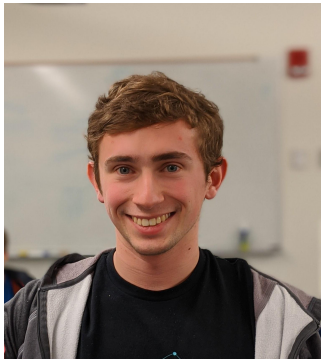
BLUESPAWN

An open-source **active defense** and EDR solution

Made with ❤️ by the UVA Cyber Defense Team
Windows Group and others

https://bluespawn.cloud

About Jake and Jack



- UVA Class of ‘20
- Computer Science Major
- Past UVA CCDC Windows Team Lead
- MetaCTF Co-Founder

- UVA Class of ‘22
- Computer Science Major
- Current UVA CCDC Windows Team Lead

The Team

- Jake Smith - Threat Intelligence and Management
- Jack McDowell - Architecture and Design
- Will Mayes - Windows OS Interactions
- Calvin Krist - Server Platform and Windows Client
- David Smith - Server Platform
- Aaron Gdanski - Linux Client

Available Tooling

What do they miss vs catch?

What do they do well or not so well?

Rise of the EDR/EPP/NGAV

Strengths:

- ✔ Robust, kernel-level monitoring across your entire fleet of devices*
- ✔ Able to quickly and easily detect, investigate and respond to threats

Limitations:

- — Somewhat of a black box for customers means trusting the vendor
- — Likely requires tuning to improve quality of alerts

* Support for non-Windows devices is ¯_(ツ)_/¯

These tools are **really** raising the bar for defenders.

Microsoft's Sysinternals

- Collection of close-sourced, "swiss army knife" tools for troubleshooting, monitoring, and analyzing Windows systems

Strengths:

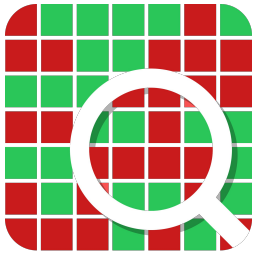
- ✔️ Comprehensive view of system activity
- ✔️ Easy to record activity
- ✔️ 1st party support

Limitations:

- ➖ Requires knowledge and intuition to know what to look for
- ➖ Doesn't map to security frameworks or clearly identify threats

Process Hacker:

- Similar to Sysinternals' ProcExp and TCPView, but open-source
- Enables easy analysis of processes, memory regions, network connections, and more
- Little to no automation



PE-sieve:

- Open-source tool by @hasherezade
- Fantastic at detecting various process manipulation attacks (injection, hollowing, etc)
- Limited to process activity

Rule Collections and Hardening Scripts

Rule Collections:  Yara Rules Project

- Infosec community is getting better at standardized sharing of detections
- Examples include YARA Rules Repos and Sigma Project

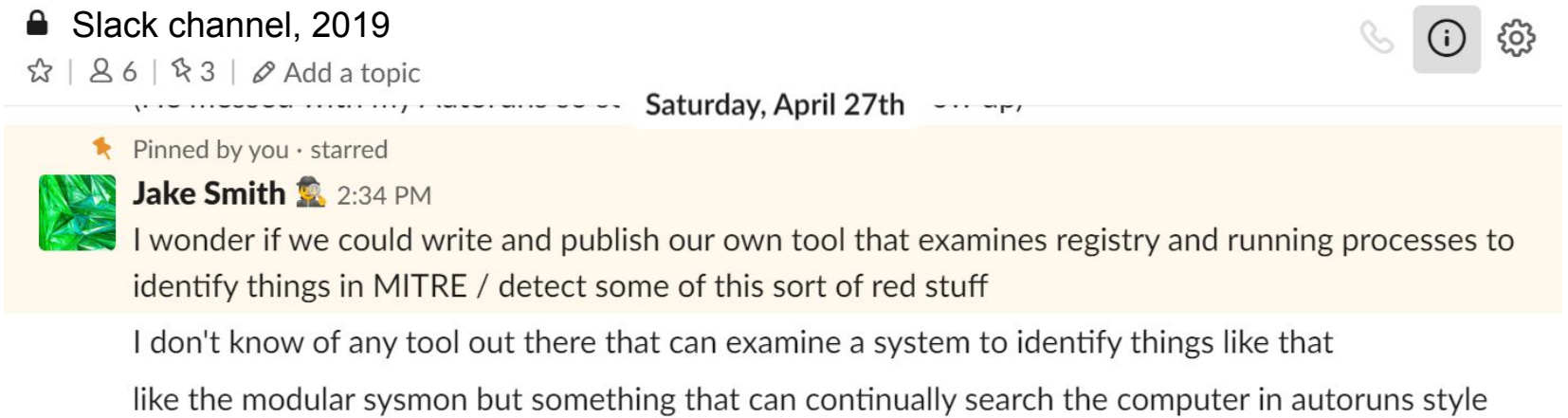
Hardening Scripts:

- Dozens of projects to try to audit or automatically harden systems
- Examples include Sysmon Configs, PowerSTIG, and random .bat/.ps1 scripts

Motivations and Goals

Why make BLUESPAWN?

“I wonder if we could build it”



The Original Idea

Active Defense Tool

Quickly identify threats and react

Know our coverage

Focus our time on less covered areas

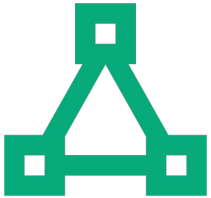
More Blue Team Software

Often little public info about defending against X technique

As Time Went On…

Continuous Active Defense

Towards monitoring the system



Networked Defense

Towards defending a fleet of systems

Proactive Defense

Towards applying mitigations / finding misconfigurations

Threat Hunting Methodology

More clearly defining Active Defense and applying it to find bad.

What is Active Defense?

- AVs and EDRs are meant to be quiet
  - Very low false positive rate
  - Generally a good thing
- BLUESPAWN is meant to be loud!
  - Value recall of detections over precision
  - Find everything!
- Careful analysis of everything on the system
  - Event logs, registry, filesystem, processes
- Bridge the gap between IR and EDR

Hunt-Centric Design

- “Hunts” for evidence of malware using MITRE ATT&CK techniques are the starting point
  - Event logs, registry, processes and files
- Monitor mode also uses these same hunts
  - Each hunt sets up its monitoring triggers
  - When a trigger is hit, the associated hunt(s) will run
- MITRE ATT&CK subtechniques are handled inside of the overarching technique’s hunt

Identifying Connections

- Malicious behavior usually has multiple components
- BLUESPAWN searches evidence for connections and associated evidence
  - Ex: registry key refers to a file
  - Builds a network of evidence
- More on this later

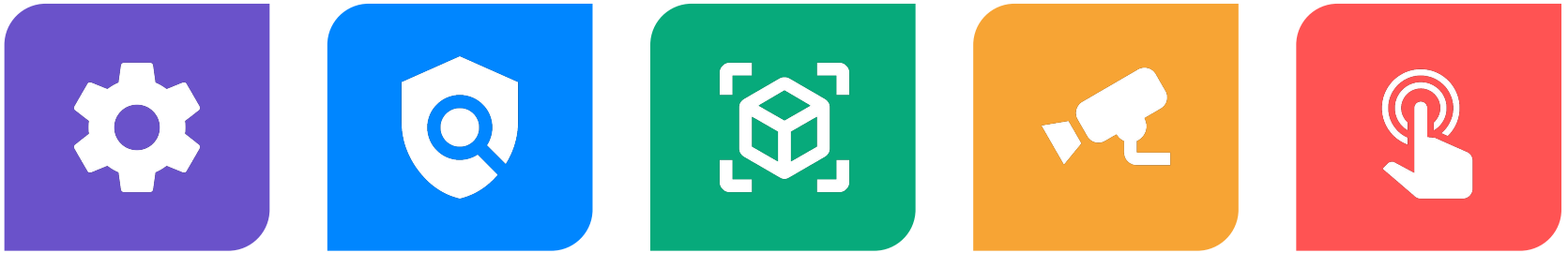
Features

Diving into BLUESPAWN's core modules and integrations

Why didn't BLUESPAWN catch X?

- BLUESPAWN is still in ALPHA development
- Development so far has focused on infrastructure
- All features presented currently exist and function as described
- Features are not without their bugs
- Features are not perfectly built out
- BLUESPAWN currently is best described as a framework

BLUESPAWN's Core Components



Mitigate Hunt Scan Monitor React

Mitigate

- Applies mitigations to the system
  - Based on DISA STIGs and MITRE Mitigations
- Can apply mitigations or simply audit
- Largely separate from the rest of BLUESPAWN
- Eventually
  - Configurable via file
  - Mitigate installed services
  - Aid in auditing

Some Definitions

- Detection - may or may not be bad
  - “Evidence” mentioned earlier
- Association - relationship between two detections
- Certainty - Metric for likelihood that a detection is bad
  - Intrinsic certainty - certainty derived from the detection itself
  - Contextual certainty - certainty derived from the context surround the detection
  - Associative certainty - certainty derived from associations to other detections

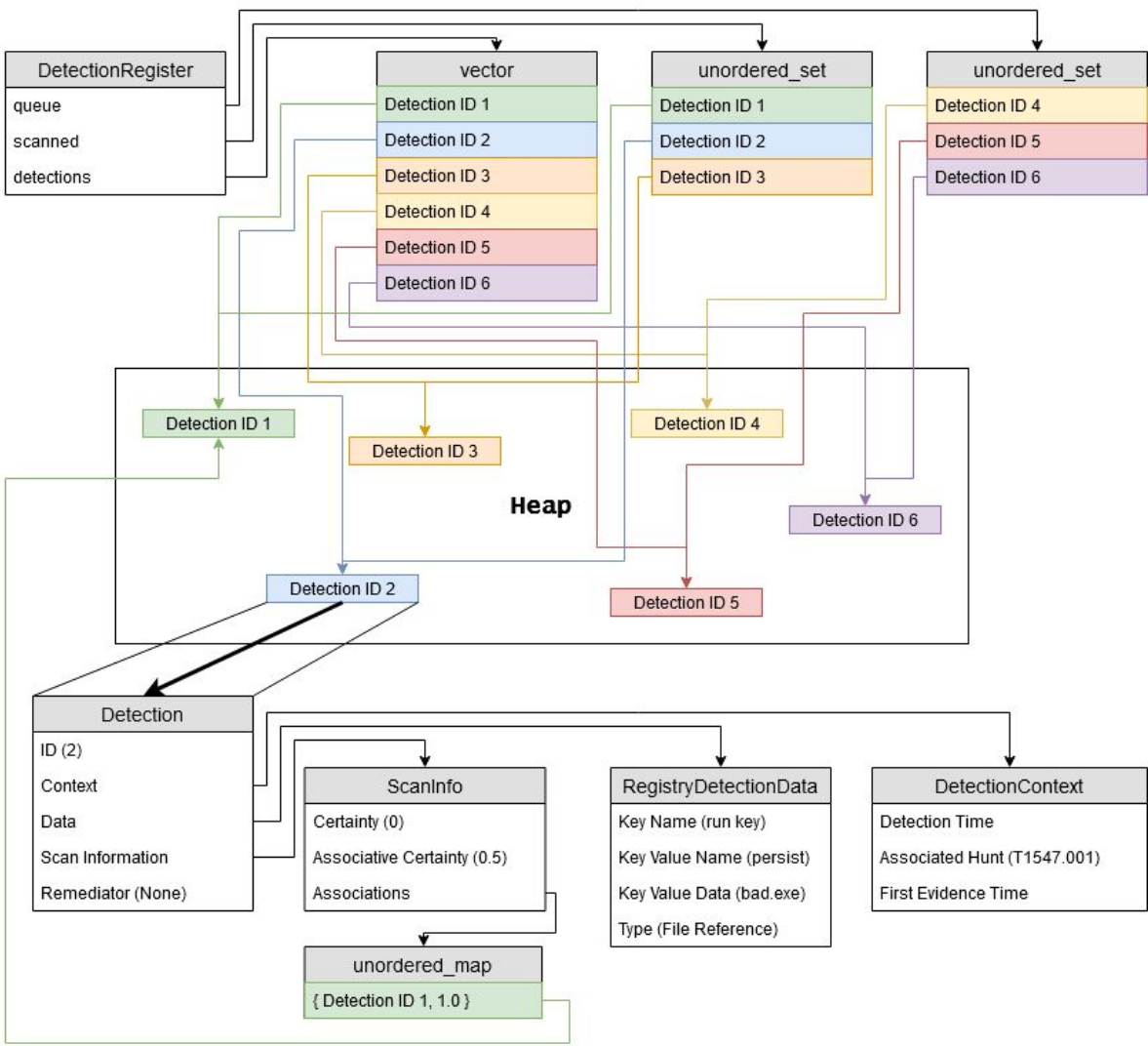
Detection Register

- Detections are sent to a detection register when created
  - Register sends back a pointer to the detection
    - May not be the same detection
- Three main storage mechanisms
  - Set of detections queued for a scan
  - Set of detections already scanned
  - Set of all detections

Detection Registration and Scanning

- If found, the detections are combined
  - Merge contextual certainties
    - Log and handle reactions
  - Merge context
- After deduplication, scans get queued
  - Calculate certainty
  - Identify associated detections
    - Register associations and repeat!
  - Log and handle reactions

Scan Internals



Hunt

- Designed to cover a MITRE ATT&CK technique
  - Example: Hunt T1547.001 scans run keys in the registry
- Checks any and all data sources available
  - Registry, filesystem, logs, OS features, and more to come
- Most items receives a “quick scan”
  - Designed to capture all detections that merit a more intensive scan
  - Detection is created if it fails the quick scan
- Hunts are broken into subtechniques and subsections

```cpp
std::vector<std::shared_ptr<Detection>> HuntT1547::RunHunt(const Scope& scope) {
    HUNT_INIT();

    Subtechnique001(scope, detections);
    Subtechnique002(scope, detections);
    Subtechnique004(scope, detections);
    Subtechnique005(scope, detections);
    Subtechnique010(scope, detections);

    HUNT_END();
}
```

Hunt Example

Features

```cpp
void HuntT1547::Subtechnique004(IN CONST Scope& scope, OUT std::vector<std::shared_ptr<Detection>>& detections) {
    SUBTECHNIQUE_INIT(004, Winlogon Helper DLL);

    SUBSECTION_INIT(WINLOGON, Cursory);
    // clang-format off
    auto userinitRegex{
        L"(C:\\\\[Ww](INDOWS|indows)\\\\[Ss](YSTEM32|ystem32)\\\\)?[Uu](SERINIT|serinit)\\.(exe|EXE),?" };
    std::vector<RegistryValue> winlogons{ CheckValues(HKEY_LOCAL_MACHINE,
        L"Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon", {
            { L"Shell", L"explorer\\.exe,?", false, CheckSzRegexMatch },
            { L"UserInit", userinitRegex, false, CheckSzRegexMatch }
        }, true, true) };
    // clang-format on

    for(auto& detection : winlogons) {
        // Moderate contextual certainty due to how rarely these values are used legitimately
        CREATE_DETECTION(Certainty::Moderate, RegistryDetectionData{ detection, RegistryDetectionType::FileReference });
    }
    SUBSECTION_END();

    SUBSECTION_INIT(WINLOGON_NOTIFY, Cursory);
    std::vector<RegistryValue> notifies{ CheckKeyValues(
        HKEY_LOCAL_MACHINE, L"Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Notify", true, true) };
    for(auto& notify : CheckSubkeys(
            HKEY_LOCAL_MACHINE, L"Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Notify", true, true)) {
        if(notify.ValueExists(L"DllName")) {
            notifies.emplace_back(RegistryValue{ notify, L"DllName", *notify.GetValue<std::wstring>(L"DllName") });
        }
    }

    for(auto& detection : notifies) {
        // Weak contextual certainty due to how rarely these values are used legitimately
        CREATE_DETECTION(Certainty::Weak, RegistryDetectionData{ detection, RegistryDetectionType::FileReference });
    }
    SUBSECTION_END();

    SUBTECHNIQUE_END();
}
```

Monitor

- Each hunt registers events
  - Events define what should trigger the hunt
  - Events can limit their scope to just a hunt subsection
- Event deduplication feature is in progress
- Eventually
  - Will use more robust ETW
  - More data sources
  - API monitor

Monitor Example

```
std::vector<std::pair<std::unique_ptr<Event>, Scope>> HuntT1546::GetMonitoringEvents() {
    std::vector<std::pair<std::unique_ptr<Event>, Scope>> events;

    // Looks for T1546.007: Netsh Helper DLL
    GetRegistryEvents(events, Scope::CreateSubhuntScope(NETSH_HELPER), HKEY_LOCAL_MACHINE,
                      L"SOFTWARE\\Microsoft\\Netsh", true, false, false);

    // Looks for T1546.008: Accessibility Features
    for(auto key : vAccessibilityBinaries) {
        Registry::GetRegistryEvents(events, Scope::CreateSubhuntScope(ACCESSIBILITY_HIJACK), HKEY_LOCAL_MACHINE,
                                    wsIFEO + key, true, false, false);
    }

    // Looks for T1546.009: AppCert DLLs
    GetRegistryEvents(events, Scope::CreateSubhuntScope(APPCERT_DLL), HKEY_LOCAL_MACHINE,
                      L"System\\CurrentControlSet\\Control\\Session Manager", true, false, false);

    // Looks for T1546.010: AppInit DLLs
    GetRegistryEvents(events, Scope::CreateSubhuntScope(APPINIT_DLL), HKEY_LOCAL_MACHINE,
                      L"Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows", true, false, false);

    // Looks for T1546.011: Application Shimming
    GetRegistryEvents(events, Scope::CreateSubhuntScope(APPLICATION_SHIM), HKEY_LOCAL_MACHINE,
                      L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\AppCompatFlags\\InstalledSDB");
    GetRegistryEvents(events, Scope::CreateSubhuntScope(APPLICATION_SHIM), HKEY_LOCAL_MACHINE,
                      L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\AppCompatFlags\\Custom");
    events.push_back(
        std::make_pair(std::make_unique<FileEvent>(FileSystem::Folder{ L"C:\\Windows\\AppPatch\\Custom" }),
                       Scope::CreateSubhuntScope(APPLICATION_SHIM)));
    events.push_back(
        std::make_pair(
            std::make_unique<FileEvent>(FileSystem::Folder{ L"C:\\Windows\\AppPatch\\Custom\\Custom64" }),
                                    Scope::CreateSubhuntScope(APPLICATION_SHIM)));

    // Looks for T1546.012: Image File Execution Options Injection
    Registry::GetRegistryEvents(
        events, Scope::CreateSubhuntScope(IFEO_HIJACK), HKEY_LOCAL_MACHINE,
        L"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options", true, false, true);

    // Looks for T1546.015: Component Object Model Hijacking
    if(Bluespawn::aggressiveness >= Aggressiveness::Intensive) {
        Registry::GetRegistryEvents(events, Scope::CreateSubhuntScope(COM_HIJACK), HKEY_LOCAL_MACHINE,
                                    L"SOFTWARE\\Classes\\CLSID", true, true, true);
    }

    return events;
```

React

- Framework for handling detections
  - Generally requires user input
- Marks remediated detections as stale
  - Other reactions may no longer be triggered
- Available options:
  - delete file, quarantine file, delete registry entry, suspend process, carve memory
- Eventually
  - More reactions to come

Example Reaction: Memory Carving

1. Kill threads with the specified memory section in the stack
2. Scan memory for pointers to memory section
   a. Patch pointee with return instruction if executable
3. Patch entrypoint and exported functions

- Effectively stops malicious image / memory section
  - Generally does not kill or restart process in question
- Threads shouldn't be able to easily re-enter malicious memory
- Experimental / in development features
  - Walk threads back instead of killing them
  - Obey calling conventions when patching

Integrations

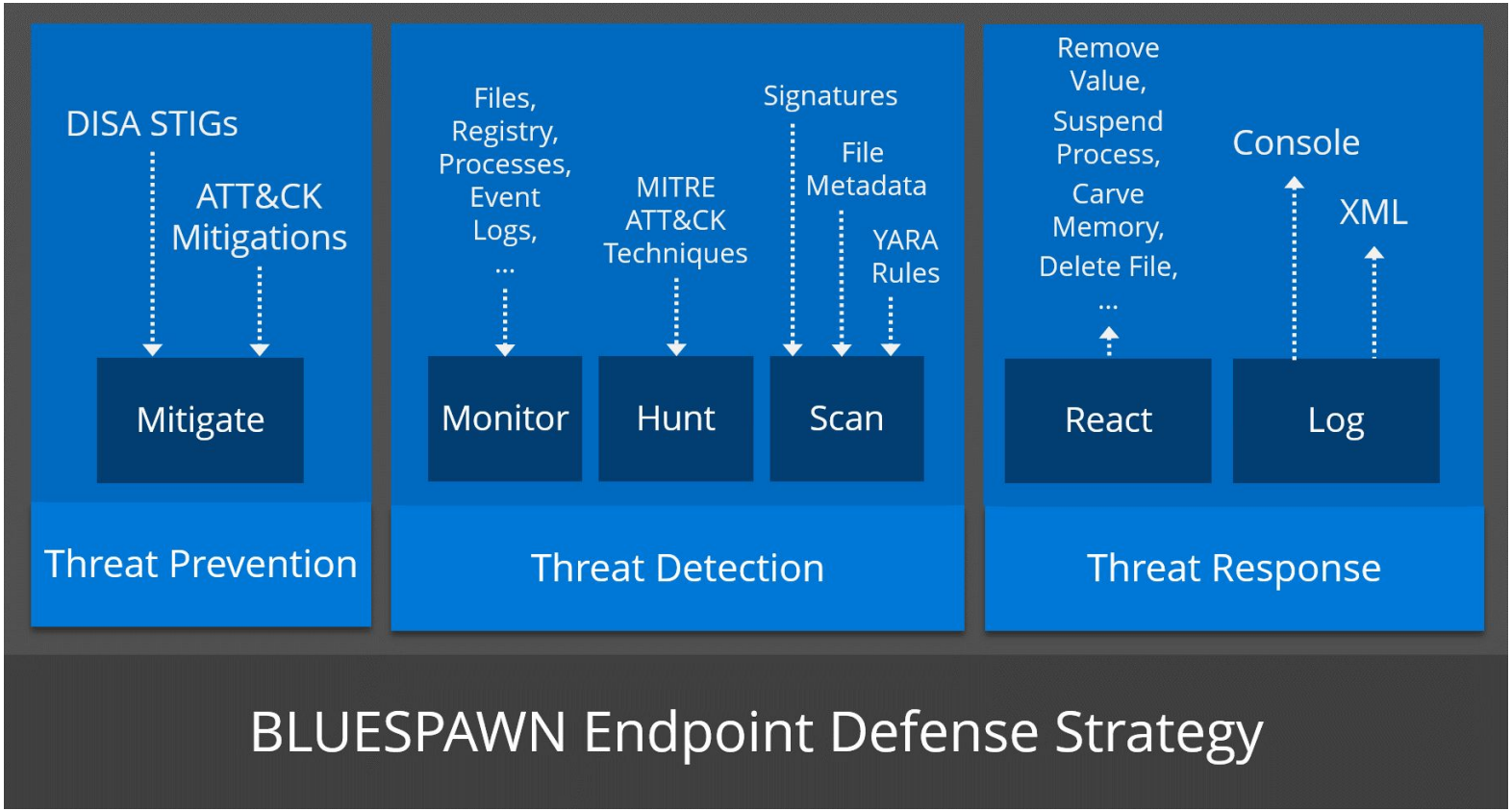
MITRE | ATT&CK®

Core methodology around which BLUESPAWN uses to look for bad and apply security settings

Community rules come prepackaged into BLUESPAWN for threat detection

Published, free guidance around securing various systems to a hardened baseline

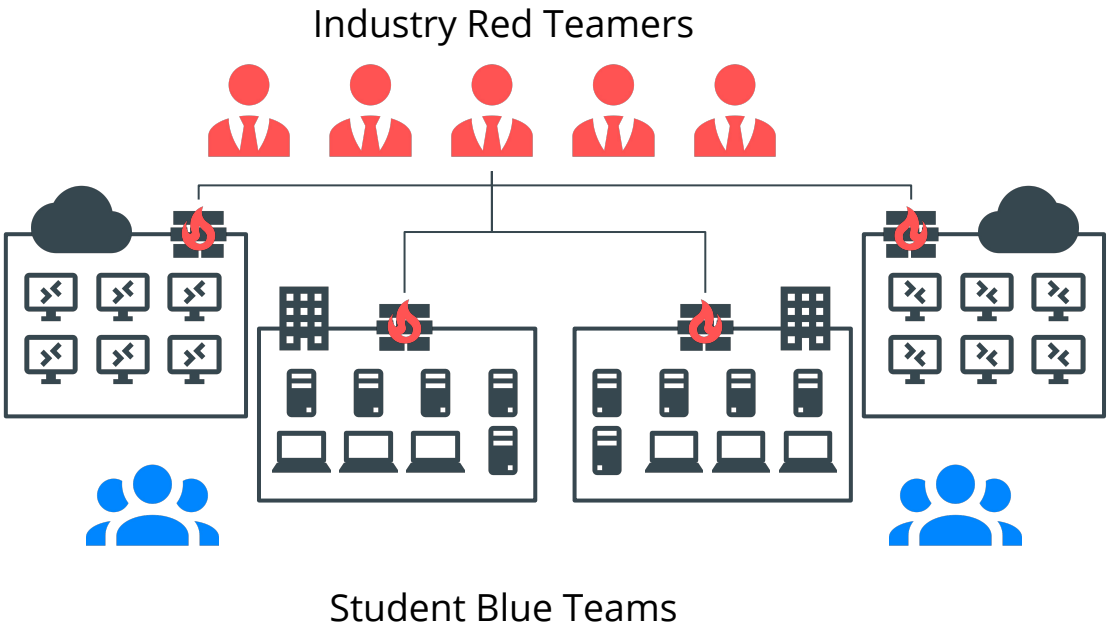
Used to scan memory and evaluate processes for malicious activity


DISA STIGs
ATT&CK Mitigations
Mitigate
Threat Prevention
Files, Registry, Processes, Event Logs, ...
MITRE ATT&CK Techniques
Signatures
File Metadata
YARA Rules
Monitor
Hunt
Scan
Threat Detection
Remove Value, Suspend Process, Carve Memory, Delete File, ...
Console
XML
React
Log
Threat Response
BLUESPAWN Endpoint Defense Strategy

Case Study and Demonstration

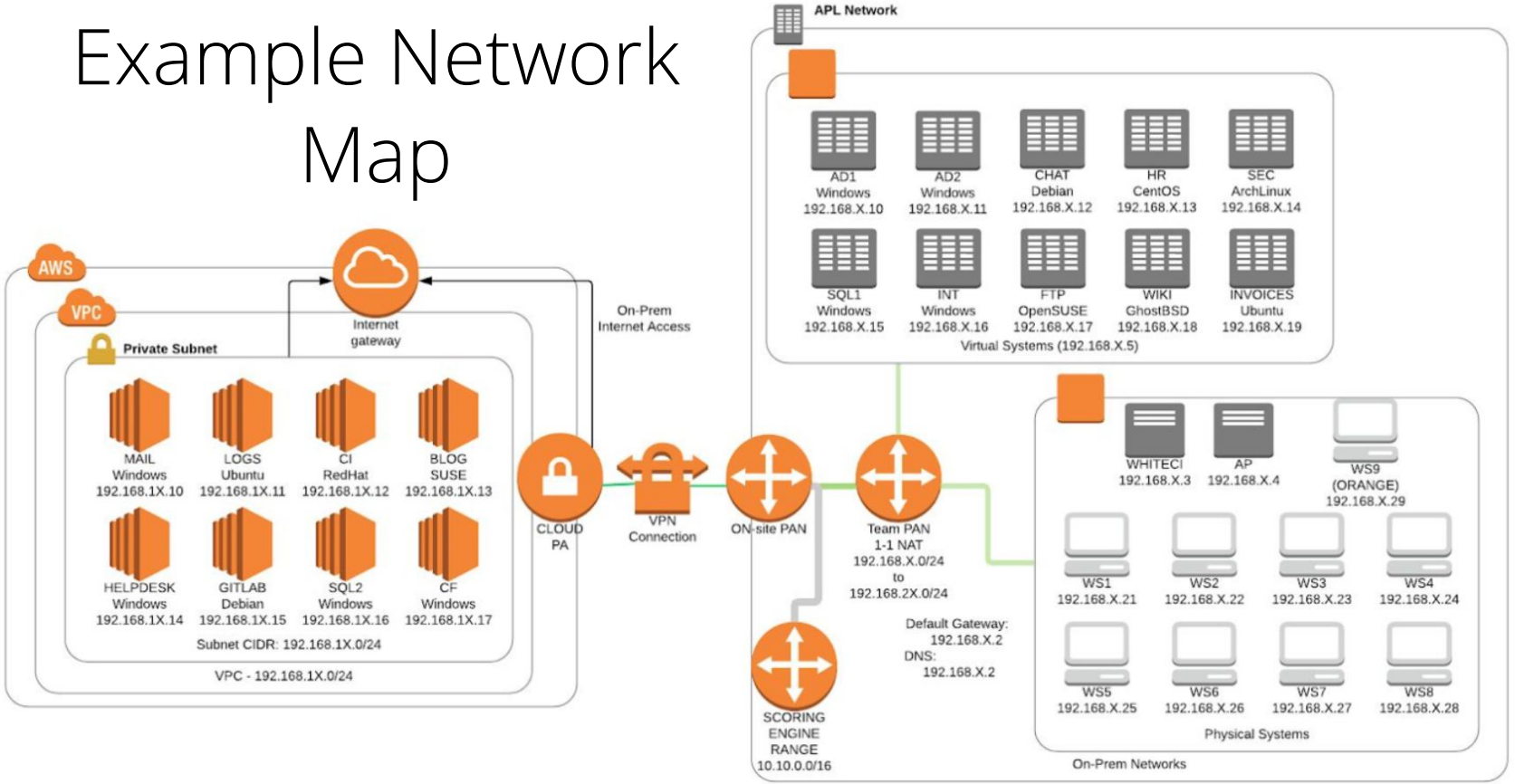
Exploring BLUESPAWN's effectiveness at the National Collegiate Cyber Defense Competition (NCCDC) and seeing it in action!

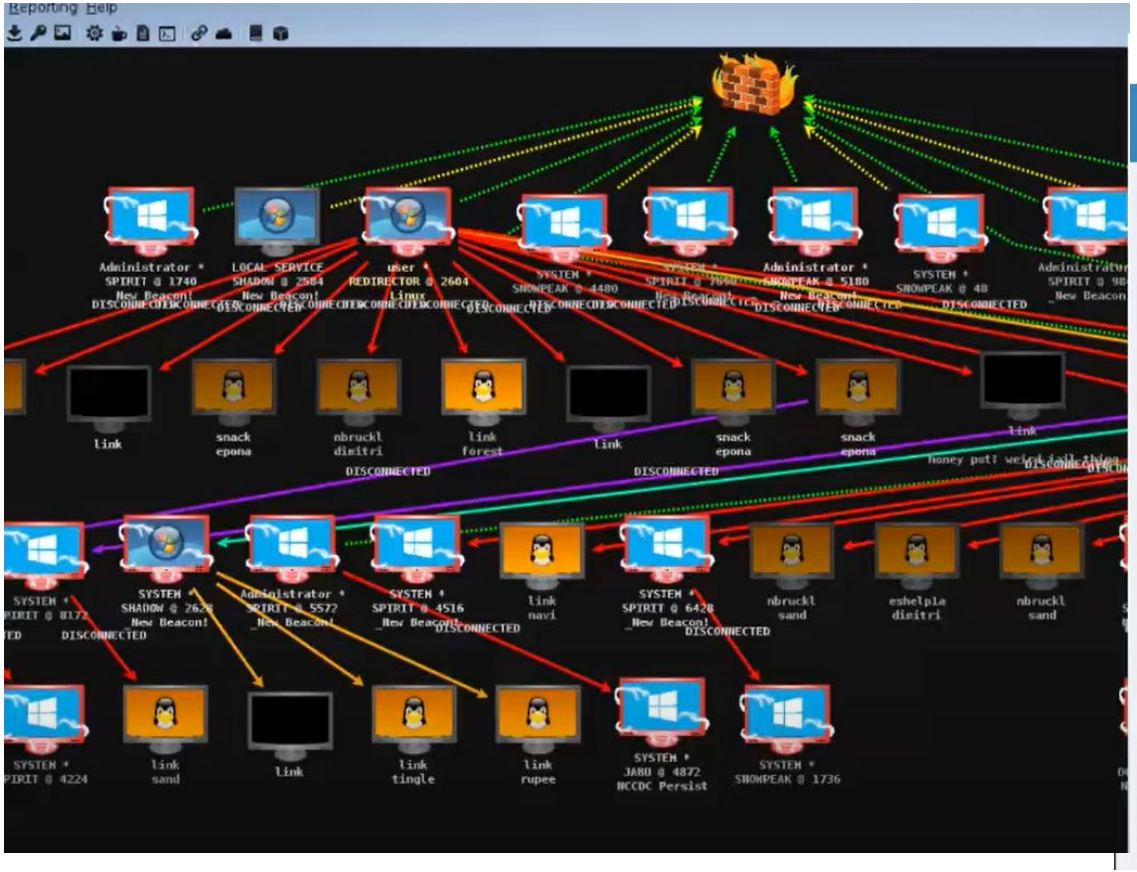
Collegiate Cyber Defense Competition (CCDC)

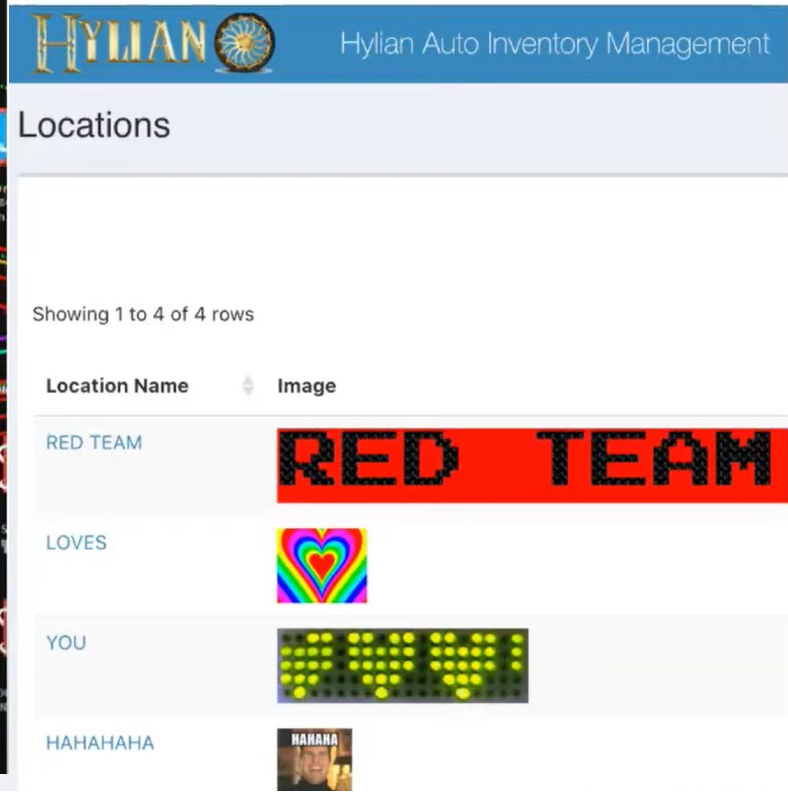


- Receive mix of Windows, Linux, ESXi, and Palo Alto FWs
- Great testing ground for defensive tools
- Red teams use Metasploit/Cobalt Strike to Custom Malware

Example Network Map

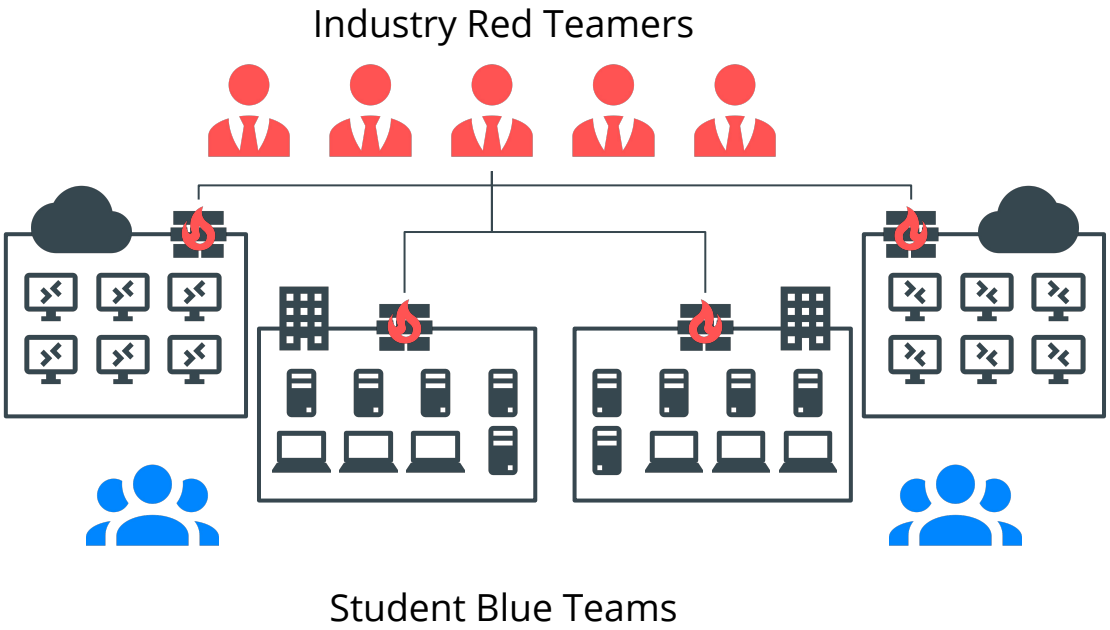




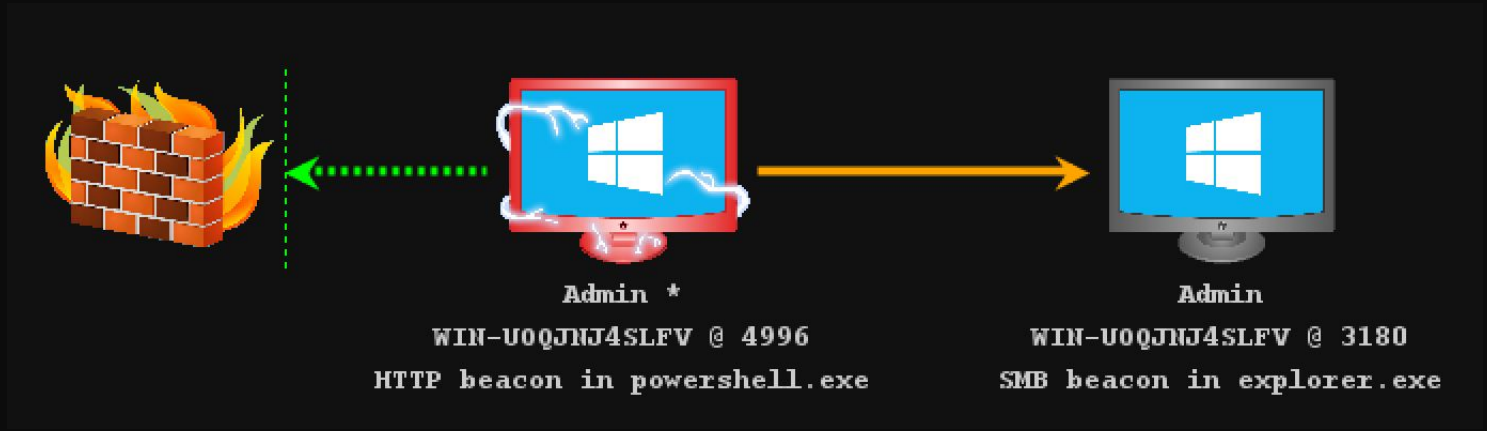


Screenshot Credit:
Dave Cowen, NCCDC Red Team Captain
Obtained from
https://www.youtube.com/watch?v=UsZhMRMGLMA in 2020

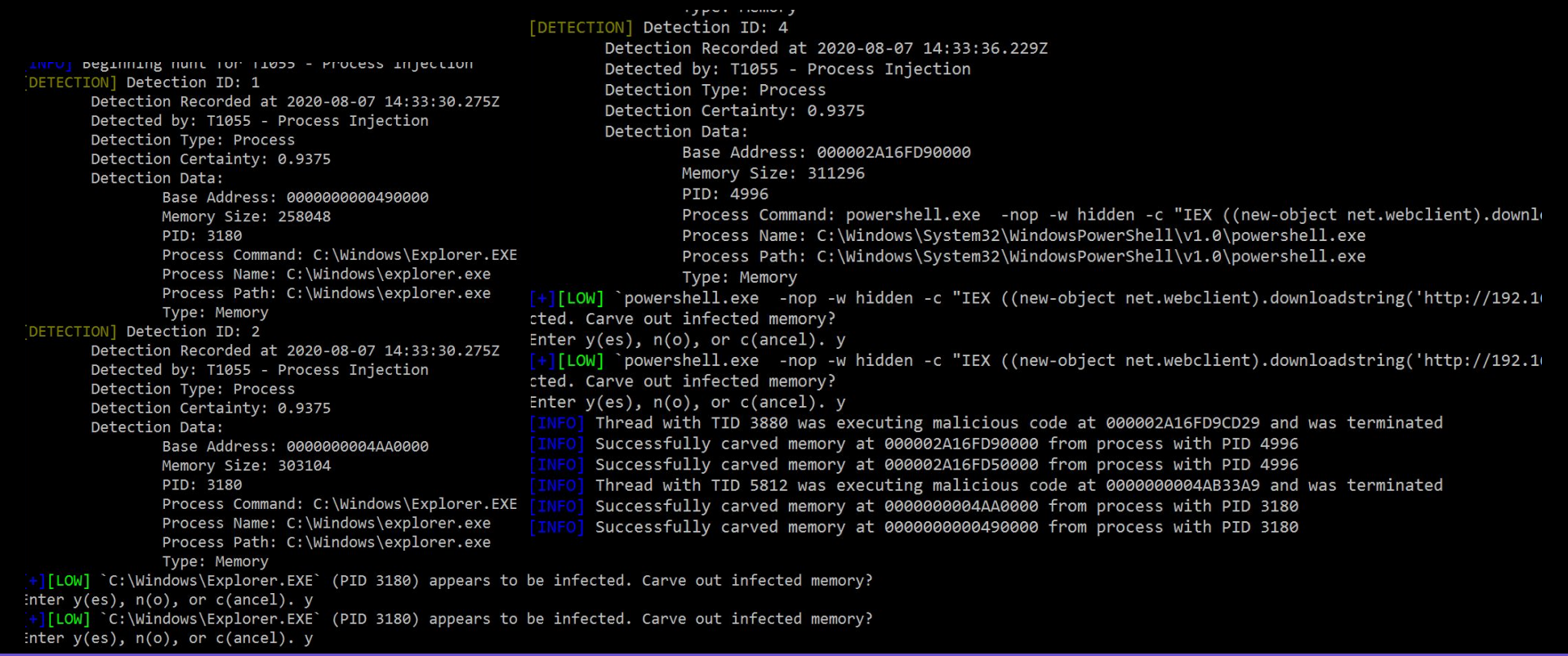
Collegiate Cyber Defense Competition (CCDC)



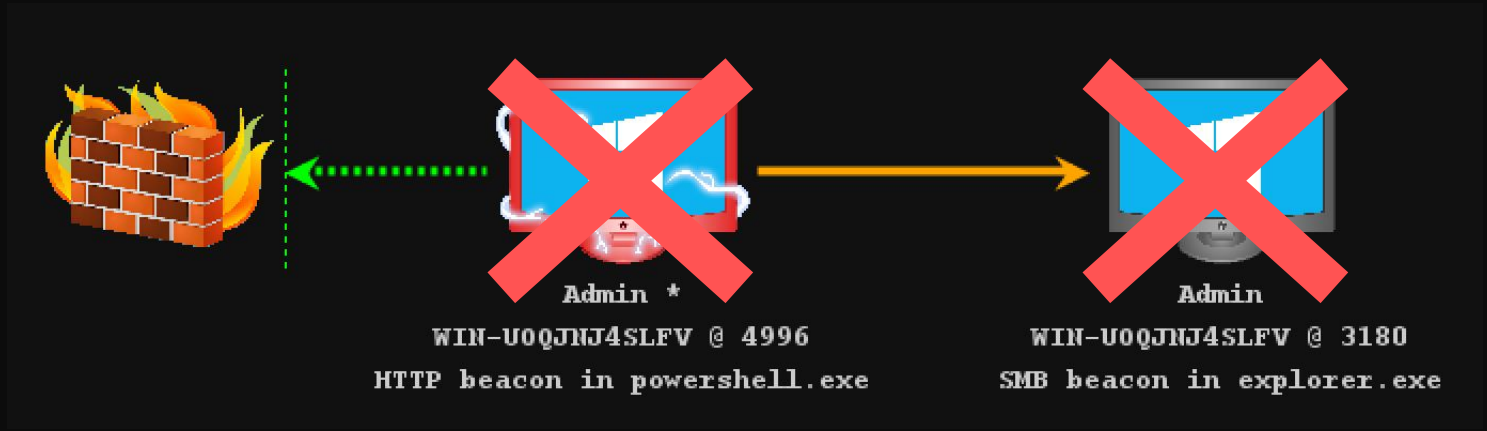
- Receive mix of Windows, Linux, ESXi, and Palo Alto FWs
- Great testing ground for defensive tools
- Red teams use Metasploit/Cobalt Strike to Custom Malware



Catching T1055 - Process Injection against Cobalt Strike



Catching T1055 - Process Injection against Cobalt Strike



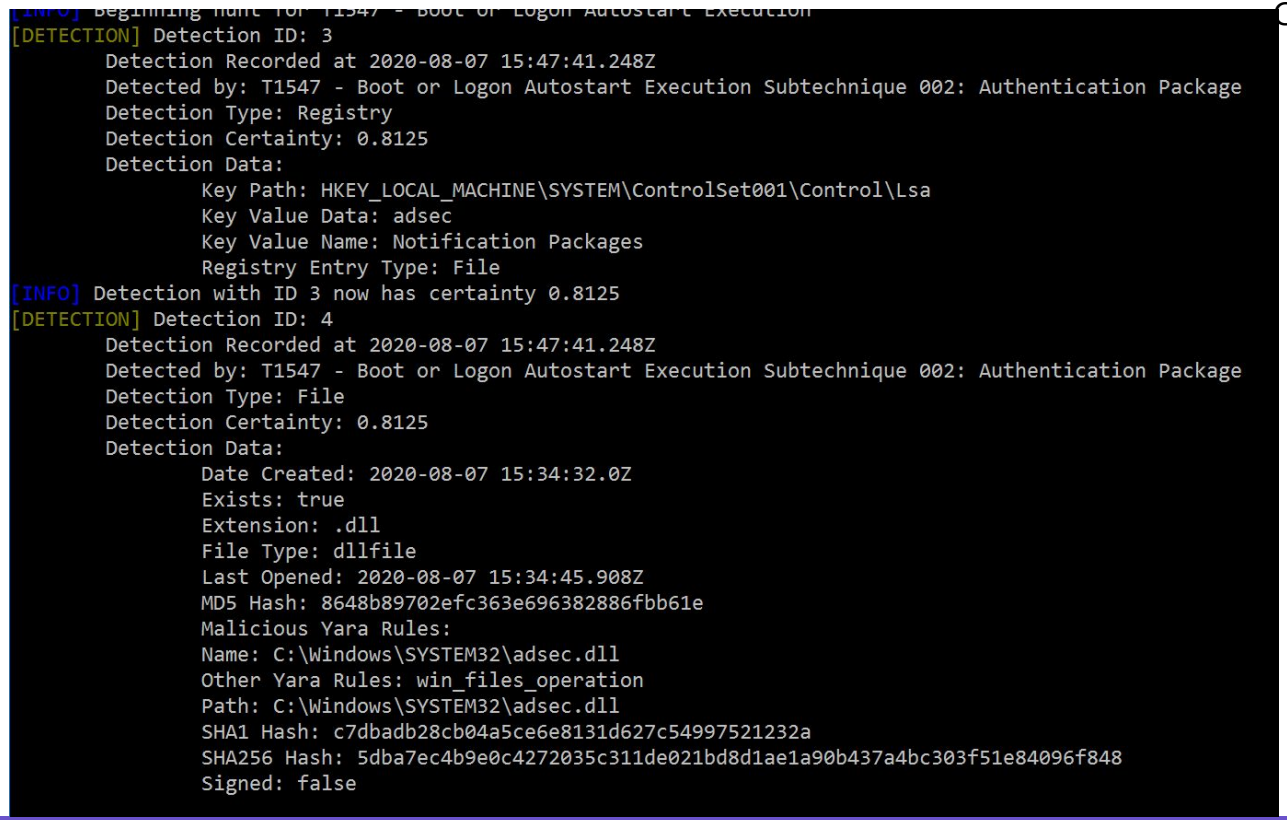
Both beacons have not stopped checking in 🙃

```
C:\Users\Administrator\Desktop>.\BLUESPAWN-client.exe --hunt -a Normal --hunts=T1546 --log=console,xml --react=remove-value

 ____  ____  ____  ____  ____  ____  ____  ____  ____
||B ||||L ||||U ||||E ||||S ||||P ||||A ||||W ||||N ||
||__||||__||||__||||__||||__||||__||||__||||__||||__||
|/__\||/__\||/__\||/__\||/__\||/__\||/__\||/__\||/__\|

[*][LOW] Starting a Hunt
[*][LOW] Starting a hunt for 1 techniques.
[*][LOW] Starting scan for T1546 - Event Triggered Execution
[INFO] Beginning hunt for T1546 - Event Triggered Execution
[INFO] Skipping T1546 - Event Triggered Execution Subtechnique 015: Component Object Model Hijacking subsection COM_HIJACK; rerun
 BLUESPAWN at Intensive to run this.
[INFO] Detections with IDs 1 and 5 now are associated with strength 1
[DETECTION] Detection ID: 1
        Detection Recorded at 2020-08-03 00:39:57.333Z
        Detected by: T1546 - Event Triggered Execution Subtechnique 008: Accessibility Features
        Detection Type: Registry
        Detection Certainty: 1
        Detection Data:
                Key Path: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe
                Key Value Data: c:\windows\system32\cmd.exe
                Key Value Name: Debugger
                Registry Entry Type: Command
[+][LOW] Registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe` co
ntains potentially malicious value `Debugger` with data `c:\windows\system32\cmd.exe`. Remove this value?
```

Identifying and remove Sticky Keys Backdoor (T1546.008)



Uncommon technique T1547.002: Malicious Authentication Package

```
C:\Users\Administrator\Desktop>.\BLUESPAWN-client.exe --hunt -a Normal --hunts=T1547 --log=console,xml

    _/_/_/    _/        _/    _/  _/_/_/_/    _/_/_/  _/_/_/      _/_/    _/          _/  _/      _/
   _/    _/  _/        _/    _/  _/          _/        _/    _/  _/    _/  _/          _/  _/_/    _/
  _/_/_/    _/        _/    _/  _/_/_/      _/_/    _/_/_/    _/_/_/_/  _/    _/    _/  _/  _/  _/
 _/    _/  _/        _/    _/  _/              _/  _/        _/    _/    _/  _/  _/    _/    _/_/
_/_/_/    _/_/_/_/    _/_/    _/_/_/_/  _/_/_/    _/        _/    _/      _/  _/      _/      _/


[*][LOW] Starting a Hunt
[*][LOW] Starting a hunt for 1 techniques.
[*][LOW] Starting scan for T1547 - Boot or Logon Autostart Execution
[INFO] Beginning hunt for T1547 - Boot or Logon Autostart Execution
[INFO] Detections with IDs 1 and 7 now are associated with strength 1
[DETECTION] Detection ID: 1
        Detection Recorded at 2020-08-03 00:54:12.354Z
        Detected by: T1547 - Boot or Logon Autostart Execution Subtechnique 001: Registry Run Keys / S
        Detection Type: Registry
        Detection Certainty: 0.5
        Detection Data:
                Key Path: HKEY_USERS\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run
                Key Value Data: C:\Windows\TEMP\BLobTVBsZqA.vbs
                Key Value Name: XnFRIQqAmF
                Registry Entry Type: Command
[INFO] Detection with ID 1 now has certainty 0.5
[INFO] Detections with IDs 7 and 8 now are associated with strength 0.75
[DETECTION] Detection ID: 7
        Detection Recorded at 2020-08-03 00:54:12.495Z
        Detection Type: Process
        Detection Certainty: 0.5
        Detection Data:
                Process Command: C:\Windows\TEMP\BLobTVBsZqA.vbs
                Type: Command
[INFO] Detection with ID 7 now has certainty 0.5
```

T1547.001: The Run Key

- Located suspicious run key
- Created detections for both key and corresponding .vbs file

Catching T1505 - a classic PHP Web Shell on our IIS server

```
[INFO] C:\inetpub\wwwroot\images\spy.php matches known malicious identifier possibleIndicator
[INFO] C:\inetpub\wwwroot\images\spy.php matches known malicious identifier webshell_PHP_r57142
[INFO] C:\inetpub\wwwroot\images\spy.php matches known malicious identifier webshell_2008_2009lite_2009mssql
[INFO] C:\inetpub\wwwroot\images\spy.php matches known malicious identifier webshell_gfs_sh_r57shell_r57shell127_SnIpEr_SA_xxx
[INFO] C:\inetpub\wwwroot\images\spy.php matches known malicious identifier webshell_c99_locus7s_c99_w4cking_xxx
[INFO] C:\inetpub\wwwroot\images\spy.php matches known malicious identifier sig_2008_php_php
[INFO] C:\inetpub\wwwroot\images\spy.php matches known malicious identifier multiple_php_webshells
[INFO] Located likely web shell in file C:\inetpub\wwwroot\images\spy.php in text eval
[DETECTION] Detection ID: 1
        Detection Recorded at 2020-08-03 00:20:39.733Z
        Detected by: T1505 - Server Software Component Subtechnique 003: Web Shell
        Detection Type: File
        Detection Certainty: 0.75
        Detection Data:
                Date Created: 2020-08-03 00:10:02.355Z
                Exists: true
                Extension: .php
                Last Opened: 2020-08-03 00:10:02.745Z
                MD5 Hash: 488c915a723ebf96c615afc96d0b3e62
                Malicious Yara Rules: obfuscatedFunctionality, possibleIndicator, webshell_PHP_r57142, webshell_2008_2009lite_20
ll127_SnIpEr_SA_xxx, webshell_c99_locus7s_c99_w4cking_xxx, sig_2008_php_php, multiple_php_webshells
                Name: C:\inetpub\wwwroot\images\spy.php
                Other Yara Rules:
                Path: C:\inetpub\wwwroot\images\spy.php
                SHA1 Hash: cd7937ffece79b2a3e0e7e7fe5b9aaa4e17799c4
                SHA256 Hash: e2f61160c8fa0f14e5f02e8a9bd8f9f855faf5c5812b8bbc3c8ce97b2a0d53f2
                Signed: false
```

BLUESPAWN @ CCDC: T1543 - Create/Modify Service

```
[DETECTION] Detection ID: 1
        Detection Recorded at 2020-08-07 15:30:29.575Z
        First Evidence at 2020-08-07 15:30:22.635Z
        Detected by: T1543 - Create or Modify System Process Subtechnique 003: Windows Service
        Detection Type: Service
        Detection Certainty: 0.5
        Detection Data:
                Display Name: Intel Graphics User Interface Service
                Service Executable: C:\Windows\system32\cmd.exe
```

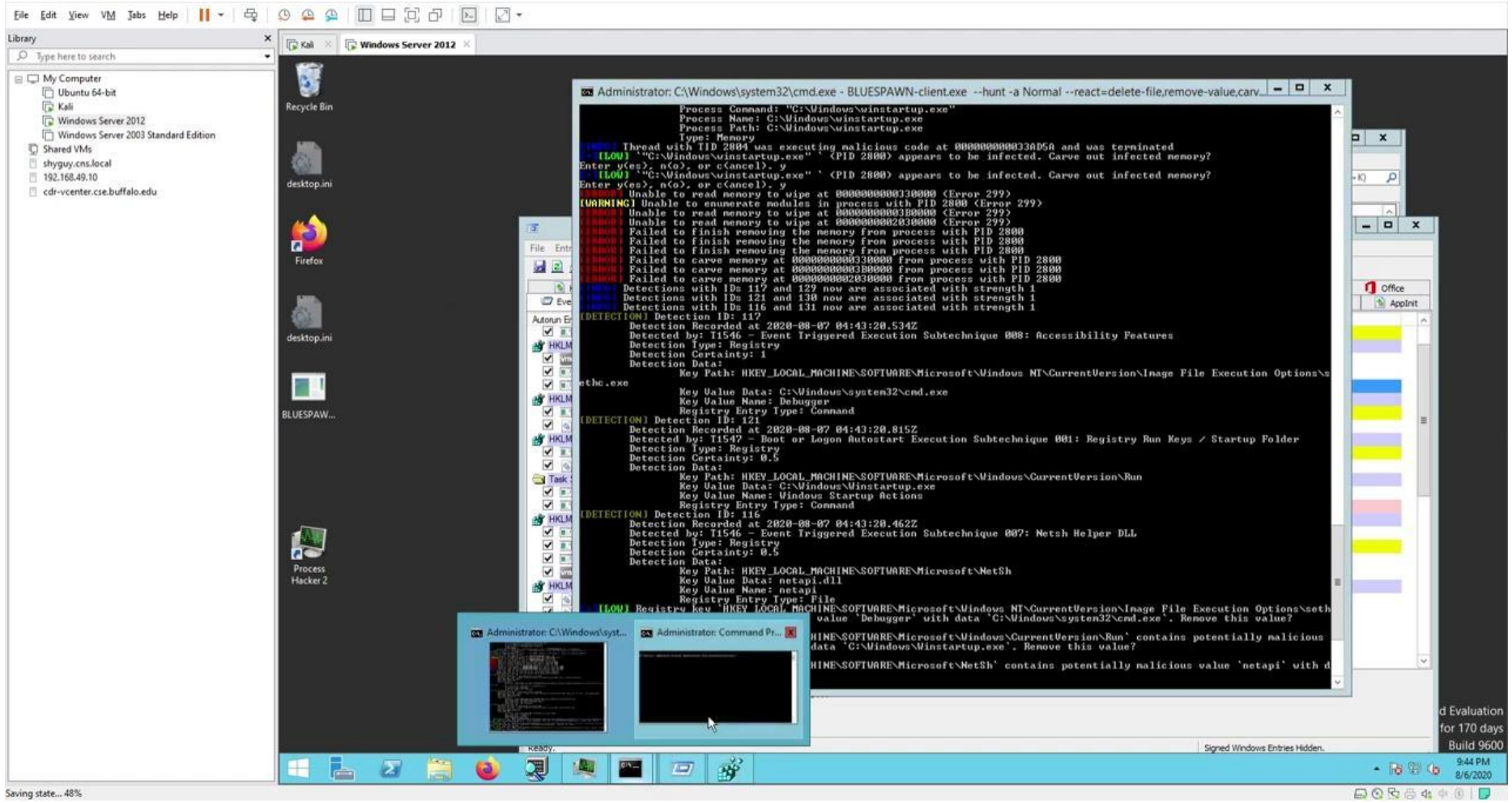
Red Team Perspective / Reflection

- Spent a decent amount of time testing against the tool specifically
- Could not rely on certain tactics as in previous years
  - “Great tool, does a great job. At some of the regions it was a real pain for our teams” - Dave Cowen, Red Team Captain
- Created counter tool called REDSPAWN to mitigate effectiveness of BLUESPAWN

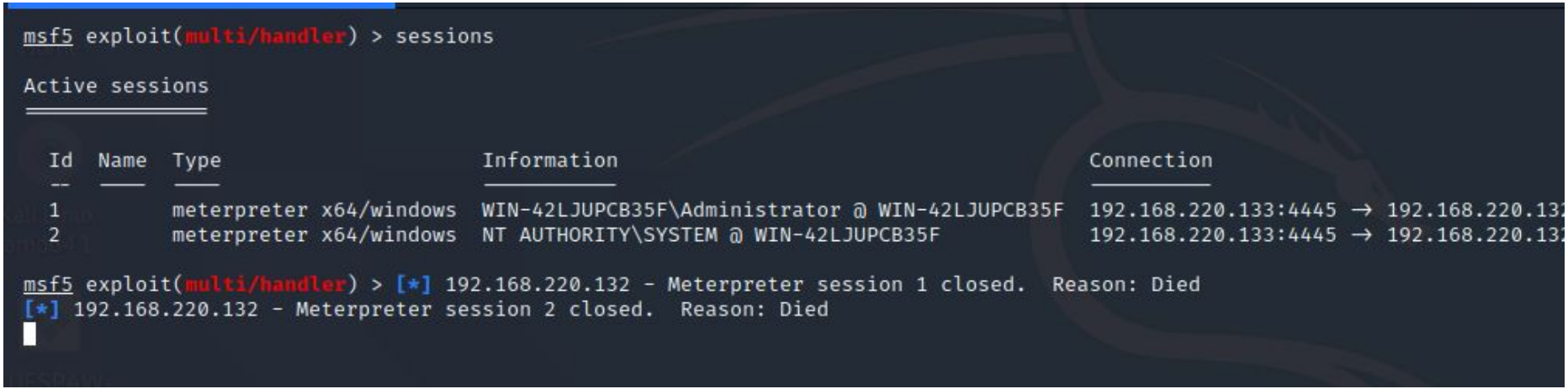
Blue Team Perspective / Reflection

- Greatly reduced time to initial detection and response
- Acted as a force multiplier - our linux focused teammates could triage their workstations without knowing much Windows
- Helpful in finding stuff that is easily missed or would require lots of manual effort to locate

Demo


File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu 64-bit
Kali
Windows Server 2012
Windows Server 2003 Standard Edition
Shared VMs
shyguy.cns.local
192.168.49.10
cdr-vcenter.cse.buffalo.edu
Kali
Windows Server 2012
Recycle Bin
desktop.ini
Firefox
desktop.ini
BLUESPAW...
Process Hacker 2
Administrator: C:\Windows\system32\cmd.exe - BLUESPAWN-client.exe --hunt -a Normal --react=delete-file,remove-value,carv...
Process Command: "C:\Windows\winstartup.exe"
Process Name: C:\Windows\winstartup.exe
Process Path: C:\Windows\winstartup.exe
Type: Memory
[INFO] Thread with TID 2804 was executing malicious code at 00000000033AD5A and was terminated
[LOW] "C:\Windows\winstartup.exe" ` (PID 2800) appears to be infected. Carve out infected memory?
Enter y(es), n(o), or c(ancel). y
[LOW] "C:\Windows\winstartup.exe" ` (PID 2800) appears to be infected. Carve out infected memory?
Enter y(es), n(o), or c(ancel). y
[ERROR] Unable to read memory to wipe at 0000000000330000 (Error 299)
[WARNING] Unable to enumerate modules in process with PID 2800 (Error 299)
[ERROR] Unable to read memory to wipe at 0000000000310000 (Error 299)
[ERROR] Unable to read memory to wipe at 0000000002030000 (Error 299)
[ERROR] Failed to finish removing the memory from process with PID 2800
[ERROR] Failed to finish removing the memory from process with PID 2800
[ERROR] Failed to finish removing the memory from process with PID 2800
[ERROR] Failed to carve memory at 0000000000330000 from process with PID 2800
[ERROR] Failed to carve memory at 0000000000310000 from process with PID 2800
[ERROR] Failed to carve memory at 0000000002030000 from process with PID 2800
[INFO] Detections with IDs 117 and 129 now are associated with strength 1
[INFO] Detections with IDs 121 and 130 now are associated with strength 1
[INFO] Detections with IDs 116 and 131 now are associated with strength 1
[DETECTION] Detection ID: 117
Detection Recorded at 2020-08-07 04:43:20.534Z
Detected by: T1546 - Event Triggered Execution Subtechnique 008: Accessibility Features
Detection Type: Registry
Detection Certainty: 1
Detection Data:
Key Path: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\s
ethc.exe
Key Value Data: C:\Windows\system32\cmd.exe
Key Value Name: Debugger
Registry Entry Type: Command
[DETECTION] Detection ID: 121
Detection Recorded at 2020-08-07 04:43:20.815Z
Detected by: T1547 - Boot or Logon Autostart Execution Subtechnique 001: Registry Run Keys / Startup Folder
Detection Type: Registry
Detection Certainty: 0.5
Detection Data:
Key Path: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Key Value Data: C:\Windows\Winstartup.exe
Key Value Name: Windows Startup Actions
Registry Entry Type: Command
[DETECTION] Detection ID: 116
Detection Recorded at 2020-08-07 04:43:20.462Z
Detected by: T1546 - Event Triggered Execution Subtechnique 007: Netsh Helper DLL
Detection Type: Registry
Detection Certainty: 0.5
Detection Data:
Key Path: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetSh
Key Value Data: netapi.dll
Key Value Name: netapi
Registry Entry Type: File
[LOW] Registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\seth
value `Debugger` with data `C:\Windows\system32\cmd.exe`. Remove this value?
HINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` contains potentially malicious
data `C:\Windows\Winstartup.exe`. Remove this value?
HINE\SOFTWARE\Microsoft\NetSh` contains potentially malicious value `netapi` with d
Administrator: C:\Windows\syst...
Administrator: Command Pr...
Office
AppInit
Signed Windows Entries Hidden.
Evaluation
for 170 days
Build 9600
9:44 PM
8/6/2020
Saving state... 48%

The shells


msf5 exploit(multi/handler) > sessions
Active sessions
Id Name Type Information Connection
1 meterpreter x64/windows WIN-42LJUPCB35F\Administrator @ WIN-42LJUPCB35F 192.168.220.133:4445 → 192.168.220.13
2 meterpreter x64/windows NT AUTHORITY\SYSTEM @ WIN-42LJUPCB35F 192.168.220.133:4445 → 192.168.220.13
msf5 exploit(multi/handler) > [*] 192.168.220.132 - Meterpreter session 1 closed. Reason: Died
[*] 192.168.220.132 - Meterpreter session 2 closed. Reason: Died

Future Work

Focusing on expanding BLUESPAWN's detection capabilities and coverage

General Approach

Coverage

- Adding new hunts & mitigations
- Adding new data sources
- More robust scanning
- Integrated automated IR

Configurability

- More ways to control how BLUESPAWN runs (ie input and output)
- Make signatures and definitions more modular

Integrity

- Closer integration with Atomic Red Team and better automated testing
- Protect BLUESPAWN from tampering
- Ensure data sources are genuine

Other BLUESPAWN Components

Linux Client

On the way!

BLUESPAWN Server

Deploy, collect logs from, and manage clients at scale

BLUESPAWN Cloud

Perform deeper analysis

Conclusion

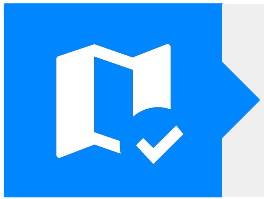
BLUESPAWN in a nutshell

- Fully open-source, active defense and EDR tool for Windows
- Demonstrates how modern-day security solutions in a transparent manner
- Integrates popular community libraries and tools such as MITRE ATT&CK, DoD STIGs, YARA, and PE-Sieve
- Enable security analysts to quickly detect, identify, and eliminate malicious activity on a system

Applying the concepts of BLUESPAWN

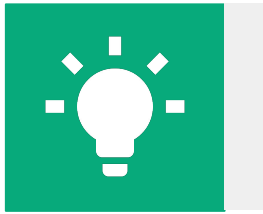
Align defenses with tactics

(ex: using MITRE ATT&CK)



Know your coverage

“Having confidence in certain lines of effort enables you to focus more in other areas”



Understand how defenses operate

Apply the hacker mindset and tear apart defensive software to learn how it works so it can be improved too

Get Involved & Questions Time

https://github.com/ION28/BLUESPAWN/

(slides are linked in Github README)

https://discord.gg/JBn7Ee

bluespawn@virginia.edu

Let us know if you:

- have ideas for features,
- want to help develop,
- found a bug, or
- just have a question!