securitykitten.github.io/_posts/2015-01-08-getmypass-point-of-sale-malware-update.md

github.com/malware-kitten/securitykitten.github.io/blob/master/_posts/2015-01-08-getmypass-point-of-sale-malware-update.md

malware-kitten

malware-kitten/
securitykitten.github.io

Jekyll theme inspired by Swiss design

0 Contributors | 0 Issues | 0 Stars | 0 Forks

layout	title	excerpt	date
category-post	Getmypass Point of Sale Malware Update	Now featuring small improvements!	2015-01-08 11:38:48 -0500

Introduction

I previously wrote about a new piece of malware called "getmypass" that was scoring 0/55 on Virustotal. The malware had an active digital signature and was able to successfully dump track data from a given process.

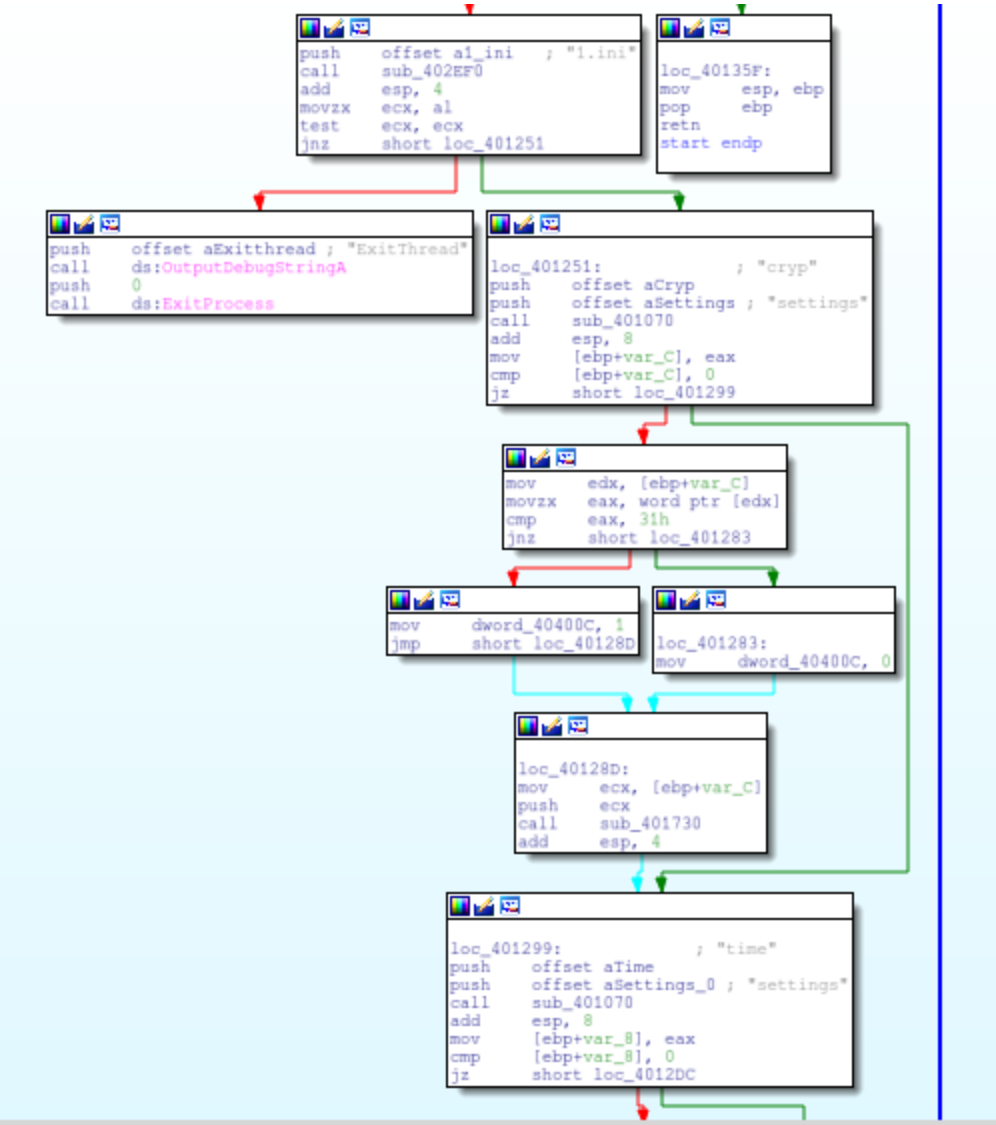
At CBTS, one thing that I frequently do is track malware families and try to trace them back to their infancy. It was interesting to me that another copy of this malware has now been uncovered and it has seen a small update. Not only is the author still writing code for getmypass, but it's now getting updated to enforce better opsec. Let's take a look.

Encryption by Default

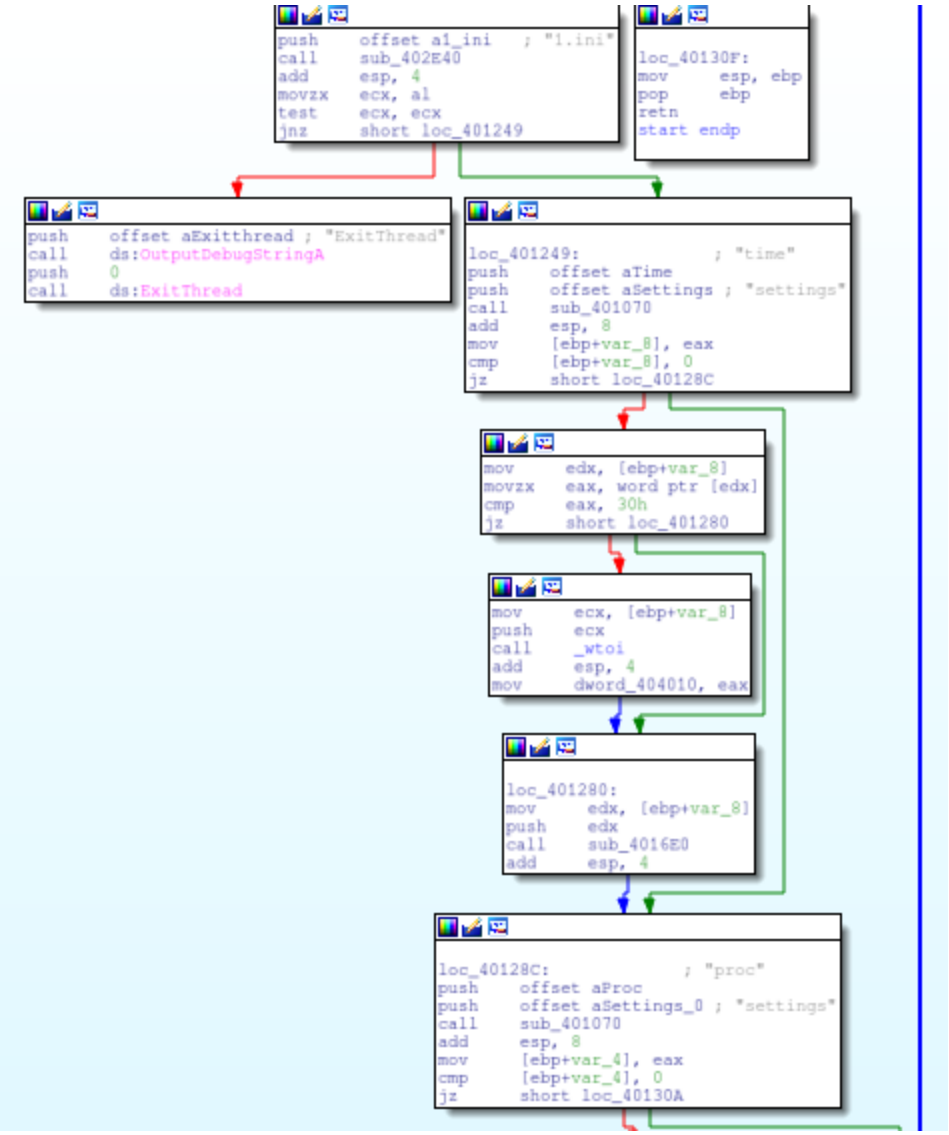
The first change in the malware is evident in the start function. In the older version of getmypass, it would look for the 1.ini file and check for the following options:

- time
- proc
- cryp

A screenshot of the old function:



In the new function, there is no longer an option for cryp, and the malware is programmed to encrypt the results by default.



The RC4 password is still unchanged and remains "getmypass". The function passing the trackdata is seen below.

```
Arg5 => 00145C70 ASCII "%B4888603170607238^Head/Potato^05051010000000000120319
Arg4 => 00145C70 ASCII "%B4888603170607238^Head/Potato^05051010000000000120319
Arg3 => 0000006D
Arg2 = 00000009
Arg1 = 00403184 ASCII "getmypass"
getmypas.00401670
```

The resulting encrypted data is written to a file rep.bin.

With a simple ruby script we can validate this RC4 routine and decrypt the contents.

{% highlight ruby %} {% raw %}

#!/usr/bin/ruby1.9.1 require 'rc4' require "base64"

f = File.new("rep.bin",'rb') decode = f.read() f.close key = "getmypass" dec = RC4.new(key) puts dec.decrypt(decode)

{% endraw %} {% endhighlight %}

And the decode:

{% highlight bash %} {% raw %}

$ ./rc4_decode.rb %B4888603170607238^Head/Potato^0505101000000000012031918051910000000?;4888603170607238=050510112031918051910?

{% endraw %} {% endhighlight %}

Support for Multiple Exfil Files

Older copies of getmypass would write to rep.bin file. This was the only output file and if there were multiple runs, the file would be overwritten.

The newer version now will move the file rep.bin to a backup file marked by the resulting value of GetTickCount.

```
push    ebp
mov     ebp, esp
sub     esp, 200h
call    ds:GetTickCount
push    eax
push    offset aRep_D_bin ; "rep_%d.bin"
lea     eax, [ebp+var_200]
push    eax
call    ds:wsprintfW
add     esp, 0Ch
lea     ecx, [ebp+var_200]
push    ecx
push    offset aRep_bin ; "rep.bin"
call    ds:MoveFileW
mov     esp, ebp
pop     ebp
retn
sub_4021F0 endp
```

The backed up file:

Conclusion

Encryption by default will now also help this malware evade tools that scour hard drives looking for structures that resemble track data. The support for multiple backup files will also enable the operators to move quickly. With the older copy, the malware authors used to have to dump processes and then backup the results file. Now they can edit the configuration file on the fly and the results get stored into a secondary file. Less clicks, less movement.

While these aren't groundbreaking changes, it's small changes like these that add up to make the malware operators more efficient and successful.