Hunting on the Endpoint

w/ Powershell

Chris Gerritz

Speaker Background

- Helped establish and led USAF's Enterprise Hunt Team.
    - ~800,000 node playground
- Founded a company that develops hunt software and capabilities.

Infocyte

Speaker

Chris Gerritz
Co-Founder, Infocyte
Twitter: @gerritzc
Github: @singlethreaded

Prior:
Chief, DCC Operations
AFCERT

U.S. AIR FORCE

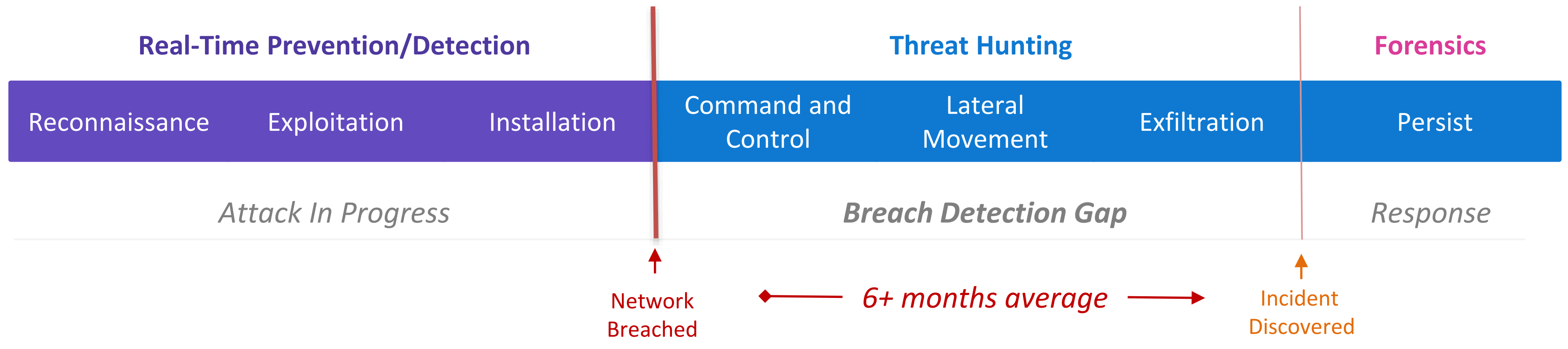
Threat Hunting

101

What is Hunt?

The proactive search for threats hiding within a network you control.

Why Hunt?



Many are breached and don't know it

The average breach goes undetected for more than 6+ months.

CARBANAK CAMPAIGN
Undetected for 114 weeks

EXCELLUS BLUE CROSS BLUE SHIELD
Undetected for 83 weeks

UNDISCLOSED RUSSIAN BANKS
Undetected for 52 weeks

PREMERA BLUE CROSS
Undetected for 46 weeks

ANTHEM
Undetected for 40 weeks

MANDARIN ORIENTAL GROUP
Undetected for 37 weeks

STARWOOD HOTELS AND RESORTS
Undetected for 33 weeks

HILTON WORLDWIDE
Undetected for 14 weeks

BUNDESTAG (GERMAN PARLIAMENT)
Undetected for 7 weeks

AMERICA'S THRIFT STORES
Undetected for 4 weeks

KOREA HYDRO AND NUCLEAR POWER
Undetected for 2 weeks

Hunt vs DFIR (tl;dr it's sort of the same, but not)

- Incident response and forensics (DFIR) tools and techniques can be used to hunt, but have some limitations:
  - 1. No bread crumb trail to follow
  - 2. Hunting requires scalability and reduced complexity
    - Especially if it's to be done iteratively (think ROI)

- **Principle of Diminishing Returns:**
  - The objective is not to perform a full forensics investigation
  - How do you know you aren't hunting snipe? (aka something that doesn't exist)

Problem w/ focused or IOC-based hunts

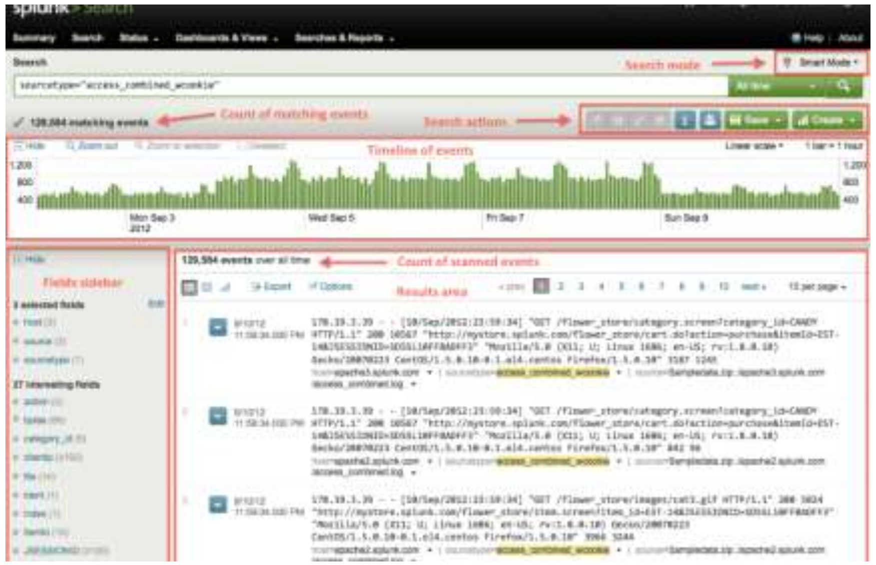
The Hunter’s Tool Bag (Examples)

- **Endpoint Solutions**
  - **Scripting (Powershell, etc.)**
  - Interactive Endpoint Hunt Solutions
  - Endpoint Response/Forensics Solutions
- **Data-Centric Solutions**
  - i.e. Elastic, Hadoop, Splunk, SEIM, etc.
  - Fed by Endpoint Detection & Response (EDR)
  - Used to store/search centralized logs/events

- **Malware Analysis**
  - PEStudio
  - Cuckoo Sandbox
- **Network Analysis Solutions**
  - passiveDNS Monitoring/Lookups
  - Wireshark (sort of?)
  - BroIDS

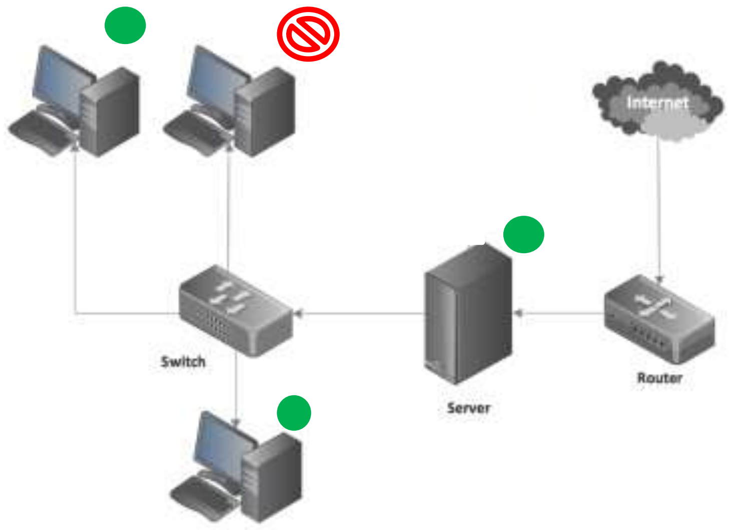
A Tale of Two Hunting Methodologies

Data-centric Analysis



<>

Endpoint Validation



- Enabled by centralized logging, long data retention + sophisticated security infrastructure and event visibility at all levels (network, host, etc.).

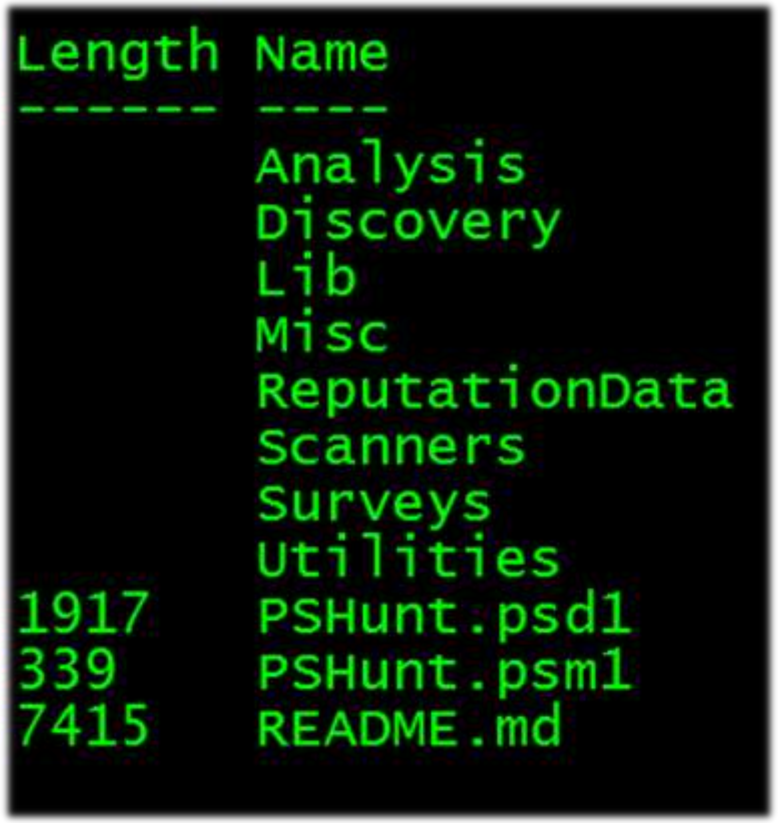
- Endpoint methodology is independent of existing security infrastructure and can be performed on almost any network (aka, the rest of us)

PSHunt

Powershell Threat Hunting Module

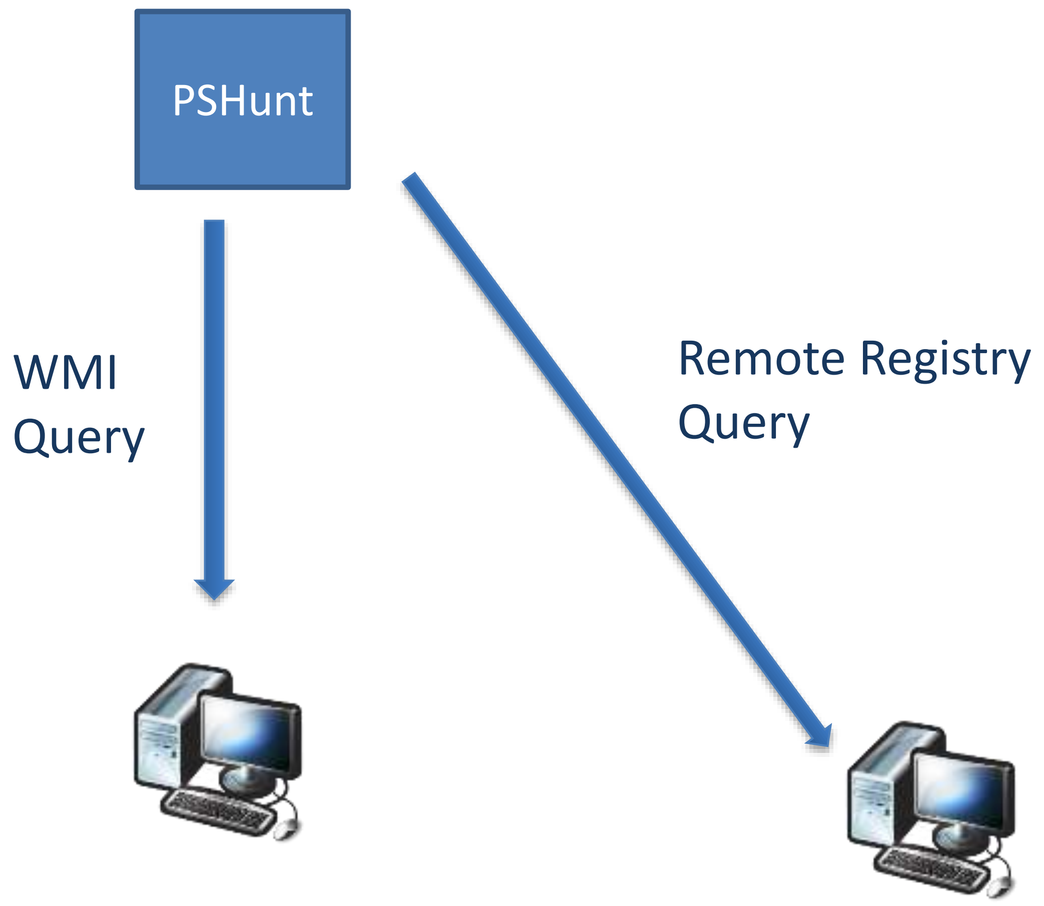
PSHunt Components/Modules

- **Scanners**
- **Surveys**
- **Discovery**
- **Utilities**
  - *Transport & Execution functions, etc*
- **Survey Analysis**
- **File Analysis**



```
Length Name
------ ----
       Analysis
       Discovery
       Lib
       Misc
       ReputationData
       Scanners
       Surveys
       Utilities
1917   PSHunt.psd1
339    PSHunt.psm1
7415   README.md
```

Scanners



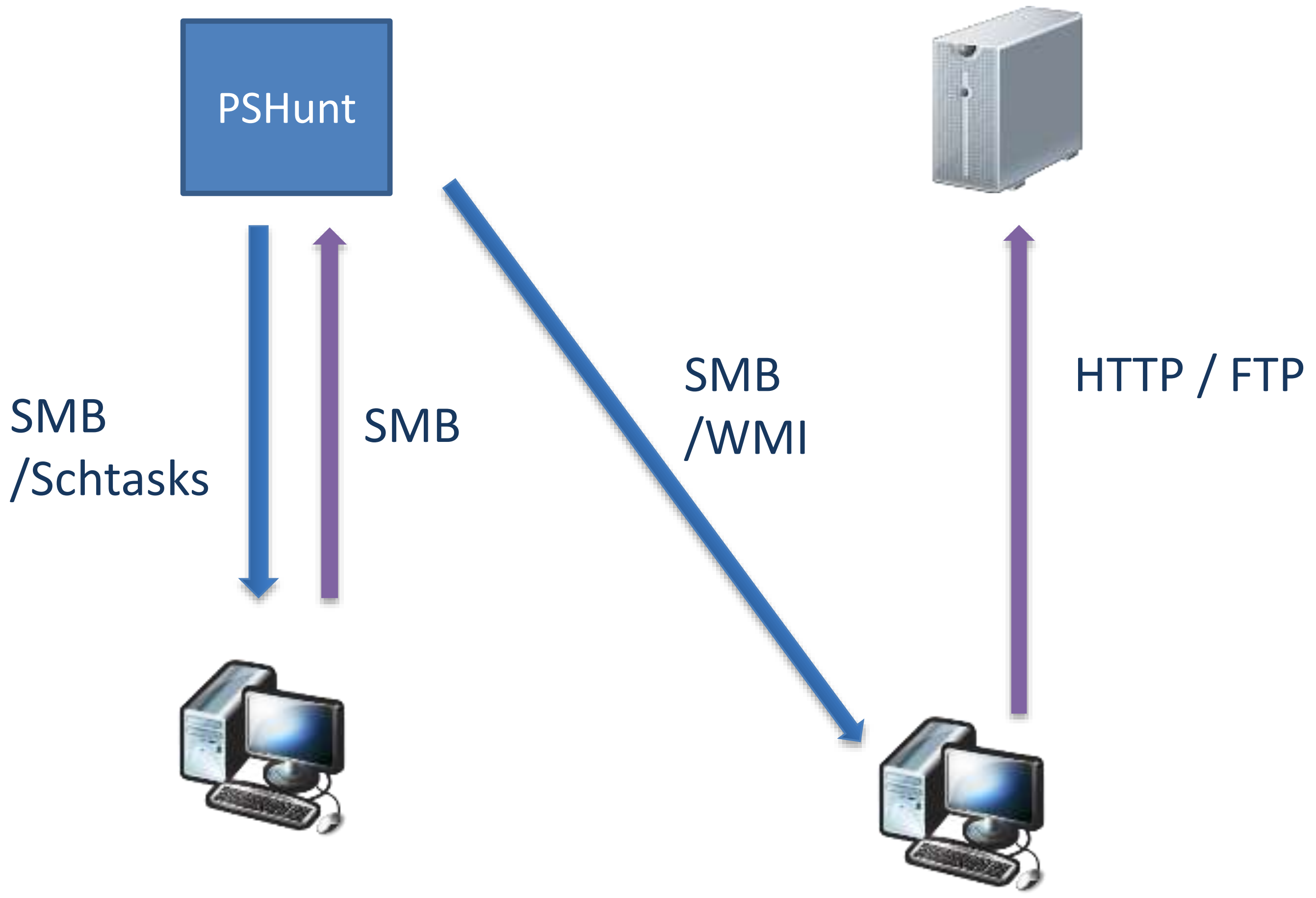
Scanners:

Description: Used to rapidly scan remote systems for a single piece of information using remote queries.

Input: Target (DNS or IP)
Output: One Line (String or CSV)

```
Invoke-HuntScan.ps1
```

Survey Deployment / Transport



Utilities [Execution]:

```
Invoke-HuntRemoteTask
```

-> Start-RemoteProcess

```
Get-HuntRemoteTaskResults
```

Download or Directly Encode Needed Libraries:

Invoke-DownloadFile
Convert-BinaryToString
Convert-StringToBinary

Remote Execution & Transport

Scanning Stuff

Execution Methods

Domain credentials are used to enumerate and access endpoints.

- WMI (Process Call Create)
- PSRemoting (Invoke-Command)
  - *Probably not enabled…* ☹
- Remote Task Scheduler (Schtasks)
- Remote Service Manager (PSExec)

Protip: type this in every windows box you see:

```
Enable-PSRemoting
```

Discovery / Testing Access

Ports and Protocols:

•TCP 22	- SSH
•TCP 135	- *WMI / RPC*
•TCP 137	- *NetBIOS (Name Resolution)*
•TCP 139	- SMB over *NetBIOS*
•TCP 445	- Server Message Block (*SMB*)
•TCP 5985	- *PSRemoting (http)*
•TCP 1025 - 5000	- Legacy Win Dynamic Range
•TCP 49152 - 65535	- Modern Win Dynamic Range

Discovery:

Test-TCPPort
Test-TCPPorts
Get-RemoteArchitecture
Get-RemotePowershellVersion
Get-RemoteOperatingSystem

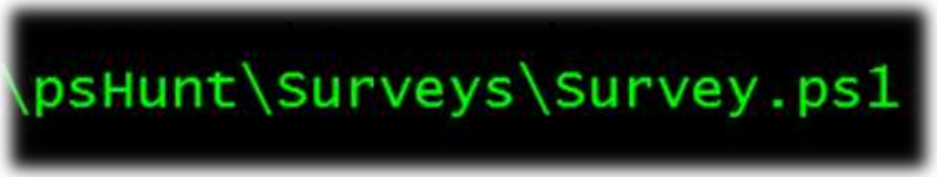
Additions:

Dsquery
Powersploit -> Recon
PowerView

Windows Host Survey

Survey: Collect from each host

- Active Processes
- Loaded Modules / Drivers
- Floating/Injected Modules
- Active Connections
- Autostarts/Autoruns
- Accounts
- Key Event Logs



```
\psHunt\Surveys\Survey.ps1
```

Description: Used to collect comprehensive information on the state of a windows host

Active Processes/Modules/Drivers

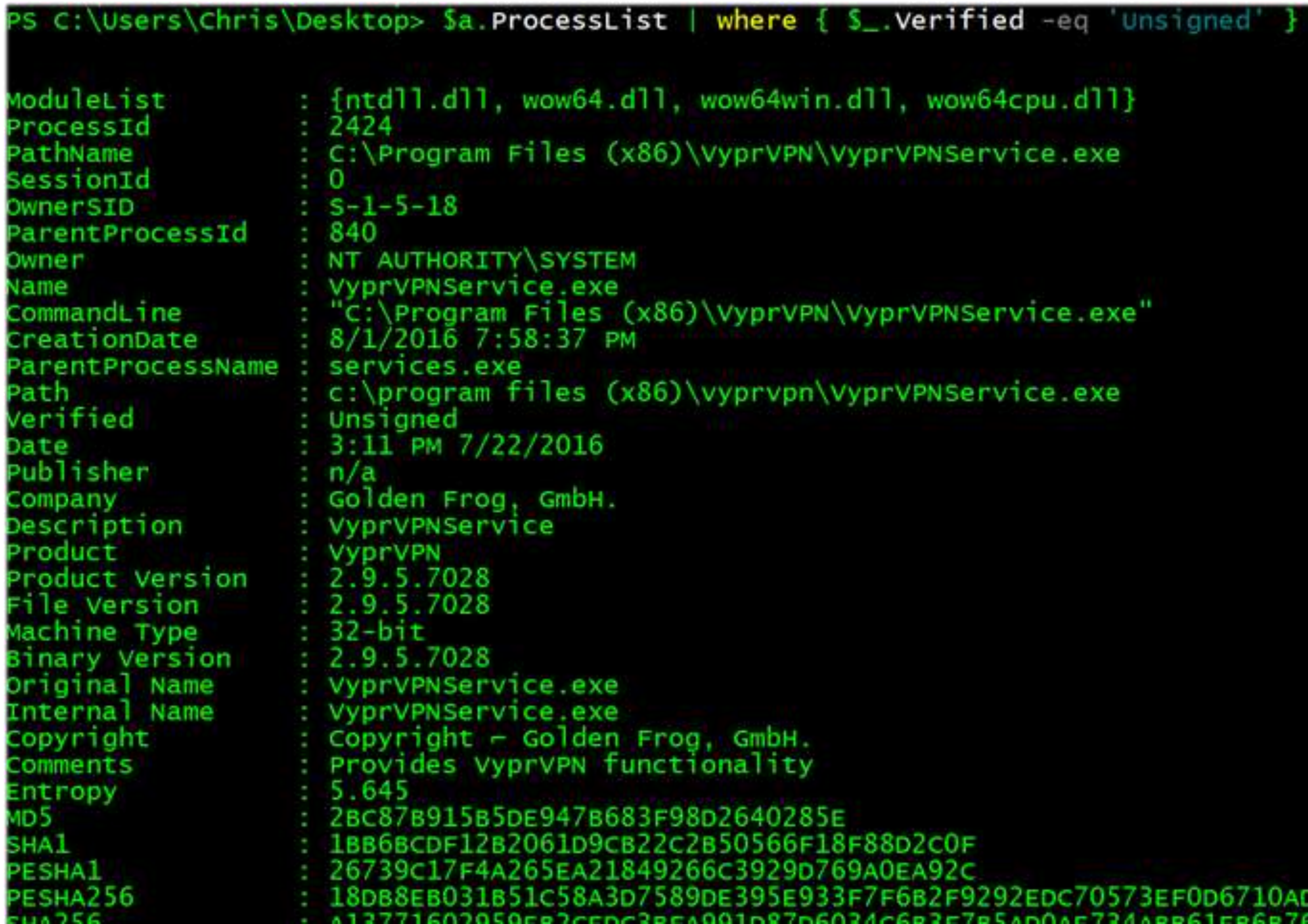
PSHunt’s **Get-ProcessList** = Get-WmiObject -Class Win32_Process

+ Get-Process –Module

+ Get-Hashes

+ Invoke-SigCheck *(Sysinternals)*

+ $Process.GetOwner()



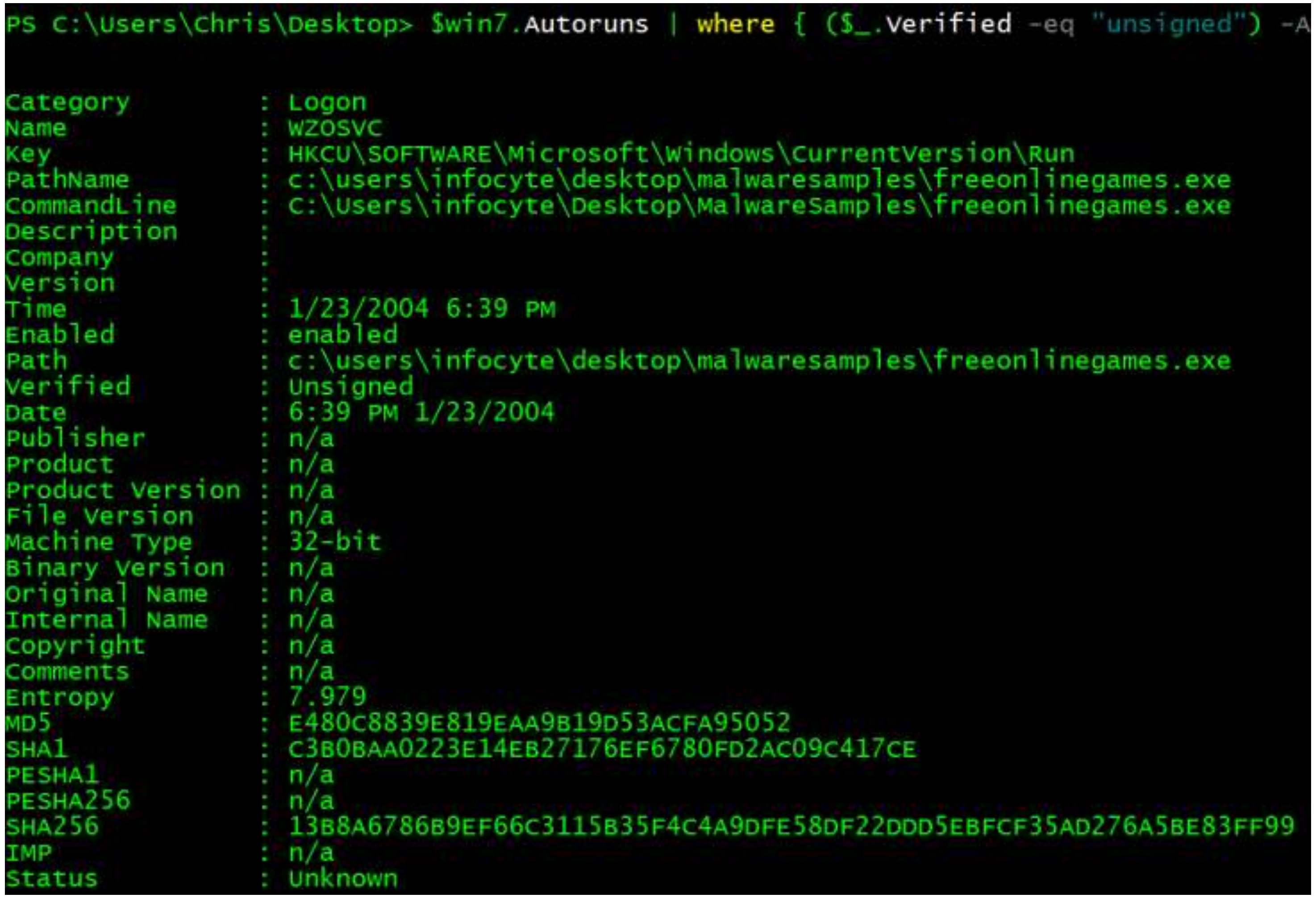
```
PS C:\Users\Chris\Desktop> $a.ProcessList | where { $_.Verified -eq 'Unsigned' }

ModuleList        : {ntdll.dll, wow64.dll, wow64win.dll, wow64cpu.dll}
ProcessId         : 2424
PathName          : C:\Program Files (x86)\VyprVPN\VyprVPNService.exe
SessionId         : 0
OwnerSID          : S-1-5-18
ParentProcessId   : 840
Owner             : NT AUTHORITY\SYSTEM
Name              : VyprVPNService.exe
CommandLine       : "C:\Program Files (x86)\VyprVPN\VyprVPNService.exe"
CreationDate      : 8/1/2016 7:58:37 PM
ParentProcessName : services.exe
Path              : c:\program files (x86)\vyprvpn\VyprVPNService.exe
Verified          : Unsigned
Date              : 3:11 PM 7/22/2016
Publisher         : n/a
Company           : Golden Frog, GmbH.
Description       : VyprVPNService
Product           : VyprVPN
Product Version   : 2.9.5.7028
File Version      : 2.9.5.7028
Machine Type      : 32-bit
Binary Version    : 2.9.5.7028
Original Name     : VyprVPNService.exe
Internal Name     : VyprVPNService.exe
Copyright         : Copyright - Golden Frog, GmbH.
Comments          : Provides VyprVPN functionality
Entropy           : 5.645
MD5               : 2BC87B915B5DE947B683F98D2640285E
SHA1              : 1BB6BCDF12B2061D9CB22C2B50566F18F88D2C0F
PESHA1            : 26739C17F4A265EA21849266C3929D769A0EA92C
PESHA256          : 18DB8EB031B51C58A3D7589DE395E933F7F6B2F9292EDC70573EF0D6710A
```

Persistence Mechanisms (Autostarts)

Implementation:
Wrapped Sysinternals
*Autorunsc**

(Note: Interacting with the registry is still a pain in the ass in Powershell.)

**currently best open source collection of autostart locations – unfortunately, it's still not comprehensive*



```
PS C:\Users\Chris\Desktop> $win7.Autoruns | where { ($_.Verified -eq "unsigned") -A


Category          : Logon
Name              : WZOSVC
Key               : HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
PathName          : c:\users\infocyte\desktop\malwaresamples\freeonlinegames.exe
CommandLine       : C:\Users\infocyte\Desktop\MalwareSamples\freeonlinegames.exe
Description       :
Company           :
Version           :
Time              : 1/23/2004 6:39 PM
Enabled           : enabled
Path              : c:\users\infocyte\desktop\malwaresamples\freeonlinegames.exe
Verified          : Unsigned
Date              : 6:39 PM 1/23/2004
Publisher         : n/a
Product           : n/a
Product Version   : n/a
File Version      : n/a
Machine Type      : 32-bit
Binary Version    : n/a
Original Name     : n/a
Internal Name     : n/a
Copyright         : n/a
Comments          : n/a
Entropy           : 7.979
MD5               : E480C8839E819EAA9B19D53ACFA95052
SHA1              : C3B0BAA0223E14EB27176EF6780FD2AC09C417CE
PESHA1            : n/a
PESHA256          : n/a
SHA256            : 13B8A6786B9EF66C3115B35F4C4A9DFE58DF22DDD5EBFCF35AD276A5BE83FF99
IMP               : n/a
Status            : Unknown
```

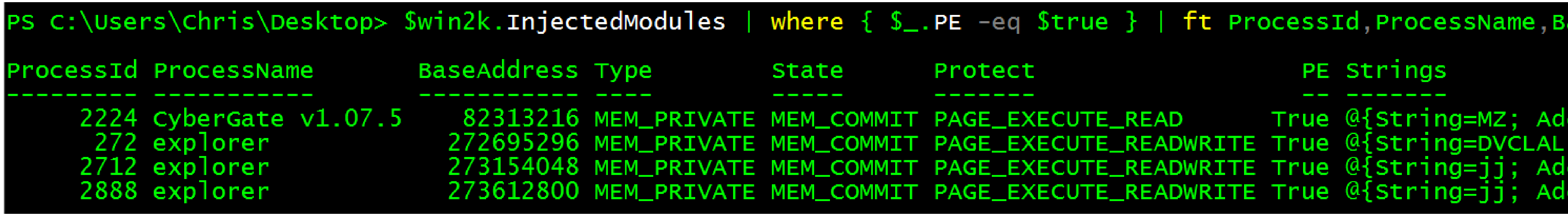
Memory-resident Malware Analysis

- Uses Matt Graeber's PSReflect Module to access Native Win32 APIs:
- **Implementation:** VirtualQueryEx walk across process memory looking for PE Headers in RWX memory.

Description: Discover DLL Injection, Process Overwrites, etc.

Uses:
PSReflect Module

```
$Kernel32::VirtualQueryEx($hProcess, $ModuleBaseAddress, [Ref] $MemoryInfo, $PageSize)
```



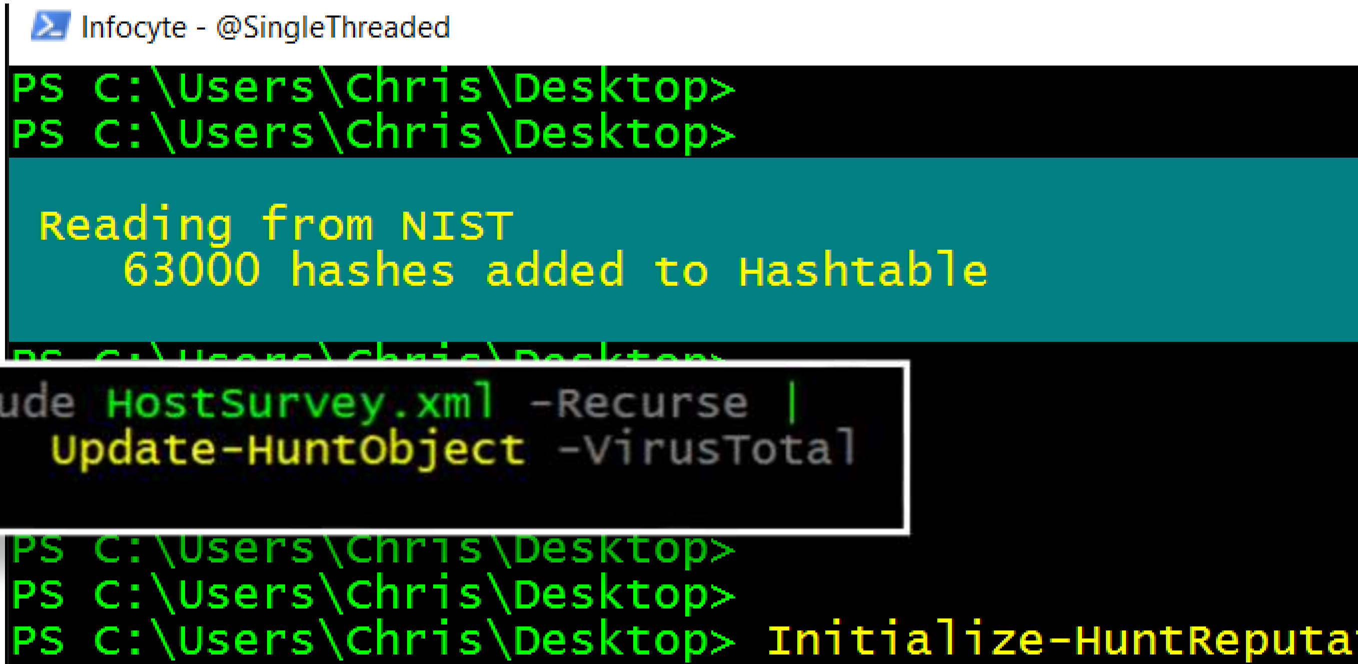
```
PS C:\Users\Chris\Desktop> $win2k.InjectedModules | where { $_.PE -eq $true } | ft ProcessId,ProcessName,B

ProcessId ProcessName        BaseAddress Type        State      Protect                PE Strings
--------- -----------        ----------- ----        -----      -------                -- -------
     2224 CyberGate v1.07.5     82313216 MEM_PRIVATE MEM_COMMIT PAGE_EXECUTE_READ      True @{String=MZ; Ad
      272 explorer             272695296 MEM_PRIVATE MEM_COMMIT PAGE_EXECUTE_READWRITE True @{String=DVCLAL
     2712 explorer             273154048 MEM_PRIVATE MEM_COMMIT PAGE_EXECUTE_READWRITE True @{String=jj; Ad
     2888 explorer             273612800 MEM_PRIVATE MEM_COMMIT PAGE_EXECUTE_READWRITE True @{String=jj; Ad
```

Survey Analysis

Survey Analysis Modules

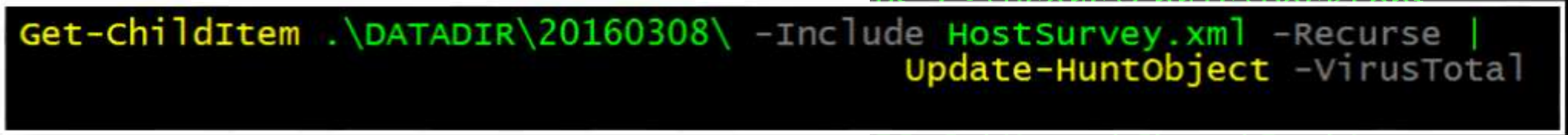
- Initialize-ReputationData
  - Loads Data into $Global:FileReputation
- Update-HostObject
  - Get-VTStatus
  - Get-VTReport
- Group-HostObjects

```
Survey Analysis:
Description: Compare Survey
Results against Reputation Data
from local store and VirusTotal.

Perform Outlier and Anomaly
Analysis
```

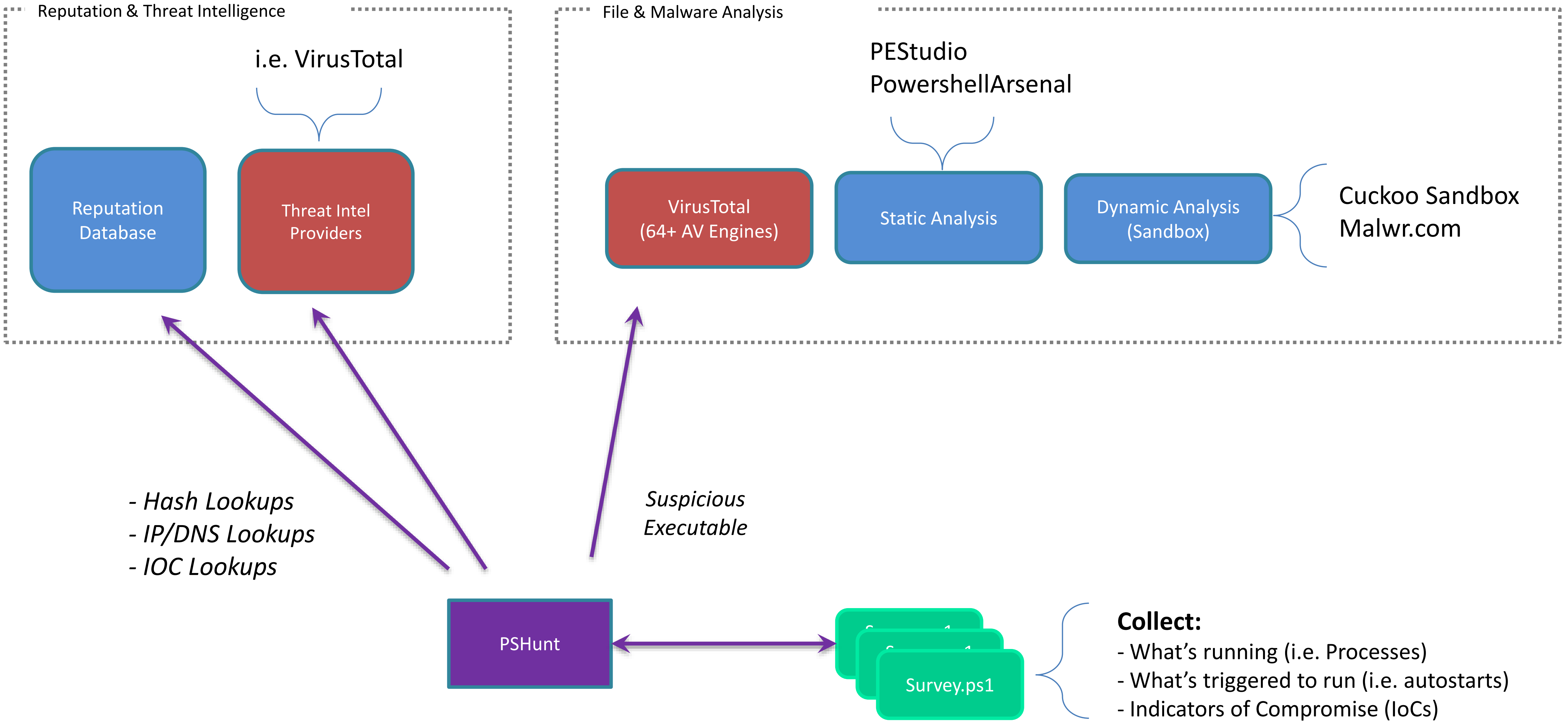


Infocyte - @SingleThreaded

```
PS C:\Users\Chris\Desktop>
PS C:\Users\Chris\Desktop>

 Reading from NIST
    63000 hashes added to Hashtable

PS C:\Users\Chris\Desktop>
PS C:\Users\Chris\Desktop>
PS C:\Users\Chris\Desktop> Initialize-HuntReputa
```



```
Get-ChildItem .\DATADIR\20160308\ -Include HostSurvey.xml -Recurse |
                                          Update-HuntObject -VirusTotal
```



Collect:
- What's running (i.e. Processes)
- What's triggered to run (i.e. autostarts)
- Indicators of Compromise (IoCs)

Finding Bad Things

Active Processes/Modules/Drivers

Some malware, even advanced types, attempt to “hide in plain sight” or within the noise of the multitude of programs running on your systems.

- ***Initial Technique*:** Hash everything and compare to a signature and threat intelligence database like VirusTotal. *This will clear all known-good and known-bads.*

- **Adv. *Technique:***

  1. Stack Remaining data *and* perform anomaly and outlier analysis

  2. Perform static/dynamic analysis on the exe of any suspicious or out-of-place processes

Digital Signatures?

Digital Signatures: Most malware is not digitally signed by a *legitimate* Certificate Authority (CA).

- Attackers may load their rogue CA into your local Trusted Root CA store at the time the malware is installed (requires root privileges)
- *Adv. Technique*:
    - Check anomalous/outlier root CA's serial number against whitelist or Google it for authenticity

- WARNING: Some may digitally sign malware with a legitimate but compromised CA which renders this technique ineffective.

  - Example: The Feb '13 attack against Bit9 targeted their CA server

Persistence Mechanisms

Required to maintain the malware through reboots and in times of dormancy.

- Scheduled Tasks, Jobs, etc.
- Registry Persistence (most common)
    - *Technique*: Hash all referenced executables in registry and compare to Threat Intel Database
- Boot Process Redirection (ie. Bootkits – very sophisticated!)
    - *Technique*: Evaluate raw MBR (first 512 bytes of disk0) for redirection to alternate boot loader

Process Memory Injection

- **DLL Injection / Process Hallowing**:
  1. Allocate chunk of unprotected Read/Write/Execute (RWX) memory inside another legitimate process.
  2. Load in a malicious DLL.
  3. Redirect an execution thread.
  4. Profit.

- *Adv. Technique:*
  - Walk Process Memory looking for PE Headers in <u>large</u> chunks of unprotected memory (Use @mattifestation's PSReflect)
    - False Positives will come from:
      1. Just-in-Time (JIT) compilers – i.e. .NET and Java Apps
      2. Security Software

That’s it for now.

More to come…

PSHunt – Powershell Threat Hunting

Follow me:

Chris Gerritz
Co-Founder, Infocyte
Twitter: @gerritzc
Github: @singlethreaded

NOTE:
PSHunt will be posted on Github this week.