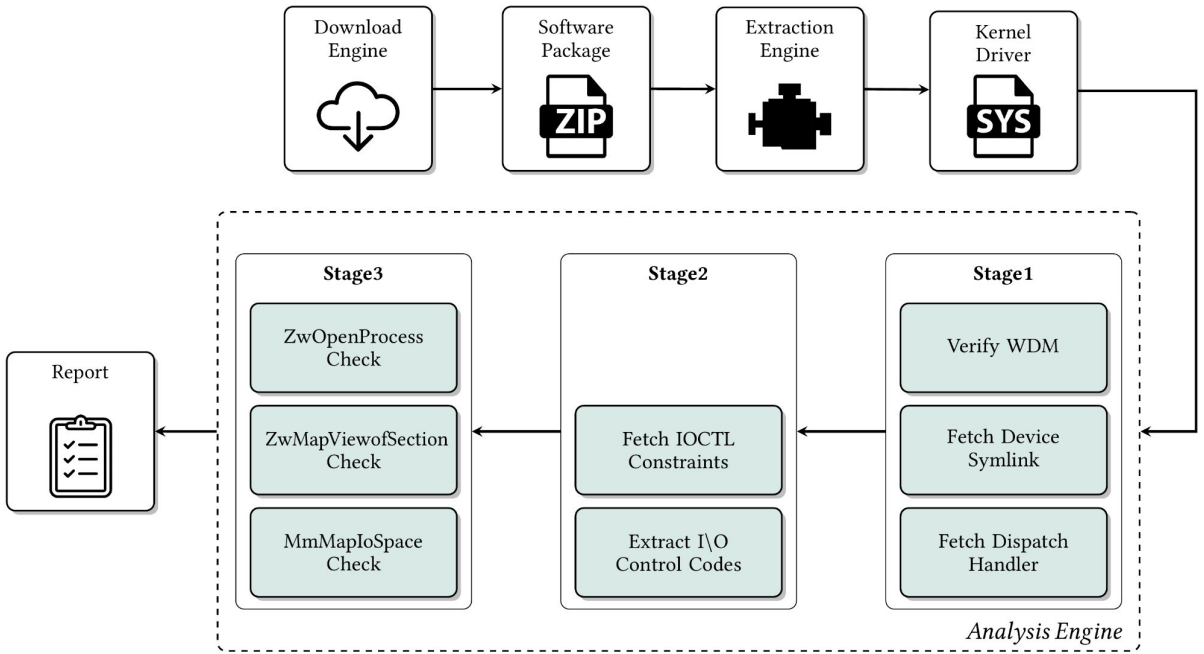


Figure 2: POPKORN System Overview.

POPKORN

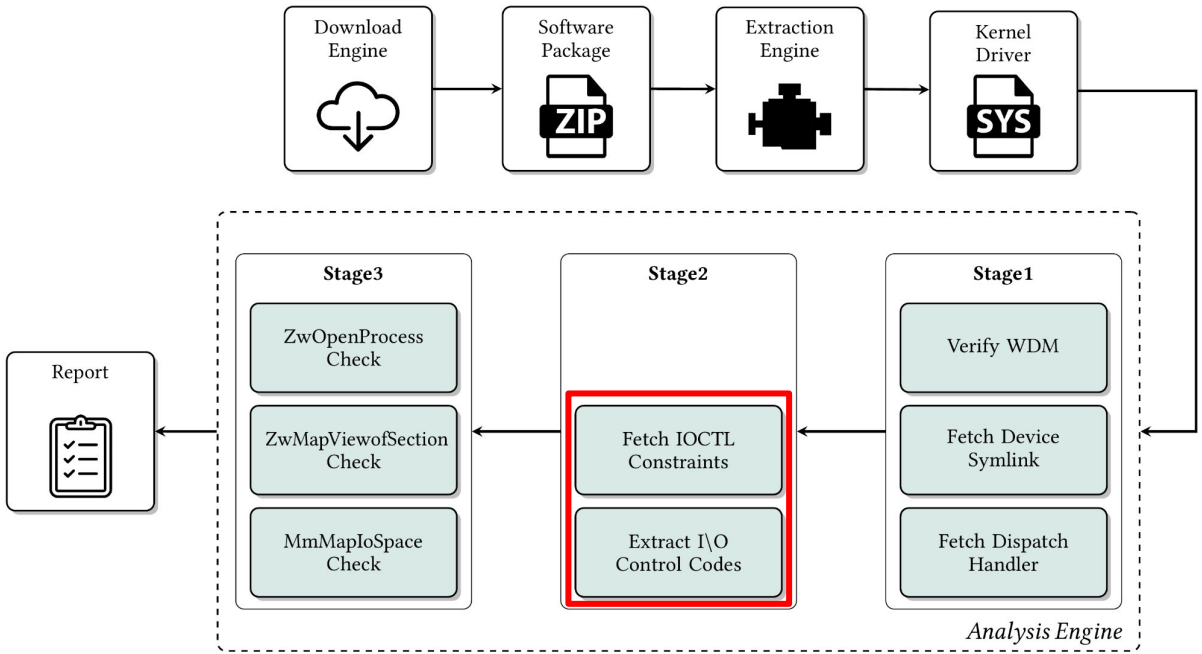


Figure 2: POPKORN System Overview.

POPKORN

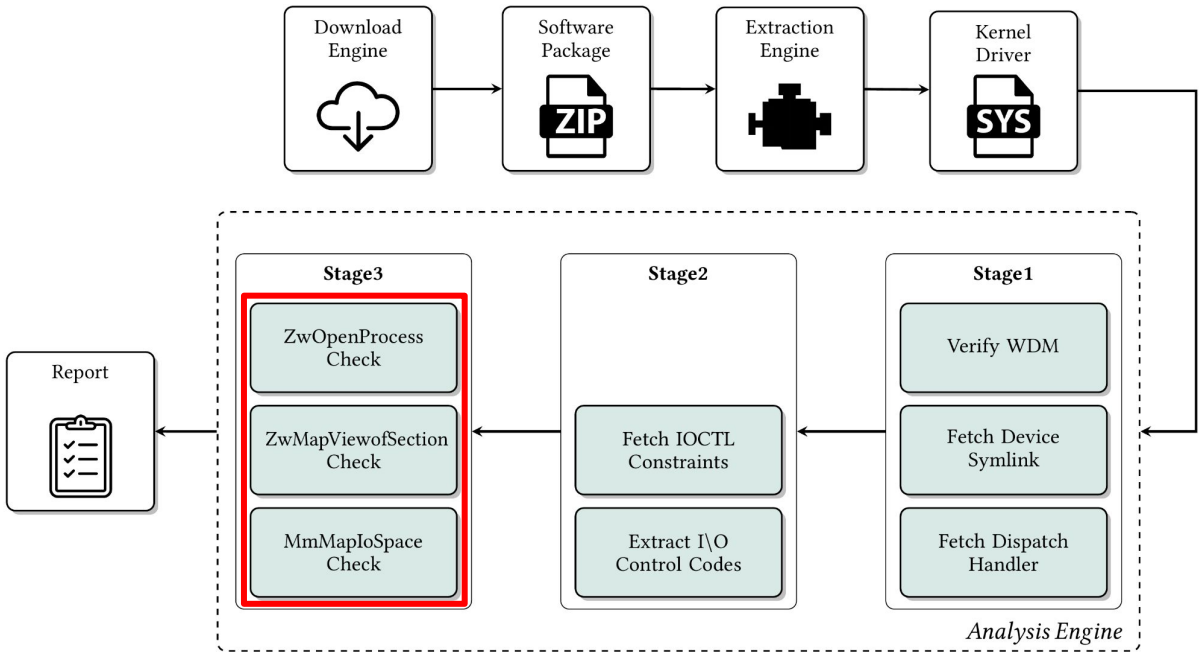


Figure 2: POPKORN System Overview.

Evaluation

- 90,000 software packages
- 5,000 Windows kernel drivers
- 3,094 WDM drivers (62%)
- 271 drivers with sink functions
  - 212 unique

Evaluation

	Using Function		Vulnerable Usages	
Kernel Function	Total	Dedup.	Total	Dedup.
MmMapIoSpace	240 (7.76%)	188	24	17
ZwMapViewOfSection	40 (1.29%)	32	17	12
ZwOpenProcess	14 (0.45%)	11	0	0
Total (Unique)	271 (8.76%)	212	38	27

Table 2: Drivers using specific target functions, and drivers found vulnerable by POPKORN, out of a total of 3,094 WDM drivers.

Evaluation - Known vulnerabilities

- False negative analysis was performed using known-vulnerable drivers from the physmem repository https://github.com/namazso/physmem_drivers
- 30 unique vulnerable drivers with

Analysis Result	#Drivers
Reported vulnerability	20 (+1*)
Timed out	6
Analysis terminated	4

Table 3: POPKORN results for `physmem_drivers`

Conclusion

Conclusion

- Implemented prototype analysis POPKORN to demonstrate how targeted analyses can scale to automated deployment

Conclusion

- Implemented prototype analysis POPKORN to demonstrate how targeted analyses can scale to automated deployment
- Detected 27 unique vulnerable kernel drivers in our real-world dataset

Conclusion

- Implemented prototype analysis POPKORN to demonstrate how targeted analyses can scale to automated deployment
- Detected 27 unique vulnerable kernel drivers in our real-world dataset
- Received two CVEs and six acknowledgements from vendors

Conclusion

- Implemented prototype analysis POPKORN to demonstrate how targeted analyses can scale to automated deployment
- Detected 27 unique vulnerable kernel drivers in our real-world dataset
- Received two CVEs and six acknowledgements from vendors
- Code and evaluation dataset on GitHub under ucsb-seclab/popkorn-artifact

Conclusion

- Implemented prototype analysis POPKORN to demonstrate how targeted analyses can scale to automated deployment
- Detected 27 unique vulnerable kernel drivers in our real-world dataset
- Received two CVEs and six acknowledgements from vendors
- Code and evaluation dataset on GitHub under [ucsb-seclab/popkorn-artifact]

	Using Function		Vulnerable Usages	
Kernel Function	Total	Dedup.	Total	Dedup.
MmMapIoSpace	240 (7.76%)	188	24	17
ZwMapViewOfSection	40 (1.29%)	32	17	12
ZwOpenProcess	14 (0.45%)	11	0	0
Total (Unique)	271 (8.76%)	212	38	27

Table 2: Drivers using specific target functions, and drivers found vulnerable by POPKORN, out of a total of 3,094 WDM drivers.

Analysis Result	#Drivers
Reported vulnerability	20 (+1*)
Timed out	6
Analysis terminated	4

Table 3: POPKORN results for `physmem_drivers`

Conclusion

lukas.dresel@cs.ucsb.edu

- Implemented prototype analysis POPKORN to demonstrate how targeted analyses can scale to automated deployment
- Detected 27 unique vulnerable kernel drivers in our real-world dataset
- Received two CVEs and six acknowledgements from vendors
- Code and evaluation dataset on GitHub under [ucsb-seclab/popkorn-artifact]

	Using Function		Vulnerable Usages	
Kernel Function	Total	Dedup.	Total	Dedup.
MmMapIoSpace	240 (7.76%)	188	24	17
ZwMapViewOfSection	40 (1.29%)	32	17	12
ZwOpenProcess	14 (0.45%)	11	0	0
Total (Unique)	271 (8.76%)	212	38	27

Table 2: Drivers using specific target functions, and drivers found vulnerable by POPKORN, out of a total of 3,094 WDM drivers.

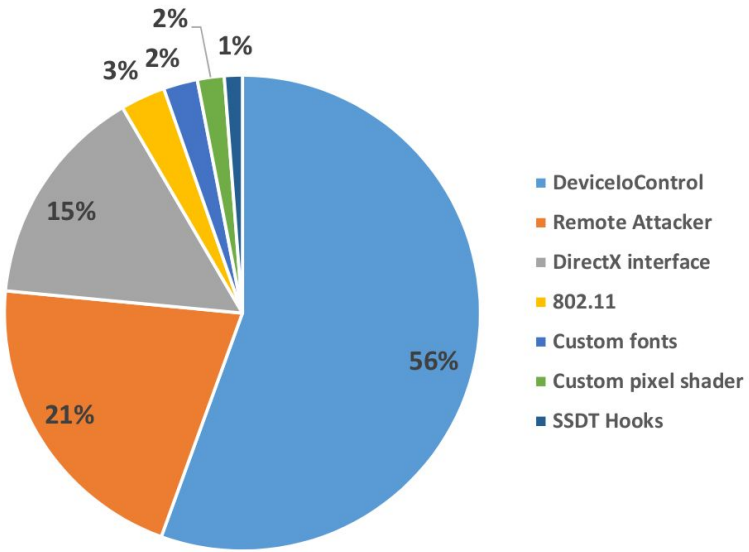
Analysis Result	#Drivers
Reported vulnerability	20 (+1*)
Timed out	6
Analysis terminated	4

Table 3: POPKORN results for `physmem_drivers`

References & Resources

- g_CiOptions in a Virtualized World - XPN InfoSec Blog
- The Swan Song for Driver Signature Enforcement Tampering | Fortinet Blog
- Microsoft signed a malicious Netfilter rootkit
- Defeating Windows Driver Signature Enforcement #1: default drivers | j00ru//vx tech blog
- BlueHat IL 2018 - David Weston - Windows Hardening with Hardware
- https://github.com/ucsb-seclab/popkorn-artifact

Kernel Driver Attack Surface in CVEs



“*A non-administrative user mode process cannot access or tamper with kernel code and data. Administrator-to-kernel is not a security boundary.*“

- Microsoft’s Security Servicing Criteria

POPKORN

```
1  NTSTATUS DeviceControlDispatch(_DEVICE_OBJECT *DeviceObject,
2                                 _IRP *Irp)
3  {
4    auto IoStackLocation = Irp->Tail.Overlay.CurrentStackLocation;
5    auto Params = IoStackLocation->Parameters.DeviceIoControl;
6    auto Buf = Irp->AssociatedIrp.SystemBuffer;
7    auto Status = STATUS_UNSUCCESSFUL;
8
9    if ( Params.IoControlCode == 0x9C402530
10     && Params.InputBufferLength >= 8
11     && Params.OutputBufferLength >= 8
12   )
13   {
14     // Map one page as determined by request
15    PHYSICAL_ADDRESS* Address = (PHYSICAL_ADDRESS*)Buf;
16
17    void* res = MmMapIoSpace(*Address, 0x1000, MmNonCached);
18
19    if (res) {
20       *(void**)Buf = res;
21       IofCompleteRequest(Irp, 0);
22       Status = STATUS_SUCCESS;
23    }
24   }
25   return Status;
26 }
```

POPKORN

```
1  NTSTATUS DeviceControlDispatch(_DEVICE_OBJECT *DeviceObject,
2                                 _IRP *Irp)
3  {
4    auto IoStackLocation = Irp->Tail.Overlay.CurrentStackLocation;
5    auto Params = IoStackLocation->Parameters.DeviceIoControl;
6    auto Buf = Irp->AssociatedIrp.SystemBuffer;
7    auto Status = STATUS_UNSUCCESSFUL;
8
9    if ( Params.IoControlCode == 0x9C402530
10     && Params.InputBufferLength >= 8
11     && Params.OutputBufferLength >= 8
12   )
13   {
14     // Map one page as determined by request
15     PHYSICAL_ADDRESS* Address = (PHYSICAL_ADDRESS*)Buf;
16
17     void* res = MmMapIoSpace(*Address, 0x1000, MmNonCached);
18
19     if (res) {
20       *(void**)Buf = res;
21       IofCompleteRequest(Irp, 0);
22       Status = STATUS_SUCCESS;
23     }
24   }
25   return Status;
26 }
```