TreasureHunter Source Code Leak Makes Payload, Builder Available to All

flashpoint-intel.com/blog/treasurehunter-source-code-leaked/

May 10, 2018

Blogs

Blog

TreasureHunter Point-of-Sale Malware and Builder Source Code Leaked

The source code for a longstanding point-of-sale (PoS) malware family called TreasureHunter has been leaked on a top-tier Russian-speaking forum. Compounding the issue is the coinciding leak by the same actor of the source code for the malware's graphical user interface builder and administrator panel.

The source code for a longstanding point-of-sale (PoS) malware family called TreasureHunter has been leaked on a top-tier Russian-speaking forum. Compounding the issue is the coinciding leak by the same actor of the source code for the malware's graphical user interface builder and administrator panel.

The availability of both code bases lowers the barrier for entry for cybercriminals wishing to capitalize on the leaks to build their own variants of the PoS malware.

Point-of-sale malware has been at the root of many breaches, including massive thefts at retailers Target in 2013 and Home Depot in 2014; in each case attackers were able to extract more than 100 million payment card and customer records from point-of-sale terminals by scraping card data before it was encrypted and sent to the payment processor.

Industry Collaboration on Detection and Prevention

TreasureHunter has been known and investigated since 2014, but until now investigators have had to reverse-engineer its code in order to analyze it. Now with the full code available, analysts have previously unseen insight into the malware’s operation. Flashpoint analysts, who discovered the source code leak in March, proactively collaborated with researchers at Cisco Talos, who reviewed and improved protections, and advanced-detection mechanisms, in an effort to disrupt potential copycats who may have their hands on the source code.

In the meantime, Russian-speaking cybercriminals have been observed on the vetted underground discussing improvements and weaponization of the leaked TreasureHunter source code. Notably, the original developer appears to be a Russian speaker who is proficient in English. Originally, this malware appears to have been developed for the notorious underground shop dump seller “BearsInc,” who maintained presence on various low-tier and mid-tier hacking and carding communities (below is a graphical representation of such an operation on the Deep & Dark Web). It’s unknown why the source code was leaked at this time.

A graphical representation of a typical cybercrime dump shop ecosystem.***Image 1:** A graphical representation of a typical cybercrime dump shop ecosystem.*

One Leak Can Spawn Many Variants

TreasureHunter behaves like many other point-of-sale malware samples. Once an attacker has access to a Windows-based server and the point-of-sale terminal, the malware is installed and it establishes persistence by creating a registry key that runs the malware at startup. It then enumerates running processes, and scans device memory looking for track data, including primary account numbers (PANs), separators, service codes, and more. It then establishes a connection with the attacker’s command and control server and sends the stolen data to the criminal.

The leak of the builder adds another dimension to the availability of the TreasureHunter payload and configurations. In the past, malware source code leaks such as the Zeus banking Trojan have spawned numerous variants, including Citadel, which cost organizations hundreds of millions in losses. PoS malware leaks have had similar effects, most notably with the 2015 leak of the Alina malware which led to the creation of the ProPoS and Katrina variants. The actor behind the TreasureHunter leak said:

“Besides alina, vskimmer and other sniffers, Treasure Hunter still sniffs (not at a very high rate, but it still does) and besides that , since now you have the source code, it can be update anytime for your own needs.”

For researchers, the availability of the source code opens the door into new avenues of analysis and proactive visibility into such activity on the underground. This affords organizations such as Flashpoint the ability to collaborate with others in the industry such as Cisco Talos in this case to improve existing protections and force attackers back to the drawing board.

Source-Code Level Insight

The code project appears to be called internally trhutt34C, and was written in pure C with no C++ features. It was compiled originally in Visual Studio 2013 on Windows XP. Based on analysis, researchers believe the developer intended to improve and redesign various features including anti-debugging, code structure improvement, and gate communication logic. With the goal of additional features to be improved, the developer hoped frustrate malware analysis and subsequent research; the actor left behind a note that said: *“We want the malware researchers screamin’!”*

A snapshot of the TreasureHunter source code.***Image 2:*** *A snapshot of the TreasureHunter source code.*
The unfinished project included continued improvement code snippets, below:

> TO DO for the next version of the client (0.2 Beta): Replace all Unicode versions of functions with ANSI versions. Now why did I ever go for wide-char in the first place?.. Improve the code structure: Replace all the if – else constructs that are rendered needless by return commands; Organize the includes; Give the code proper commenting so that I am able to modify and improve it after not having seen it for some time (if such a thing happens). Make scan exceptions and service codes configurable. Add the following commands to the gate communication logic: Download and execute for updating; Remote CMD command execution; Remote self-removal for emergency cases. Add anti-debugging: Use self-debugging by creating a child process (may be improved later by reversing the tables); Improve the MD5 function and use it to find debuggers by signatures (maybe to be added in future versions); Use GetTickCount to detect parts of code being stepped through (maybe to be added in a “heuristical” joint algorithm with the abovementioned); Upon finding a debugger, destroy the critical section and/or start creating new threads infinitely until the application crashes. Maybe also kill processes and delete debuggers and/or decompilers permanently. We want the malware researchers screamin’! Add better persistency and timeouts to gate communication. Add local saving of data if the gate can’t be reached for a certain period of time. Add the option to run the program as a service on Windows XP. Improve the code structure and add comments to avoid future confusion. Add error handling and backup restart in case of crash or heap overflow (malloc fail). Improve the Clingfish system (so that a clingfish thread doesn’t do the same thing as the main thread right after being spawned). Debug the system information extraction mechanism further (on different OS versions). Improve the track-finding algorithm to make it faster.

The stolen dump structure is as follows. The structure contains the following key elements used to collect and operate with stolen dumps, such as unique machine information and where scraped data is from:

```
typedef struct dumpsHolder {
        TCHAR *lpFileName;
        int lpFileNameLength;
        int procID;
        char *trackArr;
        int trackArrLength;
} dumpsHolder;
```

The credit card process scan works in exception mode:

```
char *scanExceptions[SCANEXCEPTIONSNUM] = {“System32”, “SysWOW64”,
“\Windows\explorer.exe”};
```

The malware focuses on scraping credit card track data, focusing on the following service codes:

```
char *serviceCodes[SERVICECODESNUM] = {“101”, “201”, “121”, “231”, “221”, “110”};
```

Registry persistence for autostart in HKLM\Microsoft\Windows\CurrentVersion\Run runs as “jucheck.”

A registry key created by the malware for persistence***Image 3:** A registry key created by the malware for persistence.*

The source code is consistent with the various samples that have been seen in the wild over the last few years. TreasureHunter\config.h shows definite signs of modification over the lifespan of the malware. Early samples filled all of the configurable fields with FIELDNAME_PLACEHOLDER to be overwritten by the builder. More recent samples, and the source code, instead writes useful config values directly into the fields. This makes the samples slightly smaller and uses fresh compiles to create reconfigured files.

Coverage and Detection with Cisco Talos

Coverage provided by Cisco Talos already exists in Cisco FirePower and AMP solutions. It is being provided here:**Cisco Threat Mitigation & Advanced Coverage:**   Snort Ruleset Identifier (available via Snort[.]org):

Snort Identifier: 29884,38573, 38574   ClamAV Signatures (available via ClamAV[.]net)

signatures: Win.Trojan.TreasureHunter*

Block has been deleted or is unavailable.