The State of the Veil Framework

@harmj0y
@ChrisTruncer

Who We Are

- **Will Schroeder** (@harmj0y)
  - Former national research lab keyboard monkey
- **Christopher Truncer** (@ChrisTruncer)
  - Florida State Graduate - Go Noles!
- Red Teamers, Pen Testers, and Security Researchers for the Adaptive Threat Division

VERIS GROUP

Overview

- Genesis
- The Veil-Framework

Evading AV	Veil-Evasion
Payload Delivery	Veil-Catapult
Situational Awareness	Veil-PowerView
Post-Exploitation	Veil-Pillage
Shellcode Generation	Veil-Ordinance
demos throughout	

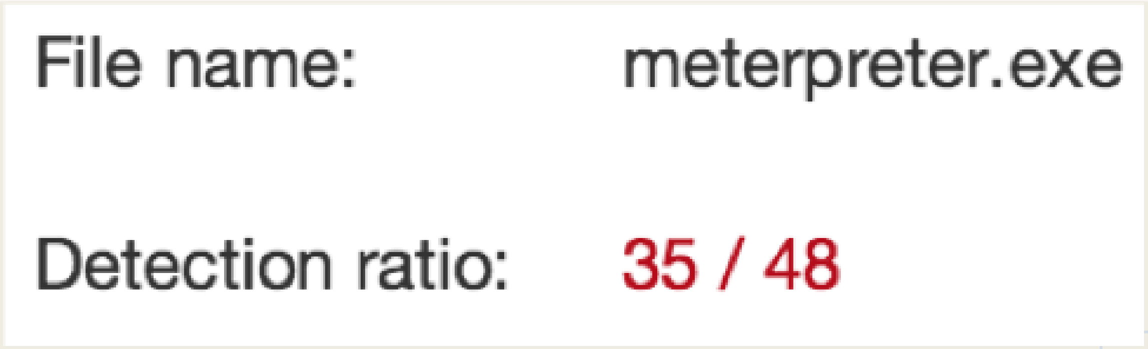
- Moving Forward
    - Veil-Framework 3.0

Genesis

Where it all began

Our Problem

- Why are are pentesters caught but malware authors aren't?



File name:	meterpreter.exe
Detection ratio:	35 / 48

Our Initial Solution

- Want a way to bypass antivirus “solutions” as easily as professional malware
- Minimize repetition
  - Don’t roll custom backdoors each assessment
- Execute our agents on targets in a way that bypasses most antivirus detection

The Veil-Framework

- A toolset aiming to bridge the gap between pentesting and red teaming capabilities

- We started with Veil-Evasion, and began to branch out to payload delivery and PowerShell exploitation

- Nothing revolutionary here, but want to bring together existing techniques and incremental research try to push things forward

Ethical Considerations

- Similar parallels to the exploit disclosure debate
- The public community is typically 5+ years behind professional malware developers
- The blackhat industry has solved this problem, why shouldn’t the whitehats as well?

HD’s Take

- ***“The strongest case for information disclosure is when the benefit of releasing the information outweighs the possible risks. In this case, like many others, the bad guys already won.”***

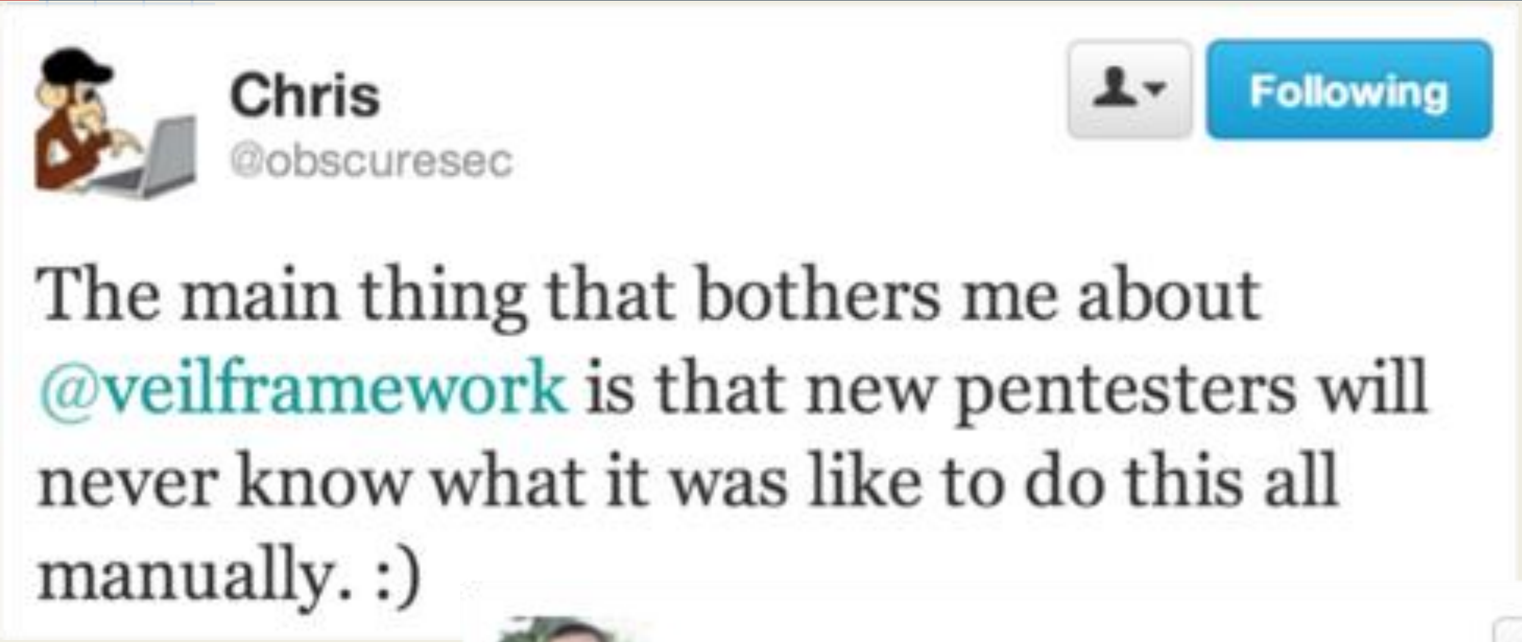
- https://community.rapid7.com/community/metasploit/blog/2009/02/23/the-best-defense-is-information

Public Reaction

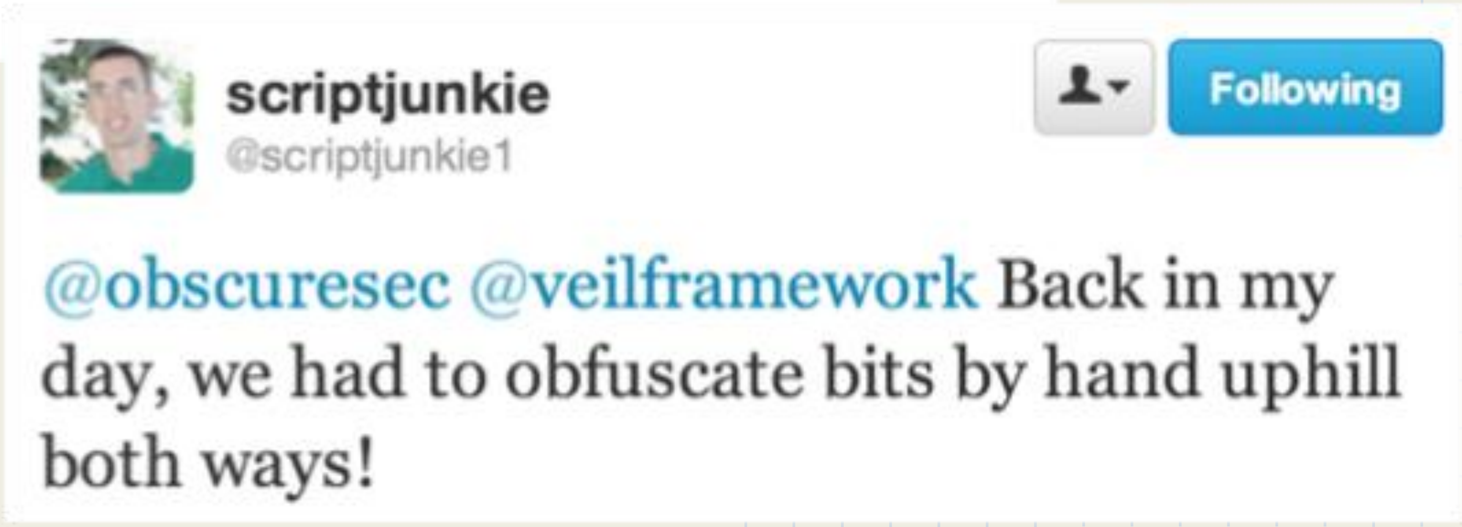
- “surely this will result in 21 new signatures for all major AVs, and then we’re back to square one?”
- “Isn’t our entire field meant to be working towards increasing security, rather than handing out fully functioning weapons?”
- “The other point here is that anything that helps to expose how **in-effective AV is at stopping even a minimally sophisticated attacker** is a good thing.”

http://www.reddit.com/r/netsec/comments/1fc2xp/veil_a_metasploit_payload_generator_for_bypassing/

Twitter Reaction


Chris
@obscuresec

The main thing that bothers me about @veilframework is that new pentesters will never know what it was like to do this all manually. :)


scriptjunkie
@scriptjunkie1

@obscuresec @veilframework Back in my day, we had to obfuscate bits by hand uphill both ways!

Veil-Evasion

Efficient
Anti-Virus
Evasion

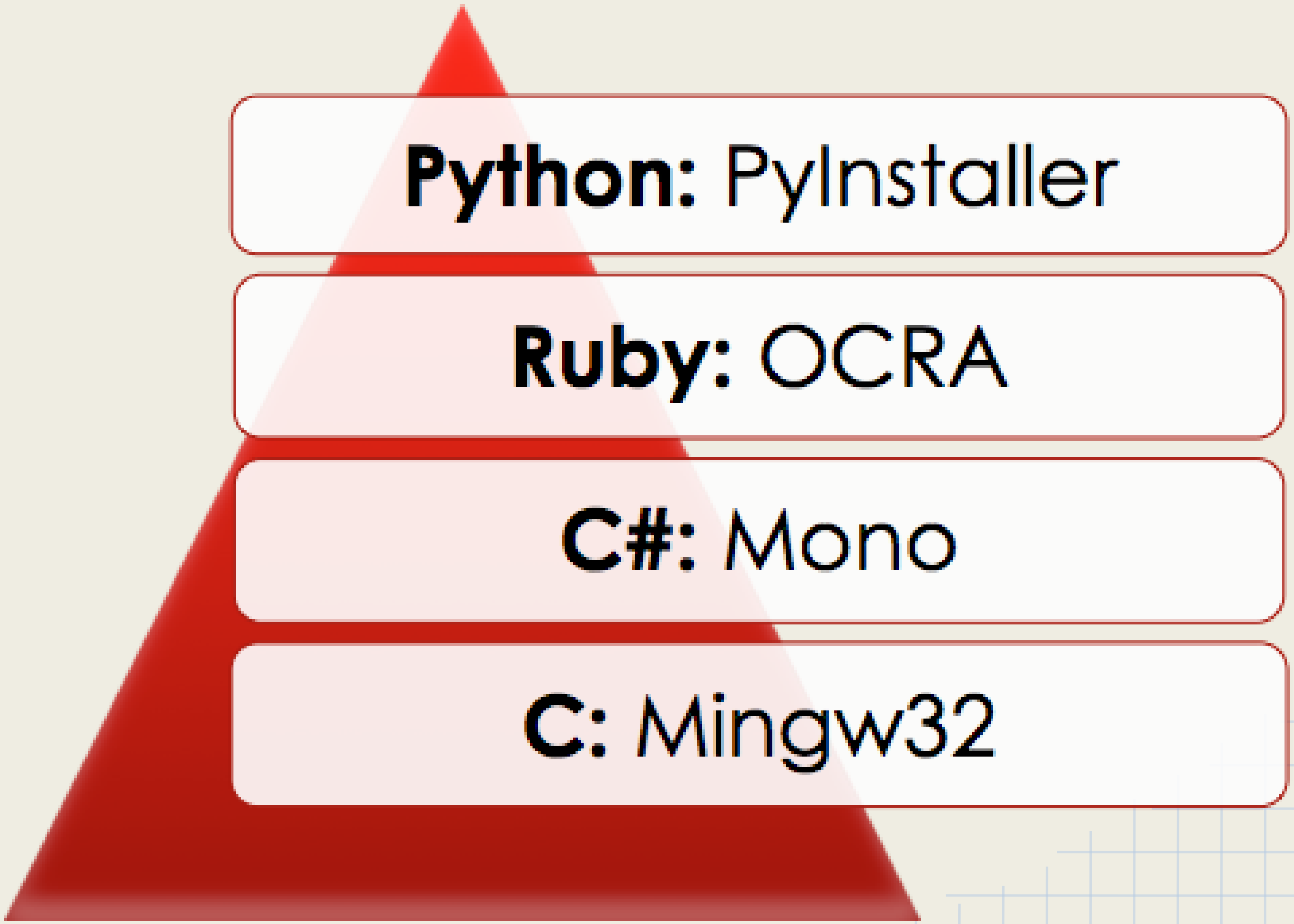
Our Approach

- Aggregate various shellcode injection techniques across multiple languages
  - Public techniques used by a variety of open-source tools
- Some shellcodeless Meterpreter stagers and “auxiliary” modules as well
- Focus on usability, automation, and the creation of a true framework

Features

- Can use either Metasploit generated or custom written shellcode
  - Metasploit Framework payloads/options are dynamically loaded
- Third-party tools can be easily integrated
  - Hyperion, PE Scrambler, Backdoor Factory, etc.
- Command line switches add in scriptability
- Check payload hashes against VirusTotal

Native Compilation



Python: PyInstaller

Ruby: OCRA

C#: Mono

C: Mingw32

Shellcode Injection 101

- **Void Pointer Casting**
  - Can’t guarantee shellcode is in an executable part of memory
- **VirtualAlloc**
  - Allocate memory as RWX, inject and execute the shellcode from the allocated section of memory
- **HeapAlloc**
  - Creates a heap object, allocates memory, injects and executes shellcode

Pwnstaller

- What if some vendors trigger on the Pyinstaller loader.exe itself?

- How about a (reasonably) obfuscated version of the Pyinstaller loader? :)
    - BSides Boston ‘14: Pwnstaller 1.0
    - https://github.com/harmj0y/pwnstaller/

- Integrated into Veil-Evasion this past May

“Pure” Stagers

- Stage 1 Meterpreter loaders don’t have to be implemented in shellcode

- Meterpreter stagers can be written in higher-level languages
  - Thanks Raffi! https://github.com/rsmudge/metasploit-loader

- Lots of varieties in Python, C, PowerShell, C# and Ruby

How Stagers Work

- **1)** a tcp connection is opened to the handler
- **2)** the handler sends back 4 bytes indicating the .dll size, and then transfers the .dll
- **3)** the socket number for this tcp connection is pushed into the edi register
- **4)** execution is passed to the .dll just like regular shellcode (void * or VirtualAlloc)

- reverse_http[s] stagers skip steps 2 and 3

V-Day

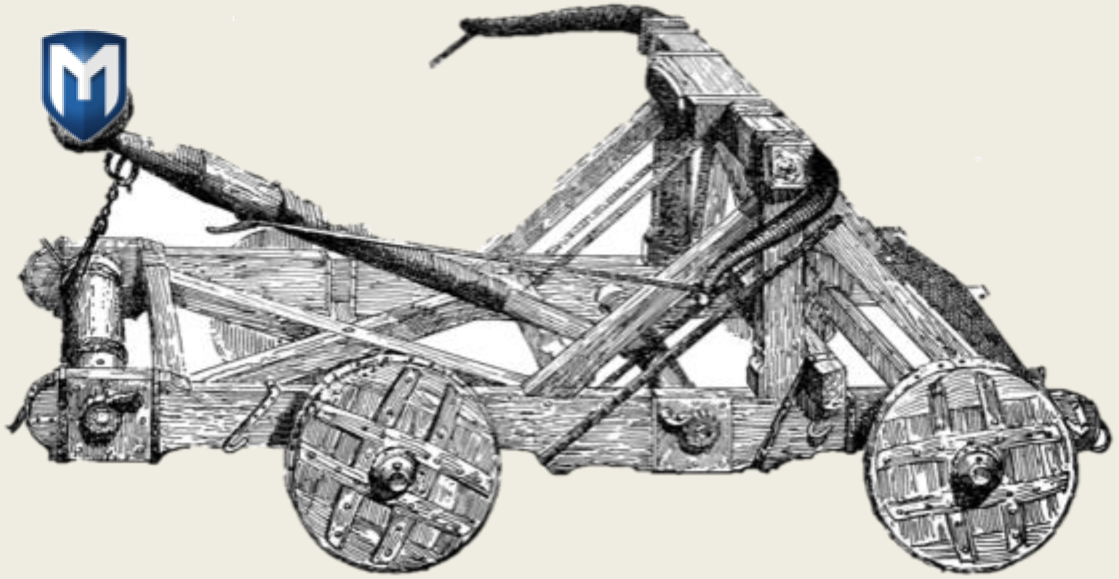
- Our release cycle, modeled on Microsoft's Patch Tuesday :)
- New modules are released on the 15th of every month
- Currently there are 34+ modules for use
  - We still have 20+ modules in a development or QA state
- We plan to keep #avloling for quite some time

Veil-Evasion Demo

Veil-Catapult

Payload Delivery

Veil-Catapult



Veil-Catapult

- After payload generation, our focus moved to payload delivery
- Features integration with Veil-Evasion to generate payloads, and can upload or host/execute binaries on targets
  - additional methods (like PowerShell) as well
- Obsoleted with the release of Veil-Pillage

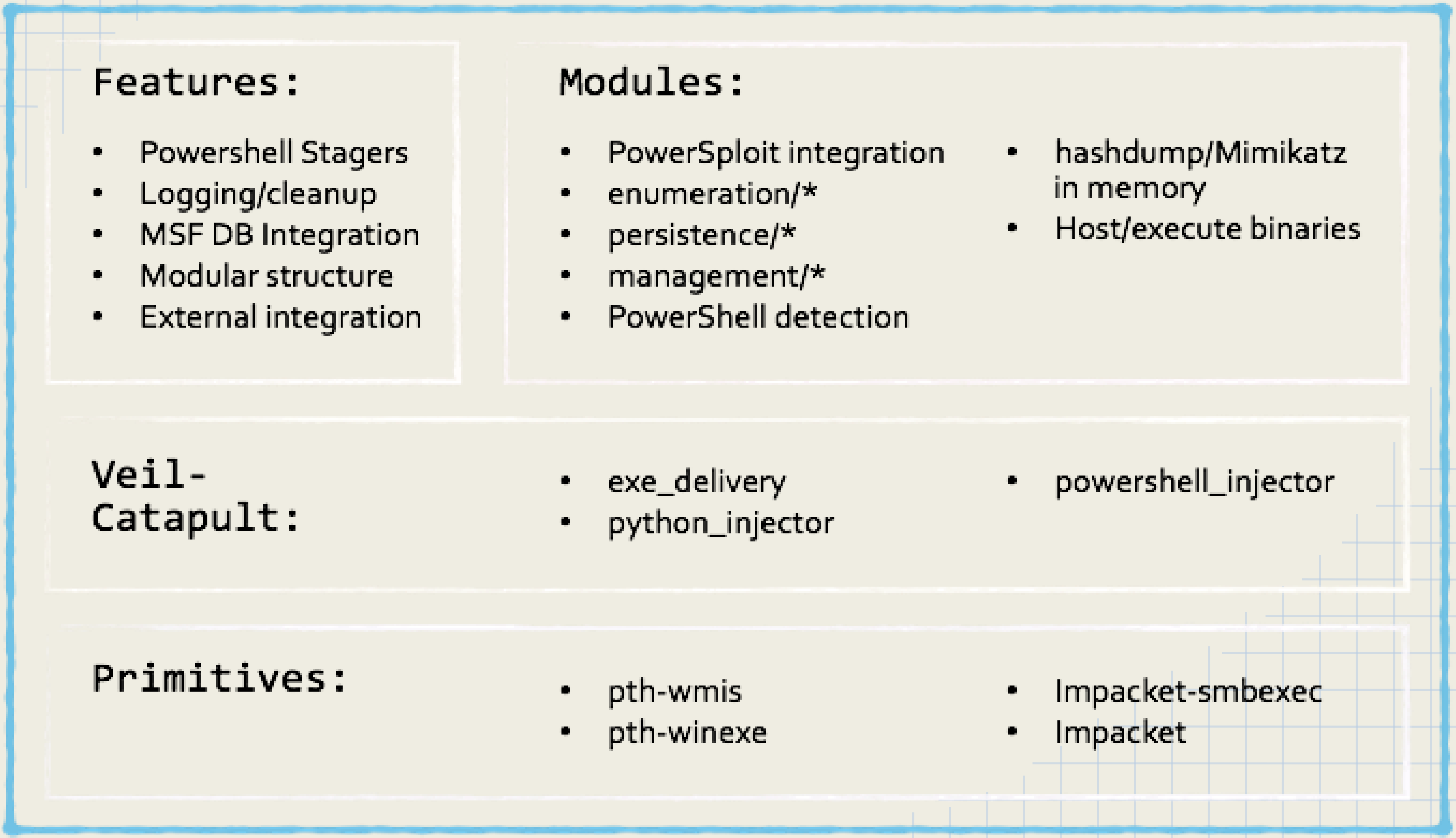
Veil-Pillage

Modular
Post-Exploitation

Features

- **Trigger Options:**
  - with a preference for stealth
  - Pillage utilizes pth-winexe, pth-wmis, and Impacket’s smbexec/smb servers for delivery and triggering
- **Modularity:**
  - want it to be easy to implement new post-exploitation techniques (common library)
  - and want to be able to easily integrate our code/techniques into other tools (cli options)
- **Completeness:**
  - automation, comprehensive logging, cleanup, etc.

Veil-Pillage



Features:

- Powershell Stagers
- Logging/cleanup
- MSF DB Integration
- Modular structure
- External integration

Modules:

- PowerSploit integration
- enumeration/*
- persistence/*
- management/*
- PowerShell detection
- hashdump/Mimikatz in memory
- Host/execute binaries

Veil-Catapult:

- exe_delivery
- python_injector
- powershell_injector

Primitives:

- pth-wmis
- pth-winexe
- Impacket-smbexec
- Impacket

exe_delivery

- Catapult functionality ported to Pillage
- Executables can be specified, or generated with seamless Veil-Evasion integration
- .EXEs are then uploaded/triggered, or hosted/triggered with a \\UNC path
    - This gets some otherwise disk-detectable .EXEs right by some AVs!

Hashdumping

- Let’s aggregate some of the best existing techniques and build some logic in:

```
if (Powershell working)          {
    Powerdump/PowerSploit }
else {
    determine_arch {
        host/execute appropriate binaries }
}
```

- Expose these techniques to the user for situation-dependent decisions

powersploit/*

- Several PowerSploit modules are included in Pillage

- A web server is stood up in the background
    - the ‘IEX (New-Object Net.WebClient).DownloadString(...)’ cradle is transparently triggered

- Makes it easy to run PowerSploit across multiple machines

Veil-PowerView

Situational Awareness with PowerShell

Veil-PowerView

- Pure PowerShell situational awareness tool
- Arose partially because a client banned “net” commands on domain machines
- Otherwise initially inspired by Rob Fuller’s netview.exe tool
  - Wanted something a bit more flexible that also didn’t drop a binary to disk
- Started to explore and expand functionality

Get-Net*

- Full-featured replacements for almost all “net *” commands, utilizing Powershell AD hooks and various API calls
  - Get-NetUsers, Get-NetGroup, Get-NetServers, Get-NetSessions, Get-NetLoggedon, etc.
- Think dsquery on steroids
- See README.md for complete list, and function descriptions for usage options

The Fun Stuff

- **Invoke-Netview:** netview.exe replacement
- **Invoke-ShareFinder:** finds open shares on the network and checks if you have read access
- **Invoke-FindLocalAdminAccess:** port of local_admin_search_enum.rb Metaspoit module
- **Invoke-FindVulnSystems:** queries AD for machines likely vulnerable to MS08-067

User-Hunting

- **Goal:** find which machines specific users are logged into
- **Invoke-UserHunter:** finds where target users or group members are logged into on the network
- **Invoke-StealthUserHunter:** extracts user HomeDirectories from AD, and runs **Get-NetSessions** on file servers to hunt for targets
    - Significantly less traffic than Invoke-UserHunter

Domain Trusts

- PowerView can now enumerate and exploit existing domain trusts:
  - **Get-NetForestDomains:** get all domains in the forest
  - **Get-NetDomainTrusts:** enumerates all existing domain trusts, à la nltest

- Most PowerView functions now accept a "**-Domain <name>**" flag, allowing them to operate across trusts
  - e.g. **Get-NetUsers –Domain sub.test.local** will enumerate all the users from the sub.test.local domain if an implicit trust exists

Veil-PowerView Demo

Veil-Ordinance

Fast
Shellcode
Generation

Veil-Evasion and Shellcode

- Veil-Evasion outsources its shellcode generation capabilities to msfvenom

- Reliance on outside tools can sometimes cause complications:
  - If msfvenom output changes, our parsing can break
    - This has happened twice :(
  - Speed - MSF can be slow to start (even when instantiating the simplified framework)

What we need

- We need a tool that generates shellcode
  - Output doesn’t change
    - Allows us to easily control what we want to parse
  - Still provide bad character avoidance
  - Speed is always nice too
- Encoders! Send us any/all python POCs!
  - We will slowly work through MSF encoders
- Feedback!

Veil-Ordnance

- 6 different payloads
  - Tried to pick from the most commonly used payloads (rev_tcp, bind_tcp, rev_https, rev_http, rev_tcp_dns, rev_tcp_all_ports)
  - All payloads were ported from MSF (read: we did not develop them)

- 1 current encoder
  - Single Byte Xor Encoder - Developed by Justin Warner (@sixdub)

Veil-Ordinance Demo

Moving Forward

Evasion Steps Forward

- Still have a large backlog of techniques and languages to release
- Looking into the generation of 64-bit payload modules
- Researching more complex shellcode-injection methods

Veil-Framework 3.0

- We’re beginning a reorganization and ground-up rewrite of the Veil-Framework
    - **Veil-Framework/Veil** will include Evasion, Catapult, Pillage, and Ordnance
    - **Veil-Framework/PowerTools** will include PowerView and PowerUp

- Will keep a common theme of evasion, interoperability, and a big UI focus

- Planning on a Spring release timeframe

Questions?

- harmj0y@veil-framework.com
  - @harmj0y
- chris@veil-framework.com
  - @ChrisTruncer
- #veil on freenode
- https://www.veil-framework.com