Thesis Portfolio

BLUESPAWN: An Open-Source, Active Defense & Endpoint Detection and Response (EDR) Software for Windows-based Systems
(Technical Report)

Investigating the Politics of Open Source Software
(STS Research Paper)

An Undergraduate Thesis

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Calvin Krist
Fall, 2020

Department of Computer Science

Table of Contents

Sociotechnical Synthesis

STS 4600 - Doc Jacques

Fall 2020

Level 1

After years of competing in collegiate cyber defence competitions at a high level, my teammates and I noticed a strange discrepancy. State of the art *offensive* technology included free, open source tools (meaning their source code is freely available), while *defensive* technology was prohibitively expensive and closed source. This resulted in a barrier to our growth as cybersecurity professionals because we could not learn how the state of the art functioned and what advanced defensive techniques they used. Thus, our technical project worked to create an open source, state of the art cybersecurity solution built around theoretical frameworks. My original inspiration for my STS research project was to investigate why this discrepancy between offensive and defensive technology existed, and in the process I learned about the politics of open source software. I found that a fascinating topic and shifted my focus to solely explore the ideology and politics of open source software and its relevance to all software developers. Open source software has grown into a multi-billion dollar industry so an exploration of this topic is important for all software developers to understand.

My technical project aimed to create an advanced, open source cybersecurity solution. The main portion of our development so far has focused on a “client”, meaning software that runs on a desktop, but a series of servers providing analytical tools is under development. The client consists of three main modes. The first is “Hunt”, wherein it searches the computer for signs of malicious activity. The module is built around the MITRE ATT&CK Framework, which provides a theoretical framework for how malicious actors attack. It proposes multiple stages of attacks, and under each lists a number of techniques that are used. The “Hunt” framework is built around a series of programs that look for traces of these techniques to identify malicious activity. By designing it around a theoretical framework, we help ensure its effectiveness against unknown actors as well as robustness against actors who test malware against our project. The second mode is “Monitor”, which constantly monitors the computer and decides if a hunt needs to be run. This is analogous to an antivirus, but the underlying hunts can find types of malware that normal consumer antivirus’ cannot. The final mode is “Mitigate”, which seeks to reduce the attack surface before an incident occurs. It searches the computer for incorrect settings and other bad practices, and then helps the operator fix them. As far as we are aware, there is no currently existing software that does this in a way that is safe to use in a business setting besides our project. This is because scripts that automatically fix bad settings might break software and cost a business millions of dollars, so an informed human operator must be involved in the process. Together, these three modes result in an advanced defensive solution that can actively monitor a system for advanced threats or assist a trained operator in seeking malware. Our project has been in development for over a year now, and has proved very successful. We used it at NCCDC, a national defensive competition, half a year ago to help us win the title. The professional hackers we fought were impressed by our project and found it made it much more difficult for them to operate. We presented it at DEFCON, a well regarded cybersecurity conference, this summer, and are now training

CCDC teams from other regions on how to use our software. Security leadership at multiple corporations are following the development of our project and are impressed by what it can achieve.

My STS research traces the development of the Internet from Cold-War era US military programs and argues that it was politically motivated. From there, it shows how there were early ideological divisions between the military and the academic researchers that assisted them, and how many examples of early computing technology in the 70's were explicitly a form of protest against the politics of the military. It then shows how the groups responsible for this protest became troubled by the increasing copyright of software, and started the "Free Software" movement in the 80's, referring to the belief that there are fundamental human rights in regards to software. In 1999, the open source movement came out of the free software community but dropped the ideological overtones. Thus, open source software is closely connected to explicitly political purposes, and in general the Internet and computing technology is a very political space. My research then argues that many of the ways that these politics influence the Internet and similar fundamental parts of society are invisible to most members, and thus technologically informed people such as software developers *must* be aware of the politics of their software. Without this awareness, it is too easy for actors such as the military to influence the development of computing technology to align with their politics, and important values such as freedom and democracy can be reduced or lost.

BLUESPAWN: An Open-Source, Active Defense &
Endpoint Detection and Response (EDR) Software
for Windows-based Systems

A Technical Report Submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia - Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Calvin Krist
Fall 2019, Fall 2020

Jake Smith (jts5np)
Jack McDowell (jnm3ecm)
Calvin Krist (czk4ja)
Will Mayes (wtm4bp)

Signature ______________________ Date 11/23/20

Approved Yonghwi Kwon Date 12/01/2020

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

BLUESPAWN: An Open-Source, Active Defense & Endpoint Detection and Response (EDR) Software for Windows-based Systems

By:

Jake Smith (jts5np)

Jack McDowell (jnm3ecm)

Calvin Krist (czk4ja)

Will Mayes (wtm4bp)

Advised by:

Yonghwi Kwon (yk2bb)

8 May 2020

Department of Computer Science

University of Virginia

Table of Contents

Abstract

In today's world, computers running Microsoft's Windows operating system remain a top target for threat actors given its popularity. While there are a number of commercial defensive cybersecurity tools and multi-purpose system analysis programs such as SysInternals, this software is often closed-source, operates in a black-box manner, or requires a payment to obtain. These characteristics impose costs for both attackers and defenders. In particular, while the restrictions prevent attackers from knowing exactly what these tools detect, defenders often end up not having a good understanding of how their tools work or exactly what malicious activity they can identify.

Building on prior work and other open-source software, our team decided to create BLUESPAWN. This open-source program is an active defense and endpoint detection & response (EDR) tool designed to quickly prevent, detect, and eliminate malicious activity on a Windows system. In addition, BLUESPAWN is centered around the MITRE ATT&CK Framework and the Department of Defense's published STIGs. We have also integrated popular malware analysis libraries such as VirusTotal's YARA to increase the tool's effectiveness and accessibility [1]. Currently, our team is developing the alpha version of the client which can already detect real-world malware. In the future, we will continue to build out the client and eventually integrate both a server component for controlling clients and a cloud component to deliver enhanced detection capabilities.

Authors' Note

The BLUESPAWN Project is released under the GNU General Public License v3.0 (GPL-3.0) license. Note that the project integrates several other third party code libraries to provide additional features/detections which are themselves published under different licenses. These projects are not necessarily affiliated with BLUESPAWN or its authors and their use does not indicate their support or endorsement of the project. Please review each of the resources referenced at https://github.com/ION28/BLUESPAWN/ for more information.

1 Introduction

Advanced Persistent Threats (APTs), Criminal Organizations, and other threat actors have been attacking Microsoft Windows systems ever since the operating system was first released. Alongside these developments, cybersecurity companies have researched and implemented increasingly sophisticated defenses. At first, anti-virus (AV) companies primarily used basic signatures, like hashes, to detect the malware. As times progressed though, attacks evolved. Defenses also advanced to include performing static and dynamic analysis, analyzing file contents, and more. In addition, secure coding principles and other security protections have begun to be built directly into the OS. For example, in Windows 8.1/10, Microsoft has implemented many new improvements such as Protected Process Light (PPL) and Virtualization Based Security (VBS) [2, 3]. Most recently, attackers have shifted towards abusing built-in features and programs to accomplish their objectives. These methods include leveraging techniques such as Process Injection, PowerShell, Run Keys, .NET binaries, LSASS Memory Dumping, Accessibility Features, Living Off the Land Binaries (LOLBAS), and Configuration/Permission weaknesses [4].

As these attacks have increased in complexity though, the tools used to defend systems have grown more elaborate. The AV market has transitioned into developing so-called Endpoint Detection & Response (EDR) and Endpoint Protection Platform (EPP) products. Some notable examples of these commercial offerings include Carbon Black EDR, Crowdstrike Falcon, CylancePROTECT, and Microsoft Defender Advanced Threat Protection (ATP) [5]. In addition, groups like the MITRE Corporation have released frameworks to codify adversary tradecraft such as MITRE ATT&CK [4]. Furthermore, the Department of Defense (DoD) has long published Security Technical Implementation Guides, otherwise known as STIGs, which detail security-oriented mitigations that can be applied to systems to enhance security [6]. Finally, alongside these innovations, the cybersecurity community has developed many other major advancements including Security and Information Event Management (SIEM) and Security Orchestration, Automation, and Response (SOAR) technologies over the past few decades.

Given today's complex environment, it is more important than ever for defenders to understand the various tactics, techniques, and procedures (TTPs) adversaries are employing. Moreover, it is also crucial to know how the tools blue teams rely on function. By better understanding these elements, defenders will be better equipped to deal with new threats, maintain sufficient defense-in-depth coverage, balance risk in their environments, and generally, move faster.

In keeping with this outlook, our team has developed BLUESPAWN, a fully open-source, active defense and EDR tool for Windows. While there are ample offensive oriented tools publicly available, there is very little on the defensive side. We aim to use this project to demonstrate how modern-day security solutions work by building our own from the ground up. In addition to being a learning tool for both students and practitioners, BLUESPAWN is designed more to be used in an "active breach" scenario by security professionals. Our idea is that anyone should be able to quickly detect, evaluate, and remediate malicious activity on a live system [1]. Finally, we show below how our software is already able to accurately identify and react to real-world malware through a case study in its use at the Collegiate Cyber Defense Competition (CCDC) and lab testing.

2 Existing Technology & Available Defensive Tooling

As mentioned above, hundreds of offensive and defensive tools exist today. Some of these are highly advanced commercial programs, while others were built by the information security (infosec) community and are generally less fully featured. Over the past decade with the advent of services like GitHub and GitLab, the open-source development community has grown tremendously. In particular, many security practitioners share the cybersecurity tools they create on these platforms. While most are primarily offensive oriented, these releases have driven security research forward and produced a number of capable solutions.

2.1 Commercial Signature-based AVs

Although less popular today, classic anti-virus software was among the first dedicated malware defenses. These solutions excelled at detecting the unchanging, custom malware popular at the time. Since defenses were weak or non-existent, malware authors did not need to be as stealthy as they do today. Furthermore, the lack of strong preventative controls and advanced security protections meant attackers had fairly wide latitude in their operations. On the other hand though, the number of attacks was orders of magnitude smaller than it was today - the industry simply did not exist like it does today.

Given this environment, popular AVs such as AVG, Norton, and McAfee could effectively rely on simple signatures [7]. Once a piece of malware was identified, they could obtain its MD5 or SHA1 hash. Then, by deploying this signature to all program installations through a “definition update,” they would be able to detect and remove malware on any system running the AV. Since the volume of malware was relatively small, this approach was largely successful. While it did not proactively identify new threats well or handle polymorphic malware, it was good enough for what it was up against.

2.2 Commercial EDR/EPP Products

As the threats continued to evolve, so too did the defenses. In order to reflect this changing landscape, Anton Chuvakin of Gartner coined the term "Endpoint Threat Detection & Response" (ETDR) in 2013. A few years later this term was shortened to "Endpoint Detection & Response" (EDR) which has carried through to today [8]. These "next-generation" solutions offer a suite of new protections beyond simple signatures. For example, these solutions can perform real-time analysis and behavior monitoring to detect novel pieces of malware and utilize more granular signatures to alert on something like a suspicious process command line. These defenses also often integrate technology like machine learning to augment and continuously improve their detection capabilities. This dynamic nature has greatly increased costs for attackers who might have their new, custom malware detected by tools in weeks, if not days or hours. Additionally, some companies have also started to employ the phrasing "Endpoint Protection Platform" (EPP). While this term is largely similar to EDR, most so-called EPP products integrate other components such as data-loss prevention (DLP) technology to provide an even more comprehensive security suite [9, 10].

Some of the most popular commercial offerings in the current market include Carbon Black EDR, Crowdstrike Falcon, CylancePROTECT, and Microsoft Defender ATP [5]. While an evaluation of each of these products is out of the scope of this paper, we'll touch on some of their key abilities - detection, hunting, and response.

First, they have the capacity to detect malware based on a variety of data sources. For example, according to a publication from Kaspersky, they combine several detection engines including standard signatures, threat intelligence, reputation scores, sandboxing, and YARA rules. In addition, they typically feed all of these measures into machine learning models to make a final decision [11]. In another publication, Crowdstrike notes their agent utilizes a kernel-mode driver to obtain raw events [12]. Unfortunately though, for the most part, these public resources do not delve into much detail on the specific data acquisition techniques and monitoring methodologies. We can, however, assess that these products integrate closely with the operating system to obtain real-time, high fidelity data. Some examples of this data include monitoring registry keys, event logs, process execution, and other OS API calls. Additionally, many solutions are taking advantage of features such as Microsoft's Antimalware Scan Interface (AMSI) [13].

Next, another major feature of these products is the ability for analysts to perform "threat hunting" across their environment. By building out strong, capable endpoint clients, vendors enable security analysts to search for malware. As an example, one could conduct a search for a specific Indicator of Attack (IOA) like a file or registry key across their systems to identify a potential compromise. Finally, if preemptive protections fail, these solutions can take action in response. Some options include killing processes, deleting files, and modifying registry key contents.

One other note is that, for the most part, these systems are very much "black-boxes." Vendors limit distribution of their EDR solutions to paying customers - they cannot just be downloaded from the internet. Next, they are obviously also closed-source which raises the research barrier. Finally, there is also often a tendency to restrict the publication of certain detection methods and for good reason. Since they do not publicize how they detect malware, malicious actors must spend more time figuring how to circumvent these controls.

2.3 Commercial Malware Sandboxes

While we will not focus heavily on malware sandboxes, they provide important insight into a piece of potential malware through dynamic analysis. Solutions like Crowdstrike's Hybrid Analysis, Any.run, and Joe's Sandbox are able to extract a number of potentially interesting execution information [14]. This data might include files created/modified/deleted, registry keys changed, processes spawned, and network connections made. All of this information can help contextualize a sample to enable either automated or manual analysis to make a decision.

2.4 SysInternals

Next, another incredibly popular and free software is the Microsoft Sysinternals Suite. This collection of tools was initially developed by Mark Russinovich in 1996 and mainly designed to "provide advanced system utilities and technical information [15]." These tools have a broad audience including system administrators, developers, and security practitioners. In the below sections we'll examine how blue teamers often utilize select Sysinternals tools to both detect and analyze malware and monitor their systems.

2.4.1 Autoruns

On modern Windows systems, there are hundreds if not thousands of auto-start locations to launch programs and scripts. While most autorun items can be configured in the registry, they can also live in files or other OS locations (like the WMI database). It should be emphasized that autoruns are an important OS feature - the average system has hundreds of active autorun objects. While many are used to launch Microsoft-signed software, third party programs such as a web browser like Chrome will establish "Run keys" to automatically re-open Chrome when a user logins in. They are also often used by programs to check for updates.

Attackers, however, also frequently utilize this built-in feature in keeping with the trend towards abusing legitimate operating system components. One such technique is covered in MITRE ATT&CK T1060 - Registry Run Keys / Startup Folder [4]. As an example of this technique in the wild, the notorious criminal group known as FIN7 was spotted configuring the following registry run key to maintain persistence: `HKCU\Software\Microsoft\CurrentVersion\Run : CtMgk2y9v0_ - explorer.exe PATH\Foxconn.lnk` [17]. As shown in the below figure, Autoruns shows an entry for this, but only if "Hide Windows is entries" is unchecked. Additionally, it does not pick up on the full command line that was present in the Registry key [16].

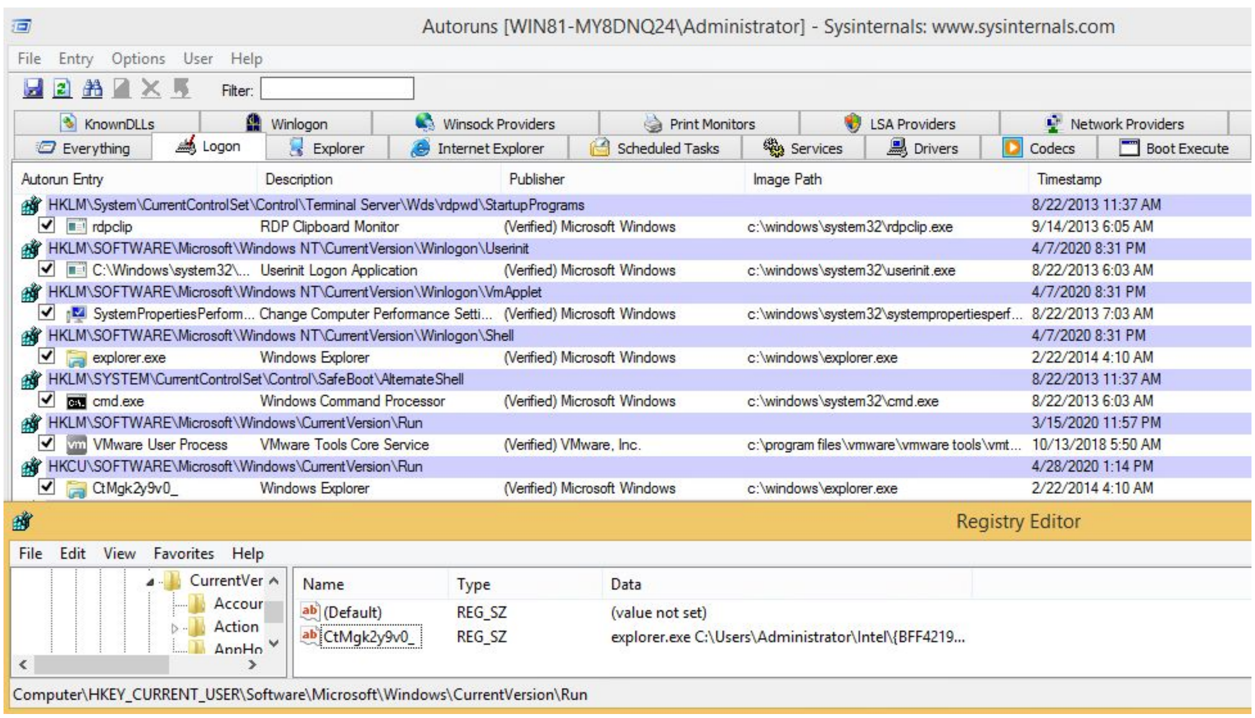


Figure 1: Autoruns and Regedit screenshot showing Run entry for "CtMgk2y9v0_"

As another example shown in Figure 2, Autoruns also has trouble displaying a standard PowerShell encoded staging command that is configured as a run key. That said, while the tool has trouble parsing command lines, it excels at detecting a regular binary and has the broadest coverage of auto-start locations on Windows.

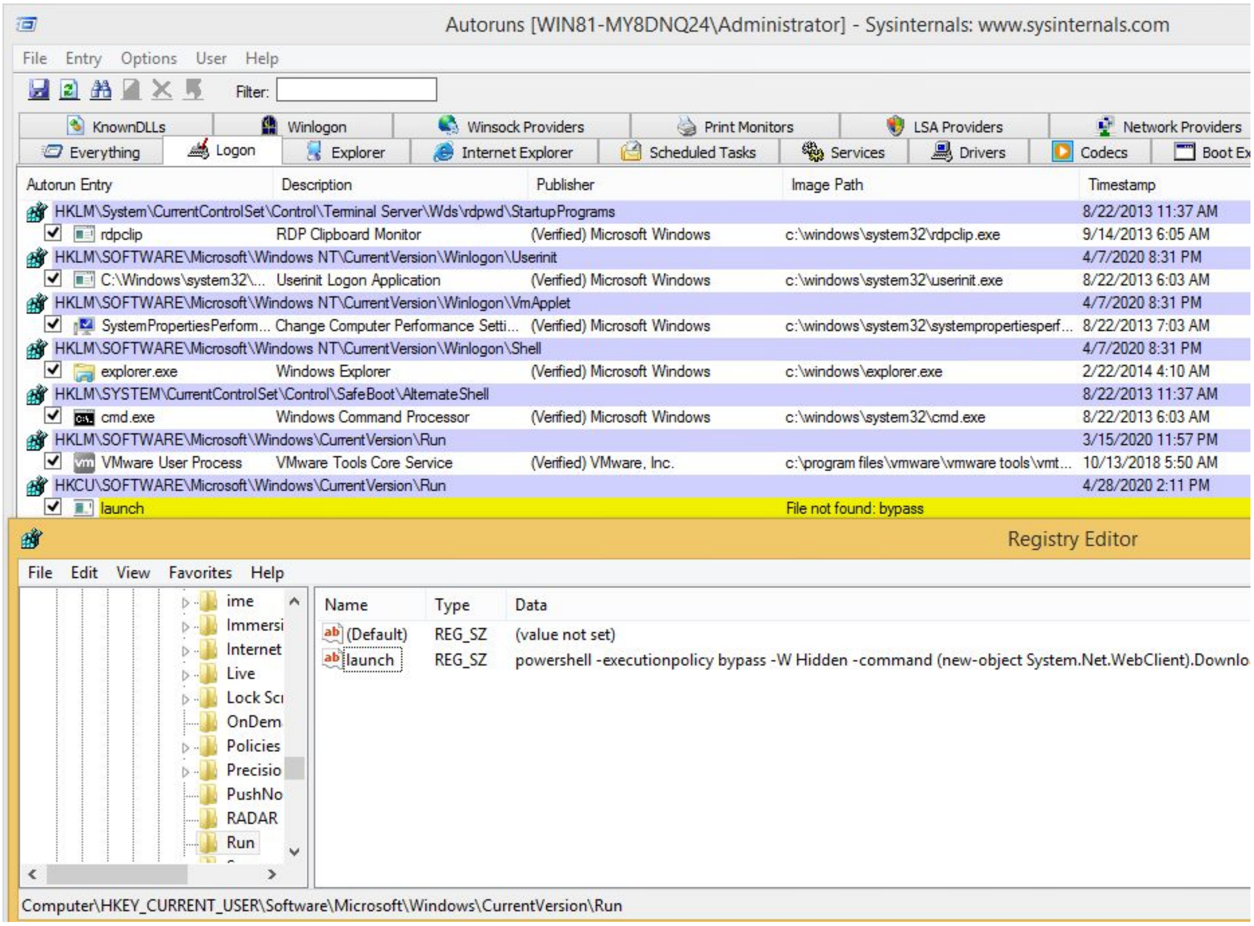


Figure 2: PowerShell run key as shown in Autoruns

2.4.2 ListDlls

Next, security analysts can use Sysinternals' ListDLLs program. This utility is able to list all of the DLLs that a particular process has loaded [18]. From a malware hunting standpoint, we might use this program to identify unsigned DLLs by running a command such as `.\listdlls.exe -u process.exe`. In the below example shown in Figure 3, we demonstrate how ListDLLs flagged items in scvhost.exe, a meterpreter beacon running directly as a standalone exe.

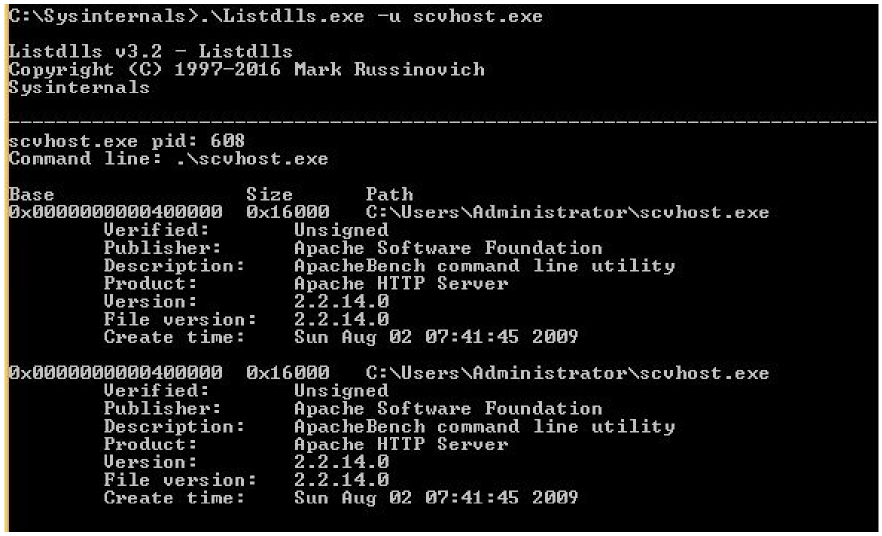


Figure 3: ListDLLs in action identifying unsigned items in a meterpreter beacon

2.4.3 Process Explorer

Another popular utility within this collection that is regularly used by all types of users (sysadmins, developers, security, etc) is Process Explorer. This program has a number of capabilities and mainly focuses on showing all process related information including handles, modules, threads, network connections, and more [19]. From a threat hunting perspective, we can employ Process Explorer to go beyond what we would ordinarily see in Task Manager to identify malware. Some examples of this would be identifying abnormal processes, looking for suspicious command lines, execution of unsigned binaries, and unusual network activity by certain processes. In Figure 4, a meterpreter session has migrated into explorer.exe which is generating network activity linked to this process. On a normal system, explorer.exe should not be making network connections.

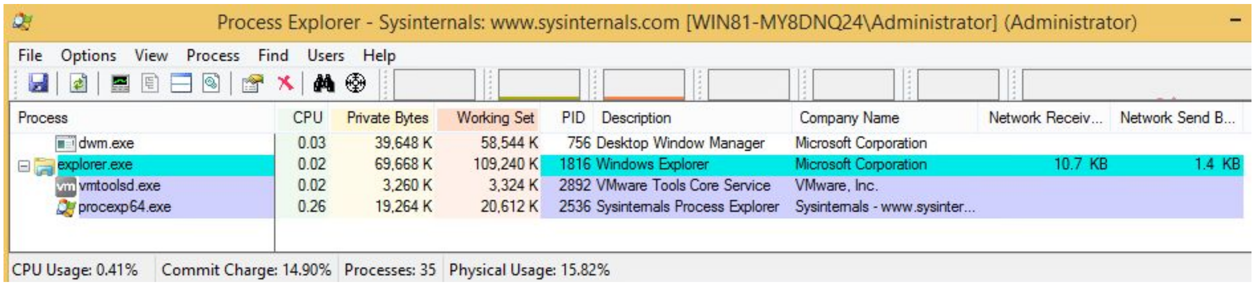


Figure 4: Explorer.exe with network activity is highly unusual and in this case, the result of a Meterpreter beacon injected into it

2.4.4 Sysmon

One of the other useful Sysinternals tools is System Monitor (Sysmon). This program provides enhanced system monitoring capabilities that go beyond the standard Windows event logs. Additionally, Sysmon is highly configurable which enables administrators to tailor logging to target specific types of activity. When combined with technologies such as Windows Event Collection or SIEM agents, teams can centrally collect these advanced logs for further analysis [20]. These abilities also make Sysmon logs an excellent source for threat hunting and forensic activity. As demonstrated below in Figure 5, the Sysmon process create function successfully recorded the full process command line for our PowerShell staging command (the same one as configured above as an autorun). From this command line, we can learn that the attacker attempted to download a binary from a remote server and execute it.

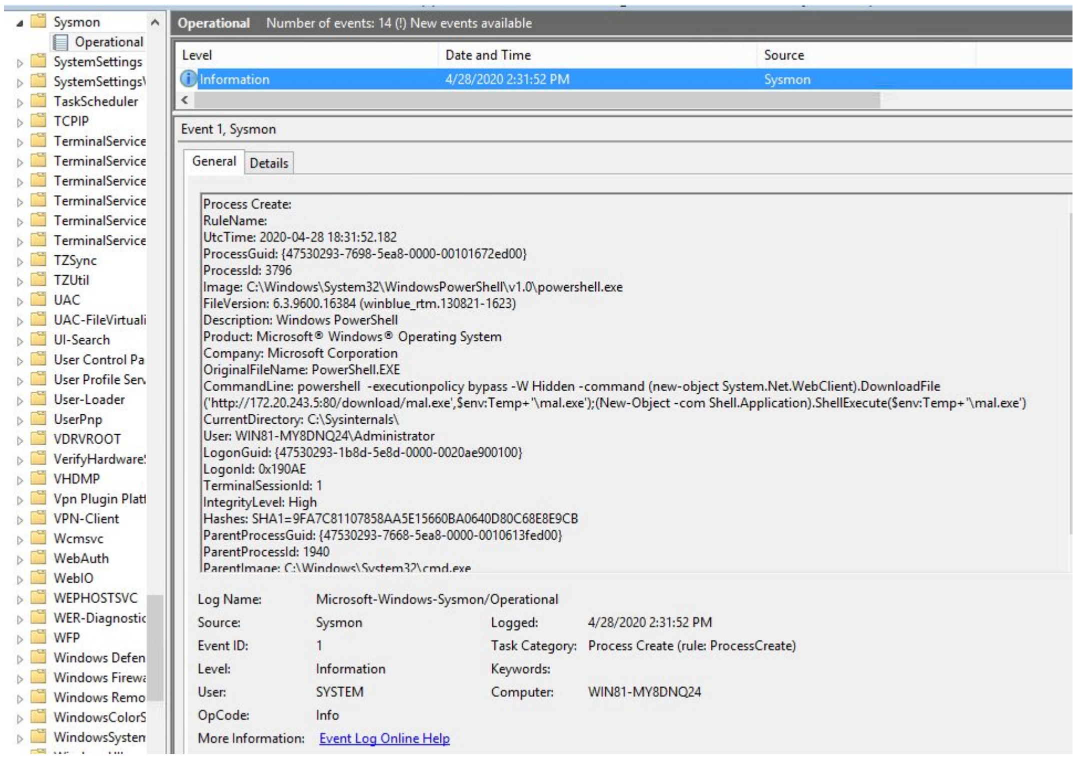
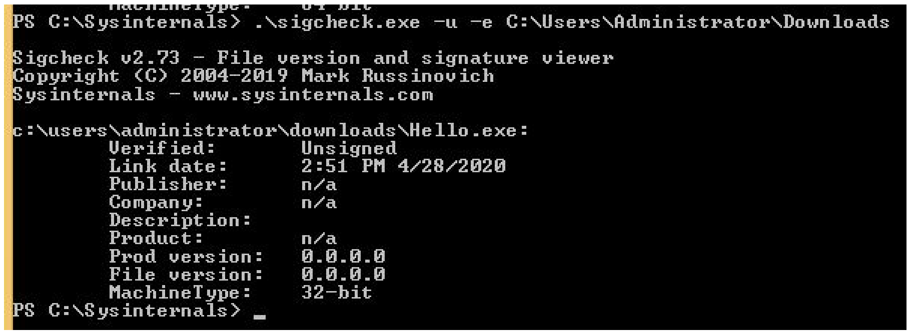


Figure 5: PowerShell execution with full command line captured by Sysmon Event 1

2.4.5 Sigcheck

Next, Sigcheck is another useful tool for security analysts and threat hunters [21]. Most operating systems have tightly integrated certificates and trust into the system. As a result, the vast majority of default binaries and libraries on the system will be signed and trusted with a chain leading back to a trusted root Certificate Authority (CA). Additionally, third party programs can be signed in a similar fashion as a form of proof the code originated with a reputable individual or company. While there are mechanisms that attackers can use to blend in to evade signature checks (for example leveraging MITRE ATT&CK T1130 - Install Root Certificate and ATT&CK T1116 - Code Signing), most standard malware will not be signed (or signed improperly) [4]. This fact enables defenders to alert on suspicious software and investigate it. Microsoft's Sigcheck provides the ability to examine a file's certificate. Additionally, it can examine the system certificate store and audit it against the Trusted Microsoft Root Certificate

List [21]. In Figure 6, we show Sigcheck identifying an unsigned binary in the user's download folder. While its lack of signature is not necessarily indicative of malicious activity, the detection can be used as the basis for further analysis.

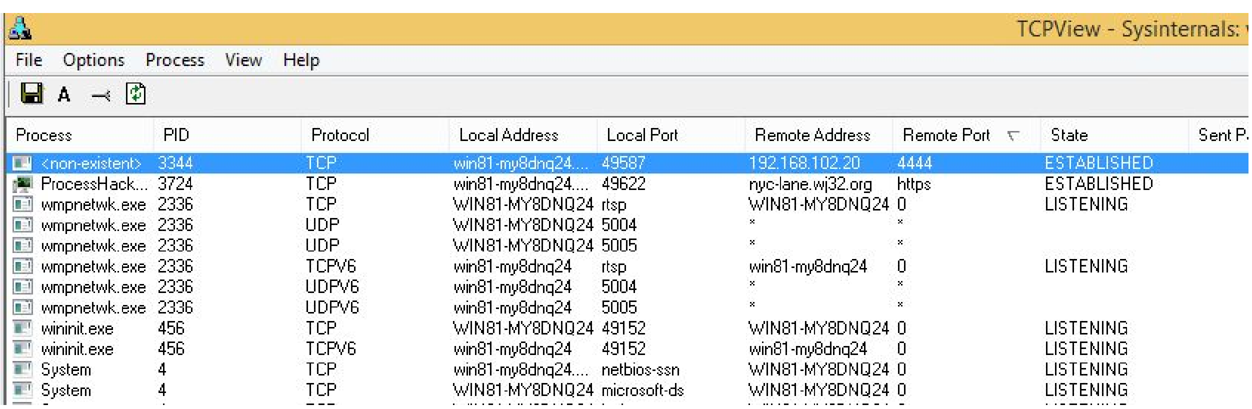


```
PS C:\Sysinternals> .\sigcheck.exe -u -e C:\Users\Administrator\Downloads

Sigcheck v2.73 - File version and signature viewer
Copyright (C) 2004-2019 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\users\administrator\downloads\Hello.exe:
        Verified:       Unsigned
        Link date:      2:51 PM 4/28/2020
        Publisher:      n/a
        Company:        n/a
        Description:
        Product:        n/a
        Prod version:   0.0.0.0
        File version:   0.0.0.0
        MachineType:    32-bit
PS C:\Sysinternals> _
```

Figure 6: Sigcheck identifies an unsigned executable in the Downloads folder.

2.4.6 TCPView

Finally, another particularly useful tool for security analysts (among others) is Sysinternals' TCPView. This program provides a way to see "detailed listings of all TCP and UDP endpoints on [your] system [22]." From a threat hunting perspective, we can use it to identify malware that may be calling back to a command and control (C2) server. While there will likely be a lot of noise on any given system, you can sort by fields such as process name or port. In the highlighted process shown in Figure 7, we see a suspicious connection over port 4444 to another system on the network.



Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State	Sent P.
<non-existent>	3344	TCP	win81-my8dnq24....	49587	192.168.102.20	4444	ESTABLISHED	
ProcessHack...	3724	TCP	win81-my8dnq24....	49622	nyc-lane.wj32.org	https	ESTABLISHED	
wmpnetwk.exe	2336	TCP	WIN81-MY8DNQ24	rtsp	WIN81-MY8DNQ24	0	LISTENING	
wmpnetwk.exe	2336	UDP	WIN81-MY8DNQ24	5004	*	*		
wmpnetwk.exe	2336	UDP	WIN81-MY8DNQ24	5005	*	*		
wmpnetwk.exe	2336	TCPV6	win81-my8dnq24	rtsp	win81-my8dnq24	0	LISTENING	
wmpnetwk.exe	2336	UDPV6	win81-my8dnq24	5004	*	*		
wmpnetwk.exe	2336	UDPV6	win81-my8dnq24	5005	*	*		
wininit.exe	456	TCP	WIN81-MY8DNQ24	49152	WIN81-MY8DNQ24	0	LISTENING	
wininit.exe	456	TCPV6	win81-my8dnq24	49152	win81-my8dnq24	0	LISTENING	
System	4	TCP	win81-my8dnq24....	netbios-ssn	WIN81-MY8DNQ24	0	LISTENING	
System	4	TCP	WIN81-MY8DNQ24	microsoft-ds	WIN81-MY8DNQ24	0	LISTENING	

Figure 7: TCPView identifies a network connection over port 4444

2.5 Process Hacker

Aside from Sysinternals, another essential tool when threat hunting on a live system is Process Hacker. This free and open-source software works similar to Process Explorer, but integrates a number of other features such as file and network monitoring into a single tool [23]. Using this view, a security analyst can quickly get comprehensive insights into both benign and malicious activity on the system. In particular in a malware context, this might include examining the publisher and verification status of an app, correlating network and disk activity to a process, and even analyzing the memory of a particular section or looking at a stack trace in a specific thread. In the following screenshot shown in Figure 8, we show the call stack of a particular thread running in explorer.exe due to a meterpreter beacon having injected into the process.

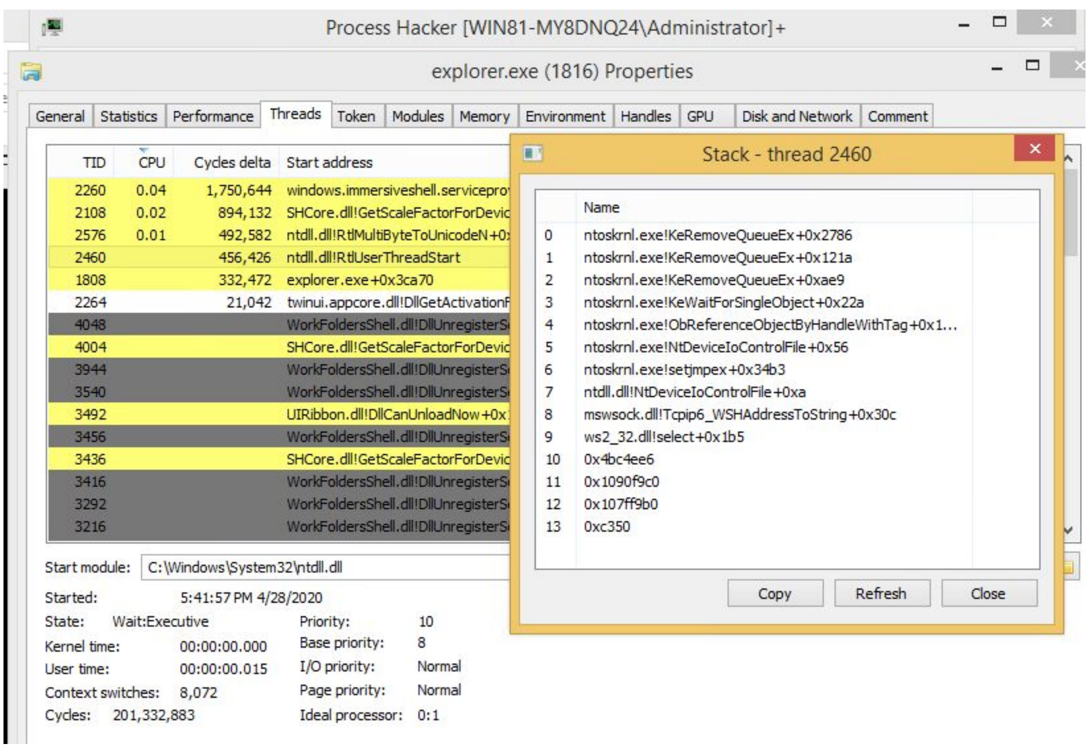


Figure 8: Examining the call stack of a process's thread using Process Hacker

2.6 PESieve and Hollows Hunter

Next, other essential programs in a threat hunter's Windows toolkit are Hasherezade's PE-sieve and Hollows Hunter, both of which are free and open-source. According to PE-sieve's README, the tool is purposely designed to hunt for malware on a system by identifying "a variety of implants including replaced/injected PEs, shellcodes, hooks, and other in-memory patches." These features make it perfect for detecting evidence of MITRE ATT&CK T1055 - Process Injection, T1093 - Process Hollowing, and T1186 - Process Doppelgänging [4]. While PE-sieve is designed to scan a single process, their Hollows Hunter project extends this functionality to be able to scan an entire system [24]. In Figure 9, we show Hollows Hunter successfully detecting a meterpreter beacon which migrated into explorer.exe.

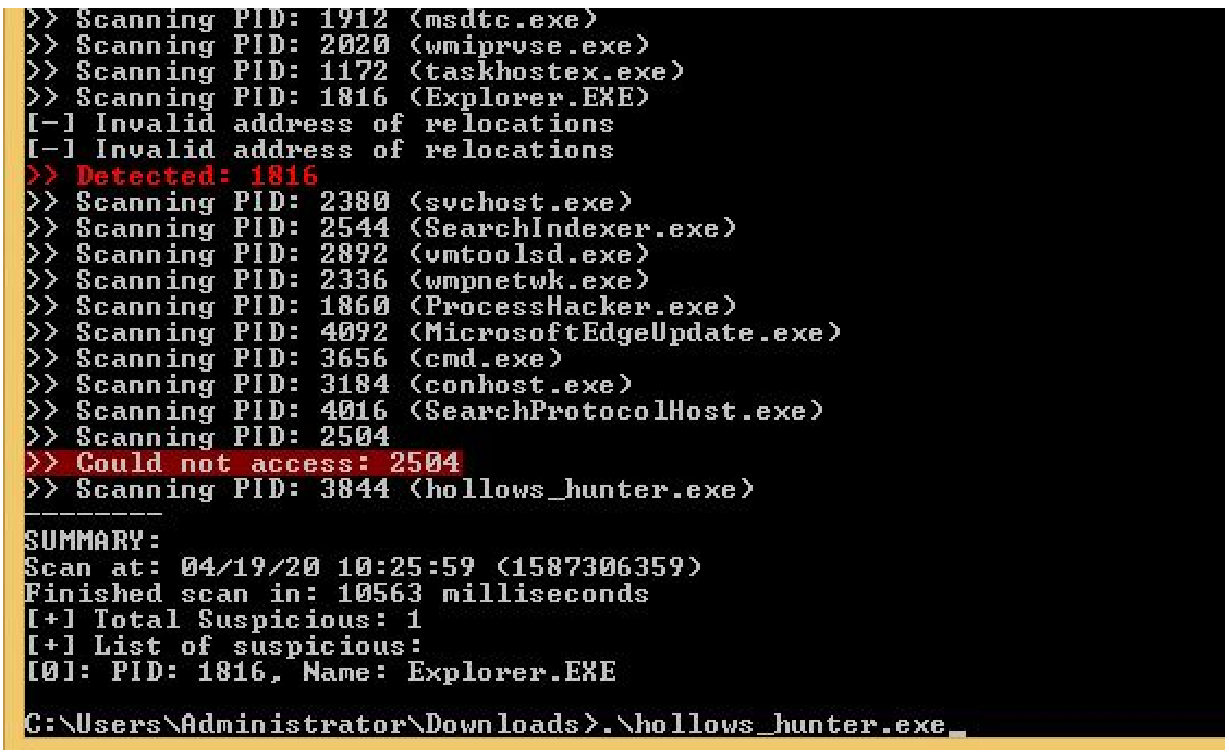


Figure 9: Hollows hunter detecting a meterpreter beacon living in explorer.exe (PID 1816)

2.7 Detections Repositories

In recent years, as the infosec community has continued to openly share lots of offensive tools, defenders have also increasingly shared cyber threat intelligence (CTI) to counter attackers. While much of this sharing is done through non-public channels such as Information Sharing and Analysis Centers (ISACs), many in the community contribute to online malware signature repositories. One such example of this is the Yara-Rules project where researchers can share YARA rules. Another increasingly popular project is Sigma which is a Generic Signature Format for SIEM Systems [25]. With these open centralized repositories, other companies, platforms, tools, and individuals can then integrate these detections into their own workflows. While many detections need to stay classified at TLP:Green, Amber, or Red to maximize their effectiveness, even these public postings can be extremely useful. In particular, they can ensure much of the “low-hanging” malware is effectively caught by defenses [26].

2.8 Security Configuration & Hardening Tools/Scripts

Finally, the last category of tools we’ll touch on in this paper are the many security hardening related scripts available. Some popular examples of these kinds of tools are the National Security Agency’s Windows-Secure-Host-Baseline and the Department of Defense’s STIG Templates and STIGViewer [27, 6]. Another notable set of scripts is Microsoft’s PowerSTIG project which aims to efficiently automate the application of a variety of published Defense Information Systems Agency’s (DISA) STIGs [28]. There are also a plethora of individual-produced hardening scripts online; however, they can be difficult to independently verify without significant work. For example, many scripts set dozens of registry keys with little context as to why or what vulnerability they help to mitigate.

3 Motivations for building BLUESPAWN

With all of this context in mind, our group set out to build another item for a defender's arsenal. Indeed BLUESPAWN is not designed to replace other existing tools, but rather augment existing capabilities. We aim to enable a blue team to move faster and better understand their coverage as well as to promote and encourage the following: analysis of the Windows Attack Surface, availability of blue team software, and methods to work with the Windows API.

3.1 Move Faster

The threats defenders face today are only growing more sophisticated as nation states and criminal actors increasingly target organizations through digital means. Crowdstrike notes that organizations must minimize attacker dwell time in their environments if they hope to prevent the most serious breaches [29]. In order to reduce that dwell time metric, blue teams need highly capable software. They need tools that can positively identify threats and respond in real-time. Furthermore, they need to continue to move away from static signature-based detection towards heuristically, behavior-based approaches. While there are a number of existing commercial or system analysis tools, we focused on helping the security analyst move quicker. In an active compromise scenario, every minute matters. To that end, we envision a world where any security analyst can detect, evaluate, and respond to the majority of possible instances of malware on a system *within minutes.* Additionally, they should also be able to secure the machine and apply proper security protections *in minutes* as well.

3.2 Know our Coverage

At the end of the day, cybersecurity is all about risk modeling. In order to minimize the risk to their environments, security teams continually assess, test, and implement new defenses and detections. One approach they can use to do this is to take a threat centric approach to their efforts. The idea behind this methodology is that if you know the tactics and techniques adversaries will use to target your networks, you can align your defenses properly. To help defenders implement this strategy, the community has worked to develop frameworks like MITRE ATT&CK [4]. And to assist with this shift in mindset, a number of security products have started to tightly integrate ATT&CK, mapping detections to their respective techniques. One gap we noticed in these solutions though, is they often provide little context as to *why* some action was classified the way it was. This problem is especially acute with the “black-box” manner in which most commercial EDRs operate. We wanted to develop something from the ground up that can properly contextualize activity to ATT&CK, and more importantly, we wanted to know exactly what threats we expected our software to catch. If we can have confidence in the status of certain lines of effort (ie, attacker techniques), we can direct our attention to other lines of effort (where we might have less coverage).

3.3 Better Understanding of the Windows Attack Surface

A popular paradigm in the defensive community is that in order to find evil, one must know normal. In order to defend something as complex as a Windows endpoint which has tens of millions of lines of code in the operating system alone, one must spend time learning about its key components. As we of the development team seek to learn about the Windows attack surface ourselves, this project represents the results of that research.

3.4 More Open-Source Blue Team Software

As we have mentioned, there is a plethora of open-source offensive security tooling available on sites like GitHub. Tools like these lower the barrier of entry for students and practitioners alike, making it easy for these groups to learn how they work. Unfortunately, though, there is not as much defensive tooling published this way. While there are many reasons for this, like the effort required to develop and maintain such software, we believe that it raises the bar of entry to blue teaming. In making our program open-source, we hope to inspire other students of the field to learn about defensive tooling and create their own tools. In addition, we hope to use this project to shed light on how EDRs work conceptually.

3.5 Demonstrate Features of Windows API

Finally, developing an EDR program requires close integration with operating system APIs. Because of this fact, we have spent hundreds of hours pouring over Microsoft documentation to learn how to interface with components such as Registry, Event Logs, and Permissions. We hope our code will be useful to other developers who also need to work with core Windows APIs or build similar programs.

4 What is BLUESPAWN

In a nutshell, BLUESPAWN is an active defense and endpoint detection & response (EDR) tool designed to quickly prevent, detect, and eliminate malicious activity on a Windows system. The software has three primary modes of operation: hunt, monitor, and mitigate. Additionally, our project has already worked to integrate many popular industry frameworks and tools such as MITRE ATT&CK, DoD STIGs, and VirusTotal's YARA [1, 4, 6, 30]. These provide the basis for our endpoint defense strategy described in section 5 as well as improve the accessibility of BLUESPAWN and its integration with other community projects.

In its first primary mode of operation, the program **hunts** for evidence of malicious behavior. Generally, this process starts with a known attacker tactic (like Persistence) and technique (like T1060 - Registry Run Keys / Startup Folder). Then, a detection is developed that is designed to identify evidence of *possible* malicious behavior through this particular method. When BLUESPAWN is then run in hunt mode, it queries for the relevant information then makes a determination whether or not a particular item is okay. For example with T1060, the program obtains a list of all Registry values stored in Run Keys (such as HKCU\Software\Microsoft\Windows\CurrentVersion\Run). For this hunt, it also needs to grab a list of files located in User Startup directories (%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup). Armed with these items that are *potentially malicious*, the program then uses a variety of methods to determine whether or not they are likely benign. These checks include whether the file is signed, whether it matches any YARA rules, or even whether it is a certain file type. For any suspicious item found, BLUESPAWN will generate a detection, alerting the user.

Hunt mode also works hand-in-hand with **reactions**. Reactions give the user flexibility in how they want to respond to a particular detection. By default, the reaction is to log it. Other ways an analyst might want to respond are also integrated though. For example, if the program generated a REGISTRY_DETECTION, one could use *remove-value* to delete the identified registry value. In the case of a PROCESS_DETECTION triggered by T1055 - Process Injection, an analyst may instead use *carve-memory*. This reaction would temporarily suspend the target process, modify any malicious threads to immediately return and exit, then resume the process, effectively removing any implant.

The next major mode is **monitor**. While a point in time analysis performed by a **hunt** works well, it is not continuous looking for malicious activity. As a result, monitor mode provides the ability to continually monitor areas of interest. It accomplishes this by, for example, monitoring for any changes to registry keys defined in Hunts. Then, when a change occurs, it will dynamically launch the relevant hunt to see if there is anything new to alert on using the aforedescribed process.

Finally, just as it is important to hunt for malicious activity, it is also important to apply strong defenses. **Mitigate** mode does just that. By mapping mitigations directly to published DoD STIGs and MITRE Mitigations, administrators can either *audit* or *enforce* specific protections [1]. For example, to protect against T1177 - LSASS Driver, one could apply M1025 - Privileged Process Integrity which configures LSASS to run as a Protected Process Light (PPL) and requires drivers loaded into LSASS to be signed. Additionally, an analyst could apply V-3479 to enable DLL Safe Search mode to limit an attacker's ability to use T1038 - DLL Search Order Hijacking to load a malicious DLL into the LSASS process [1, 4].

Along these lines, we emphasize that currently BLUESPAWN only focuses on detecting known threats really well. In the future, as we begin to integrate more advanced behavior monitoring, our tool will provide more robust protection against new threats.

5 Threat Hunting and Mitigation Approach

In the last section, we discussed the three major modes in BLUESPAWN. Here, we will explore more about the defensive methodologies behind the tool and how we approach identifying malicious activity, reacting to malware, and integrating with community tools. Overall, our approach is best described at a high level in Figure 10. Each of these areas work together to raise the overall security posture of a system, and by improving the speed, accuracy, and efficiency at which these processes happen, we can make a meaningful impact on its security.

Figure 10: BLUESPAWN's Defensive Methodology

5.1 Data Sources

The first challenge in countering threats is gathering the right data. A good array of data sources will be diverse, trustworthy, and accurate. If any one of these characteristics is missing, one's ability to effectively detect threats will be hampered in some way. With that in mind, BLUESPAWN attempts to integrate with the Windows API as closely as possible. As such, it is unacceptable to have to open a sub-process (ie spawning cmd.exe) to obtain, say, a list of users on the system. Instead, the program should use the relevant APIs to query the users on the system. This way, in order to really interfere with an operation, a program would need to hook the APIs called by BLUESPAWN in some way. Given that our software currently runs solely in user mode and does not contain a kernel driver, this is certainly possible. That said, at the end of the day, our software can not stop all threats. However, it can and does aim to raise the bar for attackers and make it more difficult to breach and persist on a system.

In the current iteration, the program has modules to interface with a number of key system components including Registry, Windows Event Logs, Files, Processes, and Permissions. In the future, this will expand to have support for COM objects (like Scheduled Tasks), User Accounts, the Windows Management Interface (WMI), the Antimalware Scan Interface (AMSI), and more [31]. As evident in the MITRE ATT&CK matrices, threats can be detected through a variety of means [4]. By having a diverse set of data sources, we increase the chances at effectively detecting a threat. For example, T1050 - New Service or T1035 - Service Execution will leave traces across all of these areas [4]. There will be event logs, entries in the Registry, files on disk, and a process running. Then, assuming the data we receive back is trustworthy and accurate, BLUESPAWN can properly evaluate a potential threat and make a decision.

5.2 Active Defense & EDR Capabilities

As we covered in the section on existing tooling, there are a number of dual-use system analysis tools available (like Sysinternals) [15]. Additionally, there are several harder to obtain commercial products that provide effective defensive capabilities primarily for the enterprise customer. One particular area we noticed a gap in though was open-source tooling for security analysts. We strongly believe that given an arbitrary system or network of systems, any security analyst should be able hunt for threats and triage any malware found. While a relatively cursory investigation such as this will not catch every threat, it does not necessarily need to. There exists significant value in the ability to *quickly* get a decent idea of the current security posture of your network.

Furthermore, we were also particularly interested in the “active breach” scenario. Assuming you now know that your network has been compromised, how can you quickly identify the vast majority of an attacker’s persistence and begin to kick them out of your environment? We refer to this concept as “active defense,” and it is one of our primary areas of focus when building out BLUESPAWN capabilities. While there will always be more experts you can call for assistance, we think it is important to equip the average network defender with the abilities to give them a fighting chance against even the more advanced attackers. Finally, our software is an EDR. This characteristic provides the capabilities to further our active defense mission and provide more general, ongoing protection against threats.

5.3 Integration with MITRE ATT&CK

Throughout this paper, we have made a number of references to the MITRE ATT&CK project and threat-centric defense [4]. One of the most important innovations beyond itemizing the most popular attacker techniques though, is ATT&CK’s success in creating a common language. In order for red and blue teams to collaborate more closely and provide the best protection for their organization, they must work together. ATT&CK has provided a strong foundation for that dialogue and cooperation to take place. By integrating heavily with this framework, BLUESPAWN builds on this work and is easily accessible to many in the industry. Furthermore, by centering our hunts around ATT&CK, as that project continues to grow, our software can grow with it. The cybersecurity landscape is incredibly dynamic; as the threats evolve, so too will the community's methods for countering them.

5.4 Integration with Department of Defense STIGs

Security compliance and testing are two other crucial areas of cybersecurity. To that end, BLUESPAWN also integrates with the excellent resources published by the DoD. Through their DoD Cyber Exchange, they regularly publish security baselines for a variety of products including Windows [6]. These baselines are the result of significant time and effort. If applied correctly, STIGs provide a solid starting point for securing a system, and by automating some of the audit and enforcement work, administrators can better protect their own systems. While STIGs are occasionally referenced in other program modes, they are primarily featured in **mitigate** mode. Our initial efforts have focused on auditing for the most critical settings (rated as High severity by the DoD), but in the future, we plan to add even broader coverage.

5.5 Integration with YARA

Finally, as of release version v0.4.3-alpha on which this paper is based, our other major integration is VirusTotal's YARA. As described by their website, YARA is a "pattern matching swiss knife for malware researchers" which helps "identify and classify malware samples [30]." In addition, the infosec community has widely embraced YARA, and there are a number of FOSS repositories of YARA signatures. Our tool integrates both YARA and a number of these detections repositories. Then, whenever a file of interest is identified by BLUESPAWN, the software scans it with the included rules as one of its checks. If a match to a "malicious" rule is found, then a FILE_DETECTION event will be triggered. By building off the existing corpus of

YARA rules, we will be able to easily integrate new signatures. Furthermore, this approach enables others to utilize their existing private rulesets with the tool for custom scanning.

6 Software Architecture

While this paper primarily focuses on the applications of BLUESPAWN as opposed to how the software works under the hood, we will touch on its architecture at a high level in a few key areas.

6.1 Architecture Diagrams of Key Components

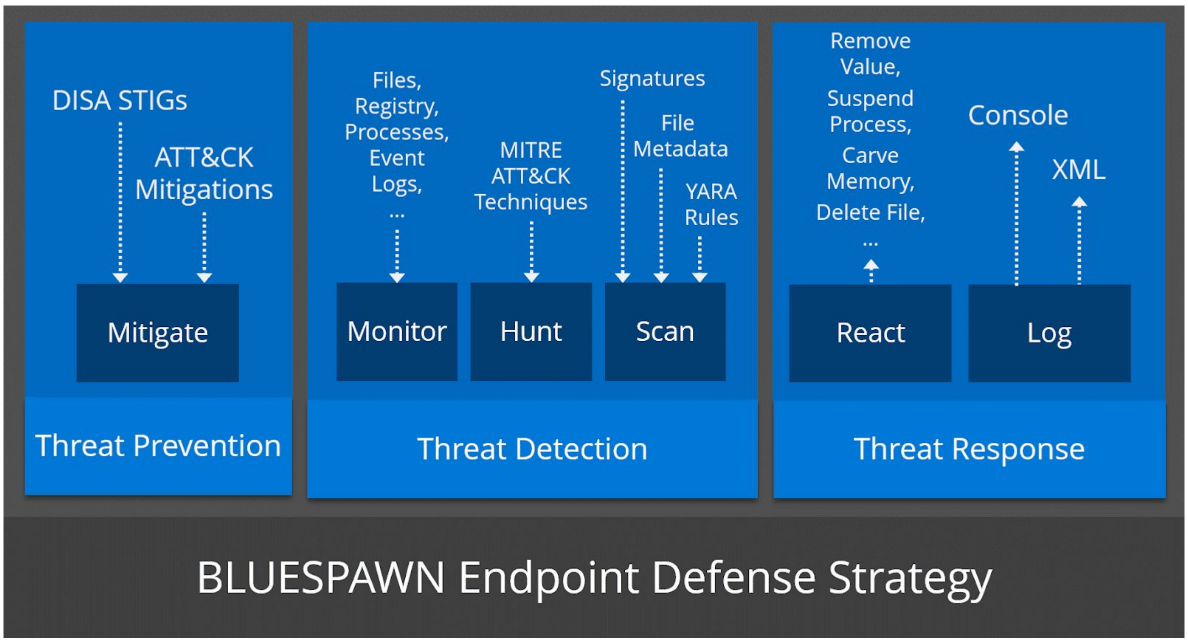


Figure 11: Major BLUESPAWN modules within overall defense strategy

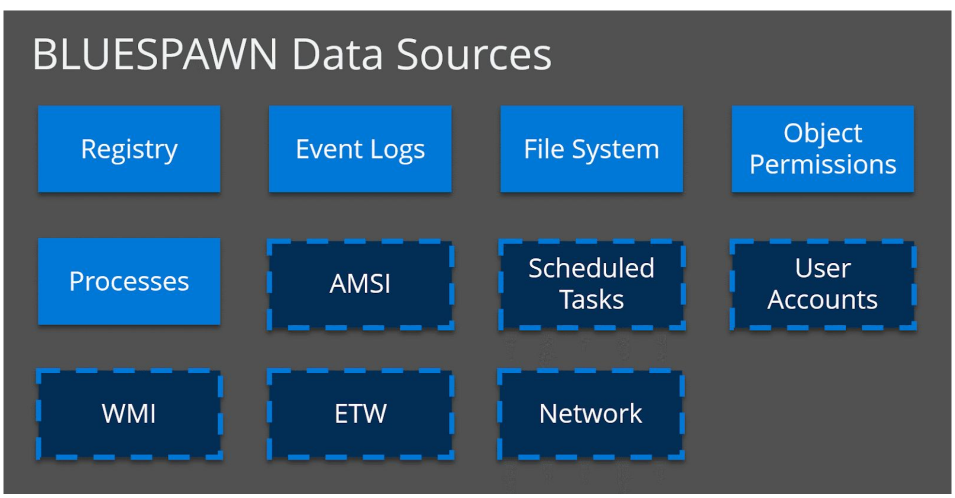


Figure 12: Current and select future BLUESPAWN data sources

BLUESPAWN Overall

Current Coverage of BLUESPAWN

filters

stages: act
platforms: windows

legend

Strong BLUESPAWN Coverage
Partial BLUESPAWN Coverage

Initial Access	Execution	Persistence	Privilege Escalatio	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command And Control	Exfiltration	Impact
T1189	T1191	T1015	T1134	T1134	T1098	T1087	T1017	T1123	T1043	T1020	T1531
T1190	T1059	T1098	T1015	T1009	T1110	T1010	T1175	T1119	T1092	T1002	T1485
T1133	T1223	T1182	T1182	T1197	T1003	T1217	T1210	T1115	T1090	T1022	T1486
T1200	T1175	T1103	T1103	T1088	T1503	T1482	T1534	T1213	T1094	T1030	T1491
T1091	T1196	T1138	T1138	T1191	T1081	T1083	T1037	T1005	T1024	T1048	T1488
T1193	T1173	T1131	T1088	T1116	T1214	T1046	T1075	T1039	T1132	T1041	T1487
T1192	T1106	T1197	T1038	T1500	T1212	T1135	T1097	T1025	T1001	T1011	T1499
T1194	T1129	T1067	T1068	T1223	T1187	T1040	T1076	T1074	T1172	T1052	T1495
T1195	T1203	T1176	T1181	T1109	T1179	T1201	T1105	T1114	T1483	T1029	T1490
T1199	T1061	T1042	T1044	T1122	T1056	T1120	T1021	T1056	T1008		T1498
T1078	T1118	T1109	T1179	T1090	T1141	T1069	T1091	T1185	T1188		T1496
	T1177	T1122	T1183	T1196	T1208	T1057	T1051	T1113	T1104		T1494
	T1170	T1136	T1050	T1207	T1171	T1012	T1080	T1125	T1026		T1489
	T1086	T1038	T1502	T1140	T1040	T1018	T1072		T1079		T1492
	T1121	T1133	T1034	T1089	T1174	T1063	T1077		T1219		T1529
	T1117	T1044	T1013	T1038	T1145	T1518	T1028		T1105		T1493
	T1085	T1158	T1504	T1073	T1539	T1082			T1071		
	T1053	T1179	T1055	T1480	T1111	T1016			T1032		
	T1064	T1062	T1053	T1211		T1049			T1095		
	T1035	T1183	T1058	T1181		T1033			T1065		
	T1218	T1037	T1178	T1222		T1007			T1102		
	T1216	T1177	T1078	T1107		T1124					
	T1072	T1031	T1100	T1006		T1497					
	T1127	T1128		T1484							
	T1204	T1050		T1158							
	T1047	T1137		T1143							
	T1028	T1034		T1183							
	T1220	T1013		T1054							
		T1504		T1066							
		T1108		T1070							
		T1060		T1202							
		T1053		T1130							
		T1180		T1118							
		T1101		T1036							
		T1505		T1112							
		T1058		T1170							
		T1023		T1126							
		T1198		T1096							
		T1019		T1027							
		T1209		T1502							
		T1078		T1186							
		T1100		T1093							
		T1084		T1055							
		T1004		T1108							
				T1121							
				T1117							
				T1014							
				T1085							
				T1064							
				T1218							
				T1216							
				T1198							
				T1045							
				T1221							
				T1099							
				T1127							
				T1078							
				T1497							
				T1102							
				T1220							

Figure 13: BLUESPAWN's combined Hunt and Mitigation coverage as of v0.4.3-alpha

While development has primarily focused on the client, future additions of the Server and Cloud components will replicate the "cloud-delivered protection, enterprise EDR" model present in commercial products. In the below diagram in Figure 14, we show a reference architecture as to what that may look like when developed.

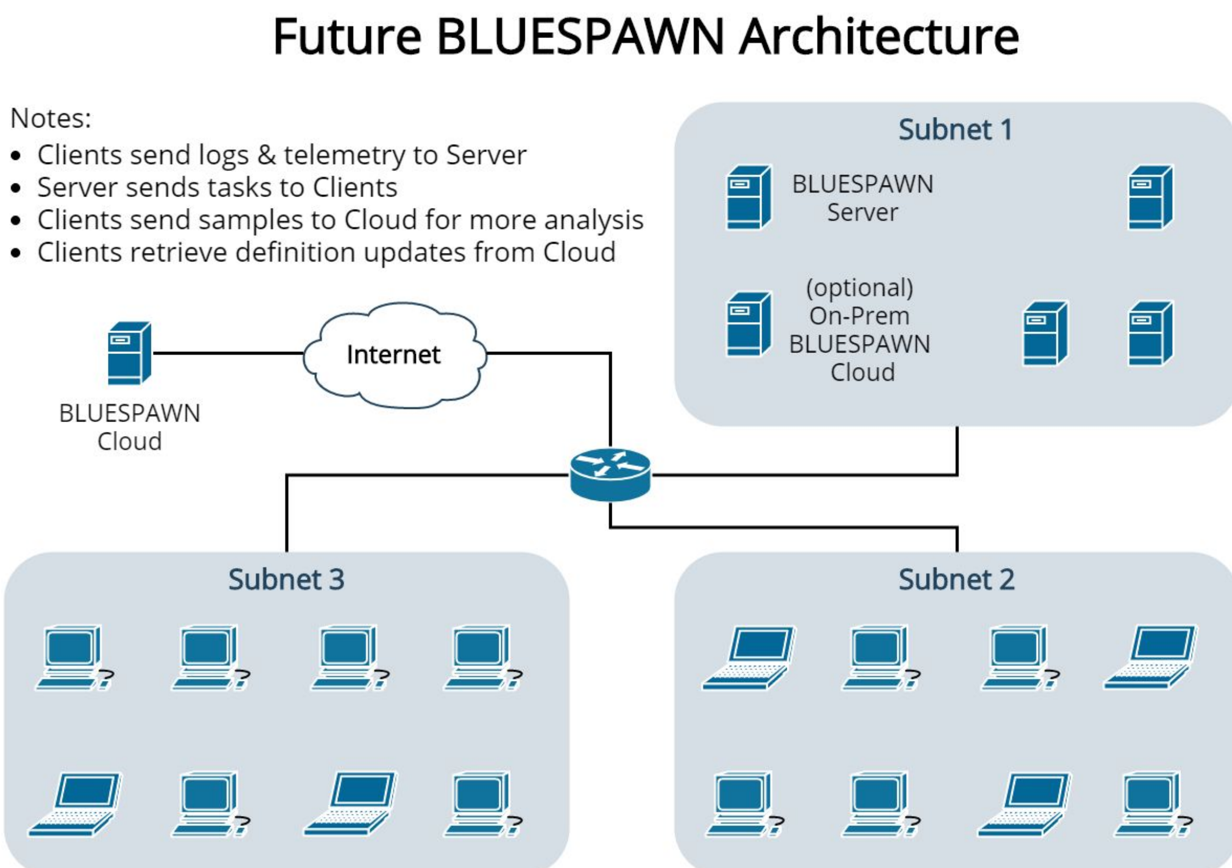


Figure 14: Example future architecture diagram which illustrates how BLUESPAWN might scale across a network

6.2 Continuous Integration & Testing

One of the unique challenges that we ran into when developing BLUESPAWN that we will cover in this paper is testing. First, this project has grown to be one of the largest codebases that any of us have significantly contributed to. Given the scope and complexity of the software, we have turned to continuous integration and testing tools. Our team has, for example, utilized GitHub Actions to automatically test new builds of the tool [1]. All of this automation is orchestrated into our git workflow which has helped to, at a minimum, ensure build consistency across our team. We have also employed collaboration tools within Visual Studio to debug issues effectively.

6.2.1 Automated and Manual Testing of an EDR

Perhaps one of the biggest challenges outside of designing the software and writing detections was/is functional testing. While the CI provides somewhat adequate “smoke testing,” it alone does not ensure hunts work as expected. By making use of Red Canary's Atomic Red Team project, we have begun to address this challenge [32]. Currently, anytime a build happens, the associated Atomic Red Team tests for supported ATT&CK Techniques are run. While these tests have been effective at catching some bugs, they do not compare to a real attack scenario. In order to mitigate this gap, we have so far performed an array of ad hoc manual testing. This approach has included running practice cyber defense simulations and testing BLUESPAWN against specific tools. For example, to test T1055 - Process Injection, we have made extensive use of Metasploit and Cobalt Strike. By more closely emulating the tools and scenarios of real attacks, we can improve detection accuracy and reduce false positives. Our current research, though, has generally found the available (public) methods to test the effectiveness of security solutions to be somewhat lacking.

7 BLUESPAWN in Action

Throughout BLUESPAWN's development, we have deployed it to a number of environments to evaluate its effectiveness. These tests include cyber defense competitions with active Red Teams to manual laboratory testing against tools such as Cobalt Strike. We found that while BLUESPAWN still generates a number of false positives and lacks the ability to identify new or more advanced threats, it shines in quickly identifying most of the low hanging fruit. Within about two minutes of downloading the tool, we were able to detect and respond to common attacker techniques like Process Injection and Autorun methods. When comparing these results with the aforementioned tools, BLUESPAWN was the only free tool that transparently identified these common threats and successfully mapped them back to MITRE ATT&CK as part of its response.

7.1 Case Study: The Collegiate Cyber Defense Competition (CCDC)

Since its creation in 2005, the Collegiate Cyber Defense Competition (CCDC) has grown to be the largest cyber defense event for students in the US. Run by the Center for Infrastructure Assurance and Security (CIAS) at the University of Texas at San Antonio (UTSA), the competition has grown to include more than 235 colleges each year [33]. In the competition, students inherit a mock business environment and defend it against real-world offensive security professionals. The event pushes teams to respond to attacks by some of the best hackers. In addition, they must keep critical services online and complete business injects despite the onslaught of attacks [34].

In the 2020 season of CCDC, the UVA Cyber Defense Team deployed BLUESPAWN at the Mid-Atlantic Qualifiers and Regionals. Our team also used the program at similar cyber defense competitions during the last few months including RIT's Information Security Talent Search

(ISTS) 2020 and University at Buffalo's Lockdown v8. The rest of this section will provide perspectives from both the Blue and Red Teams. CCDC events strictly prevent taking material out of the competition environment, so all screenshots below are from the Lockdown v8 event.

7.1.1 Blue Team Perspective and Evaluation

These cyber defense competitions are chaotic environments, and speed is of essence. At the National event, Red Team regularly breaches systems less than one minute after it starts. As a result, Blue Teams must move fast. Since they cannot expect to completely prevent Red Team from getting into their systems, they must utilize their incident response skills to detect and remove all traces of the attackers. Through our conversations with Red Team, they report generally deploying malware at three different levels. At one end of the spectrum is the malware that is fairly obvious and does not try to hide. On the other side, they deploy heavily custom malware that has never been deployed against any other target before.

In past years, our team has utilized nearly all of the tools referenced in Section 2. While we successfully have used them in combination with other software such as firewalls, we found three things. First, these tools required extensive manual effort and significant background knowledge to operate. An analyst would have to understand enough about normal system behavior to efficiently identify a malicious process, for example. Second, as with all security tools, they miss things. If the Red Team knows what tools defenders will utilize, they will spend time identifying bypasses. Finally, most of these programs are not designed with incident reporting in mind. While some can be configured to log information, it takes a lot of effort to extract all of the useful information and make sense of it - time that defenders do not have in an active breach scenario.

We cannot stress enough that BLUESPAWN was NOT found to be any sort of magic bullet for defenders. Instead, we found that the software improved our team's success with each of those three areas. First, it helped us to rapidly triage our systems. After running the program, we could be reasonably confident in a number of "lines of effort." This confidence enabled us to spend more time hunting for bad in the rest of our boxes. Furthermore, we found that the tool synchronized our response better, especially with our less experienced teammates. Typically we deploy our most experienced threat hunter to identify all traces of malware on a particular box. In a small environment, this approach works well, but fails as the network grows. BLUESPAWN helped reduce the number of cases we had to elevate to our senior threat hunter, giving everyone more time to complete their other tasks.

Next, the program expanded our coverage against threats. In the below figure, we show BLUESPAWN alerting on a malicious Windows Notification Package. This kind of malware is an example of something that may have flown under the radar before. Instead, the implant was detected and removed just minutes into the Lockdown v8 competition.


```
</hunt>
<hunt agressiveness="Cursory" categories="1" datasources="16" tactics="4" time="-664158436">
    <name>T1131 - Authentication Package</name>
    <detection type="Registry">
        <key>HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Lsa</key>
        <value>Notification Packages</value>
        <data>["adsec", "rassfm", "scecli""]</data>
    </detection>
    <detection type="File">
        <name>adsec.dll</name>
        <path>C:\Windows\System32\adsec.dll</path>
        <hash></hash>
    </detection>
</hunt>
```


Figure 15: BLUESPAWN identifies a malicious Notification Package and automatically maps it to MITRE ATT&CK T1131.

Another example of BLUESPAWN quickly detecting and removing malware is shown in the following two figures. Here the tool enumerated all Windows Services and was able to raise several suspicious ones to the analyst’s attention for further investigation. It also enabled the defender to remove the malware immediately without even leaving the current window.

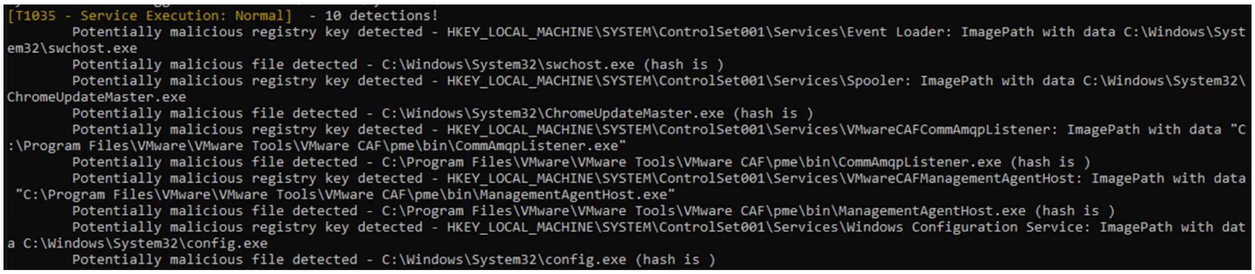


Figure 16: BLUESPAWN identifies a number of malicious Windows Services designed to blend in.

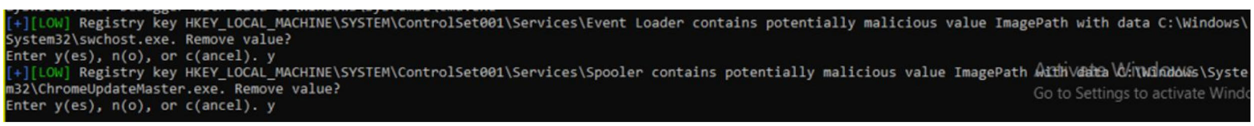


Figure 17: BLUESPAWN offers to remove the suspicious services it has identified.

While our team certainly continued to use other tools to spot and remove instances of the Red Team, BLUESPAWN was instrumental in speeding up that process from our perspective. Even though it missed plenty of later identified malware, it acted as a great starting point for incident response efforts.

Finally, the software greatly improved the speed and accuracy of incident reporting. By being able to efficiently document and map most of the identified threats to attacker techniques, we could easily create high quality reports. We were also able to assess with high confidence when various attacks happened. Even though attackers can use T1099 - Timestomp to hinder analysis, BLUESPAWN could at least say exactly when an attack was identified and remediated. When there are so many other things happening all at once, these logs proved useful for collecting the evidence we needed to make a strong case [35].

7.1.2 Red Team Perspective and Evaluation

For this section, we interviewed one of the Mid-Atlantic CCDC Red Teamers who discussed his thoughts on BLUESPAWN. One thing he noted at the outset is that advanced attackers seek to emulate their target environments as much as possible, to include security tools. As a result, one of the first steps in testing their attacks is analyzing what a defender might see on their console. This analysis also helps to determine when to utilize custom implants versus when they can operate fine mainly using open-source tools. With respect to BLUESPAWN specifically, they knew about the tool before seeing it in competition. This advance notice helped them to evaluate their techniques against it. Furthermore, since the software was open-source, they had the ability to understand exactly what aspects of their malware the tool alerted on. Overall, while it did not change their preparation process in a significant way, they noted that it was able to identify certain parts of their standard toolkit.

One area that BLUESPAWN differs significantly from other EDR programs is that it is fully open-source. In that regard, it allowed the attackers to tailor their malware to ensure it bypassed detections built into the tool. While he noted that it missed a fair amount of activity, he said that it was good at detecting known malware. In addition, he explained that the program was a great way to learn how a commercial EDR solution would work conceptually. Another thing mentioned during the conversation was the need to be transparent in how & why something was flagged as malware. He highlighted that oftentimes, for example in popular malware sandboxes or some paid tools, that they do not do enough to publish how they categorized something the way they did. While the reporting feature is still in its early stages, he saw a lot of value that BLUESPAWN could provide in showing the context around detections.

Overall, he said that though he did not notice an improvement in response time compared with other EDRs, “BLUESPAWN has a lot packed into it.” He also described some challenges for the project going forward that we would have to address. First, he encouraged us to work on building an automated definition update process. This feature would enable better protection against new threats, not requiring people to redownload the latest version each time. He also noted a lot of opportunities to continue expanding the project to target other operating systems (like Linux) and add enterprise-type features. These improvements would make BLUESPAWN a more viable solution to be utilized in real-world environments. In particular, the software could be further designed to work alongside existing solutions and provide a great way for under-resourced organizations to perform a baseline assessment against their environments. Finally, he summarized that when compared to other similar, longstanding open-source projects like ClamAV, BLUESPAWN was perhaps already a bit ahead of them [36].

7.2 Controlled Environment Testing

When developing BLUESPAWN, our team performed regular analysis against many of the most popular tools used by attackers. This testing enabled us to get a feel for how well it might do in real-world situations. In the coming sections, we will describe some of the results of these efforts.

7.2.1 Hunting for Process Injection (Cobalt Strike)

Our first set of tests to detect T1055 - Process Injection were with a licensed copy of Cobalt Strike, a program made by Strategic Cyber LLC. This tool is frequently used in penetration testing engagements, adversary emulation exercises, and real-world attacks [37].

For this setup, a Windows 10 machine patched through KB4520010 (2019-10) acted as the victim. Additionally, Windows Defender was disabled. In Figure 18, we start with three beacons running checking in at 5 second intervals. The initial payload was launched via a Stageless Scripted Web Delivery with PowerShell, calling back to an HTTP Listener. Once the initial beacon checked in, the operator used the built-in "Inject" feature to inject another HTTP Beacon into the Microsoft Edge process running on the host. Finally, an SMB Beacon that calls back through the first HTTP Beacon was injected into Explorer.exe using the same method. All beacons were running in the context of a non-administrative user account.

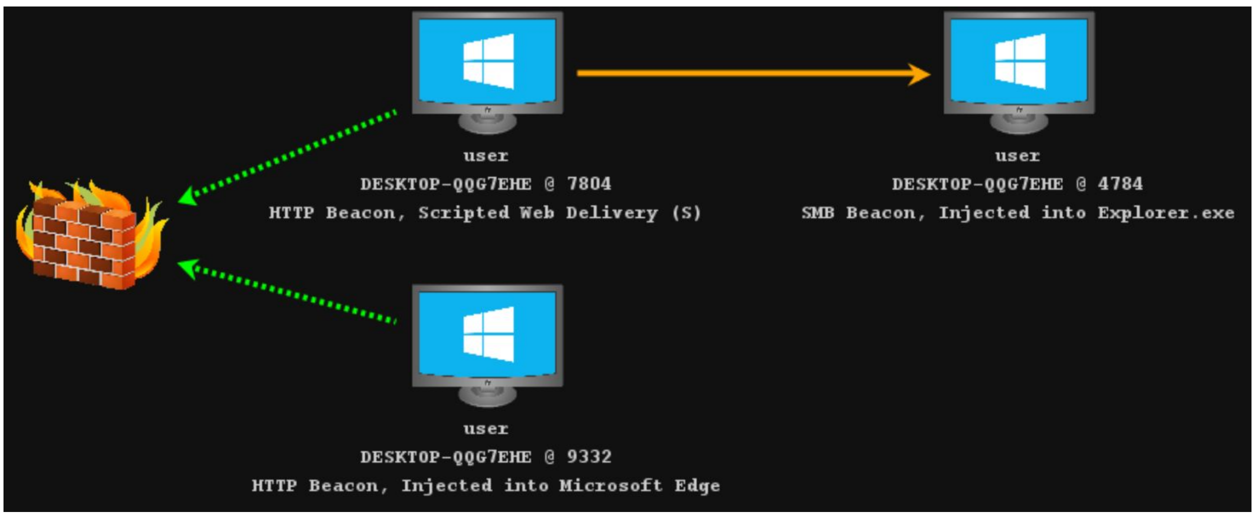


Figure 18: Graph view of Cobalt Strike beacons running on the victim.

In order to identify any evidence of the malicious activity, the analyst launched BLUESPAWN from an Administrative Command Prompt as shown in Figure 19. They did a Hunt specifically for T1055 - Process Injection. In this screenshot, one can see the tool identifying several of the beacons. In addition, the software prompted the user to terminate the specific malicious activity within each of the detected processes.

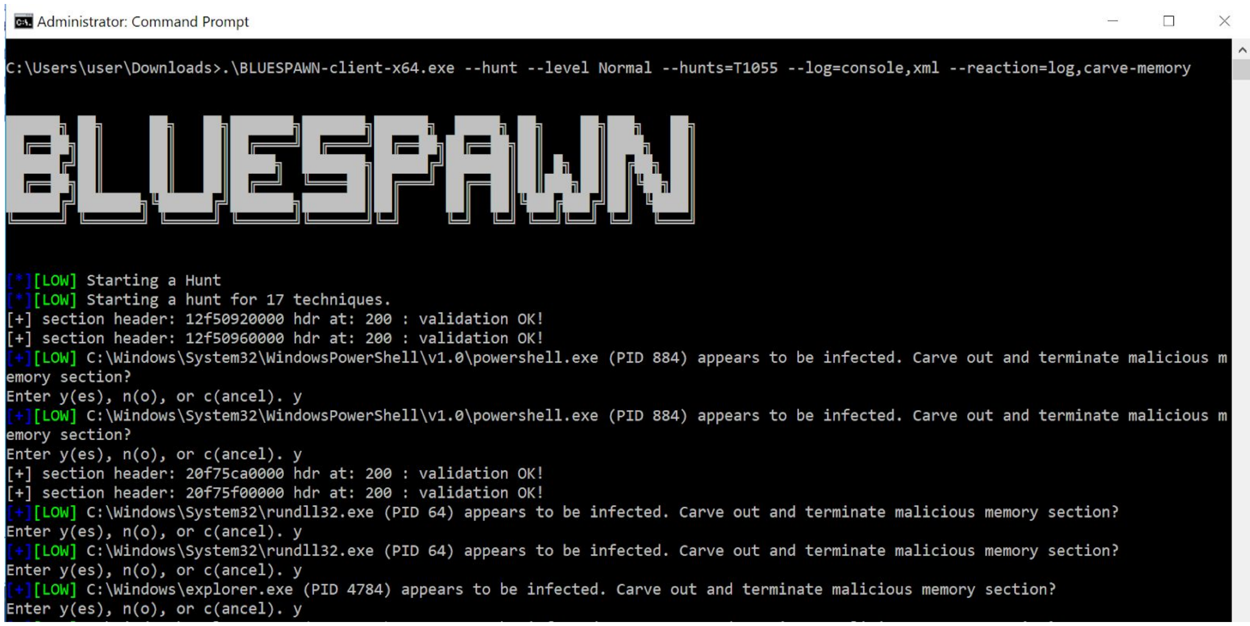


Figure 19: BLUESPAWN performs a Hunt for evidence of Process Injection

In the screenshot shown below in Figure 20, BLUESPAWN summarized all of its findings. It successfully detected each of the beacons the operator launched (the first two groups were part of the initial launch of the beacons). Furthermore, from the operator's Cobalt Strike console, each of these beacons stopped calling out and died.

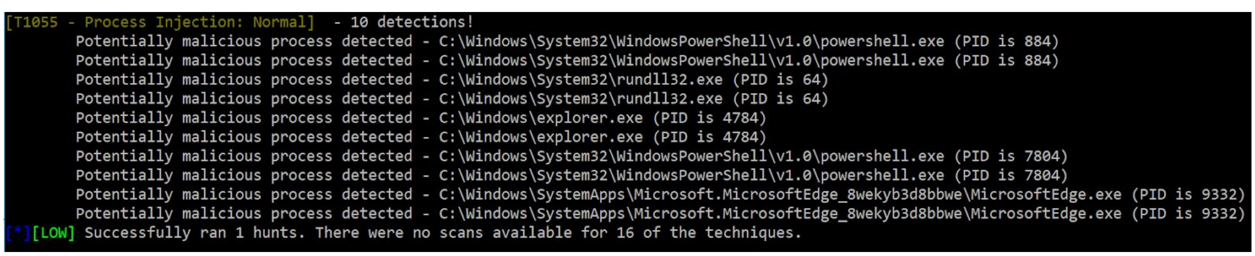


Figure 20: BLUESPAWN identifies each of the Cobalt Strike beacons on the system.

7.2.2 Hunting for Process Injection (Metasploit's Meterpreter)

Next, we tested with Metasploit using the same base environment as above [38]. The operator PSExec'd into the system using valid Administrative credentials. Once the meterpreter session called back, the operator migrated to Microsoft Edge, then to explorer.exe. In Figure 21, BLUESPAWN detected evidence of the malicious activity in both injected processes.

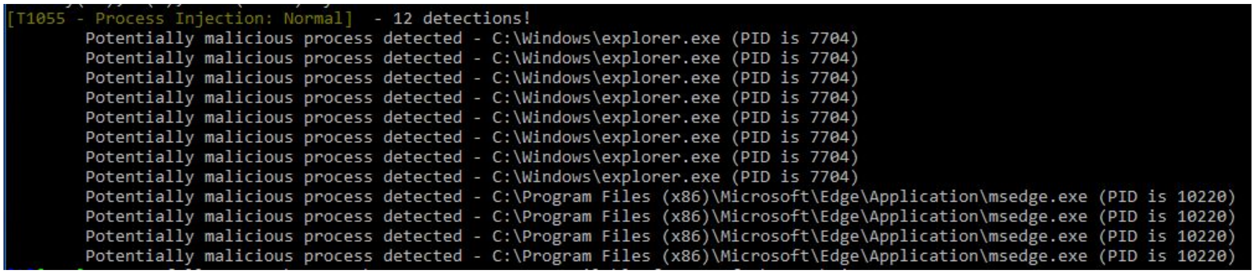


Figure 21: BLUESPAWN identifies evidence of T1055 that was launched via Metasploit.

From the attacker's perspective, commands worked perfectly initially. Once BLUESPAWN killed the threads the beacons were running in though, commands began to time out. Eventually this response led to the Meterpreter session dying as depicted in Figure 22.

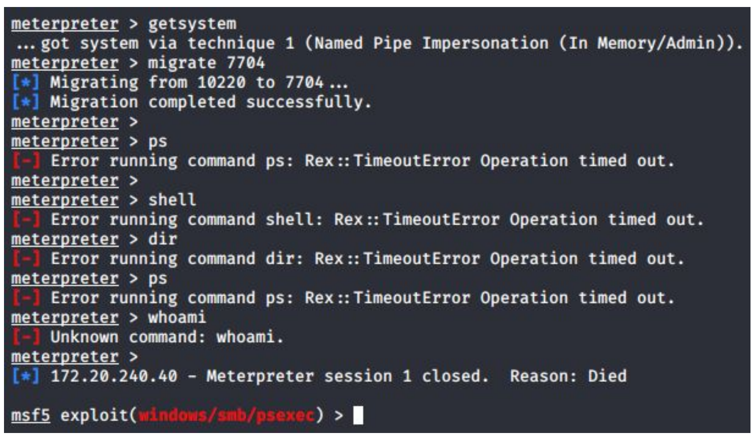


Figure 22: The attacker's Meterpreter session began timing out before dying completely.

7.2.3 Hunting for Registry-based Autorun Persistence

Attackers often leverage the Registry to automatically launch their malware as part of their persistence kit. To test an example of this scenario, we added a Debugger of cmd.exe for sethc.exe which is known as a "sticky keys backdoor [4]." The operator also used Metasploit's "registry_persistence" module to add a Run key [38]. The results illustrated in Figure 23 show BLUESPAWN detecting and removing these two particular items.

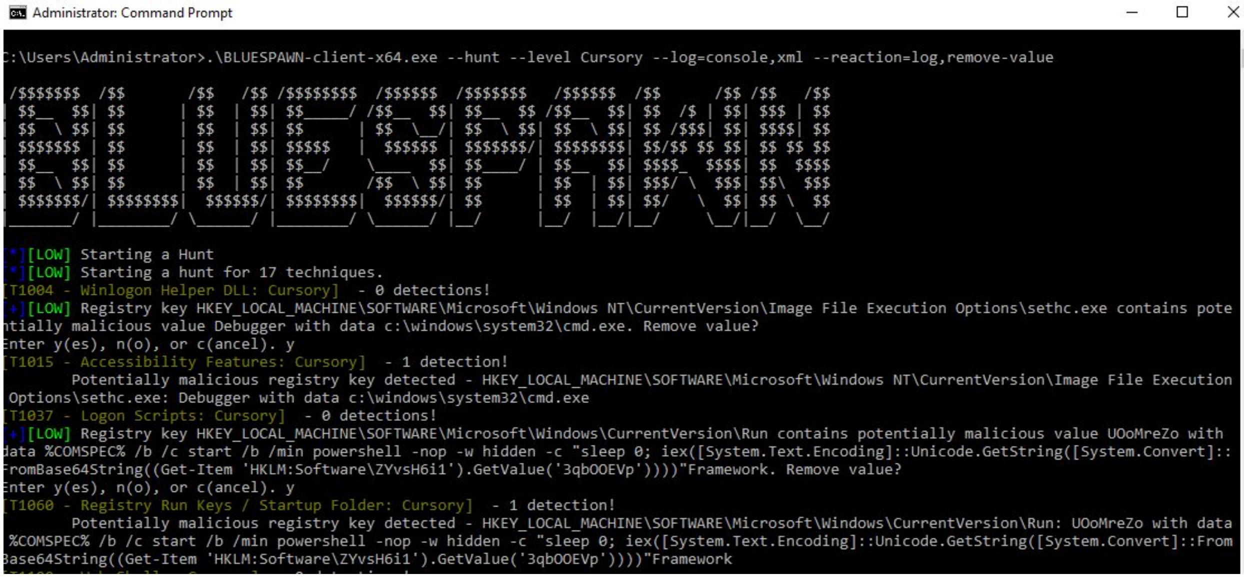


Figure 23: BLUESPAWN identifies a sticky keys backdoor (T1015) and a malicious run key (T1060)

7.2.4 Hunting for Webshells

Another popular persistence mechanism used to target web servers is web shells. These usually short snippets of code are placed in a web accessible directory and can be used by any visitor to the page to execute commands. In order to detect instances of T1100 - Web Shells, BLUESPAWN scans the web root directories for popular web server software. On Windows, this includes Internet Information Services (IIS) in C:\inetpub\wwwroot. In Figure 24, the program was able to detect a web shell using a combination of hand-crafted regexs and YARA rules.

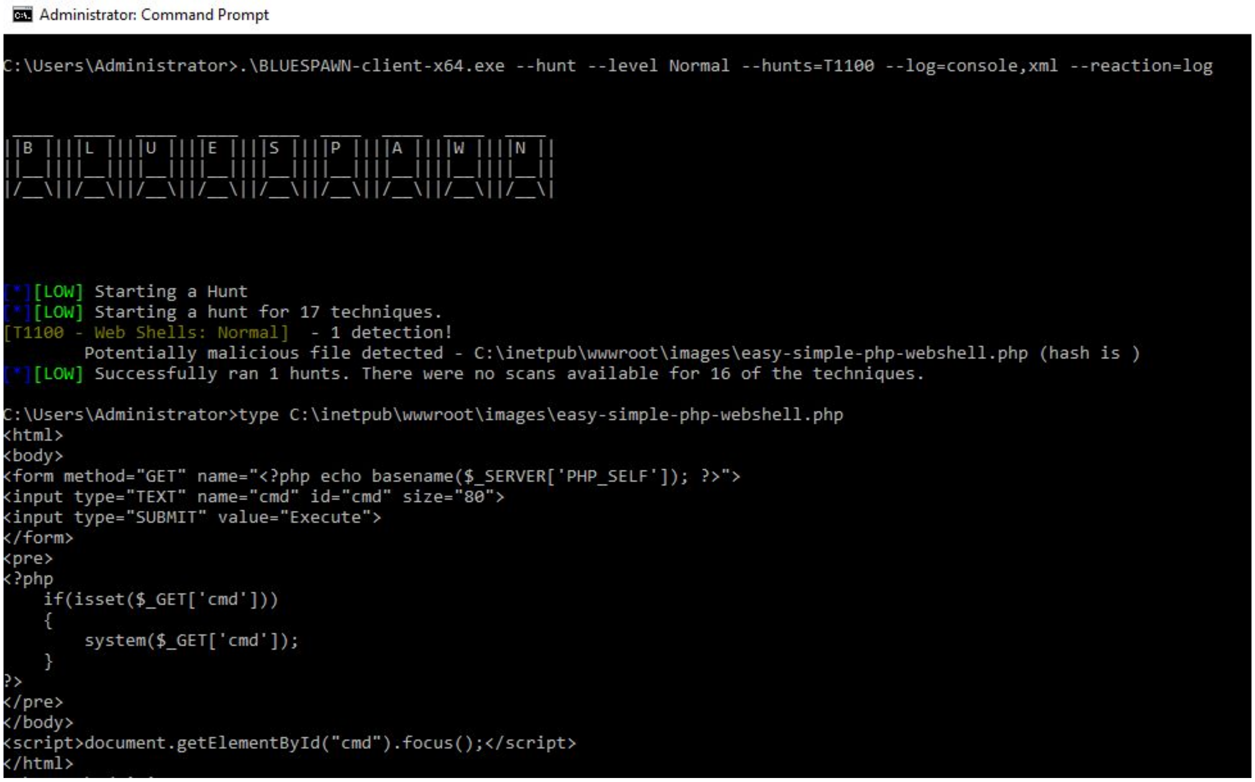
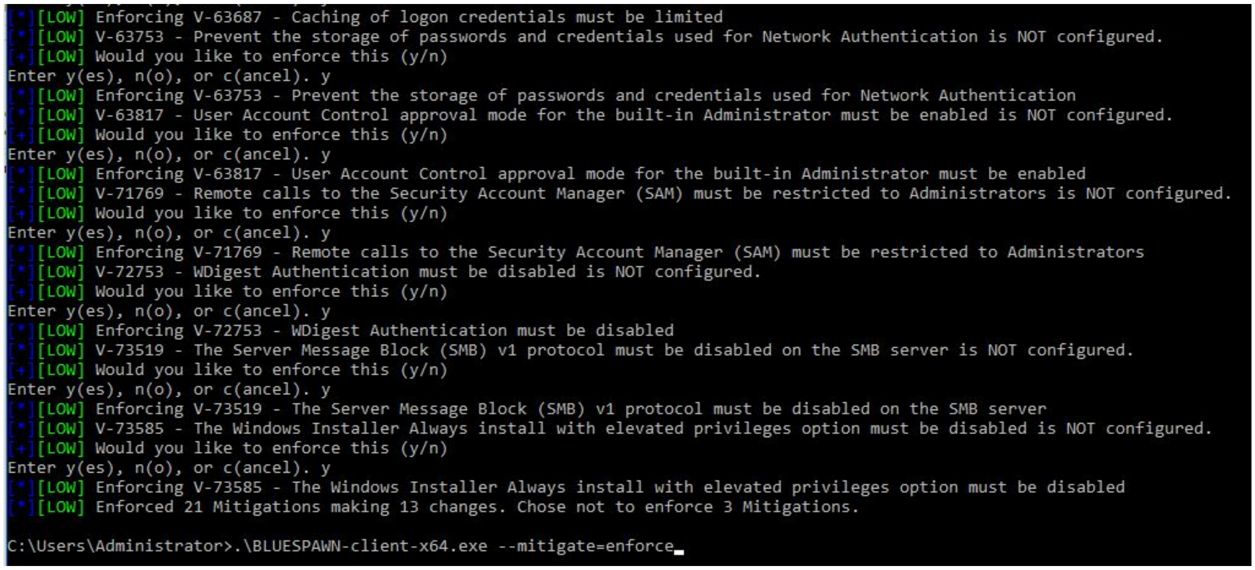


Figure 24: BLUESPAWN identifies a classic PHP web shell (T1100) [39].

7.2.5 Applying Mitigations to Improve the System's Security Posture

Finally preventing threats in the first place is just as important as being able to detect them. Mitigate mode helps an analyst do just that. By automating the auditing and application of critical security settings, one can quickly enhance a system's overall security.



```
[*][LOW] Enforcing V-63687 - Caching of logon credentials must be limited
[*][LOW] V-63753 - Prevent the storage of passwords and credentials used for Network Authentication is NOT configured.
[+][LOW] Would you like to enforce this (y/n)
Enter y(es), n(o), or c(ancel). y
[*][LOW] Enforcing V-63753 - Prevent the storage of passwords and credentials used for Network Authentication
[*][LOW] V-63817 - User Account Control approval mode for the built-in Administrator must be enabled is NOT configured.
[+][LOW] Would you like to enforce this (y/n)
Enter y(es), n(o), or c(ancel). y
[*][LOW] Enforcing V-63817 - User Account Control approval mode for the built-in Administrator must be enabled
[*][LOW] V-71769 - Remote calls to the Security Account Manager (SAM) must be restricted to Administrators is NOT configured.
[+][LOW] Would you like to enforce this (y/n)
Enter y(es), n(o), or c(ancel). y
[*][LOW] Enforcing V-71769 - Remote calls to the Security Account Manager (SAM) must be restricted to Administrators
[*][LOW] V-72753 - WDigest Authentication must be disabled is NOT configured.
[+][LOW] Would you like to enforce this (y/n)
Enter y(es), n(o), or c(ancel). y
[*][LOW] Enforcing V-72753 - WDigest Authentication must be disabled
[*][LOW] V-73519 - The Server Message Block (SMB) v1 protocol must be disabled on the SMB server is NOT configured.
[+][LOW] Would you like to enforce this (y/n)
Enter y(es), n(o), or c(ancel). y
[*][LOW] Enforcing V-73519 - The Server Message Block (SMB) v1 protocol must be disabled on the SMB server
[*][LOW] V-73585 - The Windows Installer Always install with elevated privileges option must be disabled is NOT configured.
[+][LOW] Would you like to enforce this (y/n)
Enter y(es), n(o), or c(ancel). y
[*][LOW] Enforcing V-73585 - The Windows Installer Always install with elevated privileges option must be disabled
[*][LOW] Enforced 21 Mitigations making 13 changes. Chose not to enforce 3 Mitigations.

C:\Users\Administrator>.\BLUESPAWN-client-x64.exe --mitigate=enforce
```

Figure 25: BLUESPAWN's Mitigate mode applies a variety of security settings.

7.3 Limitations and Gaps in Coverage

As shown previously in Figure 13, BLUESPAWN currently has support for just a handful of popular attacker techniques. Additionally, there are almost certainly many bypasses to the current implemented detections. Over time, detections will continue to improve though. Attackers will also adapt. What we have done so far with this project is to build a strong foundation. As we continue development, we will grow the tool's abilities to detect increasingly sophisticated threats. Mapping with the MITRE ATT&CK Framework and DoD's STIGs will also help to keep the program's mission focused on the most critical areas [4, 6].

8 Future Work

Overall, our work with this project is just getting started. We believe our efforts though have demonstrated the potential to detect real world attacks. So far, we have only made significant progress within the client portion; however, we are now starting on the server component. As we continue development, we plan to use popular commercial EDR products as a reference guide. In the future, as the threats continue to evolve, our tool will need to do so as well in order to stay effective. Additionally, as people find bypasses to the implemented detections, changes will need to be made.

8.1 Improvements to BLUESPAWN Client

As outlined in the above sections, development has primarily focused on this aspect of the project. The endpoint is the closest to the threats and thus, the best place to begin implementing our overall defensive strategy. In the coming months, we will continue to build out coverage of key data sources. These will enable detections to be built across all of the major ATT&CK techniques. Additionally, as we prepare to integrate the client with the server and cloud components, we will focus on making the client more configurable. Instead of being a standalone program, it will be a Windows Service and support custom detections.

8.1.1 Integration with the Anti-Malware Scan Interface (AMSI)

Looking beyond the most obvious upcoming features in the roadmap, the integration with AMSI will be an important step. In order to better support third party antimalware applications, Microsoft makes this set of APIs available. We can utilize these to provide real-time scanning [31]. For example, anytime an EXE requests elevation through UAC, BLUESPAWN would receive a notification, prompting a scan. AMSI also goes beyond just executables and works with PowerShell scripts, Windows Script Host, VBScript, and more [31].

While there have been a number of documented AMSI bypasses published, these APIs are a great starting point [40]. Furthermore, many of the top AV/EDR solutions rely on AMSI - and this feature will only be improved upon in future builds [13].

8.1.2 Modular, Configurable Detections

Right now, detections are written directly in C++ and compiled into the client. YARA rules are integrated in the same way, getting compiled in as a resource during build time. On one hand, this approach works well for preventing tampering. Unfortunately though, this kind of method does not scale and raises the bar to adding new signatures. Furthermore, the integration of the server and cloud components will require detections to be more customizable and configurable. As we begin to turn BLUESPAWN into a Windows Service, we will need to make these changes to keep the client lean, yet effective.

8.1.3 Heuristics, Behavioral Analysis, and Confidence Scores

Finally, another exciting space within the client development is behavioral analysis and confidence scores. For example, if we observe an unknown sample making suspicious APIs calls, should we generate a detection? What if it is launched by a process command line beginning with "`powershell.exe -windowstyle hidden -noninteractive -ExecutionPolicy bypass –EncodedCommand`"? While YARA rules focus on signaturing known quantities for the most part, dynamic detections enable automated identification of new malware. In order to do this though, one needs to be able to assign a "suspiciousness" or "malicious" score to it. A great illustration of how this can be done is looking at how a malware sandbox rates samples. Hybrid Analysis, for example, informs the user exactly what events generated the resulting threat score [41]. Another avenue tangentially related to threat evaluation is graph-based detections. If we start with the idea that one item is known malicious, what else has it touched? We can use its actions as a starting point to detect and identify other evidence of malicious activity on the system.

8.2 Creation of BLUESPAWN Linux Client

Since we initially launched the project, we have concentrated solely on Windows-based systems. These are a popular target for attackers, and there is ample online research to guide our efforts. In particular though, threat actors are also increasingly targeting other operating systems such as Linux. These targets have historically been underserved by defensive security products, primarily due to their market share (in use and attention by attackers). Now that this notion is changing, creating a Linux-based client would be a great way to expand our research & coverage. The general endpoint defense strategy would even stay largely the same, except the underlying data sources & attacker techniques would vary. While this aspect will be an incredible undertaking, there is lots of potential in this growing space.

8.3 Initial BLUESPAWN Server Development

Running BLUESPAWN on a single endpoint at a single point in time is a great way to begin a hunt for malicious activity. However, when there are hundreds, thousands, or tens of thousands of endpoints, that mentality just does not scale. In order to enable defenders to catch threats across their network, our next major step is starting to build out the server component. As we are only weeks into the initial designs for this piece, this area will rapidly evolve and likely look completely different a few months from when this paper is published. That said, as we see it now, there will be two primary components. First, the server will have a log collection, aggregation, and management engine. Most likely, this will resemble an Elasticsearch-Logstash-Kibana (ELK) setup, where clients will all ship logs to a centralized location. Elasticsearch will then index the logs while Kibana visualizes this telemetry [42]. The second half will be the management dashboard. This part will enable administrators to manage their fleet of BLUESPAWN clients across their network. It will include the ability to hunt across your environment, task clients to perform operations, and provide real-time, centralized security telemetry.

8.4 Initial BLUESPAWN Cloud Development

Finally, as we currently view the project, the third piece delivers the so-called "cloud-powered security" many vendors promise. While there is undoubtedly some hype to this component, there is strong merit to this concept. A single endpoint agent cannot be expected or tasked with fully evaluating every threat. It cannot and should not contain every signature both for operational security and performance reasons. To alleviate this problem, a cloud based element would help to offload workloads to the cloud. There, software could perform more advanced analysis of a potential threat using static and dynamic techniques. Furthermore, a cloud-based component would provide the ability to quickly update signatures for new threats. As new detections are developed, they can be pushed out as definition updates to clients, shortening the time from first sight to coverage. We stress that this feature is more on the long-term roadmap for the project, but we see the possibility for this to be hosted in the actual cloud or on-prem. This hybrid model would give organizations the flexibility to deploy BLUESPAWN reliably in a variety of environments, especially on air-gapped systems.

9 Conclusion

As the cybersecurity industry continues to evolve, the threats show no sign of stopping. Increasingly advanced defenses will be needed to stop increasingly advanced attacks. Open-source programs such as BLUESPAWN help to shed light on the historically “black-box” nature of commercial products. In addition, they can be helpful in creating a tailored approach to respond to threats, equipping *any* security professional or student with the capabilities they need to at least begin investigating a breach. Over time, as the project grows to include components like a server or cloud, the tool’s accuracy and effectiveness will increase.

10 References

[1] Jake Smith, Jack McDowell, Calvin Krist, and Will Mayes. 2020. BLUESPAWN. Retrieved from https://github.com/ION28/BLUESPAWN

[2] Kaspersky. 2018. About Protected Process Light (PPL) technology for Windows. Retrieved from https://support.kaspersky.com/common/windows/13905

[3] Microsoft. 2017. Virtualization-based Security (VBS). Retrieved from https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-vbs

[4] MITRE Corporation. 2019. Matrix - Enterprise | MITRE ATT&CK®. MITRE ATT&CK. Retrieved from https://attack.mitre.org/matrices/enterprise/windows/

[5] Paul Shread. 2020. Top Endpoint Detection and Response (EDR) Solutions. Retrieved from https://www.esecurityplanet.com/products/top-endpoint-detection-response-solutions.html

[6] Defense Information Systems Agency. SRG / STIG Tools – DoD Cyber Exchange. DoD Cyber Exchange. Retrieved from https://public.cyber.mil/stigs/srg-stig-tools/

[7] Anton Terekhov. 2017. History of the Antivirus. Hotspot Shield VPN. Retrieved from https://www.hotspotshield.com/blog/history-of-the-antivirus/

[8] Anton Chuvakin. 2013. Named: Endpoint Threat Detection & Response. Anton Chuvakin. Retrieved from https://blogs.gartner.com/anton-chuvakin/2013/07/26/named-endpoint-threat-detection-response/

[9] VMWare. What is an Endpoint Protection Platform (EPP)? | Endpoint Protection Platform Definition. VMware Carbon Black. Retrieved from https://www.carbonblack.com/resources/definitions/what-is-an-endpoint-protection-platform-epp/

[10] McAfee. What Is an Endpoint Protection Platform? | McAfee. Retrieved from https://www.mcafee.com/enterprise/en-us/security-awareness/endpoint/what-is-an-endpoint-protection-platform.html

[11] Kaspersky. 2017. A Buyer's Guide to Investing in Endpoint Detection & Response. Retrieved from https://media.kaspersky.com/en/business-security/enterprise/EDR-whitepaper.pdf

[12] Crowdstrike. STREAMING THE THREAT DETECTION AND RESPONSE LIFECYCLE WITH SPEED, AUTOMATION AND UNRIVALED VISIBILITY. Retrieved from https://www.crowdstrike.com/wp-content/brochures/Falcon-Insight-DS-PW-edits.pdf

[13] Lee Holmes. 2019. Lee Holmes on Twitter: "I love it when I hear good news! AMSI State of the Union - November 2019. @Sophos is now protecting you with its AMSI integration as well! https://t.co/0rd9sjhFAW" / Twitter. Twitter. Retrieved from https://twitter.com/lee_holmes/status/1189215159765667842

[14] Lenny Zeltser. 2019. Free Automated Malware Analysis Sandboxes and Services. Retrieved from https://zeltser.com/automated-malware-analysis/

[15] Mark Russinovich. 2020. Windows Sysinternals - Windows Sysinternals. Retrieved from https://docs.microsoft.com/en-us/sysinternals/

[16] Mark Russinovich. 2019. Autoruns for Windows - Windows Sysinternals. Retrieved from https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns

[17] FIN7 Evolution and the Phishing LNK. FireEye. Retrieved from https://www.fireeye.com/blog/threat-research/2017/04/fin7-phishing-lnk.html

[18] Mark Russinovich. 2016. ListDLLs - Windows Sysinternals. Retrieved from https://docs.microsoft.com/en-us/sysinternals/downloads/listdlls

[19] Mark Russinovich. 2020. Process Explorer - Windows Sysinternals. Retrieved from https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer

[20] Mark Russinovich and Thomas Garnier. 2020. Sysmon - Windows Sysinternals. Retrieved from https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

[21] Mark Russinovich. 2017. Sigcheck - Windows Sysinternals. Retrieved from https://docs.microsoft.com/en-us/sysinternals/downloads/sigcheck

[22] Mark Russinovich. 2011. TCPView for Windows - Windows Sysinternals. Retrieved from https://docs.microsoft.com/en-us/sysinternals/downloads/tcpview

[23] Wen Jia Liu and Steven G. 2020. Process Hacker. Process Hacker. Retrieved from https://github.com/processhacker/processhacker

[24] Hasherezade. 2020. hasherezade/pe-sieve. Retrieved from https://github.com/hasherezade/pe-sieve

[25] Florian Roth and Thomas Patzke. 2020. Neo23x0/sigma. Retrieved from https://github.com/Neo23x0/sigma

[26] Cybersecurity and Infrastructure Security Agency. Traffic Light Protocol (TLP) Definitions and Usage | CISA. US Cert. Retrieved from https://www.us-cert.gov/tlp

[27] National Security Agency. 2020. nsacyber/Windows-Secure-Host-Baseline. NSA Cybersecurity Directorate. Retrieved from https://github.com/nsacyber/Windows-Secure-Host-Baseline

[28] Microsoft. 2020. microsoft/PowerStig. Microsoft. Retrieved from https://github.com/microsoft/PowerStig

[29] Dan Larson. 2017. A Strategy to Find and Stop Attackers Before They Do Damage. Retrieved from https://www.crowdstrike.com/blog/approaching-zero-dwell-time-strategy-finding-stopping-attackers-damage/

[30] VirusTotal. 2020. YARA - The pattern matching swiss knife for malware researchers. Retrieved from https://virustotal.github.io/yara/

[31] Microsoft. Antimalware Scan Interface (AMSI) - Win32 apps. Retrieved from https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal

[32] Red Canary Co. 2020. redcanaryco/atomic-red-team. Red Canary. Retrieved from https://github.com/redcanaryco/atomic-red-team

[33] Center for Infrastructure Assurance and Security. 2006. History of NCCDC. NationalCCDC. Retrieved from http://nationalccdc.org/index.php/competition/about-ccdc/history

[34] Raytheon Corporate Communications. 2019. Raytheon: University of Virginia Defends National Cyber Title. Raytheon News Release Archive. Retrieved from http://raytheon.mediaroom.com/2019-04-26-University-of-Virginia-Defends-National-Cyber-Title

[35] UVA Cyber Defense Team - Windows Group. 2020. BLUESPAWN Reflection Interview.

[36] TJ Null. 2020. BLUESPAWN Red Team Perspective Interview.

[37] Raphael Mudge and Strategic Cyber LLC. 2020. Adversary Simulation and Red Team Operations Software - Cobalt Strike. Retrieved from https://cobaltstrike.com/

[38] Rapid7. 2020. Metasploit | Penetration Testing Software, Pen Testing Security. Metasploit. Retrieved from https://www.metasploit.com/

[39] @joswr1ght. 2019. easy-simple-php-webshell.php. Gist. Retrieved from https://gist.github.com/joswr1ght/22f40787de19d80d110b37fb79ac3985

[40] Andre Marques. 2018. How to bypass AMSI and execute ANY malicious Powershell code. zc00l blog. Retrieved from https://0x00-0x00.github.io/research/2018/10/28/How-to-bypass-AMSI-and-Execute-ANY-malicious-powershell-code.html

[41] Hybrid Analysis. 2020. Free Automated Malware Analysis Service - powered by Falcon Sandbox - Viewing online file analysis results for “searchfiles.exe.” Retrieved from https://www.hybrid-analysis.com/sample/5ff7863b8969855e695d0bf255f60e24cec10efd36b2b5f05e4cdb7e2f7ac15a?environmentId=120

[42] Elastic Co. 2020. ELK Stack: Elasticsearch, Logstash, Kibana | Elastic. Retrieved from https://www.elastic.co/what-is/elk-stack

Investigating the Politics of Open Source Software

A Research Paper Submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia - Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

By

Calvin Krist

Fall 2020

Signature ______________________ Date 11/15/2020

Approved ______________________ Date 11/20/2020

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Introduction

In 1999, Eric Raymond published *The Cathedral and the Bazaar* that posited a radical, new idea: that open source software development creates better products [18]. Open source development refers to the practice of having a programming project whose source code is freely available, and that anyone can contribute to (although their contributions may be vetted). There had been similar projects prior to this, termed “free software”, but they were tinged with ideological and political beliefs that prohibited their widespread adoption. Open source grew out of these communities and in the process threw off the ideological underpinnings. Additionally, popular belief was that difficult, large-scale, or complicated software projects required a dedicated team with forethought and planning from the top down, instead of the decentralized development model of open source software.

However, the development of the Linux operating system in the 90’s convinced Raymond that this was not true. Operating systems are one of the most complicated and difficult feats of software development, and Linux was developed using open source practices. This proved that open source development could complete with classical development methodologies, and Radmond believed it could surpass them.

It has been over twenty years since the publication of *The Cathedral and the Bazaar*, and open source development has continued to grow in relevancy. Over 13% servers worldwide run on the Linux operating system [25]. There are open source alternatives in most software categories including office productivity, photo editing, and web browsers. Multiple open source companies have been sold for billions of dollars [1][19][22], and millions of open source projects are hosted worldwide [21]. Anecdotally as a student, I have been exposed to multiple platforms to host open source software, the tools to use them, and specialized development practices for open source. I have been taught to use and believe in open source development more than traditional top-down methods. The impact of open source development cannot be understated.

However, while more and more developers continue to be taught and use open source development, and its relevancy in the open marketplace only grows, the political and historical background of it is not often discussed. This is problematic because while “open source” does not carry the same ideological meaning as “free software”, the use of these development practices can still communicate those political ideas. The people most exposed to these ideologies would be software developers, a group who can exert enormous power on the future of the Internet and what it will look like. Thus, understanding the history of open source software can provide insight into the politics of the Internet and allow software developers to more be more conscious of the meaning of their practices.

The open source community can trace its roots back to the early creation of the Internet. This paper will explore the politics and culture of the Internet and open source software using the framework of co-production, which analyzes how communities and technological artifacts evolve together and give each other meaning. Furthermore, as the discussion surrounding open source software often becomes ideological, the idea that artifacts have politics will also be employed to look at how software is embedded in political ideologies, and how those political ideologies in turn affect software developers and the development of the Internet.

This paper will start by looking at how the US military influenced the early world of computers, and why they wanted to research technology such as the Internet. The paper develops the idea that the military had a political goal to their interest in computers, and shows how this goal clashed with the researchers who helped create the Internet, leading to major ideological differences. The paper then explores how various groups protested the military's ideology, and how the target of this protest shifted from the military to corporations. This conflict eventually gave rise to the "free software" movement, and the politics of this group is analyzed. From there, the split between the "free software" and "open source" communities is explored and its long term effect on the software development community. Finally, this paper makes an argument for the importance of educating software developers on the politics of the Internet, free software, and open source software due to their ability to influence the Internet, a central aspect of most modern societies.

The Early Internet

This investigation shall begin with the era of the Cold War and the partnership between university researchers and the military, as the Internet was a result of this partnership. ARPANET, the predecessor of the Internet, came from the Advanced Research Projects Agency (ARPA), founded in 1957 out of fear of Soviet technological domination after the launch of Sputnik.

The Internet is one of the most influential inventions in human history, and as a result many histories of it have been written. Many of them choose different points to start their stories, and different moments when the Internet was "invented", highlighting both the inherent nonsensicality of choosing a singular moment as well as the different ideologies that underpin these histories. From the start, the Internet was developed as a tool of the military -- of politics -- and it has grown into a tool that can be used to spread *all* political ideologies. A history of the Internet cannot help but be political in nature.

In *Where Wizards Stay Up Late: The Origins Of The Internet*, Matthew Lyon and Katie Hafner tell the story of the Internet using a "Great Man" approach. Their story begins with BBN, the consulting company that had the initial ARPA contract for what became ARPANET. In their story, Bob Taylor, the head of the computer research office at ARPA, was annoyed because the room next to his office had three computer terminals to three different mainframes that all ran different operating systems and programs. Taylor recounts that "It became obvious that we ought to find a way to connect all these different machines" [3].

After gaining funding, Taylor enlisted Larry Roberts to assist him, a "shy, deep-thinking young computer scientist" who "had the reputation as being something of a genius" [3]. From there, the book tells the story of a group of dedicated, intelligent engineers slowly building out the building blocks of the Internet.

Another popular history of the Internet, "Brief History of the Internet" by Bruce Sterling, starts with a Cold War think-tank trying to figure out how US authorities could communicate after nuclear war. Their solution was a decentralized networking system where any single component could be destroyed without compromising the ability to communicate. Starting here, instead of with Taylor and his office, roots the Internet in the darkness of the Cold War rather than a quirky engineering problem.

Hafner and Lyon do not ignore this portion of the history, but they downplay it in comparison to Sterling. These two histories do not credit the same engineers as being the architects of the Internet because they view the Internet in fundamentally different ways. For Hafner and Lyon, ARPANET "embodied the most peaceful intentions to link computers at scientific laboratories across the country, so that researchers might share computer resources… ARPANET and its progeny, the Internet, had nothing to do with supporting or surviving war -- never did" [3].

Yet another history, called *Transforming Computer Technology: Information Processing for the Pentagon, 1962-1986* by Arthur Norberg and Judy O'Neill, focuses more on the bureaucratic factors that came together to result in funding and research for ARPANET. They consider not just ARPANET, but *all* ARPA contracts from 1962-1986 including those for new operating systems, artificial intelligence, and networking technology. This history highlights the close connection between all ARPA funding and military concerns, a fact often lost in Hafner's and Lyon's story. For example, whereas Hafner and Lyon describe the first director of the Information

Processing and Techniques Office (IPTO) as pushing it toward research for the sake of research, Norberg and O'Neill quote him as telling another military official that ARPA should only fund research that offers "a good prospect of solving problems that are of interest to the Department of Defense" [4].

They also show how many of the networking experiments that Taylor received funding for came from the IPTO's concern with using computers to create a more effective military. Norberg and O'Neill go one step further, and argue that it was due to the close ties to the military that ARPANET was successful. They argue that the incentives to develop such technology did not exist in the private sector at the time, while the military had both the interest and the funding [4].

It is important to ask the question of *why* the military cared about the Internet and what the Internet symbolized to the DOD. First, it can be easily shown that the military was one of the greatest funders and contributors to computing. In 1950, for example, the federal government (mostly its military agencies) provided 75%-80% of computer development funds [5]. In part, this funding allowed USA companies to dominate the computing sector for many years which the government doubtless cared about.

Moreso, it can be argued that computers to the US military were a mechanism to exert centralized command and control in the Soviet era of highly centralized political power. Computers allowed the military to communicate more easily, organize much larger groups, and in some cases even automate warfare. They helped the central military command exert even more influence over war, but only so long as computers were a technological advantage that the USA had over the USSR.

In *The Closed World: Computers and the Politics of Discourse in Cold War America*, Paul Edwards makes a convincing argument that military funding dramatically helped shape the direction that computing technology and thus the Internet developed in. This influence was not done without intention: rather, it was with the goal of shaping the Internet into a tool to exert centralized command and control [6]. Although his writings do not directly comment on ARPANET, it is difficult to read them and then agree with Hafner's and Lyon's view of the relationship between the military and the Internet, or even Norberg's and O'Neill's who acknowledge the connection but paint it as benign. And indeed, with the early days of ARPANET, the technology, while itself decentralized, was controlled by the military and limited to a select few universities who were invited to join the program. It supported the military's vision of a closed world subject to technological control.

This influence was apparent to many within the ARPA programs as well as leading scientists outside of it. There were arguments against it on the basis of human rights, citing the many examples of the military exerting tremendous horrors upon other humans aided by technology and computers. There were arguments on a more ideological basis, claiming that all information should be free simply because freedom and knowledge are good things. The opposition to the military's influence and control quickly took on political and ideological tones and led to major movements within the computer science field, including both the free software movement and eventually the open source movement. The politics of this argument are apparent even in the histories we tell about the Internet. Hafner's and Lyon's history de-emphasized the role of the military while praising the Internet as being a tool of peace and freedom of information, clearly showing the perspective of those opposed to centralized influence and in support of the free-flow of information. On the other hand, histories like Norberg's and O'Neill's celebrate this military influence as shown in the name of their final chapter, "Serving the Department of Defense and Nation" [4].

The Politics of the Internet

In 1968, fifty senior faculty members at MIT, the country's largest academic defense contractor, circulated a statement claiming that the misuse of science and technology were a great threat to humankind, and that due to events in Vietnam the USA government had lost their confidence in making humane and wise decisions. In early 1969, massive student protests occurred against the Stanford Research Institute (SRI), calling for an end to classified research and related military contracts. Later, some 8,000 students and faculty voted to commend the protesters for focusing attention on the nature of the research being conducted at the SRI. On a wider scale, national antiwar protests were focusing on classified research [5].

Conflict would arise within ARPA itself. The MIT Artificial Intelligence Lab helped work on ARPANET, and by this time had a well developed and distinct culture they termed "hacker" culture. Hacker culture had radically different values than the military, with an emphasis on playfulness and experimentation [7]. This also expressed itself as a disdain for rules and a value on the freedom of information. At one point, Richard Stallman, a researcher at the Artificial Intelligence Lab and eventual leader of the free software movement, carried on a "guerrilla war against the use of passwords on the system" [5]. This caused the DOD to be nervous because anyone could walk in and access ARPANET, and they threatened to cut funding unless security features were added.

This internal tension between the DOD and the academic researchers helping to develop the early Internet would be long lasting. ARPANET was increasingly used to share hacker mailing lists, slang, and inside jokes between researchers. It was a vehicle of social and recreational purposes, and it enabled the free flow of information between diverse interest groups. Eric Raymond, the eventual leader of the open source movement, wrote that “DARPA deliberately turned a blind eye to all the technically `unauthorized' activity; it understood that the extra overhead was a small price to pay for attracting an entire generation of bright young people into the computing field” [8].

This leads to yet another interpretation of the history of the Internet, given by Michael and Ronda Hauben in *Netizens: On the History and Impact of Usenet and the Internet*. In this interpretation, the meaningful story is not that of the engineers who made the technology, nor the story of the military and other bureaucratic organizations that funded ARPA and its research. Rather, the emphasis is placed on the “netizens” -- the citizens of the Internet -- who figured out what it was “really” for and made it popular [9].

Going into the 70’s, only 2% of the possible bandwidth of ARPANET was being used [5]. It was technology in large part developed by the military with cooperation from university researchers, but neither could use it how they wanted. The military had developed it for a post-nuclear war landscape, and otherwise feared the network as leading to a loss of control and chain of command. The university researchers used it to share knowledge and jokes, and while the military turned a blind eye to some of their communications there was pressure to use it only for serious work.

Then in 1979, two graduate students at Duke University created a program that allowed users of the popular Unix operating system to exchange files. They used this to create a newsletter, allow comments, and even developed email between computers connected by a modem. This was a conscious alternative to ARPANET, which even then was still limited by the DOD for military purposes. They called it Usenet, and it was "trying to give every Unix system the opportunity to join and benefit from a computer network (a poor man's ARPANET if you will)" [10].

Usenet grew rapidly, and was almost entirely used for conversation and communication. This was paralleled by developments in ARPANET. In 1972, Ray Tomlinson developed a program to send emails over ARPANET, and within a year 75% of all network traffic on the network was devoted to email. Similarly to Usenet, email -- a true purpose for this technology -- came from below, from computer users who wanted to communicate, and not from ARPA directives.

The Haubens spend much time praising Usenet and similar networks for their democracy and for allowing an "uncensored forum for debate". They place the rise of the Internet within the historical context of the 60's, and the counterculture movements and cries for democracy and freedom arising from many University environments. They emphasize how Internet standards were developed through open forums, then spread through RFCs (Request for Comments) over the network so that everyone could contribute [9]. This is, in many ways, still how Internet protocols are developed.

And it should be noted that these are not political interpretations of the present imposed on the past. Rather, within the time period the development of the Internet was already being viewed in these terms. Bay Area programmers who loved computers and politics actively worked to combine these interests. In 1972, for example, Bob Albretch launched a tabloid called *People's Computer Company*, whose cover of the first issue proclaimed "Computers are mostly used against people instead of for people; used to control people instead of to free them; Time to change all that. We need a. . . People's Computer Company" [11]. In 1973, the Berkeley Community Memory project was started which offered a free to use community bulletin board to post flyers, notes, and ads. It deliberately aimed to make computer technology benefit the community and to remove it from the relatively closed and affluent government and university control [12].

These examples, as well as Usenet, show a conscious effort to develop and use technology that embodies a different ideology and political message than ARPANET. Furthermore, the technology and artifacts developed by these groups doesn't simply *embody* this difference: they work to actively spread it, and in turn influence the programmers and computer designers of the future. Usenet was developed as an ARPANET alternative that anyone could join. This attracted those who felt that ARPANET was in the wrong, and then provided a forum where they could discuss those political differences [9]. The Berkley Community Project aimed to change not just the opinion of technologists about computers, but the greater community. Its designers were worried that computers would be viewed as a technology for the rich and powerful, and wanted to spread its power to the people [12].

This shows that from early on, the development of the Internet and associated computer technology was done with political intentions from many groups. The military viewed it as a method to strengthen themselves and the United States, and more so as a tool to further a vision of a closed, technological world. Others, from the MIT hackers to the developers of Usenet, opposed this viewpoint and instead emphasized the free flow of information, thought, and democracy and worked to subvert this control.

Free Software

One of the most significant figures to emerge from this political schism was Richard Stallman, a researcher at the MIT Artificial Intelligence Lab who helped make ARPANET and internally opposed DOD security protocols, and eventually founded the Free Software Foundation.

In the 1960's most software was "free", meaning that it was free to use, modify and distribute however anyone wished. "Free software" does not necessarily mean that the software is free to acquire, but rather "free" as in free speech, and freedom in general.

Most software was produced through research, and was thus published as part of the public domain. However, throughout the 1970's even as new networks such as Usenet created communication channels for a more open community, software was becoming more restricted. Software was becoming a commodity, and as a result more of it was being copyrighted. This was especially the case after the Copyright Act of 1976 (which didn't become effective until 1978) which made it much easier to copyright software [13].

Many of those who opposed the closed vision of ARPANET also opposed this more controlled world of software. As Richard Stallman told it in a speech to New York University in 2001, software used to be like a cooking recipe. Recipes get passed around from friend to friend, down family lines, and they get modified. Perhaps a recipe is great, but it has slightly too much salt for someone, or is too spicy, and so they change it. Stallman argued that programs were like recipes except for computers, in that both are a series of step-by-step instructions. In his academic world of the 60's, programs were often passed around and modified to suit new problems and purposes, and this was *right*. And then software copyrights got in the way [14].

As the story goes, here was a critical incident for him with a Xerox printer. He was working at the MIT Artificial Intelligence Lab, and Xerox gifted them a laser printer -- the first time anyone outside of Xerox had acquired one. However, the printer jammed frequently and they needed a way to tell when this happened. They decided to set it up so the printer would tell their operating system when it was jammed so it could be fixed. However, the printer didn't come with code, and Xerox refused to distribute it, and so they continued to suffer jamms and decreased productivity.

Then Stallman heard that someone at Carnegie Mellon had a copy of the software, and so he went and asked for it. He was denied, because this individual had promised not to share it. "I was stunned. I was so -- I was angry, and I had no idea how I could do justice to it. . . He had betrayed us." [14].

This was a radical experience for Stallman because to him, "the purpose of science and technology is to develop useful information for humanity to help people live their lives better” [14]. The withholding of that information is a betrayal to that calling, and program source code is included in information.

Around this time, his research lab at MIT was radically changing, and so he quit in 1984 and started working on GNU, a new operating system that aimed to provide an alternative to Unix. Of course, GNU was “free software” and it was distributed using the Usenet system. Anyone could download it, modify it, or use it however they wanted. Stallman established the Free Software Foundation on October 4th, 1985 as a non-profit to support the free software movement and to help distribute his GNU operating system and associated software like Emacs, a popular text editor. This foundation provided a way for him and others to make income while developing free software.

Critically in 1989 the Free Software Foundation released the GNU General Public License (GPL). This was a radically new take on copyright, and it was termed “copyleft”. Software published with the GNU GPL is free software, and all software that is derivative of GNU GPL must also be free. This license represented a legal structure for free software, counter to the common legal structure of copyright. It was made directly to contrast the existing power structures, and it is a powerful tool in the arsenal of Stallman and others who oppose the closed technological vision of computer technology.

The Free Software Foundation and the GNU GPL can be viewed as the continuation of the ideological conflict of the 60’s and 70’s. Stallman and his allies viewed the Internet and computers as tools of democracy and freedom, and while the government had certainly continued to do classified research in pursuit of their vision, the enemy was no longer the military: rather, it was the capitalist systems that insisted on copyrighting software and limiting its use. Legal systems such as copyright and software ownership served to further those ideologies, and believers in the freedom of software needed a similar alternative or else their fight would be drastically unequal. They needed the legal power offered by the Free Software Foundation and the GNU GPL.

To the Free Software Foundation and others like them, creating free software was a form of protest. Having a successful free operating system like GNU would serve to demonstrate that there were other ways of viewing computers and technology than the restrictive perspective that was quickly becoming dominant. When someone used GNU, it was at the same time supporting the freedom of software. Free software is inherently about the politics of artifacts, and thus the ideology is furthered by generating artifacts in its image. The creation of such artifacts not only protested against the status quo, but also created software whose usage could potentially convince others of the ideology.

Free Software vs. Open Source

By 1990, the Free Software Foundation had written all the parts to replace Unix they had promised -- except the kernel itself, the core of the operating system. They had compilers, text editors, graphical libraries, but not that most critical component. Then in 1991, a developer named Linus Torvalds published “Linux”, a working Unix-like kernel. By 1992 it was made free software was often known as “GNU/Linux”. The initial goal of the Free Software Foundation had been completed [16].

The Linux kernel was, in some ways, a new accomplishment. Free software had proven that it could create useful programs like Emacs, but had not yet produced something truly difficult. Operating system kernels are, arguably, the most complicated, difficult programs humans have ever created. Many people thought free source software was suitable for small projects, but that programs as difficult as kernels required hierarchies and strict development plans. They thought it required the development practices of corporations and the military, rather than the advocates of free software.

The Linux kernel proved them wrong, and in 1999, Eric S. Raymond published *The Cathedral and the Bazaar*, a seminal book that argues that “open source” software is capable of writing as good or better software as any other practice. The book compares open source development to a bazaar, where it might get messy and there’s shouting but ideas and goods can move freely, as opposed to a traditional development being like a cathedral, “carefully crafted by individual wizards or small bands of mages” [18]. The book was wildly successful, and the idea of open source development took off.

Critically, the book uses the phrase “open source”, referring to when source code is open and accessible by anyone, instead of “free software”. Raymond thought that the term “free software” was too confusing. People often thought it meant it didn’t cost anything, and it made businesses scared to produce free software because it was perceived as impossible to profit from. This rebranding was an effort to engage with the mainstream programming and business culture and eventually subvert it [17].

Stallman disagreed, claiming it wasn’t “pure” enough [17]. The phrase “open source” only indicates a development practice, a way of doing things, while “free software” indicates a value system. This has led to the growth of two separate but related communities: that of “free software”, and that of “open source”.

The free software community is still championed by the Free Software Foundation, but continues its original goals of furthering the adoption of free software and fighting a closed technological worldview. They have revised their GNU GPL twice, and continue to work on commonly used free software applications. However, the free software community hasn’t had nearly the reach that the open source community has.

Open source software is now a multi-billion dollar industry with many major companies championing its use. Red Hat, a major open source software producer, was recently acquired by IBM for $33 billion dollars [19]. Open source communities have their own licenses, such as the MIT license. These are considered “permissive” in that they don’t have the copyleft requirement, and have overshadowed copyleft licenses like GNU GPL in popularity [20]. GitHub, a website that hosts over 40 million open source projects [21], was recently acquired by Microsoft for $7.5 billion [22]. A 2020 survey by RedHat found that 95% if IT respondents say open source is a priority, and that many of them believe open source software will continue to grow. In large part, these IT leaders cite the quality of code of open source projects as the cause [23].

Critically, however, these companies and platforms all practice *open source* development because it results in better code, and do not care about the ideologies of free software. Most of these projects do count as free software, but they are not designed as such. In part, it has become so popular because it makes software developers the customers. They discover the projects online, download it and experiment with it, and integrate it into their own projects. Similarly to free software, the existence of open source software spreads its own ideology. The world increasingly looks like that of Stallman's vision, where all software is free to be downloaded and modified. However, it comes without the same values. Developers do not practice or care about open source because they care about the democracy of the Internet and the free flow of information. It is entirely possible they are not even aware of the discussion, or of how many groups actively work to create an Internet that is closed and is a tool of authoritarianism and control [24].

The Internet and Democracy

For some, the advent of the Internet and similar technologies was a momentous day for democratic values. It seemed to be a technology that worked in a free, democratic process. The very architecture that allowed it to function mimicked idyllic human societies, with networks of equals that can all communicate and freely share information. And for many people, that vision of the Internet was true and still is. It allows for diverse communities to grow, thrive, and engage with the world around them. It can allow for the free flow of information to a degree unimaginable to earlier generations. And for many, the Internet and the anonymity it offers can equalize power structures that only exist in other aspects of societies.

However, that view of the Internet is always threatened. On one hand, power structures are far from equal on the Internet. On major platforms such as YouTube, the websites have absolute power of all that happens, and groups such as traditional news organizations wield power and get benefits that independent creators do not [15]. Autocracies engage in mass censorship campaigns to control their people and limit the information they have access to, turning the technology into a tool of control and fulfilling the military's original vision. They do this in two ways; by controlling the infrastructure of the Internet, the hardware and cables and other invisible aspects that most users know nothing of [24]; and by turning computing technologies into surveillance systems, and constantly spying on their populace and punishing those who deviate.

Even when governments do not abuse the Internet, the infrastructure on which it runs and many of the major websites through which most Internet traffic goes are created and managed by corporations. Thus, non-state regulators play an important role in protecting and furthering the democracy -- or lack thereof -- of the Internet. Sometimes the public is aware of this debate, such as in the discussion over net neutrality, but more often they are not. Search engines and artificial intelligence, for example, have a long history of bias against minorities. Occasionally these flaws get caught [2], and we trust that corporations such as Google do their best to minimize these biases, but this process is largely hidden. Such biases are due to existing biases in society and datasets, and then in turn work to reinforce those biases in the public. They reduce equality, and in a system that many assume to be neutral, such as a search engine, this can have harmful impacts. More nefariously, consider if Google decided to prioritize search results or ads from certain groups over others whether for profit or political purposes. Perhaps the public would never know, and even if they did, it's possible there would be nothing to do about it.

Software developers and hardware architects are some of the few who both understand how these fundamental parts of the Internet work, and can influence their development and thus the politics they represent and encourage. There are the obvious examples, such as engineers at Google who work to ensure their search engine does not perpetuate negative stereotypes of minorities, or the engineers who help autocratic regimes censor the Internet. But software developers must also be aware of the politics of how they develop software. They should consider what rights people have to software and where they fall on the “free software” debate. They should consider if using Linux or working on an open source project is making a statement, and how others, particularly other software engineers, might interpret the ideology of their choices. They should consider what role they believe the Internet and computers in general should play in society, and how they can work to bring that vision to life. While everyone who uses the Internet can influence the politics it represents, software and hardware developers are especially privileged in how they can impact one of the most fundamental aspects of our society.

References

1. Volpi, Mike. “How Open-Source Software Took over the World.” *TechCrunch*, TechCrunch, 12 Jan. 2019, techcrunch.com/2019/01/12/how-open-source-software-took-over-the-world/.
2. Noble, Safiya. “Google's Algorithm: History of Racism Against Black Women.” *Time*, Time, 26 Mar. 2018, time.com/5209144/google-search-engine-algorithm-bias-racism/.
3. Hafner, Katie, and Matthew Lyon. *Where Wizards Stay up Late the Origins of the Internet*. Simon & Schuster Paperbacks, 2006.
4. Norberg, Arthur L., and Judy E. O'Neill. *Transforming Computer Technology: Information Processing for the Pentagon, 1962-1986*. Johns Hopkins University Press, 1996.
5. Rosenzweig, Roy. “Wizards, Bureaucrats, Warriors, and Hackers: Writing the History of the Internet.” *The American Historical Review*, vol. 103, no. 5, 1998, pp. 1530–1552. *JSTOR*, www.jstor.org/stable/2649970. Accessed 12 Nov. 2020.
6. Edwards, Paul N. *The Closed World: Computers and the Politics of Discourse in Cold War America*. MIT Press, 1996.
7. Stallman, Richard. “On Hacking.” *Richard Stallman's Personal Site.*, stallman.org/articles/on-hacking.html.
8. “The Early Hackers.” *Open Sources: Voices from the Open Source Revolution*, by Eric S Raymond, O'Reilly, 1999.
9. Hauben, Michael, and Ronda Hauben. *Netizens: on the History and Impact of Usenet and the Internet*. IEEE Computer Society Press, 1997.

10. “The Evolution of Usenet: the Poor Man’s ARPANET.” *Netizens: on the History and Impact of Usenet and the Internet*, by Michael Hauben and Ronda Hauben, IEEE Computer Society Press, 1997, p. 41.
11. Ouverson, Marlin. “(Computer) Power to the People.” *External Design*, 12 Nov. 2009, www.externaldesign.com/2001-12-05-ddj-editor-computer-power-to-people/.
12. Doub, Bo. “Community Memory: Precedents in Social Media and Movements.” *Computer History Museum*, 23 Feb. 2016, computerhistory.org/blog/community-memory-precedents-in-social-media-and-movements/.
13. “Chapter 2: Copyright of Computer Programs .” *Legal Protection of Digital Information*, by Lee A. Hollaar, Bloomberg BNA, 2016.
14. Stallman, Richard M. “Free Software: Freedom and Cooperation.” *The GNU Operating System*, Free Software Movement, 29 May 2001, www.gnu.org/philosophy/rms-nyu-2001-transcript.txt.
15. Elliott, Steven. “Youtubers Vs. Tradition Media: Trending Tab Edition.” Medium, Medium, 2 Aug. 2019, medium.com/@smonroeelliott/youtubers-vs-tradition-media-trending-tab-edition-70e2610e5fff.
16. “Overview of the GNU System.” *GNU Operating System*, Free Software Foundation, 4 Sept. 2017, www.gnu.org/gnu/gnu-history.en.html.
17. Raymond, Eric S. “Goodbye, ‘Free Software’; Hello, ‘Open Source.’” *The Cathedral and the Bazaar*, 8 Apr. 2015, www.catb.org/~esr/open-source.html.

18. Raymond, Eric Steven. *Cathedral and the Bazaar*. SnowBall Publishing, 2010.
19. Sherman, Alex, and Lora Kolodny. "IBM to Acquire Red Hat in Deal Valued at $34 Billion." *CNBC*, CNBC, 29 Oct. 2018, www.cnbc.com/2018/10/28/ibm-to-acquire-red-hat-in-deal-valued-at-34-billion.html.
20. CB Insights. "Open-Source Software Has Changed The Way Software Is Developed. Here's Where The $33B Industry Is Headed." *CB Insights Research*, CB Insights, 1 July 2020, www.cbinsights.com/research/report/future-open-source/.
21. "The State of the Octoverse." *GitHub*, Microsoft, 15 Nov. 2020, octoverse.github.com/.
22. Lardinois, Frederic, and Ingrid Lunden. "Microsoft Has Acquired GitHub for $7.5B in Stock." *TechCrunch*, 4 June 2018, techcrunch.com/2018/06/04/microsoft-has-acquired-github-for-7-5b-in-microsoft-stock/.
23. Illuminas. "The State of Enterprise Open Source." *Red Hat*, Red Hat, 2020, www.redhat.com/en/enterprise-open-source-report/2020.
24. Wagner, Ben. "Push-Button-Autocracy in Tunisia: Analysing the Role of Internet Infrastructure, Institutions and International Markets in Creating a Tunisian Censorship Regime." *Telecommunications Policy*, vol. 36, no. 6, 2012, pp. 484–492., doi:10.1016/j.telpol.2012.04.007.
25. Alsop, Thomas. "Global Server Share by OS 2018-2019." *Statista*, 13 May 2020, www.statista.com/statistics/915085/global-server-share-by-os/.

DESIGN OF AN ENDPOINT DETECTION AND RESPONSE SYSTEM FOR WINDOWS BASED ON MITRE ATT&CK FRAMEWORK AND DOD STIGS

(Technical Paper)

AN INVESTIGATION INTO THE CULTURE AND POLITICS OF OPEN SOURCE SOFTWARE IN THE CYBERSECURITY COMMUNITY

(STS Paper)

A **Thesis Prospectus Submitted to the**

Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements of the Degree
Bachelor of Science, School of Engineering

Calvin Krist
Spring, 2020

Technical Project Team Members
Calvin Krist
Jack McDowell
Jake Smith
Will Mayes

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature ______________________ Date 11/23/2020
Calvin Krist

Approved ______________________ Date 12/01/2020
Yonghwi Kwon, Department of Computer Science

Approved ______________________ Date 10/20/2020
Toluwalogo B. Odumosu, Department of Engineering and Society

Project Statement

I am developing an open source Endpoint Detection and Response (EDR) for Windows machines that aims to give corporations with large networks of computers increased security and visibility over their networks.

Technical Project Details

Introduction

An EDR application is to an entire network what an antivirus is to a personal computer. They are installed and used by security professionals to help them manage thousands of computers, deploy security policies, and to leverage centralized logging to help prevent, track, and respond to any attacks that occur. They often enable teams of less than ten to manage hundreds or thousands of computers by vastly increasing their productivity and giving them new and powerful security tools.

However, there are no open source EDR applications, and in fact there are very few open source industry-quality defensive applications. EDR applications are generally the highest standard of defensive security, but cost thousands of dollars and use proprietary services and cloud servers. I am working with a team to develop an open source EDR for Windows called BLUESPAWN that aims to counteract this.

Contributions

There are three main contributions that BLUESPAWN makes to the security community:

1. It is open source. There is very little open source defensive tooling, so this provides both a learning opportunity to other professionals and enables them to modify the program to suit their needs.
2. The main threat-hunting capabilities are modeled after an acclaimed model of malicious actors. This provides users with an explicit understanding of how BLUESPAWN works, giving them trust in its capabilities and allows them to manually pursue security BLUESPAWN does not support.
3. It adds the idea of "mitigations", security policies that can be automatically set in order to mitigate the risk of an attack.

Design Methodology

BLUESPAWN will have three components:

1. A **client** that runs on an individual machine. It hunts for malware, applies security policies, and sends logs to a central server.
2. A **local server** that can be used to control all clients on a network and to help security professionals leverage the logs for increased security.
3. A **cloud server** that functions as a database of discovered malware for all BLUESPAWN applications and is accessible to all parties.

At the moment, only the client is in development. The client is based on the MITRE ATT&CK Framework, a robust model of how advanced attackers gain entry to systems, persist despite security policies, and exfiltrate information ("What Is the MITRE ATT&CK

Framework?”). The framework defines categories, such as “Initial Access”, and then lists all possible mechanisms for initial access. BLUESPAWN designs a “Hunt” for each MITRE ATT&CK element which knows that signs of malicious activity look like. Currently, there are 17 hunts, which is less than 10% coverage of the MITRE ATT&CK Framework.

Each hunt also has an associated “monitor” mode. This defines events that can occur on the machine that are signs of likely malicious activity, and when such an event occurs the hunt is then triggered. When BLUESPAWN is in “monitor” mode, it functions much like an antivirus, except it has more methods to identify malicious activity than by matching malicious files to a database.

The final mode of the BLUESPAWN client is “mitigation” mode. This mode is modeled after Department of Defense STIGS, which list and prioritize security policies for Windows computers. For each element in a STIG, a mitigation is implemented that checks if the policy is correctly set, and if not asks a user if they would like it to be fixed. Currently, no other industry security application can do this. There are a variety of open source scripts that can apply defensive policies, but they are not clear about where the policies come from, why they are applied, and many of these scripts break on older versions of Windows machines. BLUESPAWN has been robustly tested on versions as old as Windows 2008, and offers full transparency on where mitigations come from and why they are recommended (“Security Technical Implementation Guides (STIGs).”, 2020).

Early Results

BLUESPAWN has been tested against active, professional hackers at the CCDC competition. Despite being in the early stages of development, it proved very effective at finding malware and assisting security administrators in securing the network.

STS Project Details

One of the main motivators behind BLUESPAWN was the lack of open source, defensive security software, which we felt limited our learning. This is contrasted with the large amount of open source, offensive security software. In fact, many of the most commonly used offensive software platforms are free and open source, allowing individuals to easily learn the basics of offensive security. No similar counterpoint exists for defensive security. More significantly, this often results in defensive software being sold as a service, which severely influences how companies approach their security policies. Thus, I will research if there is a cultural or STS explanation for this discrepancy.

In order to do so, I must look at the cultural and political history of the Internet, open source software, and the security community, and analyze how these cultures have changed over time, influenced different technologies, and been influenced in turn by wider worlds events or technologies. I will primarily use the framework of coproduction to do this analysis, but may also use politics of artifacts.

Coproduction is the idea that scientific ideas and technologies develop simultaneously with how they are represented, discussed, and the political institutions that surround them. In other words, beliefs regarding a technology influence the stakeholders, while the stakeholders in turn influence those beliefs (Jasanoff, “Co-production”). This has high relevance to this STS research question because software and how it is distributed and presented is a very political topic, with many formal institutions that influence the various ideologies.

For example, when discussing “open source” software there are two main beliefs. The “free software” movement is a political ideology that software should be free to modify and repurpose at will, but not necessarily monetarily free. On the other hand, the “open source” movement, which split off from the free software movement in the late 90’s, holds that open source software results in better software, but does not necessarily care about the freedom to modify software. It’s a development methodology, not a political stance (Stallman, 2007).

Both of these movements have organizations with numerous artifacts that exhibit these different viewpoints. For example, the Free Software Foundation and the GNU organization are both closely tied to the free software movement. The GNU organization produces the popular operating system Linux, and while doing so produce numerous articles and blog posts that support the free software movement. On the other hand, RedHat, developers of Ansible and the operating system CentOS, have lots of resources on how to use open source development to create cohesive teams and good programs. None of these resources take a political stance. In fact, the company largely brands itself as a proponent and supporter of open source. Thus, this should be an example that clearly illustrates how the politics of open source software have changes, resulted in new political structures being developed, and in turn those political structures produce new open source artifacts and influence the discourse, showing a clear need for a coproduction analysis.

Politics of artifacts is the idea that artifacts, such as source code, may express and further a political agenda (Winner, 1986). For example, Linux is the crowning achievement of the GNU organization, and is often held up by proponents of open source development as “proof” that the design methodology works. In other words, the Linux operating system becomes a political tool that supports the open source movement. I am not sure how useful this framework will be for my research, if only because I am not sure that there are enough security-related artifacts, and especially not many that are obviously political. Many security professionals have small, toy programs on their GitHub profiles, which may be expressions of hacker culture. These programs are often incredibly technical, but not robust and intended for industry use. However, if such research can be done, then politics of artifacts may help to more directly answer my research question than will coproduction.

There are a number of stakeholders in BLUESPAWN that should be considered while doing this research. The developers are primarily students with a passion for cybersecurity. Due to its use in a competition setting, a number of professional hackers have to test their malware against BLUESPAWN, and are indirect shareholders.

More significantly, BLUESPAWN makes use of a few other open source security projects that achieve useful goals with a more limited scope. These projects can be considered stakeholders in BLUESPAWN, and vice versa: we have made active contributions and changes to these projects to make them work better for BLUESPAWN. In fact, that interaction has direct relevance to the STS research as it highlights some of the capabilities that open source projects have that closed source ones do not, and creates explicit bonds of cooperation between different open source project groups.

There are three main research methods that will be used.

1. Reading journals, books, and other STS literature on coproduction, politics of artifacts, and the free software / open source movements.
2. Collecting empirical research based on GitHub repositories related to cybersecurity, such as how many defensive projects there are, the number of contributors, and their popularity.
3. Interviewing student developers and industry professionals on their opinion of open source, free software, and closed source proprietary software.

STS Research Schedule

The following is a tentative schedule for research starting from the end of the semester, up until completing a first draft of a research paper. This schedule assumes I will do a small amount of work over the summer but not much:

Start Date	End Date	Research Produced
4/5	7/1	Research coproduction
4/5	5/15	Start researching the history of the Internet
4/5	4/15	Start researching the history of free software and open source
4/5	6/1	Start researching the politics of open source software
6/1	8/1	Interview people on open source / free software
8/1	8/15	Conduct a survey of security projects on GitHub
8/1	9/1	First draft of STS research paper

Conclusions

I am interested in learning why there is a discrepancy between the amount and quality of open source offensive and defensive software. Coproduction will be used to analyze the cultural role of open source software, its politics, and how it might relate to the security community, while politics of artifacts might be used to more closely examine the role of artifacts, especially security artifacts, for their political influence and message.

References

Jasanoff, Sheila. “Co-Production.” Sheila Jasanoff, Harvard Kennedy School, sheilajasanoff.org/research/co-production/.

“Security Technical Implementation Guides (STIGs).” DoD Cyber Exchange, DOD, 31 Mar. 2020, public.cyber.mil/stigs/.

Stallman, Richard. “Why Open Source Misses the Point of Free Software.” GNU Operating System, Free Software Foundation, 2007, www.gnu.org/philosophy/open-source-misses-the-point.en.html.

“What Is the MITRE ATT&CK Framework?” Rapid7, Rapid7, www.rapid7.com/fundamentals/mitre-attack/.

Winner, Langdon. “Do Artifacts Have Politics?” The Whale and the Reactor: a Search for Limits in an Age of High Technology, by Langdon Winner, University of Chicago Press, 1986, pp. 19–39.