NorCERT Summit 2016, Security BSides Nola 2016

FireEye

Proactive Threat Detection: Windows

Seek and ye shall find, otherwise: compromise

BY DEVON KERR, PRINCIPAL CONSULTANT APRIL 16, 2016

SECURITY REIMAGINED

Objectives

- Learn (very little) about who I am
- Review the attack lifecycle
- Understand how host-based forensic artifacts can be used to detect threats and augment other forms analysis
- Review a few case studies
- Interrogate me (be kind)
- Do all that in one hour or less….

Introductions (/me)

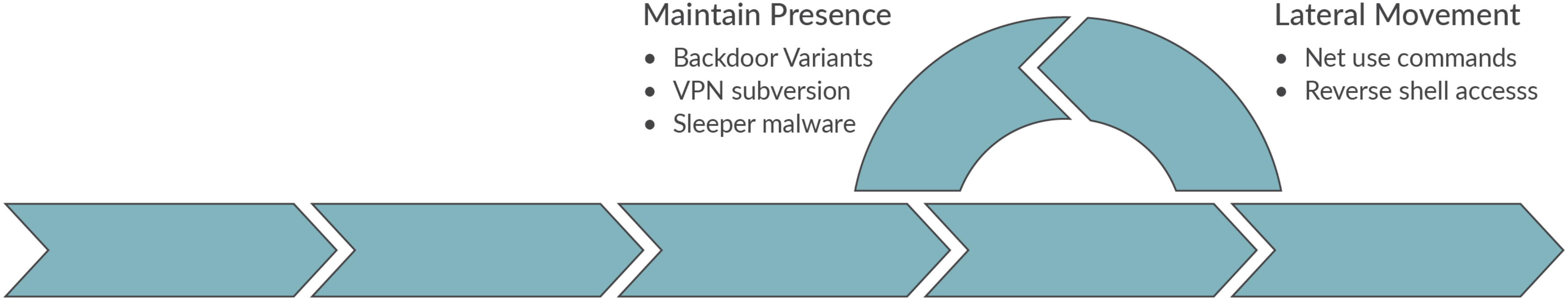
- 15 years in operations
  - Sysadmin and network engineer
- 5 years at Mandiant
  - Leading investigations and remediating intrusions
  - Research Windows CNE using native frameworks (WMI, WSH, PowerShell)
    - Delivered a bunch of talks at MIRCON, CanSecWest, SANS, other places
  - Educator (enterprise IR, proactive threat detection, creating/using IOCs, investigative techniques)

> “My presentation is much more interesting than I am.”
>
> *- Devon Kerr*

FireEye™

The targeted attack life cycle

Learn how your prey behaves



Maintain Presence

- Backdoor Variants
- VPN subversion
- Sleeper malware

Lateral Movement

- Net use commands
- Reverse shell accesss

Initial Compromise

- Spear fishing email with custom malware
- Third party application exploitation
- Web vulnerability exploitation

Establish Foothold

- Custom malware
- Command and control

Escalate Privileges

- Password cracking
- “Pass-the-hash”
- Application exploitation

Internal Recon

- Critical system recon
- System, active directory, and user enumeration

Complete Mission

- Staging servers
- Data consolidation
- Data theft

Host-based Forensic Artifacts

Index tracking on Windows

Note

- To help manage the volume of material, this section is broken up by source of evidence
  - Filesystem
  - Registry
  - Event Logs
  - Process Memory
- The general value of each artifact has been indicated during each phase of the intrusion
  - Be aware that these are not absolutes (True vs. False)
  - Value is represented using color saturation – rich colors are strong

Filesystem

	Initial Compromise	Establish Foothold	Escalate Privileges	Internal Recon	Move Laterally	Maintain Persistence	Complete Mission
Prefetch	░	░	█	█	░	░	■
INDX files	░	░	░	░	░	░	■
NTFS Change Journal	▓	░	▓	▓	░	█	█
JOB files	░	░	▓	▓	█	▓	░
BAT files	░	░	▓	█	▓	░	░
Enterprise AV logfiles	▓	░	█	░	░	▓	░
Common Staging Folders	░	░	▒	▒	░	█	█
LNK files	░	▒	░	▒	▒	▓	░
WMI CIM Repo	░	░	░	░	░	▓	░

Registry

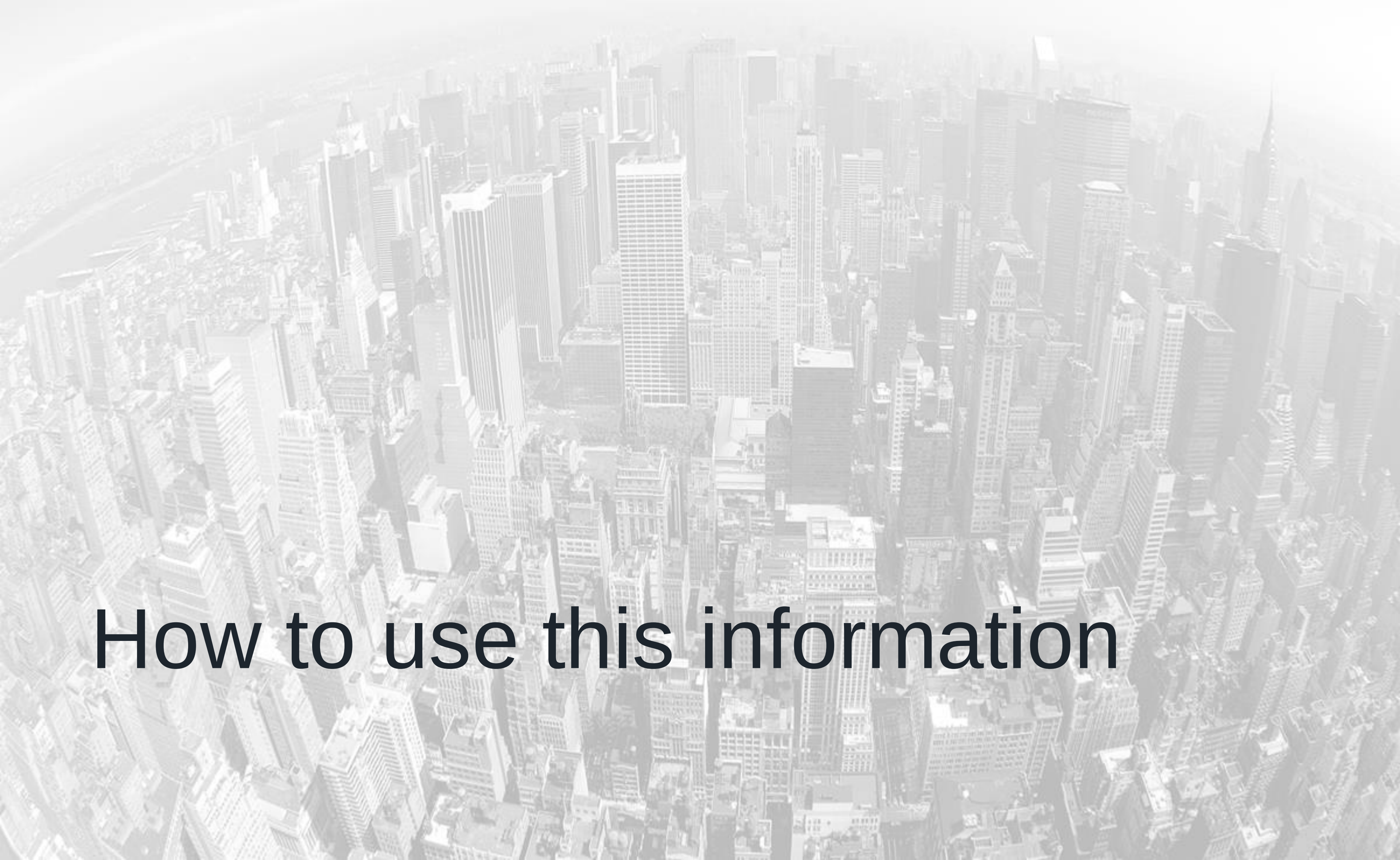
	Initial Compromise	Establish Foothold	Escalate Privileges	Internal Recon	Move Laterally	Maintain Persistence	Complete Mission
Shimcache	█	█	█	█	█	█	█
Amcache	▇	▇	▇	▇	▇	▇	▇
MUICache	░	░	▇	▓	░	▒	▒
Service Keys	░	▒	░	░	░	▇	░
Run/RunOnce Keys	░	▒	░	░	░	▇	░
LSA Secrets	░	░	▇	░	░	░	░
Shellbags	░	░	░	▓	▓	▓	▒
MRU Keys	░	░	░	▒	▒	░	▓
UserAssist	░	░	▒	▒	░	▒	▒

Event Logs

	Initial Compromise	Establish Foothold	Escalate Privileges	Internal Recon	Move Laterally	Maintain Persistence	Complete Mission
EID 592/4688 (Process tracking)	▓	■	■	▓	▓	■	▓
EID 601/4697 (Svc creation)	░	░	▓	░	░	■	░
EID 7035/7036 (Svc start/stop)	░	░	▓	░	░	■	░
EID 540/4624 (NTLM Logon)	░	░	░	░	■	░	░
EID 552/4648 (KERBEROS/Explicit)	░	░	░	░	■	░	░
EID 21/23/24/528/529/4624/4625(Terminal Services)	░	░	░	░	■	░	░
EID 602/4698 (task creation)	░	░	░	░	■	░	░
EID 556/7036 (ntdsutil/vssadmin)	░	░	■	░	░	░	░
Enterprise AV	▒	▒	▓	░	░	▒	░

Process Memory

	Initial Compromise	Establish Foothold	Escalate Privileges	Internal Recon	Move Laterally	Maintain Persistence	Complete Mission
Import hashing/analysis	░	▓	░	░	░	█	░
Unsigned binaries/drivers	▓	▓	░	░	░	▒	░
Conhost/crss strings	░	░	▒	▓	▓	▒	▓
Injected Processes	▒	▒	▓	░	░	▒	░
Network Connections	░	▒	░	░	░	▒	░
DNS Cache	░	▒	░	░	░	▒	░
Mutexes	░	▒	░	░	░	▒	░
Process Path Anomalies	░	▒	░	░	░	▒	░
Svchost.exe Artifacts	░	░	░	▒	▒	░	░



How to use this information

Develop a plan of action for today’s crisis

Often, a compromise is detected during one of the intrusion phases.

Using these tables, an investigator could prioritize which artifacts to collect.

- Here’s a for-instance: your organization receives a notification from law enforcement. It informs you that on May 15, 2015 approximately 25GB of multipart WinRAR archives were stolen from a webserver in the DMZ. The naming convention resembles “aa[1-87].jpg”.

- What can we collect that is useful during the Complete Mission phase?
  - Prefetch
  - Shimcache/Amcache
  - INDX Records
  - Common Staging Folders
  - NTFS Change Journal

Assume compromise today, tomorrow, always

Instead of waiting for a compromise to occur, what are the highest value artifacts we can collect and which phases will they be relevant?

- All phases: Shimcache/amcache, process tracking
- Initial Compromise: AV logs
- Establish a Foothold: Services keys, run keys
- Escalate Privileges: Prefetch, AV logs, LSA secrets registry key, events related to VSS
- Internal Recon: Prefetch, BAT files
- Move Laterally: JOB files, events related to logons/RDP/scheduled tasks
- Maintain Persistence: NTFS change journal, services keys, run keys, events related to services
- Complete the Mission: Prefetch, common staging folders, NTFS change journal, INDX records

Practical Examples

I caught a fish “this” big

Scenario-driven examples

All information was obtained from non-classified environments.

Note that each artifact in the presented examples was collected using an endpoint agent, however it would be relatively trivial to collect these using PowerShell, WMI, batch scripts, or other open source solutions.

Similarly, each artifact can be parsed using a wide variety of open source tools – the intent is to demonstrate the value of these artifacts without promoting a collection or analysis framework.

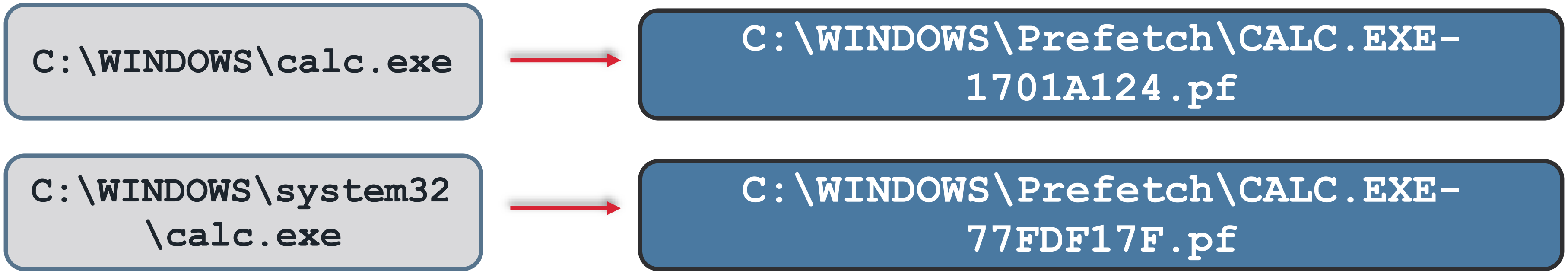
- Due to time considerations, fewer examples:
  - Using prefetch to identify credential harvesting
  - Finding PlugX with import functions

Prefetch

Overview of Prefetch

- Helps answer “what application previously ran?” and “when?”
- File system artifact located in “C:\Windows\Prefetch\”
- “path” and “hash” filename, same filename in different directories will result in different prefetch filenames
- Disabled on servers (default)
- Records the binaries loaded within ~10 seconds of execution
- Up to 128 on Win7, up to 1024 on Win8

Prefetch Naming Convention



C:\WINDOWS\calc.exe → C:\WINDOWS\Prefetch\CALC.EXE-1701A124.pf

C:\WINDOWS\system32\calc.exe → C:\WINDOWS\Prefetch\CALC.EXE-77FDF17F.pf

Valuable Prefetch Metadata

- The $SIA creation timestamp of the .pf file typically indicates the first approximate runtime
- The $SIA last modified timestamp of the .pf typically indicates the most recent approximate runtime
- Within each prefetch file, the following metadata should be consistent:
  - The # of times executed
  - The list of files accessed within ~10 seconds of execution
  - The most recent execution time

Scenario: Credential Harvesting Evidence

Scope

- Pulled Prefetch files from Windows systems in a 10k node environment

Average Data Volume

- ~34MB per system

Approximate Number of Records

- ~8,600 accessed file records each

Process and Findings

- Using keyword searches for common archive utilities, determined that command line WinRAR (rar.exe) was quite rare
    - Employees favored 7zip and integrated WinZIP, generally

- Identified three (3) systems where command line WinRAR was used
    - 1 false positive
    - 2 confirmed hits referencing “wce.exe”, “w.txt”, and “dump.bat”

Some Analysis Suggestions (101)

- Review Prefetch files for 1-3 character file names (ex. “aaa.exe”)
- Review Prefetch files for suspicious file names (ex. “rar.exe”)
- Review Prefetch files for administrative tool names (ex. “ntdsutil.exe”)
- Review accessedfiles for suspicious file names (ex. “wceaux.dll”)
- Review accessedfiles for suspicious extensions (ex. “.part.rar”)
- Review accessedfiles for suspicious directory names (ex. “RECYCLE.BIN”)
- Conduct frequency analysis of Prefetch files and distribution of accessedfiles file names

Function Imports

How do they do that?

Super high level overview of function imports

- A DLL that is loaded by an executable to provide some kind of feature or capability is an “import”
    - This is tied to one or more methods contained within the DLL
- One or more methods can represent functions like keylogging, providing a reverse shell, or interrupting a system call
    - Most can be achieved in a few different ways
- Upon execution, Windows looks at the PE import table and loads any required DLLs
    - During this process, the imports are loaded into memory and maps the various functions those DLLs support

Binary metadata blowout

- We can examine binaries (on disk and in memory) for important and useful metadata related to imports and functions:
  - The import DLL name
  - The import file size
  - The name(s) of the function(s)
- Note that sometimes an import is unpacked at runtime and may not persist on disk (I'm looking at you, "wceaux.dll")
- Combinations of functions can be great indicators of compromise
  - Much harder to change than basic metadata (which make highly volatile IOCs)

PlugX: Attributes Known

- PlugX (KORPLUG) illustrates this pretty well
- Dat metadata:
  - File size is often predictable, commonly persists via search order hijacking
    - Details we don't really need
  - Imports “msvcrt.dll”, a DLL that provides some C/C++ functionality
  - Imports the following functions:
    - Malloc – allocates memory
    - Memset – initializes memory values
    - Strcat – appends data

Let's load stuff into memory!

Scenario: PlugX Persistence

Scope

- Pulled PEInfo from binaries on Windows systems in a 17K node environment

Average Data Volume

- ~22MB per system

Approximate Number of Records

- Excluding native Windows binaries (looked up, excluded)
- Excluding non-executable files
- So many records (couple hundred thousand)

Process and Findings

- Using what we knew about PlugX
  - Searched for all files which imported “msvcrt.dll”
  - Excluded results which did not import the “malloc”, “memset”, and “strcat” functions

- Identified nine (9) systems with one binary each that matched
  - 0 false positives
  - 4 different variants with unique file metadata
    - None of the samples had public signatures at that time

Some Analysis Suggestions (101)

- Develop an understanding of which functions and imports map to common malicious capabilities
  - Look for executables, regardless of extension, which map those unusual collections of imports
- Examine how binaries are packed – like UPX and Themida - or compiled using less common compilers like Py2exe/PyInstaller
- Look at binaries compiled before 1970 or after the current year
- Examine directories for multiple files with the same root name, but different extension (ex. “abc.dll”, “abc.hlp”, “abc.exe”)
- Look for DLLs outside of “system32” and which are not listed in the “KnownDLLs” registry key
- Repeat after me: *unsigned drivers, I will know what exists in my environment*

Questions?

“What do you want to know” said no one to nobody