Email Analysis

The following files were found on Thunderbird Profile:

File Name	Type	MD5 Checksum
INBOX.msf	Mozilla Mork database, version 1.4	02713a86ad1d35ddc732e7b7e34a1462
INBOX	ASCII text	23f181d07b615d61482404f1f2f29b2e

The following message could be recovered:

To: "jones.sally1993@gmail.com" <jones.sally1993@gmail.com>
From: Biochemistry Campus IT Department <jason_halloween@protonmail.com>
Reply-To: Biochemistry Campus IT Department <jason_halloween@protonmail.com>
Subject: Important Security Update
Date: Mon, 12 Nov 2018 16:53:10 +0000

Dear user,

We have been informed of a vulnerability on the workstations connected to our campus network. This vulnerability, which is tied to improper network configurations, has been rated "Critical". If left unattended, it is likely that your personal information will be leaked to third-parties.

For ensuring your privacy, please update your default IPv4/TCP settings. You may do so automatically by executing the patch located in the attachments of this message.

-- Sid Wilkes
Technical Support - Information Technology Department

Attachment:
A single file named “main” which appears to be:

ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=28ba79c778f7402713aec6af319ee0fbaf3a8014, stripped

MD5 Checksum: 324ddc336159dd62e182e3abf12c9b0a

Artifact 1: A suspicious message found at Sally's Inbox

We were able to extract the main file from the email.

This message is suspicious, because the sender name doesn’t match the email address.

The team decided to carefully analyze the message’s header in order to extract more information about the sender identity.

It seems like the sender use **ProtonMail ISP** which claims to protect the privacy of the users: *By default, we do not keep any IP logs which can be linked to your anonymous email account. For more detailed approach on this email analysis see Header Analysis.*

A curious fact is that a file with the same name exists on `Downloads` folder (under Sally's Home Directory). We checked if the file is the same as the attachment as it's very likely that Sally downloaded it.

File Name	Type	MD5 Checksum
main	XML 1.0 document, ASCII text	1b38180828dbef5dc4a7e32589def5ae

Artifact 2: Files found on Sally's Download folder

The `main` file found at Sally's Download folder appears to be a XML file with some configuration about the desktop environment, thus revealing no interest to this investigation. *See more information on Memory Analysis section at the bottom of this page.*

Header Analysis

In order to obtain more information about the attacker we decided to take a close look at the message header.

```
Received: from mail-40136.protonmail.ch (mail-40136.protonmail.ch.
[185.70.40.136])
        by mx.google.com with ESMTPS id n7-v6si8421705wma.39.2018.11.12.08.53.26
        for <jones.sally1993@gmail.com>
        (version=TLS1_2 cipher=ECDHE-RSA-AES128-GCM-SHA256 bits=128/128);
        Mon, 12 Nov 2018 08:53:27 -0800 (PST)
Received-SPF: pass (google.com: domain of jason_halloween@protonmail.com
designates 185.70.40.136 as permitted sender) client-ip=185.70.40.136;
```

Evidence 1: First received SMTP stamp on malicious email [Artifact 1]

This reveals a dead end because the mail provider guarantees the privacy of the sender by removing the client source IP. We can only backtrack the email sender back to the ProtonMail's ISP. A possible step is to request a warrant to get the information stored at ISP based on Message ID. It is possible that ISP store the clients IP and links them to the Messages ID but not expose that information publicly.

Memory Analysis

Using the tool `Volatility`, we checked for malicious process running on Sally's system and searching for suspicious processes.

Offset	Name	Pid	PPid	Uid	Start Time UTC +0000
0xffff91d5b56b5b00	main	14919	1211	1000	2018-11-12 17:15:45
0xffff91d5b56b0000	main	14921	14919	1000	2018-11-12 17:15:45

Artifact 3: Suspicious process running on Sally's computer

These two main processes started at the same time, and one was created by the first one. The Parent Process Id (**PPid**) shows this.

Using `Volatility` tool, we were able to discover that the main process was on Sally's `Downloads` folder. We suspect that the file was replaced because the file types and checksums don't match and `fls` tool shows that this file was `realloc`.

Pid	Uid	Gid	Path
14919	1000	1000	/home/sally/Downloads/main
14921	1000	1000	/home/sally/Downloads/main

We started by searching the two processes IDs in the memory dump looking for the string "**aes**". Our goal is trying to get the key from the memory. We know that malware was running during the memory snapshot, so it is very likely that the key is stored plaintext somewhere in the memory.

```
Task: main pid 14921 rule r1 addr 0x7f12819e52f8
0x7f12819e52f8  61 65 73 6e 69 00 00 00 01 00 00 00 00 00 00 00   aesni...........
0x7f12819e5308  00 e7 72 87 12 7f 00 00 0b 00 00 00 00 00 00 00   ..r.............
0x7f12819e5318  ff ff ff ff ff ff ff ff 00 00 00 00 22 20 3e 20   ............". >.
0x7f12819e5328  6b 65 79 2e 74 78 74 00 07 00 00 00 00 00 00 00   key.txt.........
0x7f12819e5338  00 e7 72 87 12 7f 00 00 08 00 00 00 00 00 00 00   ..r.............
0x7f12819e5348  8e 2a f6 69 cb fc 16 3f 01 00 00 00 45 78 70 65   .*.i...?....Expe
0x7f12819e5358  63 74 65 72 00 5f 00 00 01 00 00 00 00 00 00 00   cter._..........
0x7f12819e5368  00 e7 72 87 12 7f 00 00 08 00 00 00 00 00 00 00   ..r.............
0x7f12819e5378  df 8e f3 ac 91 59 79 1e 01 00 00 00 4d 4f 44 45   .....Yy.....MODE
```

Artifact 4: Search results of AES in memory dump

*(You can check the entire search result at file **yarascan.14921.txt**)*

We suspect that the key was stored (even temporarily) on a file named "key.txt".

Next step is to search for that file in memory dump. Using a `YaraScan` Volatility feature, we look for "**`key.txt`**" in all memory (and not in particular process ID), and the following result appears:

```
Task: main pid 14921 rule r1 addr 0x7f127fbfd2ee
0x7f127fbfd2ee  6b 65 79 2e 74 78 74 0d 0a 5b 50 45 58 50 45 43   key.txt..[PEXPEC
0x7f127fbfd2fe  54 5d 24 20 00 00 00 00 00 00 d8 d3 bf 7f 12 7f   T]$.............
0x7f127fbfd30e  00 00 00 e7 72 87 12 7f 00 00 3d 00 00 00 00 00   ....r.....=.....
0x7f127fbfd31e  00 00 ff ff ff ff ff ff ff ff 00 00 00 00 20 75   ..............u
0x7f127fbfd32e  6e 73 65 74 20 50 52 4f 4d 50 54 5f 43 4f 4d 4d   nset.PROMPT_COMM
0x7f127fbfd33e  41 4e 44 0d 0a 6a 61 73 6f 6e 40 6f 70 74 69 70   AND..jason@optip
0x7f127fbfd34e  6c 65 78 3a 7e 24 20 50 53 31 3d 27 5b 50 45 58   lex:~$.PS1='[PEX
0x7f127fbfd35e  50 45 43 54 5d 5c 24 20 27 0d 0a 00 00 00 00 00   PECT]\$.'.......
0x7f127fbfd36e  00 00 a0 d2 bf 7f 12 7f 00 00 00 e7 72 87 12 7f   ............r...
```

Artifact 5: "key.txt" Search results in memory dump

*(You can check the entire content of this result at file **yarascan.key.txt**)*

We also found what appears to be a UNIX based bash prompt "`jason@optiplex:~$ `" Our first guest relies on the hypothesis that **PS1** is a **bash environment variable**, so we decided to scan the memory looking for the content of this mysterious variable.

```
Task: main pid 14921 rule r1 addr 0x7f127fbfd355
0x7f127fbfd355  50 53 31 3d 27 5b 50 45 58 50 45 43 54 5d 5c 24   PS1='[PEXPECT]\$
0x7f127fbfd365  20 27 0d 0a 00 00 00 00 00 00 a0 d2 bf 7f 12      .'..............
0x7f127fbfd375  7f 00 00 00 e7 72 87 12 7f 00 00 3e 00 00 00 00   .....r.....>....
0x7f127fbfd385  00 00 00 ff ff ff ff ff ff ff ff 00 00 00 00 66   ...............f
0x7f127fbfd395  63 6e 74 6c 28 29 20 61 72 67 75 6d 65 6e 74 20   cntl().argument.
0x7f127fbfd3a5  33 20 6d 75 73 74 20 62 65 20 73 74 72 69 6e 67   3.must.be.string
0x7f127fbfd3b5  20 6f 72 20 72 65 61 64 2d 6f 6e 6c 79 20 62 75   .or.read-only.bu
0x7f127fbfd3c5  66 66 65 72 2c 20 6e 6f 74 20 69 6e 74 00 00 00   ffer,.not.int...
0x7f127fbfd3d5  00 00 00 40 d4 bf 7f 12 7f 00 00 00 e7 72 87 12   ...@.........r..
0x7f127fbfd3e5  7f 00 00 3e 00 00 00 00 00 00 00 ff ff ff ff ff   ...>............
0x7f127fbfd3f5  ff ff ff 00 00 00 00 65 63 68 6f 20 22 34 37 36   .......echo."476
0x7f127fbfd405  38 33 62 39 61 39 36 36 33 63 30 36 35 33 35 33   83b9a9663c065353
0x7f127fbfd415  34 33 37 62 33 35 63 35 64 38 35 31 39 22 20 3e   437b35c5d8519".>
0x7f127fbfd425  20 6b 65 79 2e 74 78 74 0d 0a 5b 50 45 58 50 45   .key.txt..[PEXPE
0x7f127fbfd435  43 54 5d 24 20 00 00 00 00 00 00 00 00 00 00 00   CT]$............
0x7f127fbfd445  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Artifact 6: Results of PS1 search on memory

*(You can check the entire content of this result at file **yarascan.PS1.14921.txt**)*

And Jackpot! We found interesting results **[Artifact 6]**. The **echo** command found contains, potentially, the key used by the ransomware to encrypt Sally's documents.

```
47 68 3b 9a 96 63 c0 65 35 34 37 b3 5c 5d 85 19
```

Evidence 2: AES symmetric key

The key has exactly **16 bytes**, which corresponds to **128-bit key** that we were trying to find.

The team put effort on developing a script to recover the files. We found a script on the Internet[1] that allowed us to recover the encrypted files.

We changed the script to ignore the first 128 bits of each encrypted file and used those bits as the initial value for the counter.

File Name	MD5 Checksum
AS_09125_050118150001_A03f00d0.png.encrypted	6e684be3134831bd07b81b165e28010f
AS_09125_050118150001_A03f01d0.png.encrypted	de7674d7a23907429d648b4e0eeec6f6
AS_09125_050118150001_A03f02d0.png.encrypted	fed81b29f036d7999e46d4c09821980a
AS_09125_050118150001_A03f03d0.png.encrypted	2020b659af4a0b68c43cb34564ce9b3e
AS_09125_050118150001_A03f04d0.png.encrypted	b3bbe351f2331f742391a3986eab8397
AS_09125_050118150001_A03f05d0.png.encrypted	3910fae2d5659c9ada25d71e8e1f3cde
Image_Processing_with_ImageJ.pdf.encrypted	add2eb2adabae079a7bfa3f59baf9235
paper_draft.txt.encrypted	284638519f010804036a263f03423a1e

Evidence 3: List of encrypted files

Executing the following command allowed us to recover all encrypted files:

```
python aes-ctr.py -d -i ../ENCRYPTED_FILE.png.encrypted -o
../FILE.png -k AES_KEY -iv COUNTER
```

Command 1: Example of decryption tool usage

[1] https://github.com/rdomanski/AES-CTR

File Name	MD5 Checksum
AS_09125_050118150001_A03f00d0.png	b58303dd6f4026663fb1aacaccf5bf94
AS_09125_050118150001_A03f01d0.png	1e33b87269c463474f68df10d95eb67b
AS_09125_050118150001_A03f02d0.png	defa8c84d13338cf83668cf44ccbe016
AS_09125_050118150001_A03f03d0.png	32de7caaac1e191febe5c7e4d48c839a
AS_09125_050118150001_A03f04d0.png	1a6093f96040770a97dd257a3d487231
AS_09125_050118150001_A03f05d0.png	f75baf3c3f4e06d14355133a6edae13b
Image_Processing_with_ImageJ.pdf	23f432689a13006cfe0e982f8ae71459
paper_draft.txt	aa4d4b8006c1941ffa3684f26747b696

Artifact 7: List of recovered files

Malware analysis

We suspect that the attachment (main) is a **remote access tool** (RAT) that allowed to wait and receive malicious instructions. This suspicion is based on test that we performed on the malware running on a controlled environment (VirtualBox). The malware opened two ports and did not performed encryption on baited PDF at Document Folder.

We tried to decompile the executable using Snowman tool, but no interesting results were found.

We also searched for malware evidence at: `/tmp/_MEIlXS6RU` but again no relevant information was found.

Attacker Identity

Sally received an email from someone called Jason Halloween. We started by looking for "**jason**" in the memory dump.

```
0x0107d5f9  4e 6f 76 20 31 32 20 31 36 3a 32 37 3a 30 39 20   Nov.12.16:27:09.
0x0107d609  32 30 31 38 20 66 72 6f 6d 20 31 39 34 2e 32 31   2018.from.194.21
0x0107d619  30 2e 32 32 39 2e 35 10 01 00 00 00 00 00 00 50   0.229.5........P
0x0107d629  01 00 00 00 00 00 00 20 0d 0a 57 65 6c 63 6f 6d   ..........Welcom
0x0107d639  65 20 74 6f 20 55 62 75 6e 74 75 20 31 34 2e 30   e.to.Ubuntu.14.0
0x0107d649  34 2e 35 20 4c 54 53 20 28 47 4e 55 2f 4c 69 6e   4.5.LTS.(GNU/Lin
0x0107d659  75 78 20 34 2e 34 2e 30 2d 33 31 2d 67 65 6e 65   ux.4.4.0-31-gene
0x0107d669  72 69 63 20 78 38 36 5f 36 34 29 0d 0a 0d 0a 20   ric.x86_64).....
```

Evidence 4: Results of "Jason" search on memory dump

*(You can check the entire content of this result at file **volshell.lastlogin.txt**)*

```
0x7f1280379d9d  4c 61 73 74 20 6c 6f 67 69 6e 3a 20 4d 6f 6e 20   Last.login:.Mon.
0x7f1280379dad  4e 6f 76 20 31 32 20 31 36 3a 32 37 3a 30 39 20   Nov.12.16:27:09.
0x7f1280379dbd  32 30 31 38 20 66 72 6f 6d 20 31 39 34 2e 32 31   2018.from.194.21
0x7f1280379dcd  30 2e 32 32 39 2e 35 38 0d 0d 0a 6a 61 73 6f 6e   0.229.58...jason
0x7f1280379ddd  40 6f 70 74 69 70 6c 65 78 3a 7e 00 00 00 00 00   @optiplex:~.....
0x7f1280379ded  00 00 00 70 9b 37 80 12 7f 00 00 00 00 00 00 00   ...p.7..........
0x7f1280379dfd  00 00 00 08 00 00 00 10 00 00 00 01 00 00 00 00   ................
```

Evidence 5: SSH welcome message

This appears to be a welcome message that is present in most of Linux distributions when we connect via SSH.

We were able to determine that Sally's Linux Version is Ubuntu 16.04.5 LTS by extracting the `/etc/issue` file. We can now compare the two versions (Sally's Version) with version found on **Evidence 4**. The IP address **`194.210.229.58`** is, probably, the last IP where **jason** logged in to that machine from his own computer! Let's take a closer look using `volshell`:

```
0x7f12819e5684  6c 6f 67 69 6e 00 6c 75 65 00 00 00 02 00 00 00   login.lue.......
0x7f12819e5694  00 00 00 00 00 e7 72 87 12 7f 00 00 08 00 00 00   ......r.........
0x7f12819e56a4  00 00 00 00 40 bd 14 29 1d 27 35 b6 01 00 00 00   ....@..).'5.....
0x7f12819e56b4  75 73 65 72 6e 61 6d 65 00 6c 79 00 01 00 00 00   username.ly.....
0x7f12819e56c4  00 00 00 00 00 e7 72 87 12 7f 00 00 09 00 00 00   ......r.........
0x7f12819e56d4  00 00 00 00 ff ff ff ff ff ff ff ff 00 00 00 00   ................
0x7f12819e56e4  72 65 6d 6f 74 65 2e 70 79 00 79 00 02 00 00 00   remote.py.y.....
0x7f12819e56f4  00 00 00 00 00 e7 72 87 12 7f 00 00 06 00 00 00   ......r.........
```

Evidence 6: Results of "login" search in memory dump

We even suspect that `remote.py` is a script that Jason used to connected to Sally's PC and get along with the encryption process.

We decided to try some combination of useful keywords in the context of main executable, and more interesting results appears:

```
0x00d3a25e  73 61 6c 6c 79 2f 44 6f 63 75 6d 65 6e 74 73 73   sally/Documentss
0x00d3a26e  0d 00 00 00 31 34 36 2e 31 39 33 2e 34 31 2e 35   ....146.193.41.5
0x00d3a27e  37 74 05 00 00 00 6a 61 73 6f 6e 74 0a 00 00 00   7t....jasont....
0x00d3a28e  66 72 69 64 61 79 31 33 74 68 63 00 00 00 00 01   friday13thc.....
0x00d3a29e  00 00 00 04 00 00 00 43 00 00 00 73 19 00 00 00   .......C...s....
0x00d3a2ae  74 00 00 6a 01 00 64 01 00 64 02 00 64 03 00 83   t..j..d..d..d...
0x00d3a2be  01 01 7d 00 00 7c 00 00 53 28 04 00 00 00 4e 73   ..}..|..S(....Ns
0x00d3a2ce  9c 01 00 00 49 74 27 73 20 48 61 6c 6c 6f 77 65   ....It's.Hallowe
```

Evidence 7: Result of "sally/" search in memory dump

*(You can check the entire content of this result at file **Sallybarra.14921.txt**)*

```
0x7f127fbf6cfc  48 61 6c 6c 6f 77 65 65 6e 20 6d 61 6c 77 61 72   Halloween.malwar
0x7f127fbf6d0c  65 20 2d 0a 20 20 20 20 65 6e 63 72 79 70 74 73   e.-.....encrypts
0x7f127fbf6d1c  20 72 65 63 75 72 73 69 76 65 6c 79 20 61 6c 6c   .recursively.all
0x7f127fbf6d2c  20 66 69 6c 65 73 20 6f 6e 3a 20 2f 68 6f 6d 65   .files.on:./home
0x7f127fbf6d3c  2f 73 61 6c 6c 79 2f 44 6f 63 75 6d 65 6e 74 73   /sally/Documents
0x7f127fbf6d4c  00 00 00 00 c8 9b bf 7f 12 7f 00 00 30 a1 c0 7f   ............0...
0x7f127fbf6d5c  12 7f 00 00 fd ff ff ff ff ff ff ff 00 00 00 00   ................
0x7f127fbf6d6c  00 00 00 00 01 00 00 00 00 00 00 00 c0 e3 72 87   ..............r.
```

*(You can check the entire content of this result at file **Halloween.14921.txt**)*

Evidence 8: Results of "Halloween" search in memory dump

We suspect that the malware was communicating with the **146.193.41.57** IP address. This is the first hint that leads to Jason's real identity. The following steps are discovering the ISP of this address and request a warrant for discover who is Jason.