UNVEILING ADVANCED PERSISTENCE TECHNIQUES THROUGH APPLICATION SHIMMING AND COUNTERMEASURES

Akashdeep Bhardwaj[1],Naresh Kumar[2], and Shawon S. M. Rahman[3]

[1]Professor of Cybersecurity & Digital Forensics, University of Petroleum and Energy Studies, Dehradun, India
[2]Computer Science, DMPS,College of Arts and Sciences, University of Nizwa, Oman
[3]Professor, Department of Computer Science, University of Hawai‘i at Hilo
200 W. Kawili St., Hilo, HI 96720, USA

ABSTRACT

In the arms race between attackers and defenders, the significance of proactive security measures was evident. The implementation of well-considered countermeasures, which may encompass stringent access controls, regular system updates, intrusion detection systems, and behavioral analysis, emerged as vital strategies to thwart the ever-evolving landscape of APTs. Application Shimming is a tool in the Windows Application Compatibility framework that lets programs work on versions of the operating system they weren't originally made for. Due to this architecture, most programs that previously operated on Windows XP can now operate on Windows 10. Shimming takes parts from a Windows Application Compatibility database after parsing it. Shims, which were created for malware investigators, examine any entry that might have been exploited to compromise a Windows system. This research presents a framework that can compromise the target operating system along with the proposed mitigation techniques.

KEYWORDS

APT, Application Shimming, Persistence Attack, Exploit Windows, OS Pen Testing.

1. INTRODUCTION

A key characteristic that has been a part of Windows' core functionalities ever since Microsoft Windows' earliest versions is ‘Backward Compatibility’ [1]. This feature enables the use of software that was created in the past, such as when Windows XP was in use. However, since Windows 10 has been released, developers are concerned about whether Windows will still be able to run such older software. Here is where Backward Compatibility is useful. It enables software that wasn't created for the Windows OS to operate on that OS. Along with performance, stability, and manageability, one of the essential foundations of the development of Microsoft Windows operating systems is application experience and compatibility. Microsoft ensures broad software compatibility, integrating compatibility into the engineering and release process to save deployment costs and speed uptake. One such potent technological solution is the Microsoft Windows Application Compatibility Infrastructure (Shim Infrastructure) [2]. Application Shims were developed to enable backward compatibility, ensuring that programs continued to operate correctly even after modifications to Windows and its APIs [3]. These Shims give programmers the ability to repair apps that were made for earlier Windows versions and guarantee that they will function with the most recent Windows version without having to rewrite the code.

Older apps experience issues as new Windows releases and system API updates are made. Some will not install or run because their code is limited to checking for compatibility with older operating systems, such as Windows 95 [4], and they have no other option except to declare Windows 7 [5] version strings incompatible. Some depend on old paths that aren't available in more recent iterations of Windows, as evidenced by the difference between Windows XP [6] and Windows 7 paths to the user's home directory. On a case-by-case basis, some suppliers may offer firms prolonged support for these outdated products, but the majority become abandonware. Because of this problem, Microsoft developed the Application Compatibility Toolkit (ACT) [7], which the researchers can install to build repair packages for apps. One or more application fixes are compiled into a database. Following that, the fixes are installed on the target system after being read out of the database. The solutions perform a wide range of actions, including deceiving the application about the OS version, rerouting accessible paths, favoring older API functions over more recent ones, and even denying it read/write access to resources that it has access to (which can lead to some fun application sand-boxing scenarios). The application is completely unaware of what is happening.

Application Shims were created to support compatibility problems, guaranteeing that programs continued to run properly even after changes to Windows and its APIs [8]. By using the Shims, programmers can guarantee that older Windows versions will work with newly created programs without having to completely rewrite the code. The Microsoft Windows Application Compatibility Infrastructure, or 'Shim Infrastructure' [9] as they like to call it at the big house, assisted its user in obtaining such backward compatibility. The important thing to remember is that Windows maintained the same fundamental architecture during all those years of development. Microsoft continued to operate within the same framework as when it first began in the early 1990s. This indicates that some portions of the Windows 10 code that date back to Windows 95 are still present.

Application Programming Interface (API) hooking is a technique used by Shim Infrastructure. It explicitly forces linking, forcing Windows' API requests to be forwarded to other code—the Shim—by the linking mechanism. Under the Windows Portable Executable (PE) [10] and Common Object Format (COFF) [11] Specification, the data locations in this header serve as an intermediary layer between the application and the linked file. The Import Address Table is used to make calls to external binary files (IAT). As a result, a call into Windows appears to the system as the picture displayed below, as depicted in Figure 1a. Researchers may replace a reference to a procedure in the equivalent Shim code with the location of the Windows method identified in the import table, as shown in Figure 1b. When the program is loaded, this indirection takes place for files that are statically linked to.dll files. Dynamically linked.dll files can likewise be Shimmed by hooking the 'GetProcAddress' API. Initially, as seen in Figure 2. Even though the program was routed to the Shim before reaching Windows, the software that executes within the Shim still lives outside of Windows. Shim code is consequently subject to the exact security restrictions [34] as application code under Windows. The Shim code really looks like application code to Windows. Therefore, unable to utilize Shims to get around any security features built into Windows[32]. For instance, there isn't a Shim available to execute the program with elevated rights while avoiding the User Account Control (UAC) [12] questions in Windows 7. The user will have to authorize the elevation to gain administrator access with UAC enabled, regardless of whether the Shim program to demand or not demands administrator rights. The same is true for custom-written code.

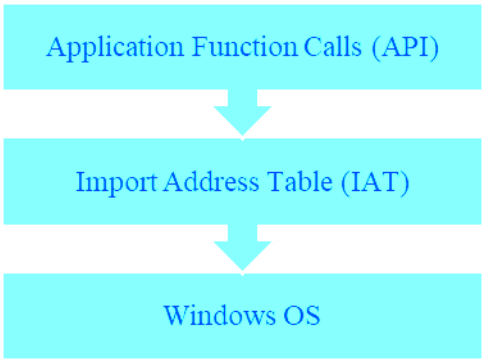


Figure 1a: App Calls reference the OS

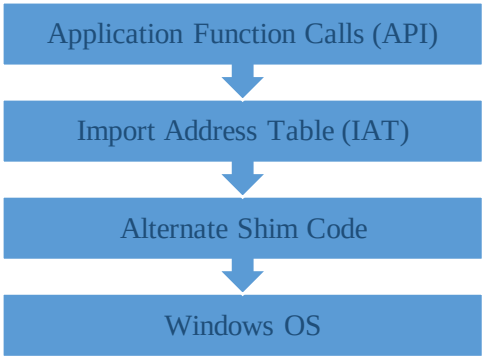


Figure 1b: App redirects to Shim prior to Windows OS

Application Shimming is a malicious technique that targets Microsoft Windows operating systems and makes use of Application Shims to elevate privileges, inject DLLs, and establish persistence, among other things. Shim Database (.sdb) files, which are typically used to resolve software compatibility issues, can be created using the Microsoft Windows Application Compatibility Framework, but they can also be abused for sinister purposes. It is possible to modify the location of the Windows function that is fixed in the import table and then replace it with a pointer to a different function in the alternative Shim code. When the program is loaded, statically linked.dll files cause this indirection. By connecting it with an API, Shim dynamically linked.dll files. Shims work as user-mode code within a user-mode application process; they cannot be used to fix kernel-mode code. For example, compatibility issues with debugging tools or other kernel-mode programs cannot be resolved with Shims. For example, several antispyware, firewalls, and antivirus software programs run in kernel mode [30][33].

Ultimately, the insights gleaned from this study shed light on the intricate interplay between attack techniques and defense strategies within the realm of APTs. As the digital landscape continues to evolve, the knowledge gained from this research serves as a steppingstone towards a more secure and resilient cybersecurity posture, ensuring the confidentiality, integrity, and availability of sensitive information and critical systems.

2. LITERATURE SURVEY

The private and business sectors have now been added to the list of targets for threats that have previously primarily targeted nation-states and the organizations that are linked to them. Each country and well-established institution dread and seeks protection against this category of dangers, often referred to as advanced persistent threats (APTs). The sophistication of nation-sponsored APT assaults will always be a defining characteristic, but the sophistication of APT attacks in the business sectors does not lessen the difficulty for the companies. We hope that this survey paper [13] will help us to learn about all the numerous APT assault stages that may be detected, as well as the threat detection techniques that should be employed to make the architecture smart and impenetrable to APT attackers that can adapt. In a technologically reliant culture, where computer devices and data have been and will remain targets of cyberattacks, especially APT actors, the requirement for cyber resilience is becoming more and more crucial. APT and nation-state attackers frequently have access to substantially greater time and resources to assist their assaults, which are typically not monetarily motivated, in contrast to normal cybercriminals. These actors also tend to be more intelligent. For instance, these threat actors frequently employ a wide variety of physical and/or cyber assault channels and continuously improve their attack strategies [29][33]. The Cyber Kill Chain concept is used in this study to

decompose any complicated assault and pinpoint the pertinent traits of such attempts. To create the taxonomy, the authors thoroughly investigated more than 40 APT operations.

To make attacks more expensive and reduce risks, proactive dynamic defenses offer a comprehensive and all-encompassing security system [28][34]. The dynamic game system proposed in this study may simulate a long-term engagement between a covert attacker and a proactive defense. The multi-stage [14] game of imperfect data, in which each participant has his or her private knowledge that is kept secret from the other players, captures the sneaky and dishonest actions. Each player plays tactically following the views they have acquired as a result of extensive observation and education.

APTs [15] enable the bulk of hacking attacks and disruption. APTs are incredibly resourceful and discreet, and work up until the victim is exploited. APTs [16] aim to inject specific automated spyware into a network or a host so that they may initiate an attack whenever they feel like it based on continuing surveillance. Due to improved, complex attack tactics and encrypted covert contact, the identification of APTs is more challenging.

A new normal is being faced by businesses of all sizes and in all sectors. More so than ever before, adversaries are savvy and relentless. Every network is subject to constant assaults[31]. However, as the first line of protection against modern, cutting-edge assaults, many businesses continue to rely on reactive threat detection and response solutions that are signature-based. This research [17] focuses on the actions and routines of the attackers that, when matched with adversarial profiles rather than only IoCs, can yield better and more durable outcomes. The article offers a novel architecture for behavior-based organized threat hunting that enables quick and reliable malware and threat cleanup on networks and systems.

APT is a sophisticated and focused assault technique that is often planned by a hacking group. An attack on a particular objective often follows a lengthy period of strategic preparation and information gathering. Focus is placed on a certain item, and tailored, precise tactics are employed to infect computers and seize sensitive data. The attack detection technique provided by this study [18] permits early identification of APT attacks.

Different adversary models are used by organizations to evaluate the risk and possible effects of attacks on their systems. Attack graphs show weaknesses and possible attacks an attacker can use to locate and exploit an organization's assets. Attack graphs make it easier to depict attack situations visually and analyze attack pathways algorithmically. MulVAL is an open-source, general framework for building logical attack graphs that have been extensively utilized by both academics and practitioners, who have also added more attack scenarios to it. This study [19] analyzes all the current MulVAL extensions and estimates the coverage of attack scenarios by mapping all MulVAL operations to MITRE ATT&CK [20] approaches.

Due to the low-and-slow attack tactics and frequent exploitation of zero-day flaws, APTs are challenging to identify. The authors introduced a data provenance analysis-based anomaly-based APT detection. From modeling to detection, the model particularly adapts its design to the special traits of APTs. To comprehend long-term behavior as the system develops, this uses a unique modeling method to enhance its detection capabilities. The authors [21] analyzed and demonstrated that the proposed model outperforms the current state-of-the-art APT detection mechanism and accurately detects real-world APT events.

In this study, the authors [22] suggested Advanced Blackholing and Stellar, the system that implements it. By improving the granularity of blackholing, enhanced blackholing improves its scalability while reducing collateral harm. Additionally, Stellar lowers the necessary amount of

collaboration to increase the efficiency of mitigation. The authors tested Stellar's flexibility and efficiency at a sizable IXP that links more than 800 networks, transfers more than 6 Tbps of data per second, and often experiences numerous network attacks. The findings demonstrated that the network attacks, such as DDoS magnification attacks, may be effectively handled while the targeted systems and applications continue to function without interruption.

Huang and Zhu [23] examined and compared two PBNEs with both whole and partial information. The findings highlight the advantages of deception for certain private assailant categories and encourage defenders to employ deception strategies to tip the knowledge imbalance in their favor. The analytical conclusions of our methodology have been supported by numerical data that demonstrate the efficacy of defensive design in effectively mitigating APTs and preventing attacks. A new type of cyber assault called APT has presented a danger to contemporary enterprises. When an APT is discovered, the organization must cope with the APT reaction issue, which entails allocating the response resources that are available to patch her unsecured hosts to lessen her possible loss.

In this work, the authors [24] proposed a unique risk management strategy to handle the APT response issue. An APT attacker might achieve its hostile objective by learning knowledge about a network's architecture and profiting from it financially. Using network traffic is one way to identify a potential APT assault. It is challenging to identify this kind of assault because of the APT attack's long-lasting nature on the network and the possibility that the system would crash owing to the enormous traffic. Therefore, in this research deep learning, Bayesian networks [25], and C5.0 decision trees are utilized to quickly identify and categorize APT assaults on the NSL-KDD data.

3. RESEARCH METHODOLOGY

Application Shimming can perform many functions, but we will be focusing on gaining a persistence shell on the Target System for now. This practical was tested in a lab-controlled environment where we have the configurations set for minimum interference. The actual real-life scenario can differ. Application Shimming has been observed being used by the financially motivated threat group FIN7 [26] (also known as the Carbanak Group) to maintain persistence with the ‘Pillowmint’ malware that targets point of sale (POS) systems. Additionally, ‘ShimRAT’ malware from the alleged Chinese-based threat actor group known as ‘Mofang’ makes use of Application Shimming persistence techniques. This research implemented the application Shimming framework to showcase this attack and mitigation as per the below-described setup and this is presented in Table 1 for other researchers to replicate and perform further research.

Table 1: Attacker & Target Setup

Attacker OS	Kali Linux version 2019.4
Tools	MSFVenom, Metasploit Framework
Target OS	Windows 10 (Build 1909)
Target Tools	Windows Assessment and Deployment Kit (Windows ADK), PuTTY.exe

The authors propose the below-mentioned research steps as illustrated in Figure 2, the framework starts by creating a malicious Dynamic Linking Library and injecting that DLL into a 32-bit binary program which is selected in step three. The infected binary is then installed on the target operating system. Once executed the attacker gains access to the target, this further results in a backdoor shell, which further leads to persistent access. Now the attacker has full control of the target system. Step six proposes and presents anti-Shimming techniques to mitigate such sim-

related attacks on legacy target systems. These steps are further illustrated, and detailed research is presented in the next section.

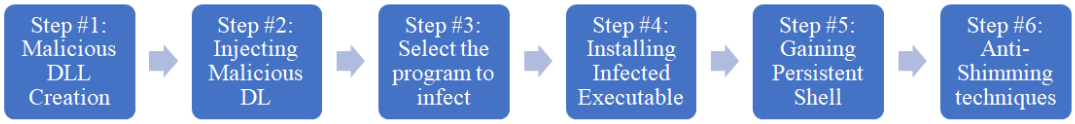


Figure 2: Proposed steps to perform Application Shimming attack

The authors also present the pseudo code to perform the application Shimming used in this research.

- Step 1: Determine the target application to be Shimmed.
- Step 2: Create a Shim database and specify the application to be Shimmed.
- Step 3: Identify the compatibility issue with the target application.
- Step 4: Write a Shim to resolve the compatibility issue.
- Step 5: Test the Shim to ensure it resolves the issue.
- Step 6: Deploy the Shim using the Shim database.
- Step 7: Monitor the target application for any further compatibility issues.

The algorithm for the pseudocode for the application Shimming is as follows:

- Initialize the target application to be Shimmed.
- Create a Shim database and add the target application to it.
- Identify the compatibility issue with the target application.
- Write a Shim to resolve the compatibility issue by intercepting calls or modifying the behavior of the target application.
- Test the Shim by running the target application and verifying that the compatibility issue is resolved.
- Deploy the Shim by adding it to the Shim database.
- Monitor the target application for any further compatibility issues. If any are found, repeat the process starting from step 3.

This algorithm provides a general outline for how application Shimming can be used to resolve compatibility issues in software. The specific implementation details will depend on the compatibility issue and the target application being Shimmed.

4. RESEARCH PERFORMED

When researching the application Shimming, some of the key areas we can focus on include:

- Understanding the concept of application Shimming and its use cases. This includes learning about compatibility issues, Shim databases, and the process of writing and deploying Shims.
- Familiarizing ourselves with different Shimming techniques and tools. There are several methods for application Shimming, including hooking, interception, and modification of executable code. It's also important to learn about the tools and libraries that are available for creating and deploying Shims.

- Researching the history and evolution of application Shimming. This can help understand how Shimming has been used in the past, and how it has evolved to meet new challenges and requirements.
- Learning about the security implications of the application Shimming. Shimming can provide a way for attackers to introduce malicious code into a system, so it's important to understand the security risks involved and how they can be mitigated.
- Studying case studies and real-world examples of application Shimming. This can help determine how Shimming has been used in different contexts, and what challenges and successes have been encountered.

Overall, the goal of our research should be to gain a thorough understanding of the application Shimming, including its benefits, limitations, and best practices, so that anyone can effectively implement it in their research and projects. The proposed application Shimming framework is executed and illustrated in detail in this section.

Step 1: Malicious DLL Creation

A decision was made to use the 'MSFVenom' [27] tool to construct a payload to start the exploitation. To obtain a shell, reverse TCP payload with Windows System as the target is utilized. The target port (LPORT) on which to receive the session from the target system after defining the LHOST for the IP address of the attacker machine. This payload was produced as a Dynamic Link Library, or DLL, with the filename inject.dll.

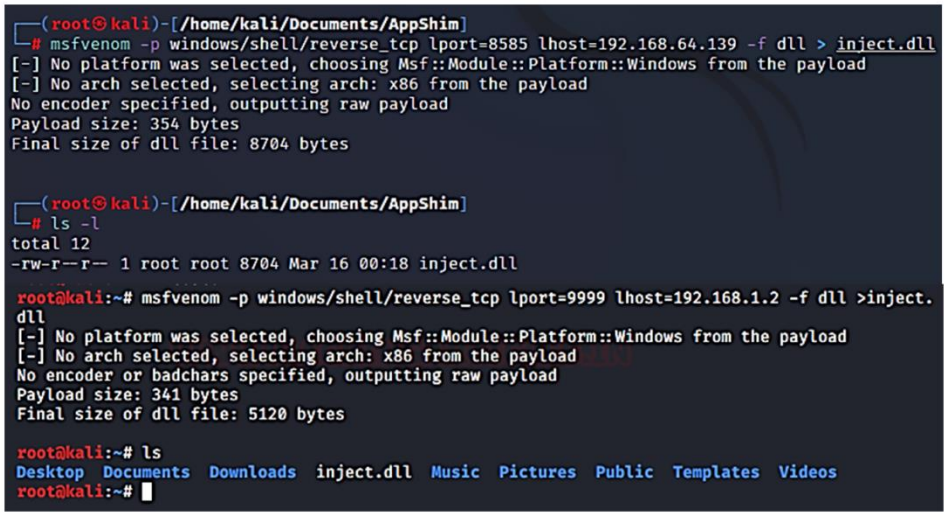


Figure 3: Msfvenom tool to create malicious DLLs (Inject.dll)

The target OS requires Windows Assessment and Deployment Kit which can be downloaded and installed as a service called Compatibility Administrator. From the Attacker Machine, the recently created DLL is transferred to the Target system. Although there are several ways to send the malicious DLL, this research used Simple HTTP Server to launch the malicious file in a website accessed by the tete system as displayed in Figure 4.

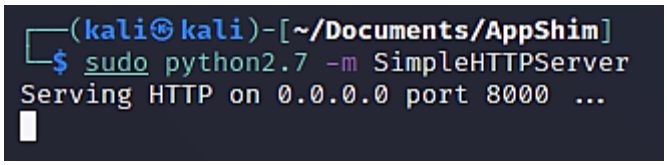


Figure 4: Send process using Simple HTTP Server

The attacker uses an attack tool (Metasploitable framework) and initiates a multi/handler exploit, with target IP and Port 8585, which waits for the DLL to be clicked on the target system as presented in Figure 5.

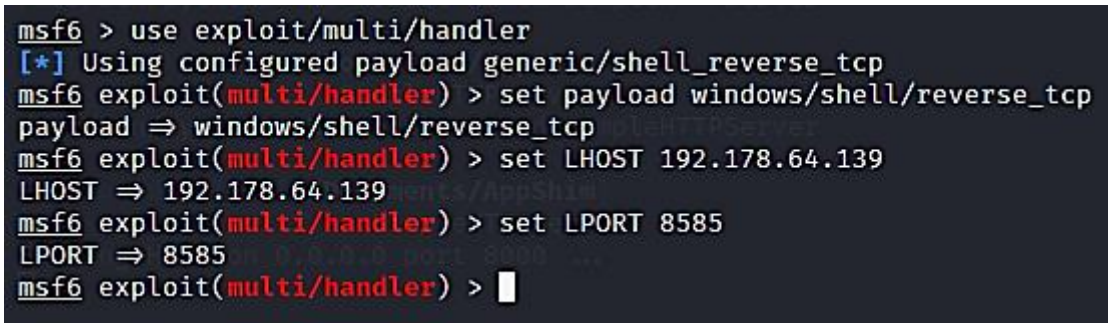


Figure 5: Multi Handler exploit initiated on attacker system

Step 2: Injecting Malicious DL

Now the attention is diverted to the target Windows 10 OS. After browsing the web server running on the Attacker Machine, it auto-downloads the malicious DLL file. In this research, the authors used a 32-bit version as it is easier to bind the DLL to it and created a new custom Database as displayed in Figure 6. The first access to the application fixes is restricted when the application compatibility toolkit is launched. These are the most popular and address several needs for legal adjustments. However, there is an undocumented command line option (/x) that unlocks the remaining patches, giving the 32-bit compatibility toolkit slightly under 900 options to choose from. The management interface enables the creation, management, and updating of Shim databases after it has been launched.

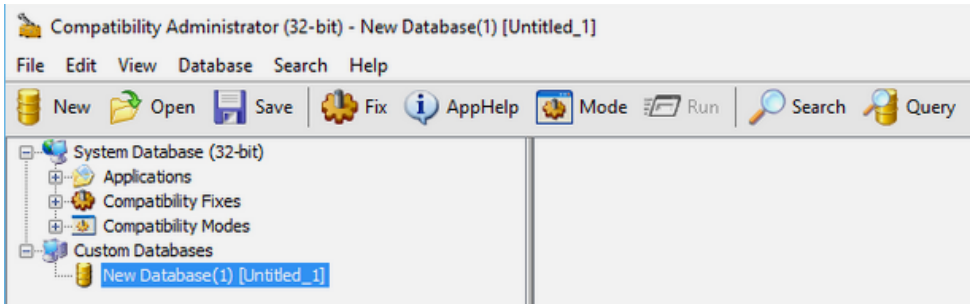


Figure 6: Compatibility Administrator Database

Now the process starts for binding the safe and original binary without malicious DLL file. On the newly created Database – right click to choose the First option in the Dropdown Menu called Create New. This leads to opening a sub-drop-down menu. Application Fix option is chosen and the Config Window Titled 'Create New Application Fix' is obtained. For this research, the

authors used the name of the Program to be fixed as ‘putty’ as shown in Figure 7 and the path of the executable to the program to inject the malicious DLL into.

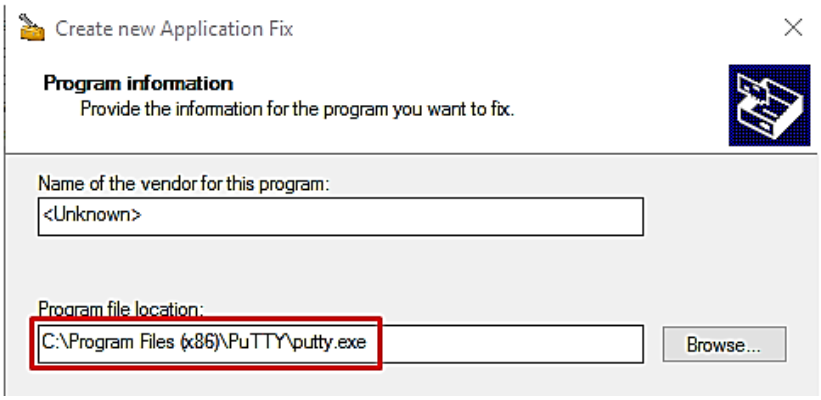


Figure 7: Creating Application Fix with the malicious program

Step 3: Select the program to infect

The unsaved Shim database hits the Fix button at the top of the window with New Database chosen (the default). The prompt for creating a new application repair will then be displayed. Here, some fundamental areas are filled in with details on the patch that is being made. Except for the path of the executable that must be Shimmed, they are filled in with false information. The compatibility modes are presented in Figure 8. If an actual executable had needed fixing, this would have been crucial. or legitimately applying the Shimming. This step can be skipped since this research isn't doing any of that; simply click the "Next" button to continue. On the following screen, there are compatibility modes. These provide ways to make older programs run on more modern hardware by fabricating the environment (OS Version, graphics modes, etc). By selecting next, skip those that are not pertinent to this example.

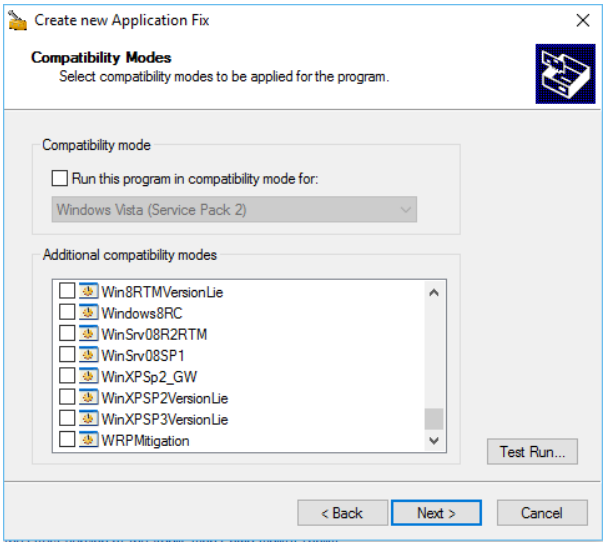


Figure 8: Compatibility Modes

Now compatibility fix which is required to be applied to the executable is done. This research chooses the ‘InjectDll’ option from the list as shown in Figure 9. After checking the box, the

'Parameters' button provides the path of our malicious DLL that had been created at the start of the exploitation.

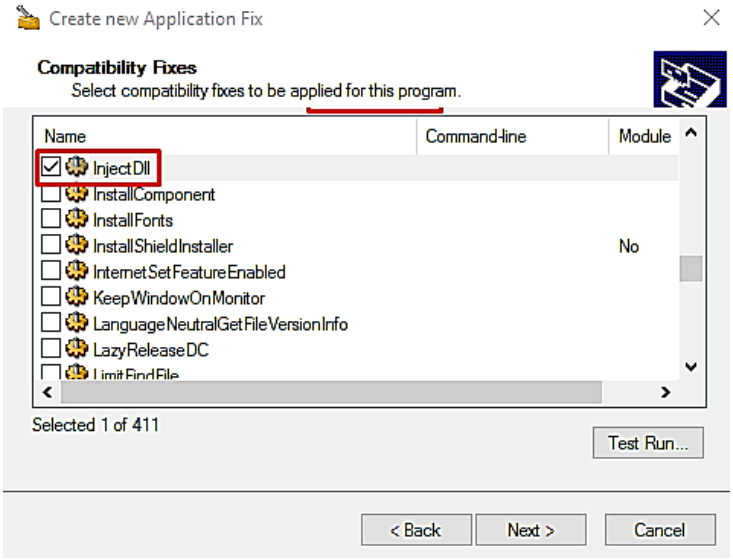


Figure 9: Various Compatibility Fixes

Select and check the box for 'Injectdll', then click the Parameters button to configure the behavior. These fixes do not have customized parameter dialogs, by moving the mouse over the fixed name, the basic description of parameters can be obtained. InjectDLL fix takes a list of paths to DLLs to be loaded during application startup. This research dropped the file on the root of the drive just

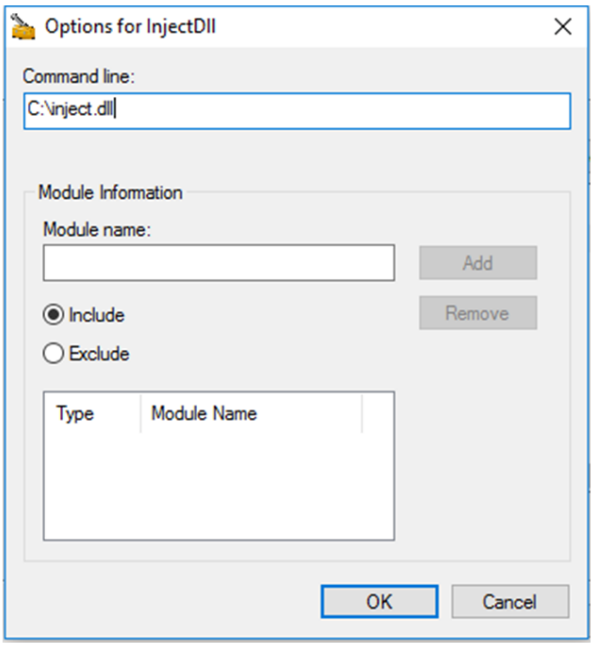


Figure 10: Options for 'Injectdll'

as an easy way as displayed in Figure 10. This opens a new window prompting the Command Line, where the path of the malicious DLL is provided.

On the config window, the Matching Information panel is clicked for the 'Unselect All' Button as no additional configurations are required to out payload, and click the Finish Button as illustrated in Figure 11.

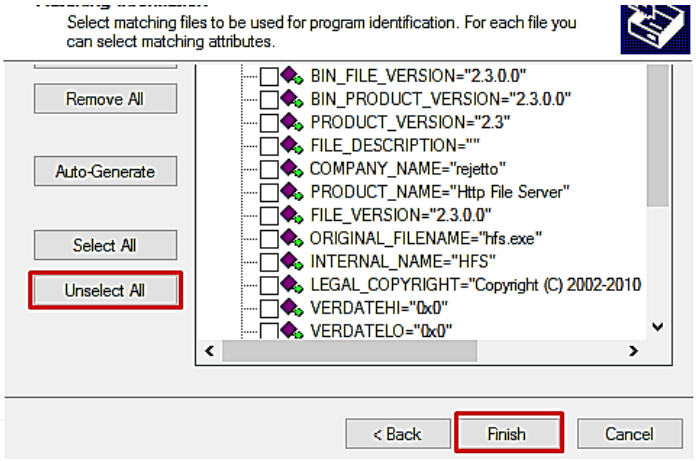


Figure 11: Unselect all configuration options

This closes the config window and back on the Compatibility Administrator window, the Save button is clicked to inject the DLL into the PuTTY executable. Once this process is completed, click Save to save and name the Shim database, which will be a .sdb file as shown in Figure 12. This Shim database is portable and can be dropped and installed on new systems, so long as the parameters that are set are fulfilled. The Shim database is named 'puttyShim'. In real-life attacking situations choose the less conspicuous name. After naming the database, the location is decided to save the AppCompat Database or the .sdb file of the complete configuration.

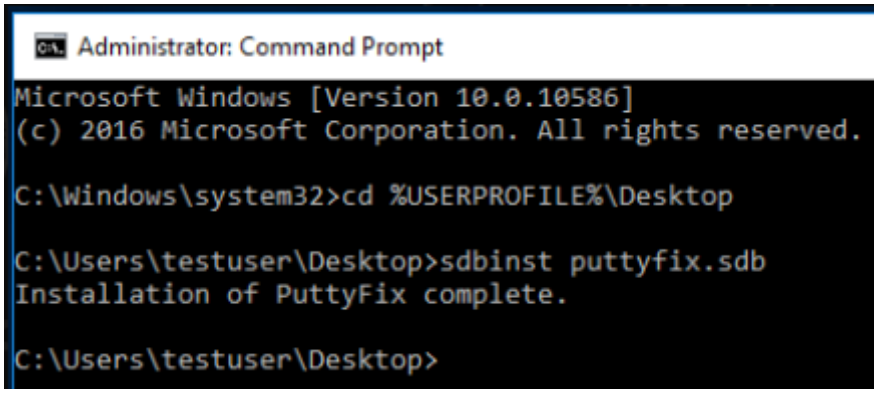


Figure 12: Saving Shim Database

Step 4: Installing Infected Executable

Now that this is done, the now infected executable is installed on the target system as displayed in Figure 13. This can be done by right-clicking on the name of the database and choosing the Install option from the drop-down button. This process initiates an installation process that will

install the infected executable as a service. From the Programs and Features section inside the Control Panel this can be verified.

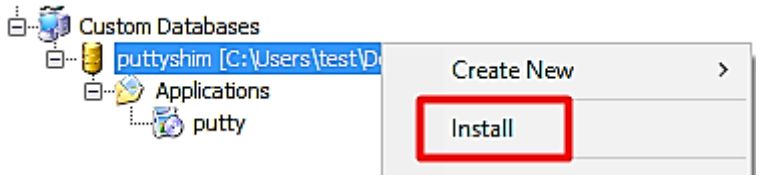


Figure 13: Installing infected executable

Keep in mind that administrative rights are necessary for this to function, therefore UAC should not have been applied. It's not especially difficult to find UAC bypasses. To show the DLL preload, the authors launched Putty as shown in Figure 14. The code must execute (popping the message box). Execution of the programme will resume normally after the DLLMain method completes.

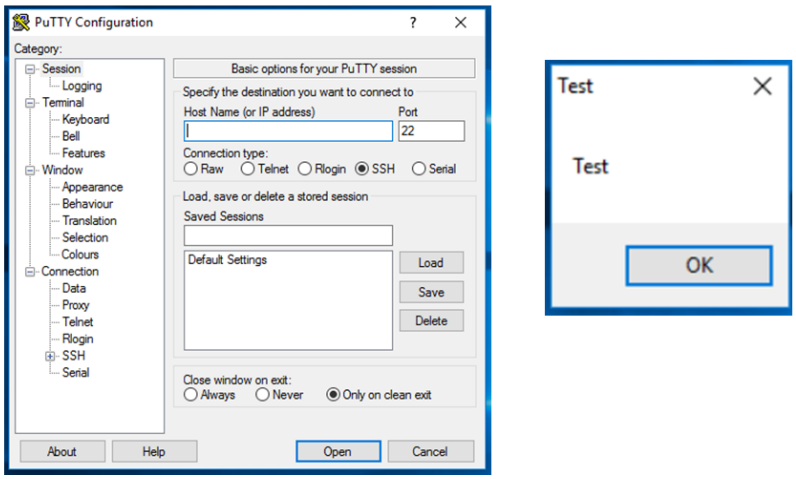


Figure 14: Launching Putty application.

This is a useful method to add to the persistence toolkit. Shim applications that are part of the daily routine for the target (parts of the Microsoft Office Suite, web browsers, etc) for reliable callbacks. This method does have a strong caveat, however, which needs to account for. The Shim name will show up in the list of installed applications on the target system if using the default Shim installation utility as shown in Figure 15.

Step 5: Gaining Persistent Shell

Now when we execute the service that we just Shimmed and installed. As soon as we have the program executed on the target machine, we will receive a shell on our attacker machine as shown in Figure 16 below. We can add the infected service in the startup service list to receive the shell every time the Target system reboots.

Step 6: Anti-Shimming techniques

There are many tools available that can detect the applications that have been Shimmed.

- **Shim-File-Scanner:** Scans Files/Folders for non-default Shims and checks registry for installed Shims
- **Shim-Process-Scanner:** Will search all processes for Shim flags and also check for the Shim App Helper

Other than that the process of Shimming creates a bloody trail that leads right to the smoking gun aka the Shimmed application. Shimming creates a trial inside the Registry at the following locations.

- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Custom HKLM\SOFTWARE\Microsoft\Windows

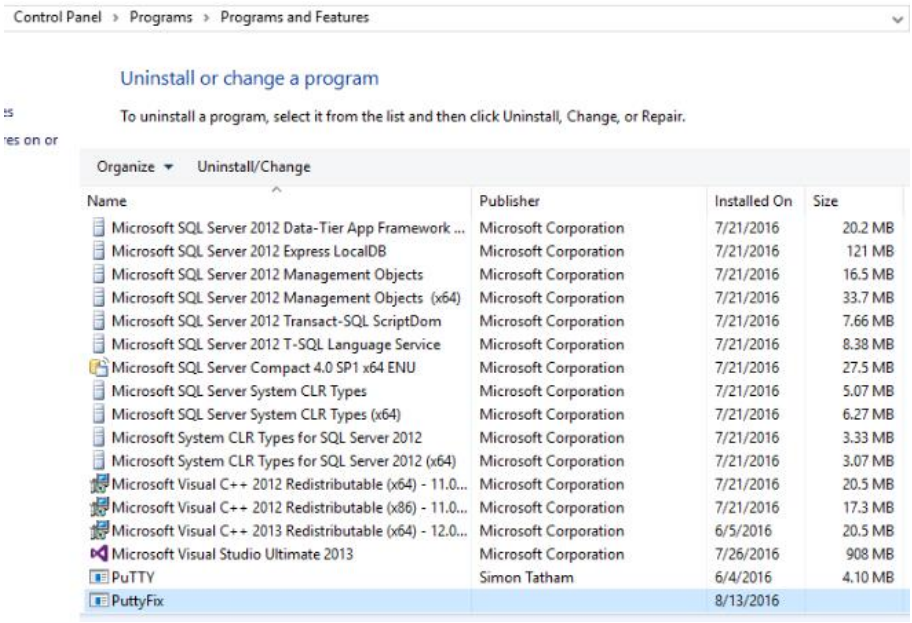


Figure 15: Shim Name displayed in list of apps

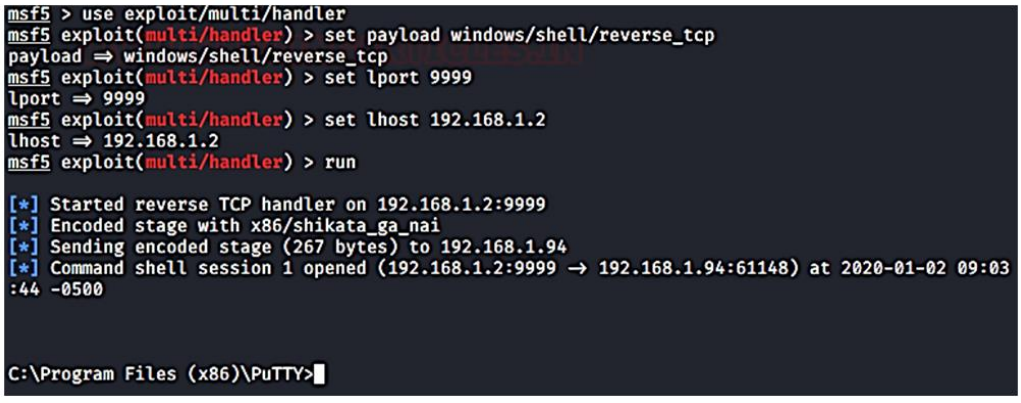


Figure 16: Shell received on attacker system

- NT\CurrentVersion\AppCompatFlags\InstalledSDB

Apart from the registry, we have some locations on the Drives where we can find evidence of the Application Shimming.

- C:\Windows\AppPatch\Custom\
- C:\Windows\AppPatch\Custom\Custom64\

We can also create custom Yara Rules and snort rules that could detect Application Shimming. There are many tools available that can detect the applications that have been Shimmed.

- **Shim-File-Scanner:** Scans Files/Folders for non-default Shims and checks registry for installed Shims
- **Shim-Process-Scanner:** Will search all processes for Shim flags and check for the Shim App Helper

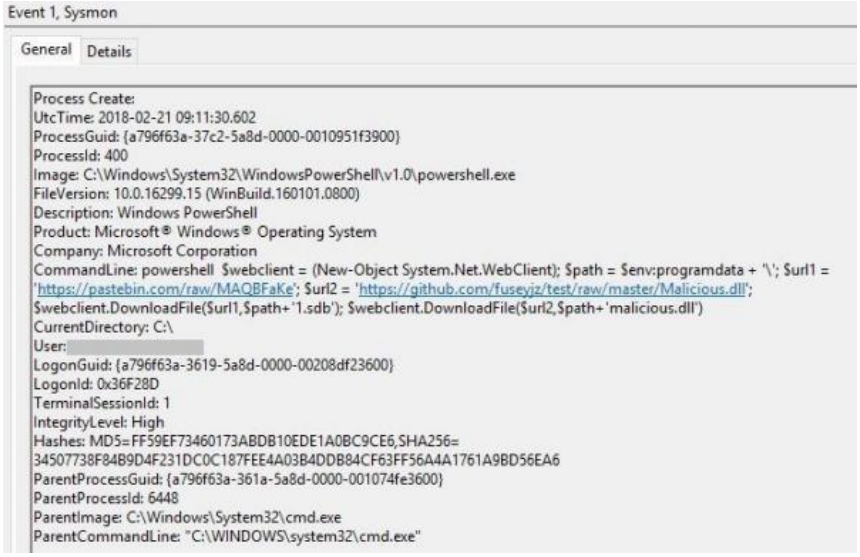
Other than that, the process of Shimming creates a bloody trail that leads right to the smoking gun aka the Shimmed application. Shimming creates a trial inside the Registry at the following locations.

- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Custom
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\InstalledSDB

Apart from the registry, we have some locations on the Drives where we can find evidence of the Application Shimming.

- C:\Windows\AppPatch\Custom\
- C:\Windows\AppPatch\Custom\Custom64\

We can also create custom Yara Rules and snort rules that could detect Application Shimming. If PS Logging is enabled, the first PowerShell command is logged with Event ID 1, and the second PS command is logged with Event ID 4104, using Sysmon to inspect events reported for this activity.



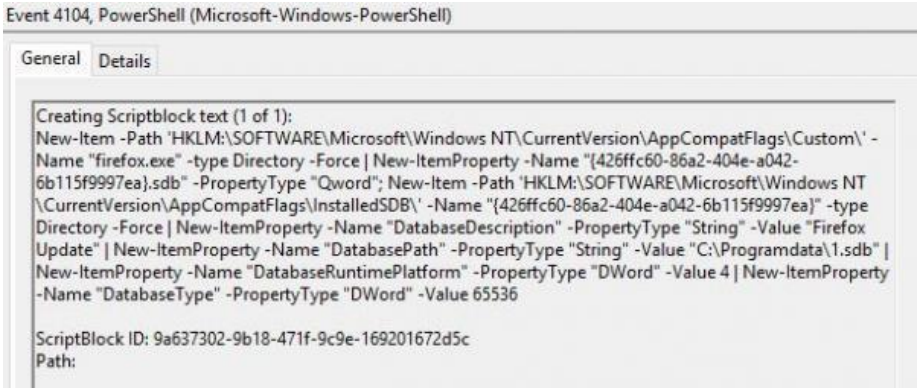


Figure 17: Anti-Shimming PowerShell techniques

The strange Firefox process that loaded our DLL and performed the payload - the calculator is displayed in Figure 17.

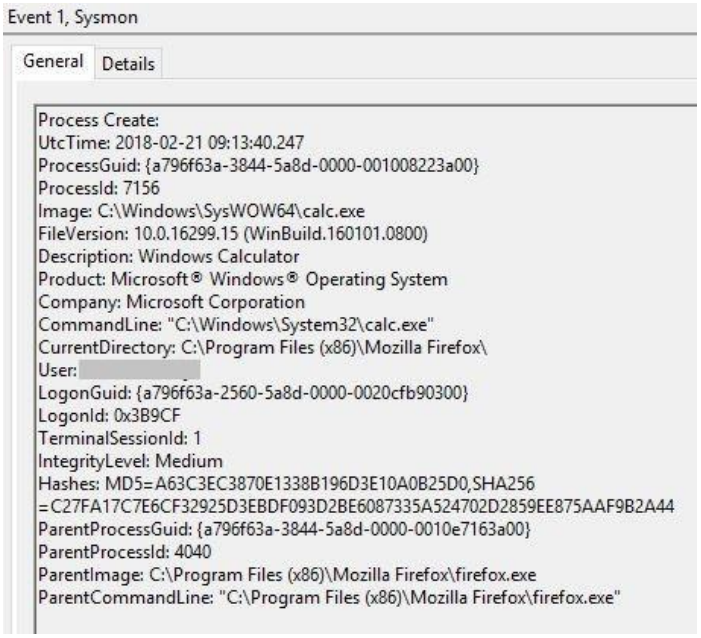


Figure 18: Malicious process loaded by Firefox

We observe an event like the one below if the attacker used sdbinst.exe to carry out the installation as displayed in Figure 18. An entry for the outbound connection made by PowerShell is also visible in the network connection event log.

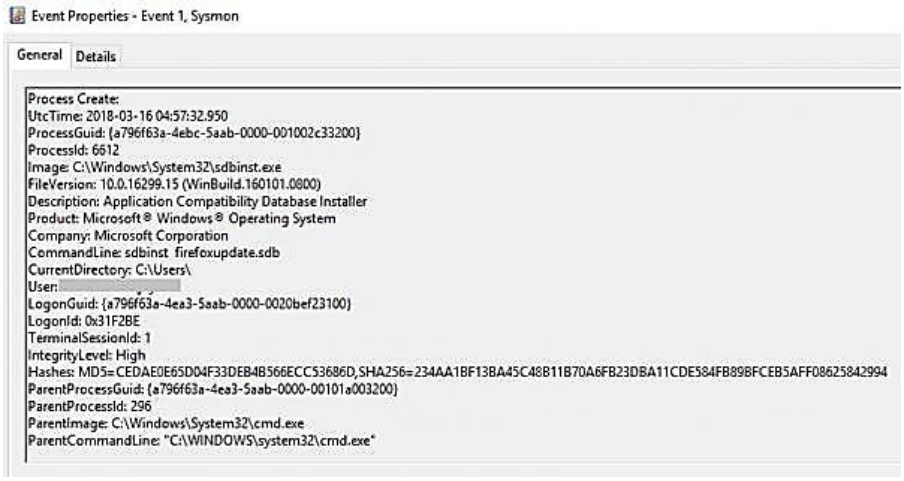


Figure 19: Attacker's process tool sdbinst.exe

The detection of abnormal DLL load events is another method of detection in this case. Now observe Firefox loading our malicious.dll in this situation. A great technique to identify suspicious DLL loading may be to use the path, name, hash, or a mix of these. The following registry locations will always have entries made if a customized Shim is installed:
HKLM → SOFTWARE → Microsoft → Windows NT → CurrentVersion →AppCompatFlags→ Custom
HKLM → SOFTWARE → Microsoft → Windows NT → CurrentVersion →AppCompatFlags→InstalledSDB

The 'DatabasePath' value from the '..InstalledSDB' registry entry contains the location of the custom SDB file. Registry key creation may be detected by Sysmon and will be noted with Event ID 12 as illustrated in Figure 19.

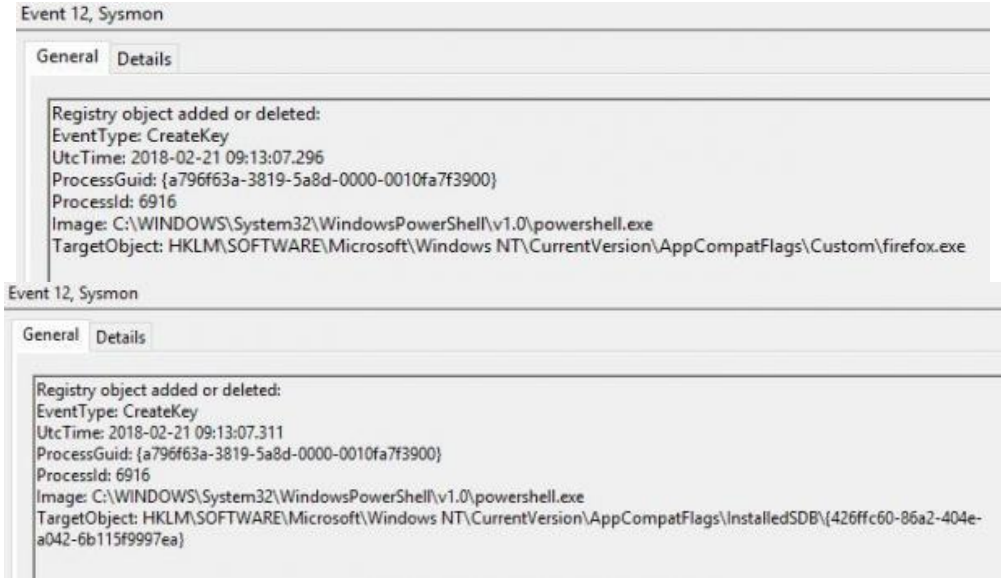


Figure 20: Registry key creation detected by Sysmon

These keys may be retrieved at scale using PowerShell, and DatabasePath's least frequency analysis can be used to identify suspicious sdb files. Using a program like "python-sdb," it is

feasible to extract the data from the SDB file for analytical purposes. By using InjectDll and referring to malicious.dll in the%PROGRAMDATA% directory from the output as presented in Figure 20 below.

It is important to note that several sorts of Shim may be employed. Our malicious DLL was injected into a 32-bit process using InjectDLL, but several real-world threats, like BlackEnergy, Gootkit, and Dridex, have utilized "RedirectEXE" to launch different processes in their place.

Additionally, the FIN7 APT group has utilized Shims in the past, which, when activated, would patch the Services Control Manager ("services.exe") process in order to run a secondary payload saved in the registry. This topic is covered in greater detail here.

```
<DATABASE>
  <TIME type='integer'>0x1d144275b804000</TIME>
  <COMPILER_VERSION type='stringref'>3.0.0.9</COMPILER_VERSION>
  <NAME type='stringref'>Firefox Update</NAME>
  <OS_PLATFORM type='integer'>0x1</OS_PLATFORM>
  <RUNTIME_PLATFORM type='integer'>0x25</RUNTIME_PLATFORM>
  <DATABASE_ID type='guid'>426ffc60-86a2-404e-a042-6b115f9997ea</DATABASE_ID>
  <LIBRARY>
  </LIBRARY>
  <EXE>
    <NAME type='stringref'>firefox.exe</NAME>
    <APP_NAME type='stringref'>Firefox</APP_NAME>
    <VENDOR type='stringref'>Firefox</VENDOR>
    <EXE_ID type='hex'>a560db14-20b3-4072-a86c-998a0321cd0b</EXE_ID>
    <APP_ID type='hex'>ce351825-4d10-4852-838c-3ffbfe35e74a</APP_ID>
    <MATCHING_FILE>
      <NAME type='stringref'>*</NAME>
      <COMPANY_NAME type='stringref'>Mozilla Corporation</COMPANY_NAME>
      <PRODUCT_NAME type='stringref'>Firefox</PRODUCT_NAME>
      <PRODUCT_VERSION type='stringref'>58.0.2</PRODUCT_VERSION>
      <BIN_FILE_VERSION type='integer'>0x3a0000000219d3</BIN_FILE_VERSION>
      <BIN_PRODUCT_VERSION type='integer'>0x3a000000020000</BIN_PRODUCT_VERSION>
      <FILE_VERSION type='stringref'>58.0.2</FILE_VERSION>
    </MATCHING_FILE>
    <SHIM_REF>
      <NAME type='stringref'>InjectDll</NAME>
      <COMMAND_LINE type='stringref'>C:\Programdata\malicious.dll</COMMAND_LINE>
    </SHIM_REF>
  </EXE>
</DATABASE>
```

Figure 21: Ppython-sdb program output

5. CONCLUSIONS

This research paper delved into the realm of cybersecurity by investigating Advanced Persistence Techniques (APT) through the innovative use of Application Shimming, while simultaneously exploring effective countermeasures to mitigate such threats. The experimental environment, featuring Kali Linux as the attacker and Windows 10 as the target, provided a comprehensive platform to analyze the intricate dynamics between attack and defense. Using tools such as MSFVenom, Metasploit Framework, Windows ADK, and PuTTY.exe, this study unveiled the potential vulnerabilities within modern computing systems. The findings underscore the critical importance of vigilance in safeguarding against APTs, as the demonstrated techniques highlighted the ability of malicious actors to exploit hidden pathways within applications for persistent unauthorized access.

Furthermore, this research underscored the necessity of collaboration between cybersecurity professionals and application developers to enhance the robustness of software against potential exploitation. By embracing a holistic approach that encompasses secure coding practices, threat

modeling, and continuous vulnerability assessments, organizations can effectively fortify their defenses and preemptively address vulnerabilities that could be leveraged by APT actors.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] "Microsoft Excel | Microsoft Wiki | Fandom." https://microsoft.fandom.com/wiki/Microsoft_Excel (accessed: Aug. 16, 2023).

[2] "Demystifying Shims - or - Using the App Compat Toolkit to make .." https://techcommunity.microsoft.com/t5/ask-the-performance-team/demystifying-Shims-or-using-the-app-compat-toolkit-to-make-your/ba-p/374947 (accessed: Aug. 07, 2023).

[3] "Quick introduction to Windows API, A. M. Steane." https://users.physics.ox.ac.uk/~Steane/cpp_help/winapi_intro.htm (accessed: Jul. 07, 2023).

[4] "Windows 95 - NETWORK ENCYCLOPEDIA." https://networkencyclopedia.com/windows-95/ (accessed: Jul. 16, 2023).

[5] "Windows 7: Getting Started with Windows 7." https://edu.gcfglobal.org/en/windows7/getting-started-with-windows-7/1/(accessed: Jul. 07, 2023).

[6] "Windows XP: The Windows XP Desktop." https://edu.gcfglobal.org/en/windowsxp/the-windows-xp-desktop/1/ (accessed: Jul. 25, 2023).

[7] "Application Compatibility Toolkit (ACT) - Win32 apps | Microsoft Learn." https://learn.microsoft.com/en-us/windows/win32/win7appqual/application-compatibility-toolkit--act- (accessed: Jul. 07, 2023).

[8] "Hunting for Application Shim Databases - F-Secure Blog." https://blog.f-secure.com/hunting-for-application-Shim-databases/ (accessed: Jul. 16, 2023).

[9] "About Microsoft Windows Application Compatibility Infrastructure .." https://docs.flexera.com/adminstudio2019/Content/helplibrary/ASAppCompatInfrastructure.htm (accessed: Jun. 07, 2023).

[10] "PE Format - Win32 apps | Microsoft Learn." https://learn.microsoft.com/en-us/windows/win32/debug/pe-format (accessed: Jun. 25, 2023).

[11] "Common Object File Format (COFF)." http://osr507doc.sco.com/en/topics/COFF.html

[12] "How User Account Control works - Windows Security | Microsoft Learn." https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/how-it-works(accessed: Jun. 16, 2023).

[13] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities," IEEE Communications Surveys and Tutorials, vol. 21, no. 2, pp. 1851–1877, Apr. 2019, doi: 10.1109/COMST.2019.2891891.

[14] L. Huang and Q. Zhu, "A dynamic games approach to proactive defense strategies against Advanced Persistent Threats in cyber-physical systems," Comput Secur, vol. 89, p. 101660, Feb. 2020, doi: 10.1016/J.COSE.2019.101660.

[15] K. Kaushik, S. Tayal, A. Bhardwaj, and M. Kumar, "Advanced Smart Computing Technologies in Cybersecurity and Forensics," Advanced Smart Computing Technologies in Cybersecurity and Forensics, Nov. 2021, doi: 10.1201/9781003140023/Advanced-smart-computing-technologies-cybersecurity-forensics-keshav-kaushik-shubham-tayal-akashdeep-bhardwaj-manoj-kumar.

[16] S. SibiChakkaravarthy, D. Sangeetha, and V. Vaidehi, "A Survey on malware analysis and mitigation techniques," Comput Sci Rev, vol. 32, pp. 1–23, May 2019, doi: 10.1016/J.COSREV.2019.01.002.

[17] A. Bhardwaj, K. Kaushik, A. Alomari, A. Alsirhani, M. Mujib Alshahrani, and S. Bharany, "BTH: Behavior-Based Structured Threat Hunting Framework to Analyze and Detect Advanced Adversaries," Electronics (Basel), vol. 11, no. 19, p. 2992, Sep. 2022, doi: 10.3390/ELECTRONICS11192992.

[18] W. L. Chu, C. J. Lin, and K. N. Chang, "Detection and Classification of Advanced Persistent Threats and Attacks Using the Support Vector Machine," Applied Sciences 2019, Vol. 9, Page 4579,

vol. 9, no. 21, p. 4579, Oct. 2019, doi: 10.3390/APP9214579.

[19] D. Tayouri, N. Baum, A. Shabtai, and R. Puzis, "A Survey of MulVAL Extensions and Their Attack Scenarios Coverage," Aug. 2022, doi: 10.48550/arxiv.2208.05750.

[20] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats," Jan. 2020, doi: 10.14722/ndss.2020.24046.

[21] "MITRE ATT&CK®." https://attack.mitre.org/ (accessed: May 07, 2023).

[22] C. Dietzel, M. Wichtlhuber, G. Smaragdakis, and A. Feldmann, "Stellar: Network Attack Mitigation using Advanced Blackholing," CoNEXT 2018 - Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, pp. 152–164, Dec. 2018, doi: 10.1145/3281411.3281413.

[23] L. Huang and Q. Zhu, "Analysis and computation of adaptive defense strategies against advanced persistent threats for cyber-physical systems," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11199 LNCS, pp. 205–226, 2018

[24] L. X. Yang, P. Li, X. Yang, and Y. Y. Tang, "A Risk Management Approach to Defending Against the Advanced Persistent Threat," IEEE Trans Dependable Secure Comput, vol. 17, no. 6, pp. 1163–1172, Nov. 2020, doi: 10.1109/TDSC.2018.2858786.

[25] J. H. Joloudari, M. Haderbadi, A. Mashmool, M. Ghasemigol, S. S. Band, and A. Mosavi, "Early detection of the advanced persistent threat attack using performance analysis of deep learning," IEEE Access, vol. 8, pp. 186125–186137, 2020, doi: 10.1109/ACCESS.2020.3029202.

[26] "FIN7, GOLD NIAGARA, ITG14, Carbon Spider, Group G0046 .." https://attack.mitre.org/groups/G0046/ (accessed: Jun. 25, 2023).

[27] "MSFvenom - Metasploit Unleashed." https://www.offsec.com/metasploit-unleashed/msfvenom/

[28] Porche, Joshua and Rahman, Shawon S. M., "Security Culture, Top Management, and Training on Security Effectiveness: A Correlational Study with CISSP Participants", International Journal of Computer Networks & Communications (IJCNC), Vol.15, No.2, March 2023, DOI: 10.5121/ijcnc.2023.15205, https://aircconline.com/ijcnc/V15N2/15223cnc05.pdf

[29] Phibbs, Christina L., and Rahman, Shawon S. M. 2022. "A Synopsis of "The Impact of Motivation, Price, and Habit on Intention to Use IoT-Enabled Technology: A Correlational Study", Journal of Cybersecurity and Privacy, Special Issue Cyber-Physical Security for Critical Infrastructures, 2, no. 3: 662-699. https://doi.org/10.3390/jcp2030034

[30] Roberts, Gerrianne and Rahman, Shawon S. M., "Does Digital Native Status Impact End-User Antivirus Usage?" (May 19, 2021). International Journal of Computer Networks & Communications (IJCNC), Vol.13, No.2, March 2021, DOI: 10.5121/ijcnc.2021.13207, https://ijcnc.com/2021/04/15/ijcnc-07-16/

[31] Vaishnavi D., Mahalakshmi D., Shawon Rahman M.S. "Digital Image Forgery Detection Using Ternary Pattern and ELM"; In: Jeena Jacob I., Kolandapalayam Shanmugam S., Piramuthu S., Falkowski-Gilski P. (eds) Data Intelligence and Cognitive Informatics. Algorithms for Intelligent Systems. Springer, Singapore. pp 77-86, 2021https://doi.org/10.1007/978-981-15-8530-2_5

[32] Loukaka, Alain and Rahman, Shawon; "Security Professionals Must Reinforce Detect Attacks to Avoid Unauthorized Data Exposure"; International Journal of Information Technology in Industry (ITII), Web of Science (Thomson Reuters) Indexed Journal, vol. 8, no.1, 2020 https://doi.org/10.17762/itii.v8i1.76

[33] Dharmalingam, Vaishnavi and Rahman, Shawon; "Towards Cloud of Things from Internet of Things"; International Journal of Engineering and Technology, Vol. 7, No 4.6, 2018, Pages: 112-116

[34] Schneider, Marvin and Rahman, Shawon; "Identifying Protection Motivation Theory Factors that Influence Smartphone Security Measures"; IEEE BigData 2021, Workshop on Big Data for CyberSecurity (BigCyber- 2021), December 15-18 2021, DOI: 10.1109/BigData52589.2021.9671882 https://it-in-industry.com/itii_papers/2021/9121itii01.pdf

AUTHORS

Akashdeep Bhardwaj: Dr. Akashdeep Bhardwaj is working as Professor (Cyber Security & Digital Forensics) and Head of Cybrsecurity Center of Excellence at University of Petroleum & Energy Studies (UPES), Dehradun, India. An eminent IT Industry expert with over 27 years of experience in areas such as Cybersecurity, Digital Forensics and IT Management Operations, Dr. Akashdeep mentors graduate, masters and doctoral students and leads several IT Security projects. Dr. Akashdeep has Post-Doctoral from Majmaah University, Saudi Arabia, Ph.D. in Computer Science, Post Graduate Diploma in Management (equivalent to MBA), and an Engineering Degree in Computer Science. Dr. Akashdeep has published over 120 research publications (including research papers, patent, copyrights, authored & edited books/chapters) in international journals. Dr. Akashdeep worked as Technology Leader for various multinational organizations during his time in the IT industry. Dr. Akashdeep is certified in Cybersecurity, Complance Audits, Information Security, Microsoft, Cisco and VMware technologies.

Naresh Kumar Dr Kumar is an assistant professor at Computer Science, DMPS, University of Nizwa, Oman. He received his Master's and Ph.D. degrees from the Indian Institute of Technology India in 2012 and 2019 respectively. He holds 7+ years experience of teaching Graduate, Master's, and Ph.D. students at various reputed universities around wide globe. He has researched for 10 years and visited reputed international organizations (NTU Singapore, ECE Paris) to deliver presentations on his research domain. In his master's degree, he achieved a thorough knowledge of Computer Science and its applications. He has a good knowledge of applied Mathematics. His Ph.D. area was focused on Computer Vision and Artificial Intelligence Problems. He published in Q1 and Q2 journals in Elsevier and Springer. He has been a reviewer in International journals and IEEE Transactions and delivered expert lectures at many international conferences and seminars. His research area belongs to Computer Vision, Artificial Intelligence, Data Science, and Video and Image analysis with various spectrums.

Shawon Rahman: Dr. Shawon Rahman is a tenured Professor in the Department of Computer Science and Engineering at the University of Hawaii-Hilo (UH Hilo). Dr. Rahman has over 17 years of teaching experience (bot h in-class and online format)in undergraduate and graduate courses. He has published over 130 peer-reviewed papers and conducted numerous professional activities in conferences and journals. Dr. Rahman has published over 70 peer-reviewed papers (including over 40 journal articles) in the area of Cybersecurity and related domains. He has awarded and managed several federal and other grants including NSF, USDA, DOE, etc. Dr. Rahman's research interests include Cybersecurity, Information Assurance and security, STEM Outreach, Digital Forensics, Cloud Computing, Internet of Things Security, Web Accessibility, Robotics, Software Testing, and Quality Assurance. He has served as the dissertation chair and main supervisor, produced 15 Ph.D. graduates, and mentored numerous students. He is serving a few journals in the capacity of editor-in-chief, editor, guest editor, or associate editor. Dr. Rahman has served many conferences as the program chair, local chair, or session chair. He is a member of many professional organizations including IEEE (senior member), ACM, CSTA, ASEE, ASQ, ISCA, and ISACA. For more info please visit https://hilo.hawaii.edu/faculty/rahman/