BARE KNUCKLED ANTIVIRUS BREAKING

Bálint Varga-Perke
Silent Signal Kft.

E-mail: vpbalint@silentsignal.eu
Web: www.silentsignal.eu
Budapest, 2018.01.08.

Contents

Background

Endpoint security software is the most basic element of virtually every defensive system from home users to large enterprises. In order to have a detailed view on the protected system and to be able to counter malware appearing in the context of any system component these products require unrestricted privileges to function properly. This of course introduces new risks as it has been demonstrated numerous times in the past[1].

In modern Windows systems privileged features are generally implemented in kernel drivers and user-mode services running with high privileges (usually SYSTEM). Additionally, low-privileged processes provide user interfaces through which malware scans can be initiated, the software can be configured, etc. Since these user interfaces usually allow to disable software features and hide running malware different "self-defense" (or "tamper protection") mechanisms are implemented on in parallel with standard OS security features that prevent interference even with the low-privileged components.

In this research we demonstrate a self-defense bypass method that can be used against endpoint security products of multiple vendors. Next, we demonstrate that self-defense may hide exploitable attack surface that allows local privilege escalation.

During the research the following products were tested:

- Kaspersky Labs
  - **Kaspersky Free (18.0.0.405)**
  - **Kaspersky Antivirus (17.0.0.611)**
  - Kaspersky Endpoint Security (10.3.0.6294)
- Symantec
  - **Symantec Norton Security Deluxe (22.10.1.10)**
  - **Symantec Endpoint Protection (14.0.3752.1000.105)**
- Bitdefender
  - **Bitdefender Antivirus Plus 2018 (22.0.8.99[2])**
  - **Bitdefender Gravityzone (Endpoint Security Tools 6.2.25.953)**
- Comodo
  - Comodo Internet Security Premium (10.0.1.6258)
- Trend Micro
  - Trend Micro Maximum Security (22.10.1.10)

Generic self-defense bypass was demonstrated against all of the above products. Local privilege escalation exploits were demonstrated against products set in bold[3]. The described techniques may be applicable to a wider range of products.

---

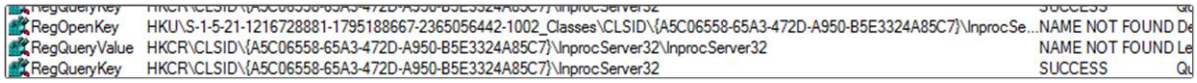
[1] https://www.google.hu/search?q=site:https://bugs.chromium.org/p/project-zero/+"antivirus"
http://joxeankoret.com/download/breaking_av_software_44con.pdf
https://cansecwest.com/csw08/csw08-alvarez.pdf
http://www.securityfocus.com/news/11426

[2] File version of bdusers.dll

[3] Trend Micro was also found to be vulnerable by an independent researcher, see Quarantine Restoration

Self-Defense

The starting point of this research was a self-defense bypass technique we discovered while analyzing the COM hijacking technique as demonstrated by James Forshaw to exploit Oracle VirtualBox for privilege escalation[4]. In short, the technique builds on the fact that while COM objects are usually registered in the HKLM registry tree, users can register alternative DLL's in their respective HKCU tree. These user-specific registrations take precedence in the COM loading process, the lookup looks like this in Process Monitor:



First, the CLSID is looked for in the HKU subtree corresponding to the process owner, if not found, Windows proceeds to the HKCR tree where in this case the object is registered. Since users are allowed to write to their corresponding HKU subtree applications can be forced to load user-specified libraries.

In case of the CVE-2017-3563 VirtualBox vulnerability it was also demonstrated that signature checks of the loaded DLL can be bypassed by loading the Microsoft signed `scrobj.dll` library that allows the attacker to invoke dynamic, unsandboxed JScript and eventually .NET code. With this code the memory of the hijacked process can be arbitrarily manipulated, API calls can be made etc.

What should be noted is that the above technique is especially useful in cases when a low-privileged process is granted special rights to control high-privileged system components. As we saw in the previous section, endpoint security software operates in a really similar architecture. The first question is: can this technique be applied to endpoint security software, too?

Our tests showed that the proof-of-concept code published by James Forshaw can be used without modification (after replacing the CLSID's to be hijacked) to achieve code injection to the user-level components of all tested products, with self-defense enabled. The tests also showed that only the antivirus engine of Kaspersky Labs detected our injected scriptlets (.SCT files - code injection proof-of-concepts and full privilege escalation exploits) as potentially malicious. In case of Kaspersky products, detection could be avoided by providing a http:// URL in the registry, and serving the file from a remote network location.

The second question is whether the ability to execute code in the UI processes of the target products allows access to functionality that lets an attacker cross the security boundaries of the operating system. In case of VirtualBox, the ability of the user process to access kernel memory via a driver is an obvious attack surface, as it is documented by the developers[5]. While in some cases self-defense bypasses in themselves are considered as security issues (e.g. CVE-2017-6331), the potential for OS privilege escalation is less obvious in case of endpoint security software. In the following section, we will explore common software functionality that could allow such attacks.

[4] https://googleprojectzero.blogspot.hu/2017/08/bypassing-virtualbox-process-hardening.html
https://bugs.chromium.org/p/project-zero/issues/detail?id=1103
[5] https://www.virtualbox.org/browser/vbox/trunk/src/VBox/HostDrivers/Support/SUPR3HardenedMain.cpp#L39

Crossing Privilege Boundaries

In this section, we explore features generally available in endpoint security products that may be useful to cross privilege boundaries of the operating system and achieve privilege escalation. While configuration changes that allow easier deployment of malware may be practically useful, we are not covering these methods as they are mostly trivial.

For the feature to be abusable the following Requirements must meet:

R1. The feature should be available to the attacker. (Access)

R2. The action executed through the feature should be performed with high user privileges. (Elevation)

R3. The attacker should be able to manipulate the subject of the action. (Control)

Our research focused on cases where Access or Control could be gained after self-defense is bypassed. We will see that it is possible to circumvent important security measures by only manipulating processes running under our own user account. We will also see that in some cases all requirements meet by default, and full control over the processes of the target application is not required.

The following subsections provide examples meeting these criteria, while the Exploitation section describes practical implementations of attacks based on one of the identified features.

1. Quarantine Restoration

In order to reduce the impact of potential false positive detections, endpoint security products allow some users to restore quarantined files. The right to restore files is handled differently by each product, and may be configured by a central policy.

The quarantined files can belong to any user, and encryption keys for quarantined files need to be protected. Therefore, the restoration process is not done by the UI process running with the privileges of the current user. Instead, this process signals another process that usually runs with SYSTEM privileges to do the restoration. Thus, if not implemented with care, the restored file is created with SYSTEM privileges, providing venue for privileged file-writes that can easily result in local privilege escalation on Windows operating systems.

For requirement R1. we observed the following behaviors with different endpoint security products:

A1. The feature is always available for every user.

A2. The access to the feature can be configured for different users and privileges.

a. Policy is enforced by low-privilege component

b. Policy is enforced by a high-privilege component

A3. The feature is always available to administrators only.

In case of A1. exploitation is trivial. Several products implement a configurable system conforming to A2.a which is exploitable if self-defense is bypassed. Examples of these scenarios are detailed in the Exploitation section. In case of A2.b and A3. exploitation of the Quarantine feature is not possible

According to our experiences, requirement R2. is usually met, probably as a result of design decisions: allowing users to handle false positive detections affecting system components or the product itself[6]. Alternative solutions could employ impersonation (as we will see in the case of KES), request elevation (UAC) or simply perform restoration from the low-privileged process to avoid unintended high-privileged file writes and prevent exploitation.

R3 can be met in several ways, because multiple edge cases must be handled during the restoration process:

[6] https://www.theregister.co.uk/2015/05/07/avast_false_positive_snafu/
http://www.zdnet.com/article/sophos-antivirus-detects-own-update-as-false-positive-malware/

E1. Regular file exists at the destination path

E2. Destination directory no longer exists

E3. The destination contains links or junctions

E4. The destination is on a remote computer

In any of these cases the application may require user input (thus giving Control) to determine the path to which the file will be restored. It's important to note that the above scenarios can come up multiple times during the restoration process. For example, E1. can come up after the software lets the user choose an arbitrary new destination path while handling E2. During our research, we found that both E1. and E2. can trigger application features that allow users to choose arbitrary destination paths for the files to be restored. While additional checks may be implemented during the handling of these edge cases, only the checks performed by high-privileged components can be effective security measures, otherwise they can be disabled after successful self-defense bypass. Examples of the latter are also included in the Exploitation section.

Additionally, Florian Bogner published an independent research dubbed as AVGater[7] that also abuses the AV quarantine restore feature and relies on NTFS directory junctions to trick the high-privileged AV components to write to sensitive file system locations (E3).

Forcing high-privileged processes to access remote locations (E4.) can result in leak of user credentials, as it was demonstrated by James Forshaw of Google Project Zero[8] and by the "Hot Potato" attack of FoxGlove Security[9]. Interestingly, the Project Zero research also targeted an endpoint security product (Windows Defender). This previous research demonstrated that high-privileged remote file access is achievable with generally unrestricted product features (user initiated scans) so this exploitation path was no further researched.

2. Other potential vectors

Quarantine restoration proved that self-defense can hide exploitable attack surface if protective application policy is enforced by the low-privileged component of the target (A2.a). During our research we also experimented with additional, potentially exploitable application features:

Some products may allow users to **manually move files to the quarantine** – this may allow using the product to implement custom security controls, and to crowdsource information about previously unknown malware. This feature (together with restoration) may allow arbitrary file reading with SYSTEM privileges. We wanted to implement such an exploit for Symantec Endpoint Protection, but contrary to the quarantine restoration feature (that is exploitable as described in section 1.2 of the Exploitation part), the software implements impersonation when manually adding items to the quarantine, which is a secure solution. It could be demonstrated though, that access checks are performed from the low-privileged process, after which the file path can be modified before it is passed to the high-privileged service.

As a superset of quarantine-based attacks, any file access features may be susceptible. These features commonly include **secure file removal** ("file shredding"), **encryption** and **import/export of application data** (logs, settings, etc.). For example, in case of Kaspersky home products, the "Settings export" feature can be used to overwrite arbitrary files with SYSTEM privileges if the attacker controls the UI process. Although the contents of the resulting file can only be controlled in small parts, data of other users can be corrupted (loss of integrity or availability), and even complex attacks relying on file polyglots or fail-open configurations may be implemented.

[7] https://bogner.sh/2017/11/avgater-getting-local-admin-by-abusing-the-anti-virus-quarantine/
[8] https://bugs.chromium.org/p/project-zero/issues/detail?id=222&redir=1
[9] https://foxglovesecurity.com/2016/01/16/hot-potato/

All the above vectors rely on standard application features, and are aimed to circumvent file system access controls. Features unrelated to direct file access (such as system-wide network settings, SSL/TLS interception) may also open venues for interesting attacks. Discovering exploitable functionality not intended for application users (e.g. debug features, verification of updates) may yield even more powerful vectors.

The bottom line is that quarantine restoration is just an example of exploitable attack surface. The outlined requirements are applicable outside of the scope of the quarantine, security critical checks are implemented in low-privileged components in case of other features too. These components are only protected by application-specific self-defenses, and their security can't be guaranteed by the operating system. Thorough exploration of this attack surface requires further research.

Exploitation

The user-mode components of endpoint protection software can be thought of in a client-server model where the UI process of the product acts as the client and the high-privileged service as the server. In case of common client-server applications (like web applications) implementing security checks at client-side is generally considered as bad practice. However, in case of our targets, self-defense may be taken as a security boundary that allows moving such checks (and thus complexity) away from the main component.

As we will see, different endpoint security products perform "client-side" checks only to protect sensitive features. The following subsections describe the implemented security features, and the techniques we developed for their bypass after in-process code execution was achieved via COM hijacking. After self-defense is breached the products allow performing privileged actions without performing further checks in higher-privileged components. We will also see that some products don't implement protections in the UI process either, making exploitation trivial.

1. Symantec

1.1. Symantec Norton Security Deluxe

In case of Norton Security, only administrative users (the user Administrator or members of the Administrators group) can access sensitive actions such as quarantine restore. The user interface that allows access to these settings runs as a process of the currently logged in user (`NS.exe`).

`NS.exe` is responsible for checking if the current user is an administrator. It performs this check via the `ccLib.dll` library that performs a series of calls to the `GetTokenInformation()` API with the current process token. One of the checks compares the group SID's associated with the current token with the SID of the `BUILTIN\Administrators` group. For this check `ccLib.dll` constructs the group SID by invoking the `AllocateAndInitializeSid()` API as follows (module base is at 0x10000000)[10]:

```
.text:1006006F      push    ebx
.text:10060070      push    edi
.text:10060071      or      [ebp+TokenHandle], 0FFFFFFFFh
.text:10060078      lea     eax, [ebp+pSid]
.text:1006007E      push    eax   ; pSid
.text:1006007F      xor     ecx, ecx
.text:10060081      mov     word ptr [ebp+pIdentifierAuthority.Value+4], 500h
.text:10060087      push    ecx   ; nSubAuthority7
.text:10060088      push    ecx   ; nSubAuthority6
.text:10060089      push    ecx   ; nSubAuthority5
.text:1006008A      push    ecx   ; nSubAuthority4
.text:1006008B      push    ecx   ; nSubAuthority3
.text:1006008C      push    ecx   ; nSubAuthority2
.text:1006008D      push    220h  ; nSubAuthority1
.text:10060092      push    20h   ; nSubAuthority0
.text:10060094      push    2     ; nSubAuthorityCount
.text:10060096      lea     eax, [ebp+pIdentifierAuthority]
.text:10060099      mov     [ebp+pSid], ecx
```

[10] ccLib.dll version 15.0.0.80
SHA-256: bea0fa64f24faf5f27a87fb8ff7b9632646d0565f871664a200969224e827406

```
.text:1006009F      mov     bl, cl
.text:100600A1      mov     dword ptr [ebp+pIdentifierAuthority.Value], ecx
.text:100600A4      push    eax   ; pIdentifierAuthority
.text:100600A5      mov     [ebp+var_A8], ebx
.text:100600AB      call    ds:AllocateAndInitializeSid
.text:100600B1      test    eax, eax
.text:100600B3      jnz     short loc_100600FF
```

As we can see, two sub-authorities are set. One of these, `nSubauthority1` is set to the value 0x220 that (together with `nSubAuthority0`) represents the `BUILTIN\Administrators` group. By patching this value to 0x221, the library will check the membership to the `BUILTIN\Users` group instead, that will result in a successful lock-out of the administrative features of the user interface, including Quarantine restore.

Norton Security allows trusted users (administrators) to restore quarantined files. Since the quarantined files can belong to any user, the restoration process is not done by the UI process running with the privileges of the current user. Instead, this process signals another instance of `NS.exe` running with `SYSTEM` privileges to do the restoration. The restored file is thus created with `SYSTEM` privileges.

This process can be easily abused because Norton Security lets the user choose an arbitrary restoration path if a file already exists at the original path of the quarantined file (edge case E1.).

1.2. Symantec Endpoint Protection

For our tests, Symantec Endpoint Protection (SEP) was installed with an installation package generated by Symantec Endpoint Protection Manager with its default policy. This policy allows regular users to restore files from the quarantine, and there is no option to configure it other ways. SEP also allows adding arbitrary files to the quarantine and choosing arbitrary restore locations. However, the UI process (`SymCorpUI.exe`) of SEP also performs privilege checks before signaling the high-privileged endpoint component (`ccSvcHst.exe`) to restore files by attempting to create an empty file in the destination restore directory. This check is of course insufficient if an attacker can control the UI process and fake the results of the privilege check or manipulate the path sent to the high-privileged process.

In the `SymCorpUI.exe` process the `CliProxy.DLL` library provides the IPC interface to `ccSvcHost.exe`. During the restore process this DLL is invoked first at offset 0x36530[11]:

```
.text:00036530      push    ebp
.text:00036531      mov     ebp, esp
.text:00036533      sub     esp, 8
.text:00036536      mov     eax, ___security_cookie
.text:0003653B      xor     eax, ebp
.text:0003653D      mov     [ebp+var_4], eax
.text:00036540      mov     eax, [ebp+arg_0]
.text:00036543      push    esi
.text:00036544      mov     esi, [ebp+restore_path]
.text:00036547      push    edi
.text:00036548      mov     edi, [eax+150h]
.text:0003654E      call    sub_345B0
```

[11] File version: 14.0.3752.1000
SHA-256: d0fd4d5a7b340b125595665f60f083c39f6e5fceedc3766d1a649226679d1bc5

```
.text:00036553      test    eax, eax
.text:00036555      jnz     short loc_3656E
.text:00036557      pop     edi
.text:00036558      mov     eax, 20000046h
.text:0003655D      pop     esi
.text:0003655E      mov     ecx, [ebp+var_4]
.text:00036561      xor     ecx, ebp
.text:00036563      call    @__security_check_cookie@4 ; __security_check_cookie(x)
.text:00036568      mov     esp, ebp
.text:0003656A      pop     ebp
.text:0003656B      retn    10h
.text:0003656E loc_3656E:
.text:0003656E      mov     edx, [ebp+arg_4]
.text:00036571      lea     eax, [ebp+var_8]
.text:00036574      push    eax
.text:00036575      push    [ebp+arg_C]
.text:00036578      mov     ecx, edi
.text:0003657A      push    esi   ; file name
.text:0003657B      push    0
.text:0003657D      mov     [ebp+var_8], 0
.text:00036584      call    restore_func0_
```

At 0x36544 the address of the destination path string passed as function argument is stored in the `ESI` register. From 0x36553 the return value of a function call is checked, and the function exits if an error is reported, otherwise execution is resumed at the 0x3656E basic block that proceeds with the restoration, using the value in `ESI`. Our exploit patches the error handler branch: we remove the call to the stack cookie check and use the freed space to load the address of a new destination path string (allocated from the injected .NET code) into the `ESI` register:

```
                                    be12345678  mov esi, 0x78563412
85c0        test eax, eax           85c0        test eax, eax
7517        jne 0x1b                7512        jne 0x1b
5f          pop edi                 5f          pop edi
b846000020  mov eax, 0x20000046     b846000020  mov eax, 0x20000046
5e          pop esi                 5e          pop esi
8b4dfc      mov ecx, dword [ebp - 4] 8b4dfc     mov ecx, dword [ebp - 4]
33cd        xor ecx, ebp            33cd        xor ecx, ebp
e84a350000  call 0x355f
                                    8be5        mov esp, ebp
8be5        mov esp, ebp            5d          pop ebp
5d          pop ebp                 c21000      ret 0x10
c21000      ret 0x10
```

This way `ccSvcHost.exe` will receive a different path that was checked by the UI process. Of course, more complex logic can be implemented by performing a `CALL` to some shellcode instead of a simple `MOV`.

2. Kaspersky Labs

2.1. Home Products

Kaspersky Antivirus, Free Antivirus and Internet Security products allow quarantine restore for any user. No file system ACL checks are performed and users can choose arbitrary restore locations for quarantined files in case the original directory of the quarantined item no longer exists. This makes local privilege escalation trivial.

Aside of the quarantine, several product features are present which may be protected with a password. The password itself is checked by a high-privilege process (`avp.exe`), but it is the UI process (`avpui.exe`) that passes the MD5 hash of the provided password (through standard Windows RPC) and checks if `avp.exe` reports success or failure. The check is performed by `avpuimain.dll`[12]

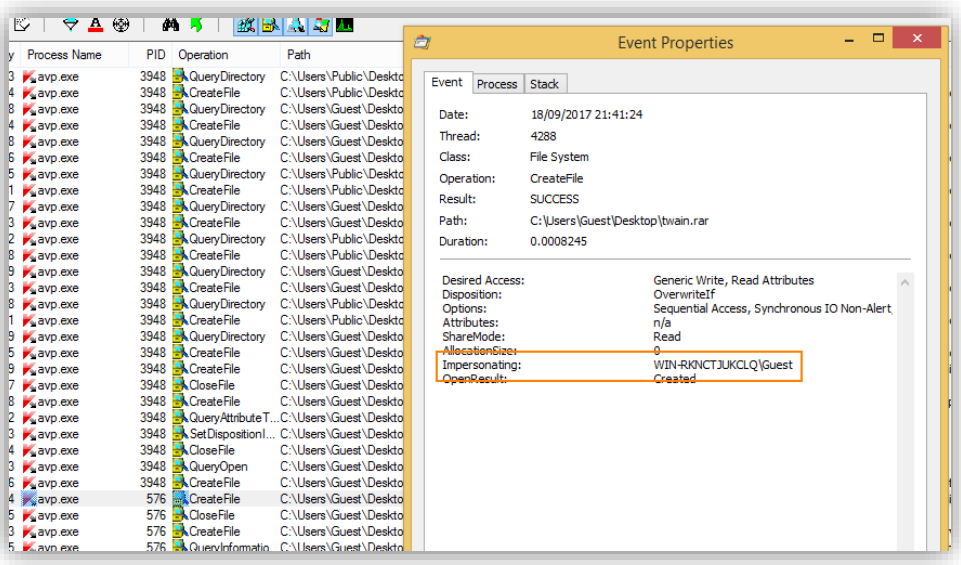
```
.text:0000EECD      mov     eax, [ebp-1Ch]
.text:0000EED0      lea     edx, [ebp-3Ch]
.text:0000EED3      push    edx
.text:0000EED4      push    eax
.text:0000EED5      mov     ecx, [eax]
.text:0000EED7      call    dword ptr [ecx+0Ch]
.text:0000EEDA      mov     esi, eax
.text:0000EEDC      pop     ecx
.text:0000EEDD      pop     ecx
.text:0000EEDE      test    esi, esi
.text:0000EEE0      jns     short loc_EF61
...
.text:0000EF61 loc_EF61:                ; CODE XREF: sub_EDA5+13Bj
.text:0000EF61      xor     eax, eax
.text:0000EF63      cmp     esi, 1
.text:0000EF66      setnz   al
.text:0000EF69      mov     [ebx], al       ; Set success/failure
.text:0000EF6B      test    al, al
.text:0000EF6D      jz      short loc_EF79
.text:0000EF6F      cmp     byte ptr [ebp+10h], 0
.text:0000EF73      jz      short loc_EF79
.text:0000EF75      mov     byte ptr [edi+18h], 1
```

After code injection is performed via COM hijacking, the attacker can patch offset 0xEF66 so that EAX will have a positive value. This way the UI will accept any password (but the correct one), and the attacker can access functionality like exclusion lists of settings import/export (see Other potential vectors) or he can simply disable protection – the `avp.exe` process will not perform further security checks.

[12] DLL version: 18.0.0.495
SHA-256: 37fe075f4eb8795b9b2fc61abd96a0f87f67a4f154e5dd9b40d7d9de65a030bb

2.2. Business Products

After succeeding with multiple home products, we tried the same techniques against Kaspersky Endpoint Security, which is the endpoint agent installed by the business products of the vendor. As it turned out, in this case, the privileged avp.exe process performs impersonation while processing the request of the UI process:



It seems that while home products can be trivially exploited, Kaspersky has a secure solution in its business products.

3. Bitdefender

3.1. Bitdefender Antivirus Plus 2018

In case of Bitdefender Antivirus Plus 2018 only administrative users (the user Administrator or members of the Administrators group) can access sensitive options such as quarantine restore. The user process `seccenter.exe` is responsible for checking if the current user is an administrator. It performs this check via the `bdusers.dll` library that performs a series of calls to the `CheckTokenMembership()` API with the current process token. The API is always called via a wrapper function that creates a SID object based on the function arguments and compares it with the provided token handle by invoking `CheckTokenMembership()` (module base is at 0x180000000)[13]:

```
.text:0000000180005140     mov     [rsp+arg_10], rbx
.text:0000000180005145     push    rdi
.text:0000000180005146     sub     rsp, 80h
.text:000000018000514D     mov     rax, cs:__security_cookie
.text:0000000180005154     xor     rax, rsp
.text:0000000180005157     mov     [rsp+88h+var_10], rax
.text:000000018000515C     xor     edi, edi
.text:000000018000515E     mov     [rsp+88h+pIdentifierAuthority.Value+4], 500h
.text:0000000180005165     lea     rax, [rsp+88h+SidToCheck]
.text:000000018000516A     mov     [rsp+88h+IsMember], edi
.text:000000018000516E     mov     [rsp+88h+pSid], rax ; pSid
```

[13] DLL version: 22.0.8.99
SHA-256: c3b020178b1c85c99af58f61aaff149588bb008835252fc774fecf4686ed2139

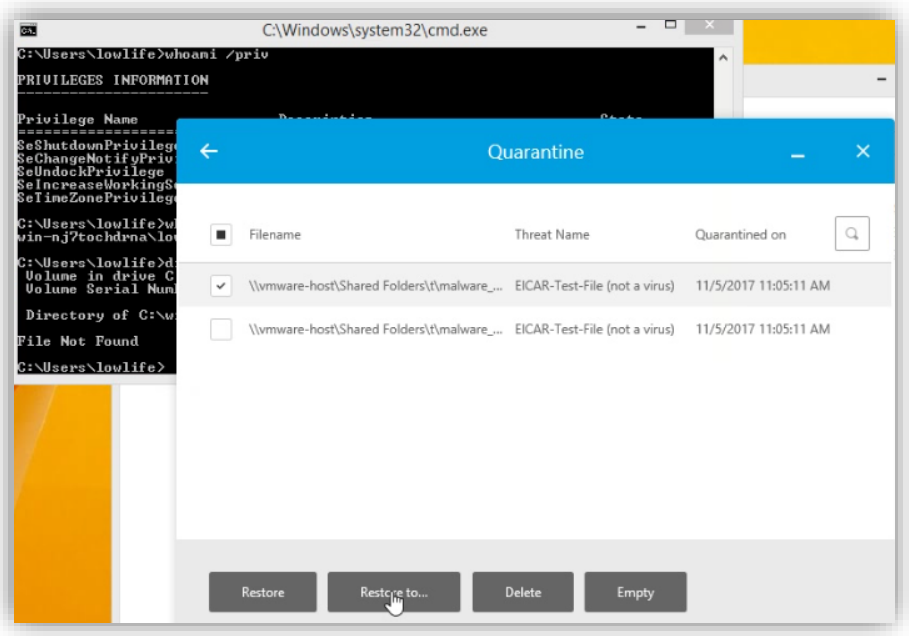
```
.text:0000000180005173     mov     rbx, rcx
.text:0000000180005176     mov     [rsp+88h+nSubAuthority7], edi ; nSubAuthority7
.text:000000018000517A     lea     rcx, [rsp+88h+pIdentifierAuthority]
.text:000000018000517F     mov     [rsp+88h+nSubAuthority6], edi ; nSubAuthority6
.text:0000000180005183     lea     r8d, [rdi+20h]  ; nSubAuthority0
.text:0000000180005187     mov     [rsp+88h+nSubAuthority5], edi ; nSubAuthority5
.text:000000018000518B     mov     r9d, edx        ; nSubAuthority1
.text:000000018000518E     mov     [rsp+88h+nSubAuthority4], edi ; nSubAuthority4
.text:0000000180005192     mov     dl, 2           ; nSubAuthorityCount
.text:0000000180005194     mov     [rsp+88h+nSubAuthority3], edi ; nSubAuthority3
.text:0000000180005198     mov     [rsp+88h+nSubAuthority2], edi ; nSubAuthority2
.text:000000018000519C     mov     dword ptr [rsp+88h+pIdentifierAuthority.Value], edi
.text:00000001800051A0     mov     [rsp+88h+SidToCheck], rdi
.text:00000001800051A5     call    cs:AllocateAndInitializeSid
.text:00000001800051AB     test    eax, eax
.text:00000001800051AD     jz      short loc_1800051DA
.text:00000001800051AF     mov     rdx, [rsp+88h+SidToCheck] ; SidToCheck
.text:00000001800051B4     lea     r8, [rsp+88h+IsMember] ; IsMember
.text:00000001800051B9     mov     rcx, rbx        ; TokenHandle
.text:00000001800051BC     call    cs:CheckTokenMembership
.text:00000001800051C2     mov     ecx, [rsp+88h+IsMember]
.text:00000001800051C6     test    eax, eax        ; PATCH: inc rcx
.text:00000001800051C8     cmovz   ecx, edi        ; nop
.text:00000001800051C8                             ; nop
.text:00000001800051C8                             ; nop
.text:00000001800051CB     mov     [rsp+88h+IsMember], ecx
.text:00000001800051CF     mov     rcx, [rsp+88h+SidToCheck] ; pSid
.text:00000001800051D4     call    cs:FreeSid
.text:00000001800051DA
.text:00000001800051DA loc_1800051DA:
.text:00000001800051DA     mov     eax, [rsp+88h+IsMember]
.text:00000001800051DE     mov     rcx, [rsp+88h+var_10]
.text:00000001800051E3     xor     rcx, rsp
.text:00000001800051E6     call    sub_180005EC0
.text:00000001800051EB     mov     rbx, [rsp+88h+arg_10]
.text:00000001800051F3     add     rsp, 80h
.text:00000001800051FA     pop     rdi
.text:00000001800051FB     retn
```

To bypass the check for administrative privileges, the DLL (loaded via COM registration) will patch offset 0x51C6 to set the RCX register that will later determine the return value of the function while removing the original checks of the return values of the `CheckTokenMembership()` API. The patch code can be

seen as comment in the above listing from offset 0x51C6. This way all the functions that use the patched wrapper will see that any token is member of any given group, including Administrators. This way all features available for administrators will be available to the regular user of the attacker. After privilege checks are bypassed, the quarantine restore feature can be abused because Bitdefender lets the user choose an arbitrary restoration path if the folder the file was quarantined from doesn't exist anymore (edge case E2.).

3.2. Bitdefender Gravityzone

Unlike its counterpart intended for home use, the endpoint agent of Bitdefender Gravityzone (Bitdefender Endpoint Security Tools - BEST) - configured with the default – allows any user to restore quarantined files to arbitrary file system locations:



This means that while Bitdefender's home product at least tries to cover this attack surface, the business version doesn't even require self-defense bypass to be exploited. It's worth noting though that the BEST client provides a highly restricted set of functionalities, most configurations and actions can only be initiated from the cloud console by design.

Conclusions

In this paper we demonstrated a generic self-defense bypass method against several endpoint security products. While the utilized COM hijacking technique was known for a long time[14], its applicability to endpoint security products was not recognized by the affected vendors during this period.

It's important to note that COM hijacking is just one technique for self-defense bypass. Since the self-defense mechanisms generally don't operate within the operating systems security model, we shouldn't see them as a security boundary that can be relied on to protect sensitive application functionality.

We demonstrated that if the assumption about the robustness of self-defense is broken, endpoint security products expose an architecture that can be exploited to achieve local privilege escalation. Just like the utilized self-defense bypass technique, the most significant vector for privilege escalation (quarantine restoration) has been present in the affected products for several years. Interestingly, this vector was just recently publicized in parallel with our research (under the AVGater "brand") indicating a convergence of research in this field. In addition to the findings of AVGater, our research showed that product features may be exploitable even if their use is prohibited by application configuration. This is because the affected products implement security checks in low-privileged ("client-side") components, which aren't meant to be protected by the operating system from interference by the process owner. We also found that other product features may also be abused to cross OS privilege boundaries although these vectors seem less powerful.

Researching multiple vendors showed that different products of the same vendors implement the same features in both secure and insecure ways. This may indicate differences in the quality assurance processes of the respective product teams or even conscious design decisions (or acceptance of risk).

We've been monitoring the exploitability of the targeted products since July 2017. Our additional findings and demonstration materials of the described vulnerabilities will be continuously updated on our blog:

https://blog.silentsignal.eu/2018/01/08/bare-knuckled-antivirus-breaking/

[14] https://attack.mitre.org/wiki/Technique/T1122