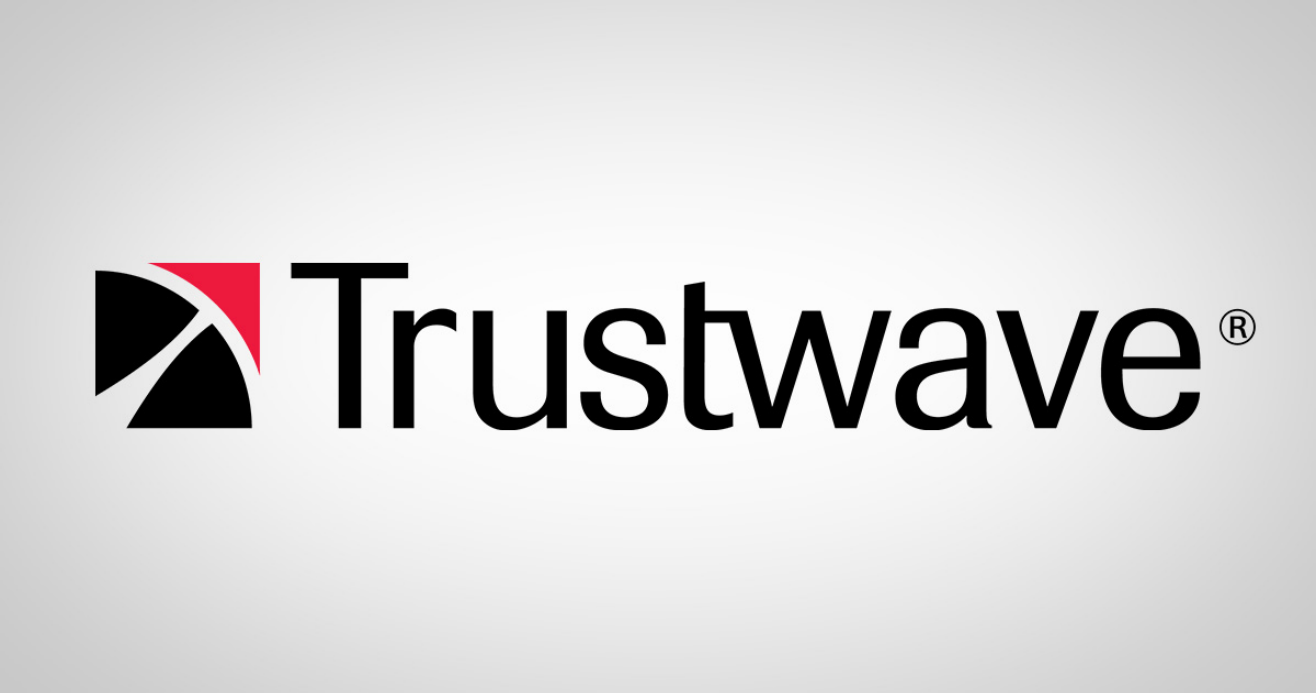
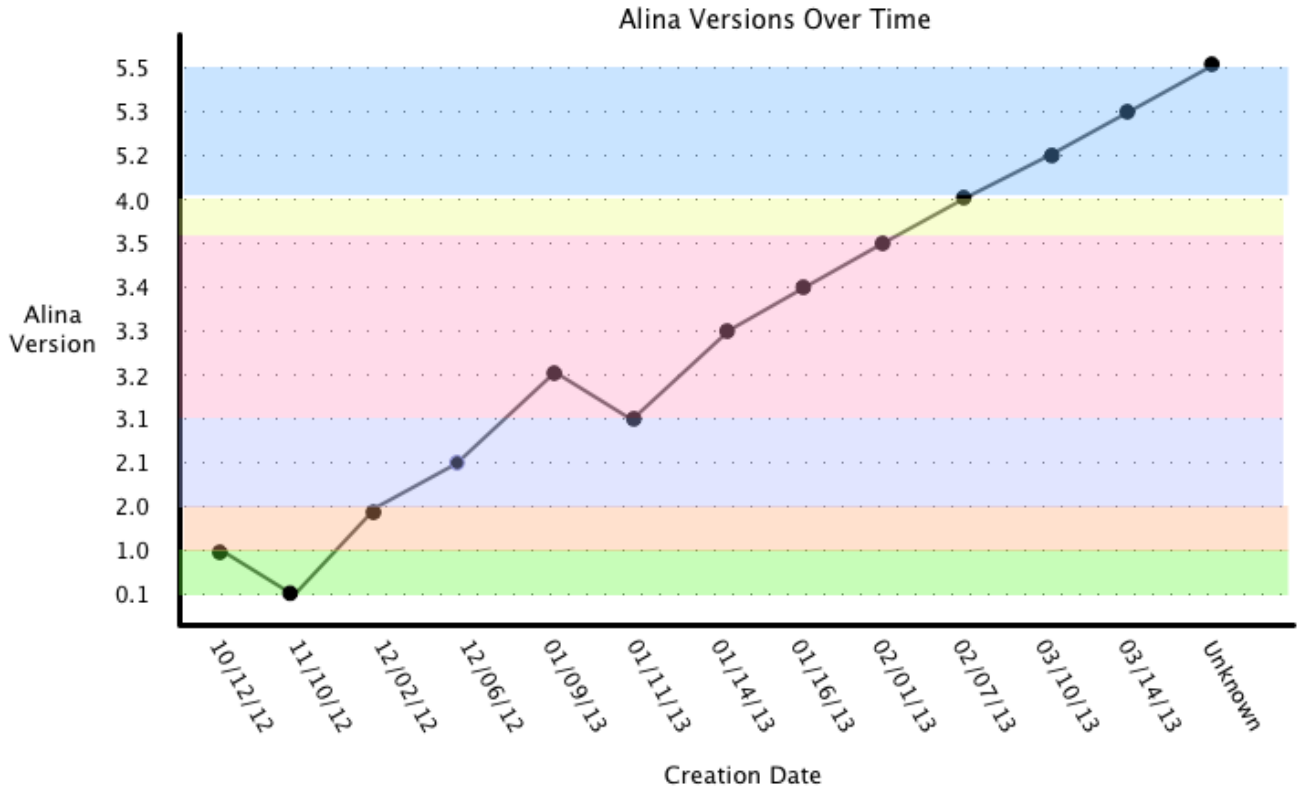
Alina: Following The Shadow Part 2



This will likely be the final blog post in this series on the Alina Point of Sale (POS) malware family. If you're just now joining us, please be sure to check out my previous blog posts on this topic, which cover the intricacies of version 4.0, as well as information about how this malware has evolved with respect to exfiltration and command and control (C&C).

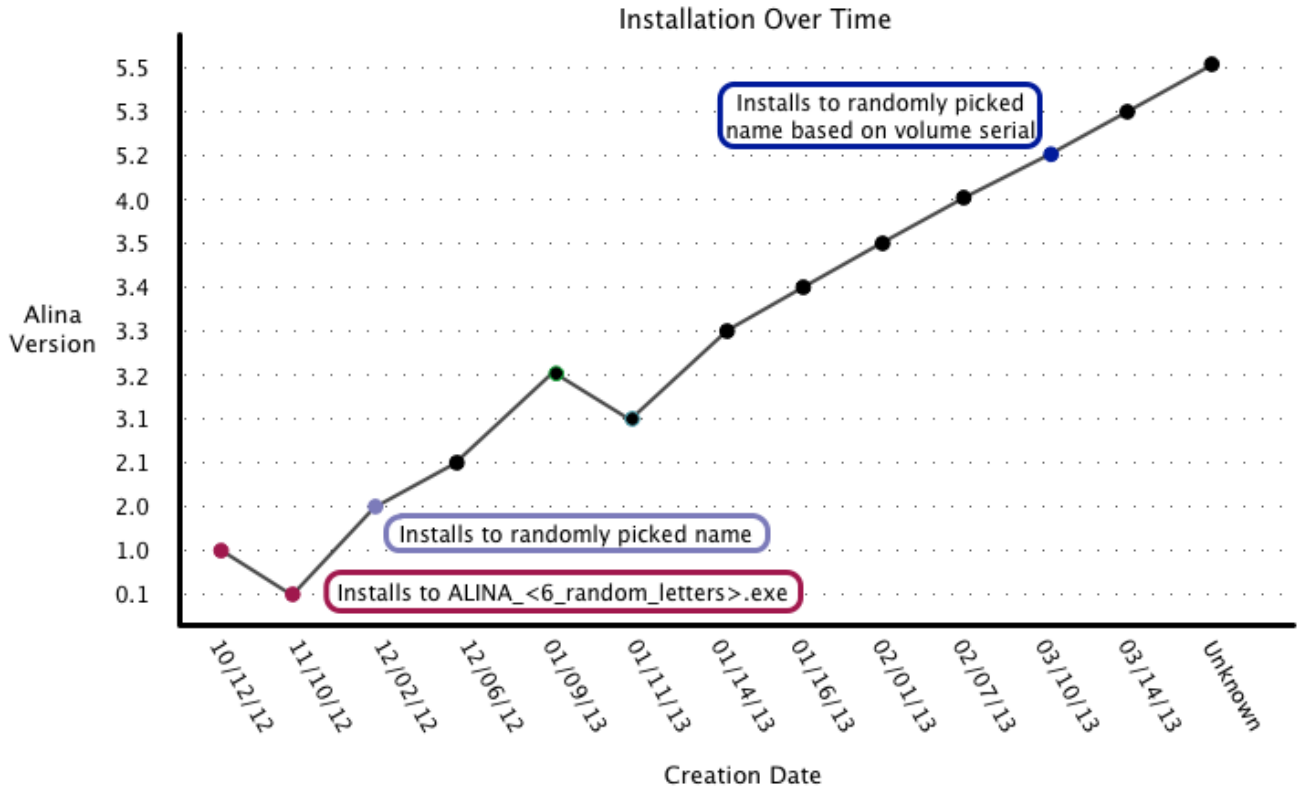
- Alina: Casting a Shadow on POS
- Alina: Following The Shadow Part 1

For this final part, I'm going to focus on how this malware is installed, what protections the author has placed on the malware to prevent Anti-Virus detection and/or reverse engineering of it, and how Alina aggregates track data. I may also throw in some other random tidbits of information that I've encountered depending on how long this blog post goes. My last one in particular was quite lengthy, so I'm going to do my best to avoid that this time around. We're going to be looking at the same versions as before. I've included the timeline graph below as a reference for readers.



Installation

Overall the malware authors were fairly consistent with regards to how this malware is installed, as opposed to previous characteristics that we've looked at, where many changes were incorporated in a large number of revisions. I've illustrated the changes witnessed throughout various versions below:

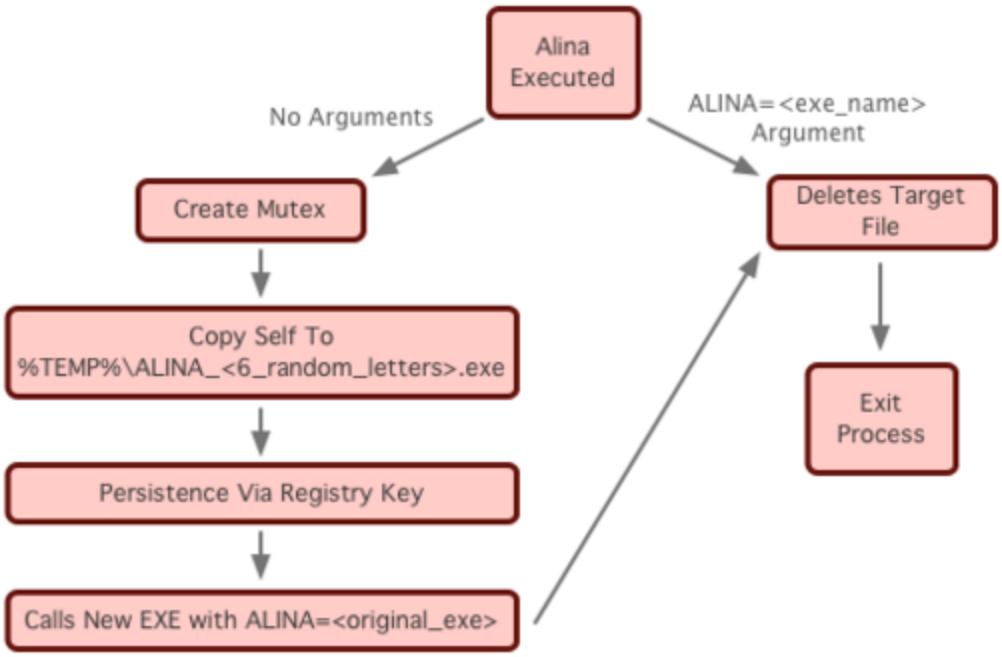


v0.1 / v1.0

For the early versions of Alina, the authors followed a simple process for malware installation. In essence, the malware looks to see if it has been supplied with the 'ALINA=<exe_name>' argument, where '<exe_name>' refers to an executable path. If this argument is supplied, Alina will not perform the installation procedure, but will instead simply delete this executable name before exiting.

If Alina started without this argument, it will begin the installation procedure. Alina copies itself to the following path:
%TEMP%\ALINA_<6_random_letters>.exe

It then modifies the HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ALINAhuahs registry key, and writes the location of the previously copied executable. This is a simple persistence technique that is encountered regularly when dealing with malicious samples. Finally, Alina will call the new executable with the 'ALINA=' argument, and point it to itself, ensuring the original file is deleted. I'm done my best to demonstrate this process visually below:



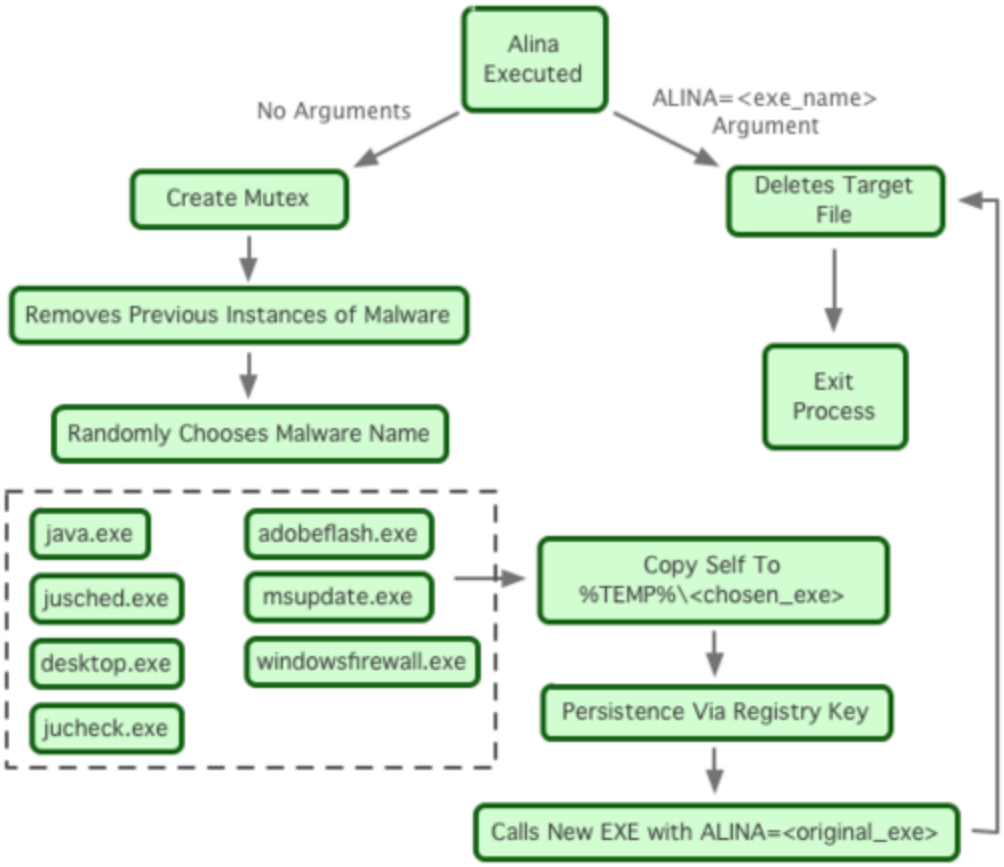
v2.x / v3.x / v4.x

Starting with version 2.0, the authors of Alina decided that they needed something a bit stealthier regarding how the malware is installed. Specifically, instead of installing the malware to 'ALINA_<6_random_letters>.exe', they instead decided to choose from a pool of seven potential malware names. When Alina is installed, it chooses to copy itself to one of the following names within the %TEMP% directory:

- java.exe

- jusched.exe
- jucheck.exe
- desktop.exe
- adobeflash.exe
- msupdate.exe
- windowsfirewall.exe

Persistence once again utilizes the Run registry key, however, the specific name coincides with the chosen malware name. For example, if 'windowsfirewall.exe' was chosen, the malware would install to the 'windowsfirewall' registry key. Additionally, when these versions of Alina are installed, the malware will look for previously installed instances and remove them. Once again I've tried to visualize this below:



v5.x

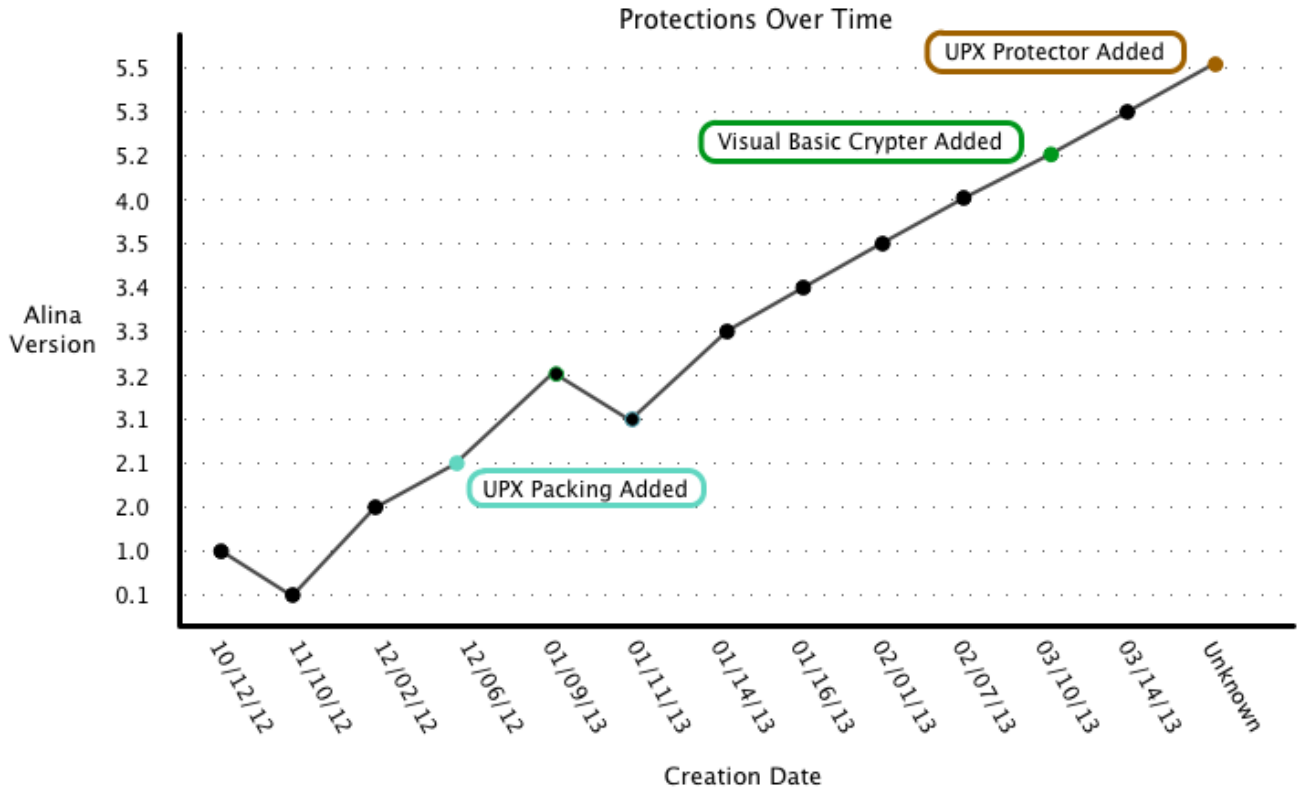
With version 5, we see a number of overall changes to Alina (many of which were covered in the previous post). With regard to installation, we see the authors shift away from a completely random choice of malware names. This likely has to due with the fact that every time Alina ran on the victim machine in versions 2.x-4.x, the malware would essentially reinstall, and often chose a different name than previously. It's likely the authors of Alina wished instead to chose a random name originally, and then stick with it for the remainder of the malware's existence. In version 5.x, Alina has increased its pool of potential malware names to eleven. Additionally, instead of randomly choosing them every time, Alina utilizes

the victim's volume serial number to decide which name to choose. This ensures a random name to begin with, but also ensures that it is consistent every time the system reboots. The following pool of potential malware names is utilized:

- defender.exe
- explorer.exe
- svchost.exe
- scvhost.exe
- ctfmon.exe
- rundll32.exe
- cmd.exe
- csrss.exe
- dasHost.exe
- services.exe
- Taskmgr.exe

Protections

Looking at how the protections for Alina have evolved over time has been quite fascinating. It's quite common to see malware that targets Point of Sale devices not employ common techniques seen in the wild, such as packing, crypting, and anti-debugging. As we look at Alina over the course of a few months, it becomes clear that the authors attempted to combat some threat that we can only speculate towards. It's possible that Anti-Virus began detecting the samples, which led to them adding protections. Additionally, they may have been fearful of reverse engineers gaining insight into the inner workings of their malware, which prompted them to make changes.



UPX

I'm sure a number of the people reading this blog are familiar with UPX, or the Ultimate Packer for eXecutables. UPX is one of the most popular, if not the most popular, packers on the market. Given the fact that it's so popular and (somewhat more importantly) free, it makes complete sense that the authors of Alina chose to pack their malware using this product. Not only does it reduce to overall file size of the malware, but it also prevents simple detection mechanisms, such as searching for strings within the binary. UPX began being used with versions 2.1 and above.

Visual Basic Crypter

Starting with version 5.2, we begin seeing a large leap with regard to protections. Specifically, we begin seeing a crypter written in Visual Basic being thrown on top of Alina. While packers have historically been created with the primary purpose of speeding up executables, crypters main purpose in life is to make my life difficult. In other words, crypters were built primarily to obfuscate binaries. One of the interesting side-affects of utilizing a crypter is that in some cases it will actually increase the rate of detection by the Anti-Virus community. An Anti-Virus company may not have a specific detection in place for a given family of malware, but they may have a signature in place for a crypter that is typically utilized for malicious purposes.

As an example, Alina version 2.0, which is not packed at all, is only detected by 30 Anti-Virus companies (VirusTotal). Conversely, Alina version 5.2, which makes use of the crypter is detected by 33 Anti-Virus companies (VirusTotal). This isn't a comprehensive test of

course, and there are a number of reasons why version 5.2 may be detected more than 2.0, but I'm simply using it to illustrate my point.

UPX Protector

UPX Protector is a utility that was developed to hinder UPX from being easily unpacked. Utilizing a simple command-line utility, UPX can be trivially unpacked. UPX Protector attempts to prevent this by corrupting the header (thus the reason we were unable to get a PE timestamp in version 5.5 in the previous blog post), cloaking sections, modifying the entry-point, etc. This protection was put into place starting with version 5.5.

Aggregating Track Data

For the most part, the task of aggregation of track data was fairly consistent for most of Alina's history. Up until versions 5.x, the process looked like this:

> Create an array of processes to look at via calls to CreateToolhelp32Snapshot(), Process32First(), and Process32Next(), ignoring processes in the following blacklist:
>
> - explorer.exe
> - chrome.exe
> - firefox.exe
> - iexplore.exe
> - svchost.exe
> - smss.exe
> - crss.exe
> - wininit.exe
> - steam.exe
> - devenv.exe
> - thunderbird.exe
> - skype.exe
> - pidgin.exe

- Loop through every process, and read pages of memory via calls to VirtualQueryEx() and ReadProcessMemory(), targeting RAM with read/write privileges.
- Apply a number of regular expressions against this read memory, targeting Track 1 and Track 2 data.
- Exfiltrate any data discovered and begin this process from the beginning.

Starting with version 5.x, the authors decided to spawn a new separate thread for each process they targeted. This meant that each process was constantly having its memory read, which decreased the chance that the authors would miss any track data being processed. The downside to this, however, was that it made the malware extremely noisy

and increased the chance of detection on the victim. This is just one example of how the authors weighed the consequences of their decisions and determined that it would be better to have a higher rate of success versus a greater chance of detection.

Random Thoughts / Points of Interest

Overall it's been very interesting to see Alina grow over the months in many different ways. As I mentioned originally in my first blog post, memory dumpers targeting POS devices are nothing new, however, the trend towards automation and C&C has been interesting to say the least. Looking back, I find it very interesting to see some of the processes in the blacklist, such as steam.exe, skype.exe, pidgin.exe, etc. If I had to speculate, I'd argue that seeing these processes demonstrates a lack of sophistication on the author's part. I wouldn't be surprised to discover that the author was running these programs in his development environment, and decided to add Steam, Pidgin, Thunderbird, etc. to the list in order make things easier on their end. Alternatively, I've heard reports of Alina showing up on a number of end-user machines (as opposed to POS devices), and it's certainly possible that these processes were added simply because they are commonly found on end-user devices. However, these are only sporadic rumors. I can only say for certain is that Trustwave has encountered Alina on a number of forensic cases affecting the Food and Beverage industry. It is very likely that this malware is being used for attacks on other industries as well; simply due to the way the malware works.

Per some comments we've received from the prior posts—I'd like to emphasize that Alina is very generic in nature. While it targets track data, and is most likely found on POS devices, it isn't limited to these machines, or any POS vendor in particular.

At this point I'm going to wrap things up as I've once again made this post longer than I anticipated. I hope you've enjoyed reading about Alina as much as I have reversing it. Thanks for reading!