C2 (Command and Control) systems

Thomas Hendrickson and Tom Hebb

What is a C2 system?

- Offensive tool to take persistent control of target computers
- Initially placed on targets using exploits, social engineering, etc
  - Initial attack vector is often limited in duration or capabilities
  - With a C2 system in place, scope and duration of access is nearly unrestricted
- Used to
  - Exfiltrate data (download files and configuration) from targets
  - Take over targets' computing resources (e.g. DDoS, bitcoin mining)

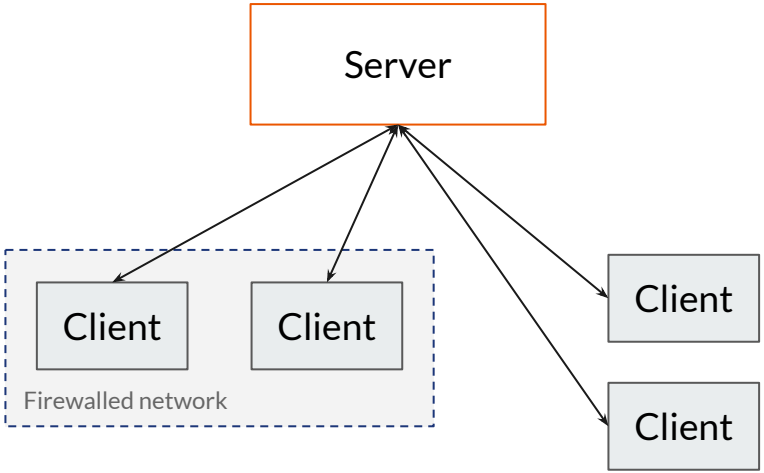
Who uses C2 systems?

- Penetration testers (good guys!)
  - Private contractors
    - Praetorian Cybersecurity (our senior design sponsor!)
    - SpecterOps (owned by founder of Cobalt Strike, a popular white-hat C2 system)
  - Corporate and military internal red teams
- Botnet masters (bad guys!)
  - Mirai (used in the attack on Dyn DNS in 2016)
- Government agencies (???)
  - CIA C2 systems leaked by WikiLeaks in 2014; NSA almost certainly has similar
  - Israeli Unit 8200 (Duqu, Stuxnet)

https://wikileaks.org/vault7/document/Assassin_v1_4_Users_Guide/ - CIA "Assassin" C2 system

C2 Architecture

- Server
  - Commands multiple clients, each running on an infected target
- Client
  - Supports, at a minimum, network communication and remote code execution.
  - May include other features or be extensible using modules



Typical client capabilities

- Filesystem traversal, upload/download
- Execute shell commands
- Load and execute of arbitrary code
- **Encrypt and/or disguise network traffic**
- **Hide self and loaded modules from detection**
  - Avoid disk operations and suspicious syscalls
  - Embed self within legitimate processes on the system

Network Evasion

Traversing Firewalls

- Problem: Firewall blocks all incoming and most outgoing connections
- Solution: Initiate connections from client
  - Connect to server through proxies to avoid lots of traffic to a single IP
  - Use common ports, like 80 (HTTP) and 443 (HTTPS)

Remaining Undetected

- Problem: Persistent sessions are easy to detect
- Solution: Use periodic check ins
  - All commands are asynchronous
  - Client checks in to receive new commands and deliver the results of old ones
  - Checks-ins can be scheduled to happen at randomized times

Remaining Undetected

- Problem: C2 traffic can be fingerprinted and blocked
- Solution: Disguise as other protocols
  - If on port 80, send a fake HTTP header before the data
  - If on port 443, use TLS just like HTTPS does
  - Can disguise as SMTP, DNS, etc, with encrypted data hidden where the payload usually goes

Memory Evasion

But first: DLLs!

What is a DLL (dynamic-link library)?

- Library of shared code that is made available to (“linked with”) programs with they run
- Contains code and data that can be used by more than one program at the same time
- Dynamic linking advantages

https://support.microsoft.com/en-us/help/815065/what-is-a-dll

How to make a .dll?

- (Nearly) Any compiled language can compile to a shared library instead of an executable
- For C, pass special linker options
- For Rust →

```rust
extern crate libc;

use libc::uint32_t;

#[no_mangle]
pub extern "C" fn mul_two(n: uint32_t) -> uint32_t
{
    2 * n
}
```

11,0-1 All

How to use a .dll?

- output: “result: 10”

```rust
extern crate libloading;

fn main() {
    let lib = libloading::Library::new("math.dll").unwrap();
    unsafe {
        let mul_two: libloading::Symbol<unsafe extern "C" fn(n: u32) -> u32> =
            lib.get(b"mul_two\0").unwrap();
        println!("result: {}", mul_two(5));
    }
}
```

Where are they used?

- Windows API
- Programming language implementations
  - Rust
  - Python
- etc...

https://support.microsoft.com/en-us/help/815065/what-is-a-dll
https://msdn.microsoft.com/en-us/library/windows/desktop/ee663297(v=vs.85).aspx

Process Hacker Demo

Avoiding Detection

- Antivirus solutions might be running on a target
- Be as silent as possible
- Client lifetime / C2 goals

Memory Evasion Techniques

- Remain in Memory
- Correct Memory Permissions
- DLLs
  - Scrubbing known strings
  - Don't look like an injected .dll

0x2410000	Private: Commit	964 kB	RWX
0x1f20000	Private: Commit	396 kB	RWX
0x1d90000	Private: Commit	936 kB	RWX
0x220000	Private: Commit	128 kB	RWX

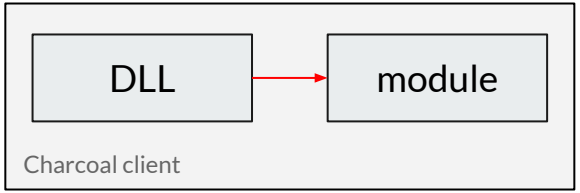
notepad.exe (2452) (0x2410000 - 0x2501000)

```
00000000 4d 5a e8 00 00 00 00 5b 52 45 55 89 e5 81 c3 74 MZ.....[REU....t
00000010 17 00 00 ff d3 81 c3 85 80 0e 00 89 3b 53 6a 04 ............;Sj.
00000020 50 ff d0 00 00 00 00 00 00 00 00 00 00 00 00 00 P...............
00000030 00 00 00 00 00 00 00 00 00 00 00 00 f8 00 00 00 ................
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 ........!..L.!Th
00000050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f is program canno
00000060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 t be run in DOS 
00000070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 mode....$.......
00000080 f4 1f 93 1a b0 7e fd 49 b0 7e fd 49 b0 7e fd 49 .....~.I.~.I.~.I
00000090 f6 2f 1c 49 9d 7e fd 49 f6 2f 22 49 af 7e fd 49 ./.I.~.I./"I.~.I
000000a0 f6 2f 1d 49 0b 7e fd 49 cd 07 1d 49 3f 7f fd 49 ./.I.~.I...I?..I
```

https://www.endgame.com/blog/technical-blog/hunting-memory

In-Memory DLL Loading

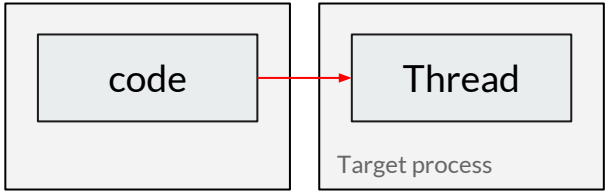
- Loading and using a DLL
- Reflective DLL injection
  - Stephen Fewer
  - Unload a DLL into memory
  - But requires ReflectiveLoader()
- Monoxgas' sRDI
  - Convert any dll to memory loadable module



https://github.com/monoxgas/sRDI - DLL shellcode injection technique
https://github.com/dismantl/ImprovedReflectiveDLLInjection - Reflective injection into remote processes

Process Injection

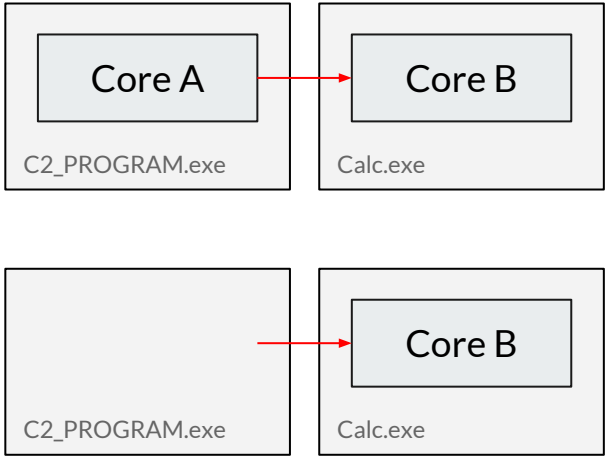
- Various injection techniques
  - SetRemoteContext (Process Hollowing)
  - SetWindowsHookEx (CIA yay!)
  - CreateRemoteThread
- DLL injection into remote processes
  - Convert to shellcode
  - Open target process
  - Allocate Memory
  - Copy to target process
  - Start remote thread



http://blog.deniable.org/blog/2017/07/16/inject-all-the-things/

Process Migration

- DLL injection into remote processes
  - Convert to shellcode
  - Open target process
  - Allocate Memory
  - Copy to target process
  - Start remote thread
  - Kill original program



Real World C2

C2 in the Wild

- Meterpreter
  - C2 component of the free and open-source Metasploit project
- Cobalt Strike
  - Commercial C2 system intended for pentesters—requires paid license
- CIA Assassin, AfterMidnight, etc
  - US state-sponsored C2 systems with documentation leaked by WikiLeaks
- For-profit botnets (e.g. Mirai)
  - Written by malicious actors and often reverse-engineered and documented by security researchers

https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/ - Meterpreter
https://www.cobaltstrike.com/ - Cobalt Strike

Ethical Concerns

- As an open-source product, could easily be used maliciously
- Features have open-source (proof-of-concept) implementations and have already been integrated into proprietary, state- and commercially-sponsored toolkits
- Meterpreter (open source)
- Cobalt Strike (licensed software)

Our Demo

Questions?