



# EGO MARKET

When Greed  
for Fame Benefits  
Large-Scale Botnets

---

**Masarah Paquet-Clouston**

*Cybersecurity Researcher, GoSecure Inc.  
Master's Student in Criminology, Université de Montréal*

**David Décary-Héту**

*Assistant Professor, School of Criminology,  
Université de Montréal*

**Olivier Bilodeau**

*Cybersecurity Research Lead, GoSecure Inc.*

**Thomas Dupuy**

*Malware Researcher, ESET Canada Research*

# Introduction

Cybercrime is an evolving phenomenon and offenders are continuously developing new techniques to gain unauthorized access into computer systems. This paper will showcase just how ingenious botmasters have become by analyzing a specific botnet, the Linux/Moose botnet, and the illicit online market it thrives in. The Linux/Moose botnet was first discovered by the ESET research team in 2015 and their analysis of the botnet was published in a **technical report**. Following the publication of this report, the botnet operators<sup>1</sup> transitioned to a new version of their Linux/Moose botnet infrastructure, but essentially kept the same end-goal: social media fraud, which can be defined as the process of creating false endorsements of social networks accounts in order to enhance a user's popularity and visibility. This can be achieved by *liking* posts (or any similar endorsement) or *following* a user.

Linux/Moose stands out among all other botnets for three reasons. First, it is part of a new generation of Internet of Things (IoT) botnets that run on embedded systems such as routers rather than computers. It is therefore much stealthier and difficult to detect since no antivirus and little security software monitor these devices' traffic and behavior. Second, rather than sending instructions to the bots it has compromised, the botnet uses the bots only as proxies to hide the origin of the requests it sends to social media websites exclusively. The bots therefore do not need much computational power; their bandwidth and their "clean" IP address is what the botnet is after. Lastly, the botnet specializes in social media fraud, a very different activity from other botnets, which usually send spam, commit ad fraud and launch distributed denial of service (DDoS) attacks.

To investigate the Linux/Moose botnet, we infected several honeypots around the world. We performed a man-in-the-middle attack to decrypt the botnet's traffic, analyzed its operations and studied the illicit online market where social media fraud services are bought and sold. This paper presents the results of our months-long investigation and blends a technical understanding of the botnet with a social assessment of its activities. The first section presents the latest updates of the Linux/Moose botnet and the multiple steps that were required to successfully infect honeypots. It also presents how a man-in-the-middle attack (mitm) was performed on the botnet's traffic. The second section presents the botnet's activity on social media networks, the illicit online market in which social media fraud services are bought and sold and the potential revenue generated by the botnet.

---

<sup>1</sup> For simplicity and clarity, we use the term "botnet operators" in plural, but note that we do not know how many individuals are behind the botnet.

# Catching and Stripping Linux/Moose

This section covers the new features that were introduced in the Linux/Moose botnet following the 2015 ESET report. This is further described in a [blog post released by ESET](#). It also describes how the honeypots that caught the malware were built and how the botnet's traffic was successfully decrypted.

## Linux/Moose Features and Changes

### Uniquely Focused on IoT

The Linux/Moose malware is an embedded system threat with no x86 architecture variant; it can run only on embedded Linux systems using MIPS or ARM architectures. This focus on embedded systems is rare as even the well-known embedded threats like *LizardStresser* (Linux/Gafgyt) and [the recently discovered Mirai](#) have x86 variants. Since our industry's best tools are built to detect and clean x86 threats, we can only deduce that the botnet operators aimed to proactively stay under the radar.

### Operators Reacting to Publications

After ESET's 2015 [publication](#), the Linux/Moose's command and control (C&C) servers went dark. This was to be expected due to [the extensive press](#) that the report gathered and because the C&C servers were hosted on IP addresses hardcoded in the malicious binary files. There was therefore no easy way for the operators to relocate their infrastructure.

### Obfuscated C&C IP addresses

In September 2015, ESET was able to collect a new executable binary of what turned out to be a new version of Linux/Moose. The analysis of the new sample suggests that the botnet operators read the ESET report and modified their malware accordingly. This illustrates how easily offenders can adapt once their activities are exposed. The first modification in the malware relates to how it handles the IP address of the C&C server. The IP address is no longer located inside the binary. It is now passed as an argument to the executable. This means that out-of-context hunting by gathering files from Virus Total (VT) is no longer a viable means to track the botnet.

```
[...]  
# echo -n -e «\x00\x00\xe6\x13\x02\x00...» >> /tmp/crondd  
# chmod +x /tmp/crondd  
# /tmp/crondd 763473758  
Loading modules...  
Modules are loaded
```

**Figure 1** *Updated infection mechanism*

As Figure 1 shows, the IP address is not passed as an argument in its usual dot-decimal format in the new version of the malware. It is slightly obfuscated by being represented in a 32-bit Integer format. The IP address is also XORed with a static value. This prevents the extraction of the IP address of the C&C servers by analysts monitoring only the network traffic. Figure 2 presents Python code to convert a captured Integer value into a dot-decimal IPv4 address.

```
import socket, struct  
argument = 763473758  
argument = argument ^ 0xF789AC9E  
print(socket.inet_ntoa(struct.pack("<I", argument)))
```

**Figure 2** *Python script to decode the obfuscated IP addresses of C&C server*

### New Proxy Port

The botnet operators also updated the TCP port they use to proxy their traffic. The old SOCKSv4 port used by the botnet was 10073 and the new one is 20012. It would be tempting to scan the entire Internet on that port and fingerprint the SOCKSv4 signature in order to estimate the botnet's size. However, the bots are configured to respond only to IP addresses that are on a whitelist controlled by the C&C server. In retrospect, this was a logical evolution for the malware as monitoring network traffic on a single non-standard Internet port is very easy in a large-scale environment, like ISPs.

### Updated Bot Enrollment Process

Another significant change in the new version of the malware relates to the bot enrollment process. We found that simply mimicking an infected device and contacting the C&C was no longer sufficient to be considered as a bot by the botnet operators. The operators introduced correlation checks between the infecting party and the victim in their C&C server. This makes their tracking more tedious.

## A New HTTP-like Network Protocol

Finally, the network protocol between the bots and the C&C server was updated. It appeared to be completely redesigned. The previous one was raw binary over port 80 or 81, which is shady since HTTP is a plain text protocol, as shown in Figure 3a. The new protocol, shown in Figure 3b, cleverly mimics HTTP traffic on port 80 and the server displays a very common "It works!" page. However, under closer scrutiny, instead of reimplementing an entire new protocol, it appears the operators just added some layers above the old one so that it can hide in plain sight.

```
00000000 1c 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 .....
00000010 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 .....
00000020 00 00 00 00 00 00 00 00 .....
00000000 00 00 00 00 0a 00 00 00 b9 00 00 00 .....
00000010 05 00 00 00 0a 00 00 00 55 9f ed 6b 5d be 8c dd ..... U..k]...
00000020 58 02 00 00 00 00 00 00 00 00 00 00 39 00 00 00 X.....9...
00000030 4d f7 b1 1f 00 00 00 00 55 9f ed 6b 01 00 00 00 M.....U..k...
00000040 55 9f ed 6c 00 00 00 00 c0 a8 01 03 01 00 00 00 U..l.....
00000050 4d f7 b1 24 00 00 00 00 4d f7 b2 b1 00 00 00 00 M..$....M.....
00000060 55 9f ed 6f 00 00 00 00 6d c9 94 c9 00 00 00 00 U..O....m.....
00000070 6d c9 94 f1 00 00 00 00 d9 17 02 30 00 00 00 00 m.....0....
00000080 d9 17 07 85 00 00 00 00 5d be 8b 93 00 00 00 00 .....}.....
00000090 52 92 3f 0f 00 00 00 00 6d c9 94 88 00 00 00 00 R.?....m.....
000000A0 55 9f ed 6f 00 00 00 00 5d be 8f 3c 00 00 00 00 U..O....}<....
000000B0 5d be 8b 7b 00 00 00 00 6d ec 59 d0 00 00 00 00 ]..{....m.Y....
000000C0 5d be 8e 71 00 00 00 00 d9 17 02 4f 00 00 00 00 ]..q....0....
000000D0 cf f4 43 c1 00 00 00 00 d9 17 02 fb 00 00 00 00 ..C.....
000000E0 d9 17 02 fd 00 00 00 00 d9 17 02 fc 00 00 00 00 .....
000000F0 d9 17 02 1e 00 00 00 00 d9 17 02 2f 00 00 00 00 ...../....
00000100 d9 17 02 31 00 00 00 00 d9 17 02 34 00 00 00 00 ...l....4....
00000110 4f b0 1a 8e 00 00 00 00 d9 17 02 f9 00 00 00 00 0.....
00000120 5d be 8c dd 00 00 00 00 7f 00 00 01 00 00 00 00 ].....
00000130 1b 7c 29 1f 00 00 00 00 2a 77 ad 8a 00 00 00 00 .|)....*w.....
00000140 1b 7c 29 1f 00 00 00 00 1b 7c 29 21 00 00 00 00 .|)....|)!....
00000150 1b 7c 29 34 00 00 00 00 67 ee d8 15 00 00 00 00 .|)4....g.....
00000160 67 ee d8 1a 00 00 00 00 67 ee d8 1c 00 00 00 00 g.....g.....
00000170 67 ee d8 1d 00 00 00 00 67 ee d8 1e 00 00 00 00 g.....g.....
00000180 67 ee d8 1f 00 00 00 00 67 ee d8 17 00 00 00 00 g.....g.....
00000190 1b 7c 29 21 00 00 00 00 1b 7c 29 34 00 00 00 00 .|)!....|)4....
000001A0 1b 7c 29 0b 00 00 00 00 67 ee d8 16 00 00 00 00 .|)....g.....
000001B0 67 ee d8 18 00 00 00 00 67 ee d8 19 00 00 00 00 g.....g.....
000001C0 d9 17 0c 7c 00 00 00 00 6d ec 56 12 00 00 00 00 ...|....m.V....
000001D0 d9 17 07 d3 00 00 00 00 c0 7e b8 ea 00 00 00 00 .....~.....
000001E0 67 ee d8 d9 00 00 00 00 67 ee d8 d8 00 00 00 00 g.....g.....
000001F0 67 ee d8 da 00 00 00 00 08 00 00 00 06 00 00 00 g.....
00000200 54 03 1b 00 51 c2 06 00 00 00 54 03 1e 0d 59 c2 T...Q...T...Y.
00000210 0c 00 00 00 6c 03 08 0e 07 11 16 07 08 09 72 c2 ....l.....r.
00000220 08 00 00 00 43 3c 2a 06 16 17 4f c2 0c 00 00 00 ...C<*.0.....
```

**Figure 3a** Old protocol



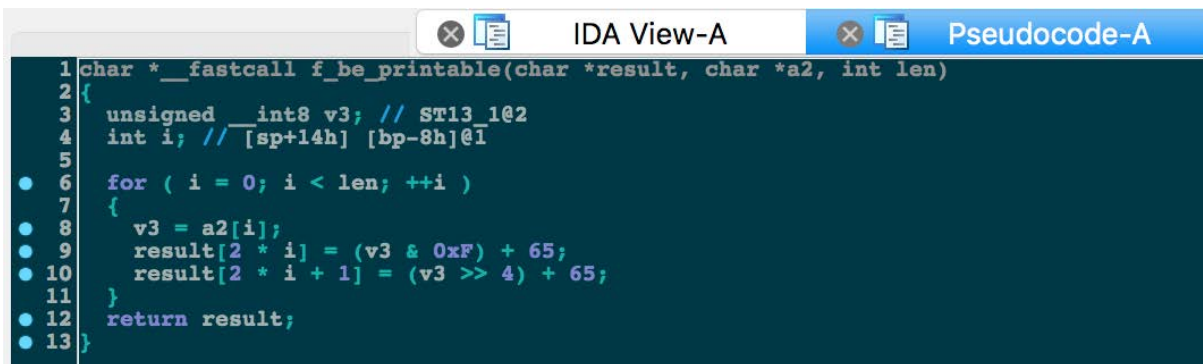
```

00000000 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a GET / HTTP/1.1..
00000010 48 6f 73 74 3a 20 77 77 77 2e 63 68 61 6c 6c 70 Host: ww w.challp
00000020 6f 6b 2e 63 6e 0d 0a 55 73 65 72 2d 41 67 65 6e ok.cn..U ser-Agen
00000030 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 t: Mozil la/5.0 (
00000040 57 69 6e 64 6f 77 73 20 4e 54 20 36 2e 32 3b 20 Windows NT 6.2;
00000050 72 76 3a 34 30 2e 30 29 20 47 65 63 6b 6f 2f 32 rv:40.0) Gecko/2
00000060 30 31 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f 0100101 Firefox/
00000070 34 30 2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 40.0..Ac cept: */
00000080 2a 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 *.Accep t-Langua
00000090 67 65 3a 20 65 6e 2d 55 53 2c 65 6e 3b 71 3d 30 ge: en-U S,en;q=0
000000a0 2e 35 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 .S..Acce pt-Encod
000000b0 69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66 6c 61 ing: gzi p, defla
000000c0 74 65 0d 0a 43 6f 6f 6b 69 65 3a 20 50 48 50 53 te..Cook ie: PHPS
000000d0 45 53 53 49 44 3d 42 43 41 41 41 41 42 41 42 41 ESSID=BC AAAABABA
000000e0 41 41 41 41 46 42 47 41 44 42 41 41 45 47 45 47 AAFAFBGA DBAAEGEG
000000f0 41 41 41 41 4d 4d 4d 4d 41 41 41 41 41 41 41 41 AAAAMMM AAAAAMMM
00000100 41 41 41 41 41 41 41 41 41 41 41 41 45 41 45 41 AAAAAAAA AAAAEAEA
00000110 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAA AAAAAAAA
00000120 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAA AAAAAAAA
00000130 41 41 41 41 50 50 3b 20 6e 68 61 73 68 3d 31 3b AAAAPP: nhash=1;
00000140 20 63 68 61 73 68 3d 30 0d 0a 43 6f 6e 6e 65 63 chash=0 ..Connec
00000150 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 tion: ke ep-alive
00000160 0d 0a 0d 0a .....
00000000 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.
00000010 0a 44 61 74 65 3a 20 54 75 65 2c 20 32 37 20 53 .Date: T ue, 27 S
00000020 65 70 20 32 30 31 36 20 31 34 3a 33 37 3a 32 32 ep 2016 14:37:22
00000030 20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20 41 70 GMT..Se rver: Ap
00000040 61 63 68 65 2f 32 2e 32 2e 32 32 20 28 55 62 75 ache/2.2 .22 (Ubu
00000050 6e 74 75 29 0d 0a 53 65 74 2d 43 6f 6f 6b 69 65 ntu)..Se t-Cookie
00000060 3a 20 50 48 50 53 45 53 53 49 44 3d 4b 41 41 41 : PHPS SID=KAAA
00000070 41 41 45 42 45 42 41 41 41 41 4a 43 4a 43 41 41 AAEBEBA AAJCJCAA
00000080 41 41 4b 41 4b 41 41 41 41 41 4b 41 4b 41 41 41 AAKAKAAA AAKAKAAA
00000090 41 41 41 4d 44 4d 4c 41 44 4e 4c 4e 41 41 41 41 AAAMDMLA DNLNAAAA
000000a0 41 41 49 46 4b 46 43 41 41 41 41 41 41 41 41 41 AAIFKFCA AAAAAAAA
000000b0 41 41 41 41 41 41 41 41 41 41 4c 49 4a 43 47 4a AAAAAAAA AALJCGJ
000000c0 48 4d 44 50 41 41 41 41 41 41 41 41 41 41 41 41 HMDPAAAA AAAAAAAA
000000d0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAA AAAAAAAA
000000e0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAA AAAAAAAA

```

**Figure 3b** New protocol

The botnet operators also wrapped the old binary protocol with some minor field positioning changes in the `Cookie:` and `Set-Cookie:` HTTP headers. However, those fields have restrictions on special characters so another layer of wrapping was required. Instead of using base64 they implemented some custom and easily reversible obfuscation. The function which decodes this obfuscation is presented in Figure 4.



```

1 char *__fastcall f_be_printable(char *result, char *a2, int len)
2 {
3     unsigned __int8 v3; // ST13_1@2
4     int i; // [sp+14h] [bp-8h]@1
5
6     for ( i = 0; i < len; ++i )
7     {
8         v3 = a2[i];
9         result[2 * i] = (v3 & 0xF) + 65;
10        result[2 * i + 1] = (v3 >> 4) + 65;
11    }
12    return result;
13 }

```

**Figure 4** Decryption routine

## Updated Yet Still the Same

Through all of these changes, the feature sets and goals of the malware didn't change. The Linux/Moose malware itself is still doing brute force login attempts on the Internet for weak Telnet credentials and is still mainly used to proxy traffic with its SOCKSv4 proxy. This enables it to leverage clean IP addresses in order to reach social networking sites without being flagged as spam or, at least, to reduce the chances of detection.

This evolution was easy to foresee as it is aimed at avoiding the **Indicators of Compromise (IoCs) released with the ESET paper**. Malware operators implemented the minimum number of changes to avoid detection and stay under the radar of the security community. We provide new IoCs in the form of file hashes, IP addresses, snort signatures and Yara rules in the **appendix section**.

## Building IoT Honeypots

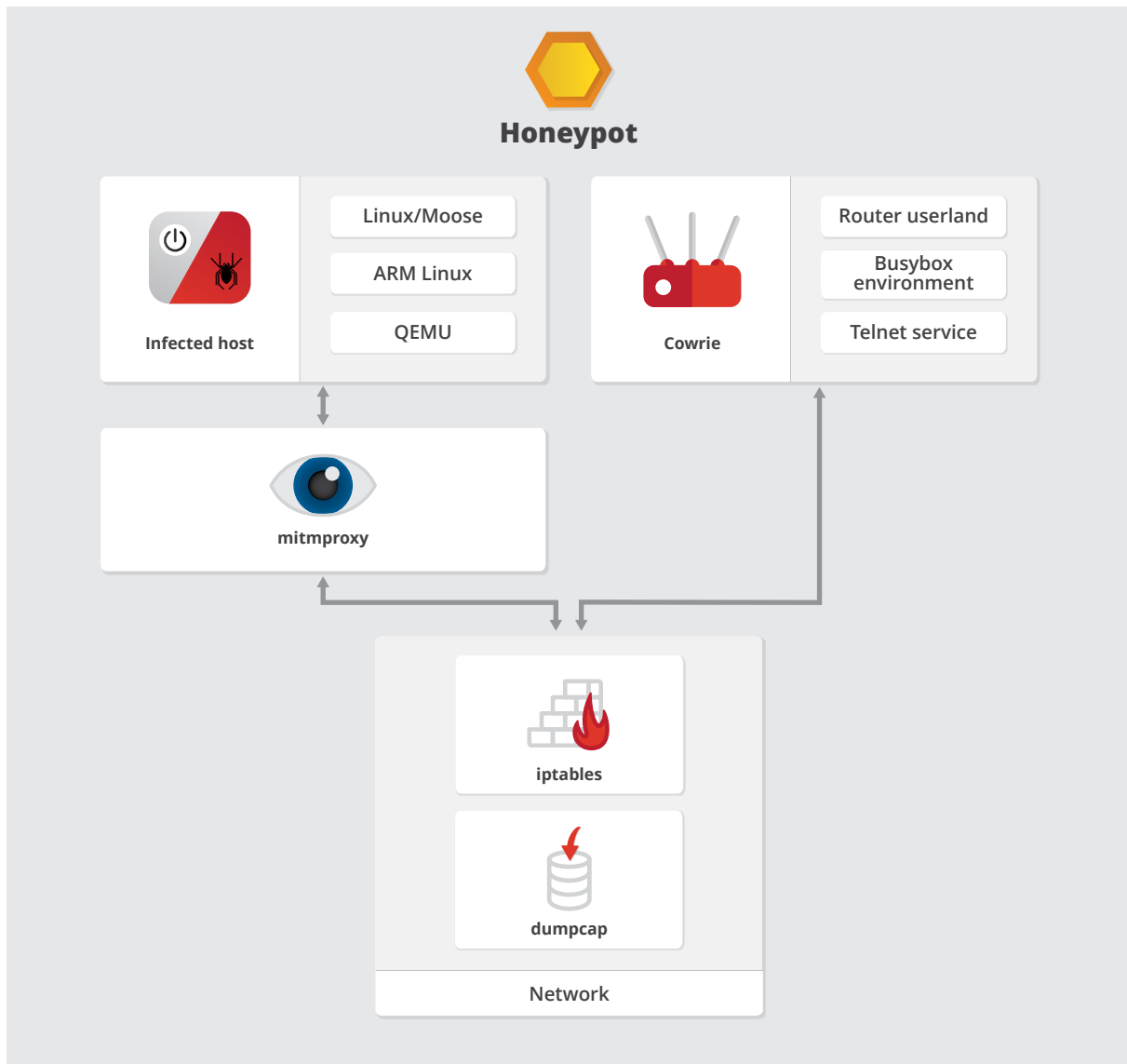
Monitoring and understanding the botnet's activities was a difficult task since the malware resides only on embedded Linux devices with no graphical user interfaces (GUIs). This limited our monitoring options to two approaches: infect dedicated hardware with out-of-band monitoring or create a honeypot package with a QEMU virtual machine to emulate the embedded device. We opted for the latter for a few reasons, easier management and the ability to remotely deploy on the cloud being the most important ones.

### IoT Honeypot Architecture

We built the honeypot environment by combining several existing tools and doing a little bit of development. First, we selected the Cowrie Honeypot and contributed its Telnet protocol support<sup>2</sup> enabling the malware to infect the honeypot with its usual credential brute forcing technique. Cowrie was configured to mimic a consumer router using the ARM architecture. The environment also included full packet capture performed by dumpcap, a QEMU setup with an ARM Linux distribution to execute the binaries and a mitmproxy install. Figure 5 displays a diagram of our honeypot's architecture.

---

<sup>2</sup> Details regarding our Telnet contribution to Cowrie can be found at: <https://github.com/micheloosterhof/cowrie/pull/222>



**Figure 5** *HoneyPot architecture*

### Deployed Worldwide

Several instances of this system were deployed worldwide. We monitored around twenty individual honeypots with varying levels of uptime over the course of several months. For safety reasons, the exposed Telnet was directed at the Cowrie honeypot and not at the QEMU system. We manually infected the QEMU systems with recent Linux/Moose samples. Once the QEMU systems were infected, traffic with the Command and Control (C&C) infrastructure was immediately observable and HTTPS proxy traffic started flowing in after a few hours of infection.

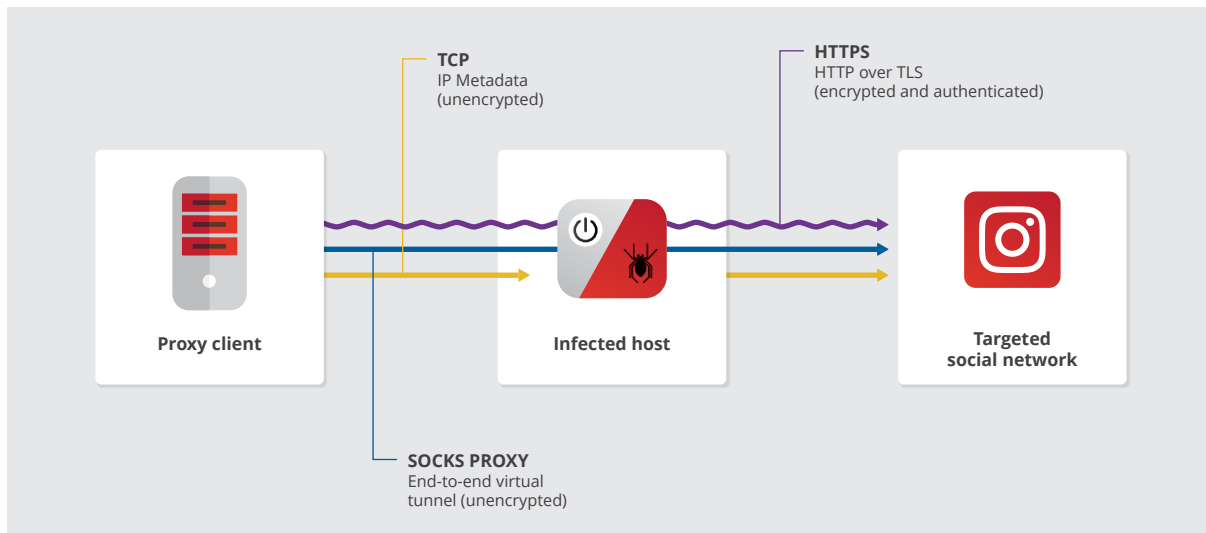


## Decrypting the Botnet's HTTPS Traffic

The Linux/Moose botnet's aim is to conduct social media fraud. To do so, the C&C needs to send requests to social media websites to create accounts, follow users and like their posts. Rather than instruct the bots to make these requests, the C&C infrastructure of the botnet sends all the requests itself and uses its bots only as proxies.

A proxy, by its nature, can be considered as an authorized man-in-the-middle attack since it acts as a middleman and forwards traffic. In this case, from the malware authors' perspective, the middleman is an infected process hosted on a distrusted server. In general, a man-in-the-middle attack consists of intercepting a traffic flow between two parties to eavesdrop or alter the conversation.

The use of bots as proxies obfuscates the true origin of the botnet's requests and is intended to bypass the anti-fraud countermeasures of social media websites. Figure 6 illustrates how the botnet operators leveraged the malware's proxy to hide the true source of the traffic.



**Figure 6** How the botnet uses its bots to relay traffic

Our honeypots relayed the botnet's proxy traffic for nine months at different time periods and with varying levels of activity. The proxy traffic consisted of a majority of TLS/SSL encrypted HTTP traffic, commonly referred to as HTTPS, that we needed to decrypt in order to understand the botnet's behavior. We attempted two types of man-in-the-middle attacks to decrypt the botnet's traffic.

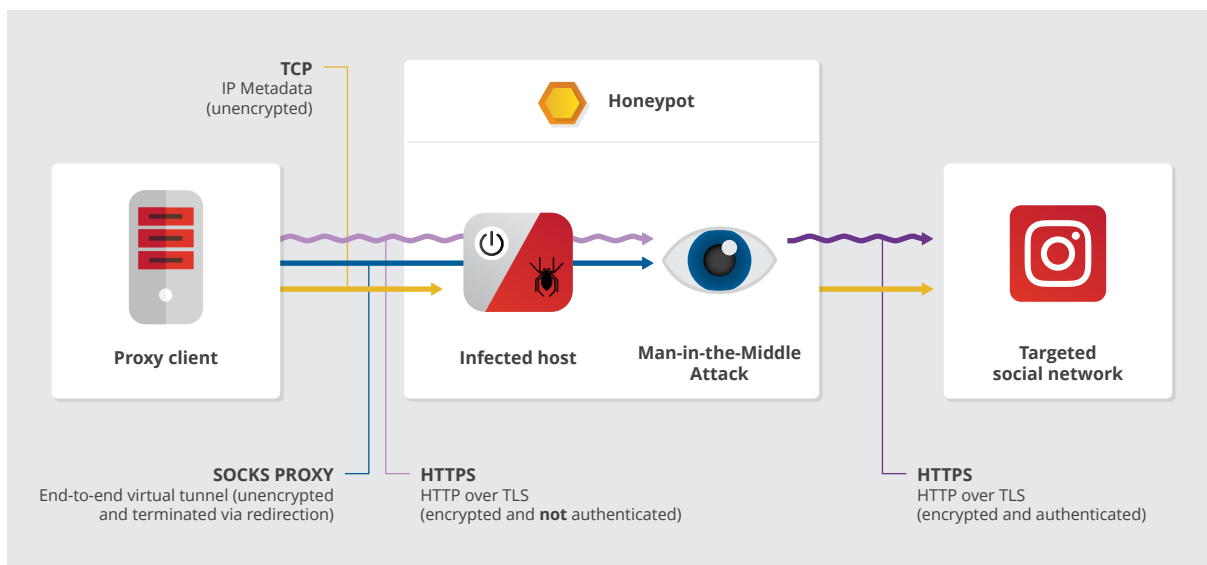
### SSL Stripping Attack

The first attack—known as an SSL/TLS stripping attack—was prioritized because it was less likely to generate errors on the client side and thus reduce the odds of tipping off the malware operators about our investigation. This technique intercepts the HTTPS requests originating from the botnet and rewrites their `Location:` header with `http://` rather than `https://`—hence the “stripping” attack. This should have in theory created unencrypted traffic between the botnet and the social media websites. However, the attack failed.

We believe this is due to the way the TLS upgrades are performed by social media websites. For every request, the social media websites replied with “Location:” headers that pointed to an “https://” URL, no matter if the initial request included an HTTP or HTTPS “Location:” header. This resulted in a request loop where our honeypots would request web pages over HTTP and only receive a redirect towards the same page but over HTTPS, thereby generating no real traffic data.

### Man-in-the-Middle Attack

The second attack sought to impersonate the social networks’ servers to the botnet. To do so, we terminated the TLS connection from the C&C to the social media websites at the honeypots and created a new connection from the honeypots to the social media websites. The following diagram depicts the attack, where the first HTTPS connection is terminated and a new one created:



**Figure 7** *Interception of HTTPS traffic*

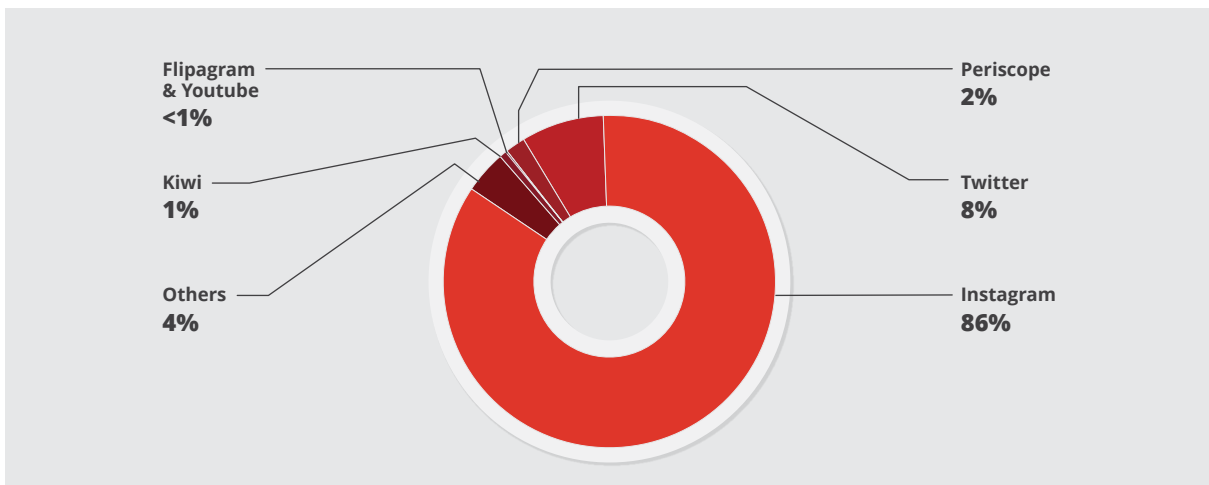
The HTTPS protocol is designed to prevent this type of attack through the use of certificates for authentication and the use of verification chains that are validated by browsers. Our attack on a browser would have raised a “Certificate Error” message since we replaced the social media websites’ certificates with our own that was generated on the fly and with no validation by a certificate authority. We expected that the botnet operators would care little about such errors considering that the Internet is a wild place where several issues can tamper with the certificates’ chain of validity. Our expectation turned out to be correct and the attack was successful. The attack gave us valuable information on the botnet’s activities, such as which websites were targeted and the requests made on each. This is how we confirmed that Linux/Moose was still conducting social media fraud. Indeed, even though we had strong assumptions that the botnet was conducting such fraud due to the common name (CN) fields extracted from the certificates present in the encrypted traffic, only the decrypted traffic could undeniably confirm them.

# Understanding Linux/Moose's Activities

Months of monitoring the Linux/Moose botnet and the illicit online markets where social media fraud services are advertised allowed us to survey the actions undertaken by the botnet and the ecosystem in which it evolves. Through this monitoring, we managed to define the type of fraud committed by the botnet, the profile of its customers as well as its potential revenue. The following section will present all three successively.

## The Targets of Linux/Moose

To begin, 95% of the traffic coming from the Linux/Moose C&C was directed towards various social network websites on which social media fraud was performed. This was achieved by sending requests to log in on the websites, create accounts and endorse other accounts. Figure 8 illustrates the overall distribution of the requests relayed by our honeypots from February to August 2016.



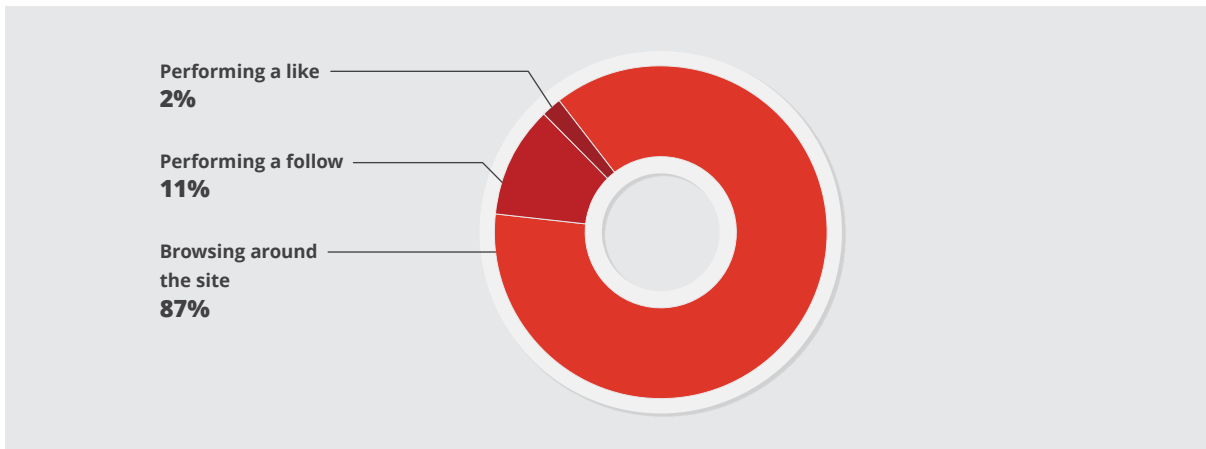
**Figure 8** *Distribution of Linux/Moose traffic on different social media websites*

About 86% of Linux/Moose traffic was directed towards Instagram, 8% towards Twitter, and less than 3% towards Periscope, Kiwi, Flipagram and YouTube. The *Other* category includes requests sent to Gmail or Yahoo for email account creation, requests on Websta—an Instagram viewer—and requests to lookup IP addresses.

The prominent proportion of requests directed towards Instagram was consistent across all our honeypots, indicating that most of the social media fraud services/sales orchestrated by the botnet is geared towards this social network. Instagram is a popular social network that crossed the **500 million users mark as of June 2016**. Its users mostly interact by sharing pictures and videos with their network of friends.

## Conducting Social Media Fraud on Instagram

Conducting social media fraud, at least on Instagram, is not a very straightforward activity. Indeed many “preliminary” requests need to be sent to the social media website before the botnet can perform a like or a follow. Of the thousands of requests relayed by our honey-pots, only 11% requested to follow an account and 2% to like a post, as shown in Figure 9. The vast majority of requests (87%) sought to legitimize the follow and like requests sent by the botnet and evade the anti-spamming filters.

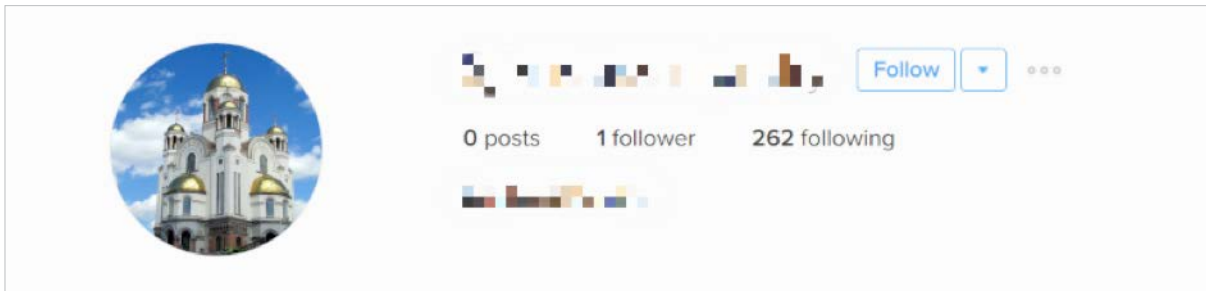


**Figure 9** *Distribution of requests on Instagram*

We found various *modus operandi* that the botnet employed to evade the anti-spamming filters and appear more “human”. Before following an account, the botnet would first log in on one of its accounts. It would then surf the social media website, making a dozen requests where the botnet searched for and viewed the account that will be followed, visited the feed of its own account and visited its inbox. These preliminary requests seem to be necessary to ensure that the botnet’s requests are not blocked by Instagram. They also appear to be quite successful as the Linux/Moose botnet’s success rate when attempting to follow accounts is 89%, thereby demonstrating the ability of the botnet to deliver on its promises to its customers.

### Linux/Moose’s Ephemeral Fake Accounts

While the Linux/Moose botnet has an effective method for following users and liking posts, its ability to keep its Instagram account active is much more limited. We found that out of the 1,732 Instagram accounts Linux/Moose used during the six months of the study, 1,247 (72%) were suspended by Instagram. Linux/Moose operators therefore appear to be investing a lot in their ability to follow accounts and like posts but much less in maintaining these accounts and actions through time. The botnet’s customers are likely to see a sharp decline in their followers and likes over time as Instagram removes the spamming content. The rate of account suspension may be due to the lack of content on Linux/Moose’s accounts.



**Figure 10** *Example of Linux/Moose fake accounts*

Linux/Moose's accounts use random numbers and letters as their usernames; plants, buildings, landscapes or animals as profile pictures; have no posts or followers; and follow up to 822 accounts. Surprisingly, the accounts controlled by the botnet don't even try to follow each other. Flagging such accounts as spam appears to us to be easy since little effort is made to make them look legitimate. The botnet operators appear not to be interested in ensuring the longevity of their accounts; their services are therefore rather ephemeral. It is possible that the fastidious nature of following and liking posts are taking time away from creating more legitimate-looking accounts. Customers of the Linux/Moose botnet should therefore not expect to keep their followers and likes for long.

## Linux/Moose's Market

Given the unknown size of the Linux/Moose botnet, it was not possible to estimate the number of customers that purchased social media fraud services from it. We still managed to identify a sample of customers from our honeypot's traffic, which is presented in the following section. These customers found the service on any of thousands of websites that advertise social media fraud and our analyses also include a survey of these websites. The survey includes an up-to-date evaluation of the costs of purchasing social media fraud services on Instagram. Finally, using the data gathered throughout the investigation, we present the potential revenue for each of the Linux/Moose bots.

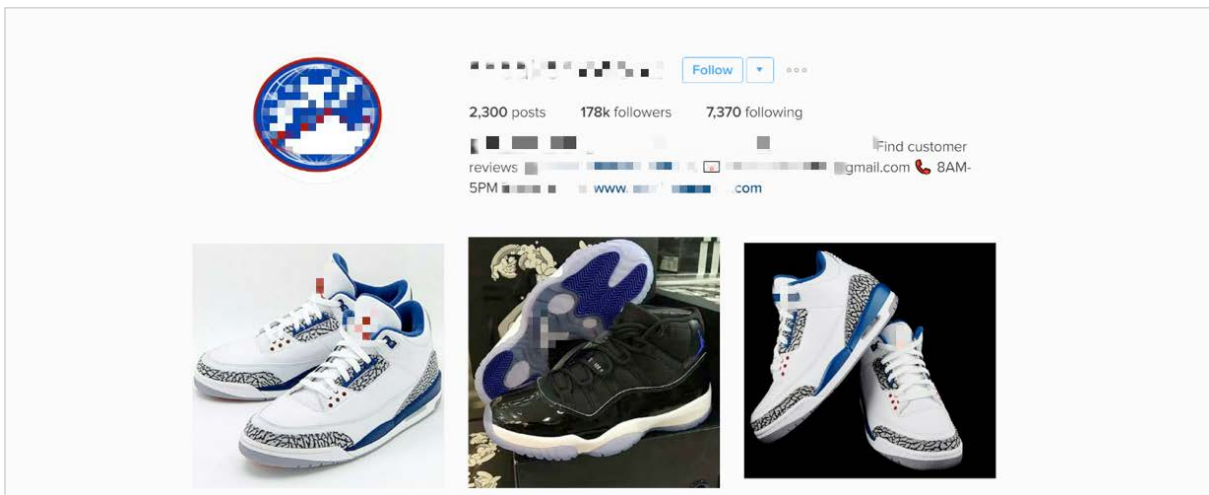
### Customers of Social Media Fraud: The EGO Market

When looking for customers, we focused solely on the accounts which were followed by the botnet. As the botnet sometimes followed celebrities to legitimize its activities, it was not possible to simply extract, from the traffic, the names of all of the accounts that had been followed by the botnet. Instead, the ratio between the average number of likes for posts and the number of followers was generated for each account. An account with 150,000 followers who posted a picture that had fewer than 50 likes or comments was a signal that the account's followers were likely fake and coming from the botnet. Due to time constraints, 500 profiles were investigated and 450 (90%) of them matched our criterion for being a customer.

Through our analysis, we found business-related accounts, among which were many:

- magazines,
- interior design enterprises,
- food-related companies,
- car shops,
- clothes shops,
- makeup shops,
- electronic shops, and
- shoe shops (as shown in Figure 12).

Most of these businesses were either local and established in a specific city or present solely on the web, with no shopping establishment customers could refer to. Overall, we found that most of the business profiles flagged as potential customers were not large corporations or well-known people.



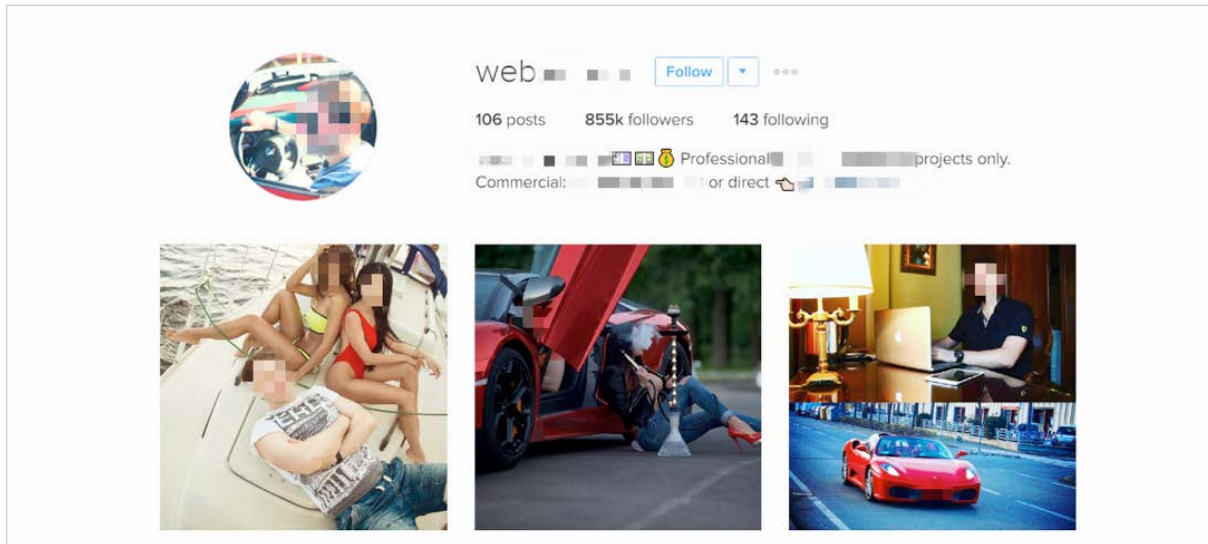
**Figure 11** *Example of a business that potentially bought social media fraud*

Also, we found many accounts that were business-related, but centered around individuals. For example, we found multiple:

- hairdressers,
- tattooists,
- photographers,
- entrepreneurs,
- businessmen, and
- web designers.



Figure 11 shows the profile of a web designer looking for business but centering his profile on his own life instead of the service he offers.



**Figure 12** Example of a profile from an individual who potentially bought social media fraud

Among profiles owned by individuals, there were also many aspiring celebrities such as:

- models,
- singers,
- musicians,
- actors,
- bloggers,
- television presenters, and
- designers.

Finally, we found many individuals who potentially bought social media fraud for their own benefit, sharing pictures about their life, the food they ate, where they went and the like. Most of the profiles owned by individuals were self-centered.

Thus, many accounts were related to individuals and highly self-centered, which could be a consequence of the Instagram social network allowing only posts of pictures or videos. We decided to call the market we found through the analysis of the Linux/Moose botnet potential buyers, the *EGO Market*. The botnet provides a service mostly to individuals or businesses looking to increase their public profile and present themselves as famous and well-known when, in reality, they might not be. The market demand for this fraud is shaped by people's greed for fame, from which large-scale botnets profit.

### Supply of Social Media Fraud

Assessing the type of customers of social media fraud indicated that most potential buyers were normal people who probably bought the service through a website found using a search engine like Google. Using Google, we found websites, freelancer platforms and SEO forums where social media fraud was offered. Due to the profiles of the customers we found in the traffic, we decided to focus on the websites and one freelancer platform that were easily accessible through Internet searches.

We found hundreds of websites and numerous ads on freelancer platforms offering different social media fraud services. Follows and likes—or similar endorsements—were offered on all sorts of social media websites such as Facebook, YouTube, Instagram, Twitter, LinkedIn, Vines, Pinterest, Kiwi, Periscope and more.

Most websites offered a specific price for each service based on (1) the quantity offered and (2) the social network for which the service was sold. The services were offered in “bundles” that accounted for specific quantities, ranging from 100 follows/likes up to millions. The price per service decreased as the quantity offered in the bundles increased. Visually, most websites gathered the services offered on a social media website and their subsequent prices into one specific page, as shown in Figure 13. Most websites were quite diversified, with 60% offering services on more than three different social networks.

100 Followers	500 Followers	1,000 Followers	5,000 Followers	10,000 Followers	50,000 Followers
<b>\$2.95</b>	<b>\$6.95</b>	<b>\$9.95</b>	<b>\$39.95</b>	<b>\$64.95</b>	<b>\$249.95</b>
Instant delivery guaranteed	Instant delivery guaranteed	Instant delivery guaranteed	Instant delivery guaranteed	Instant delivery guaranteed	Instant delivery guaranteed
Quality profiles	Quality profiles	Quality profiles	Quality profiles	Quality profiles	Quality profiles
100% safe	100% safe	100% safe	100% safe	100% safe	100% safe
Buy Now	Buy Now	Buy Now	Buy Now	Buy Now	Buy Now

**Figure 13** *Example of Prices for Instagram followers bundles*

The freelancer platform differed from the websites as multiple providers of social media fraud could post all sorts of ads related to social media fraud. We found 94 sellers advertising for social media fraud on a single freelancer platform we monitored in February 2016. They offered—via their ads—social media fraud for specific social media websites, such as: “I will Add 3000 Permanent FACEBOOK Likes” or “I will add 700 google circles or followers”. Aggregating all sellers according to their ads, we found that 54% of freelancers offered services on only one single social network and 75% of them offered services on one or two social networks. Most of these sellers were therefore quite specialized. This suggests freelancers and websites differ as freelancers’ ads are more specialized and websites are more diversified.

### Social Media Fraud Prices

Since the Linux/Moose botnet specialized in social media fraud on Instagram, we decided to present in this paper only the prices related to that social media website. Mixing both websites and freelancers, we gathered information on 84 suppliers of social media fraud on Instagram. The number could, however, be *de facto* lower since one supplier may use many platforms to gain more visibility while selling online. Average prices for follows, likes and comments on Instagram are presented in the table below.

**Table 1** *Price distribution of social media fraud on Instagram*

	AVERAGE PRICE (US\$)	STANDARD DEVIATION
<b>1,000 Follows</b>	\$15.98	18.87
<b>10,000 Follows</b>	\$112.67	103.29
<b>1,000 Likes</b>	\$19.54	25.06
<b>10,00 Likes</b>	\$158.99	174.45
<b>100 Comments</b>	\$72.25	52.51

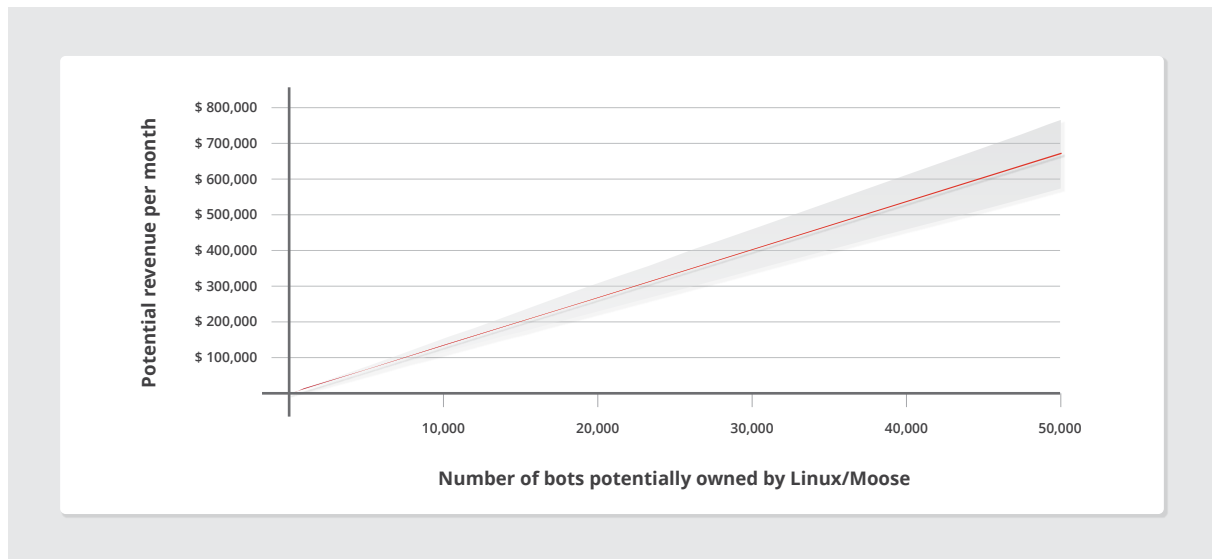
The price for 1,000 follows averaged \$15.98 (\$0.016/follow). This was 25% more expensive than the price per follow with a purchase of 10,000 follows (\$112.67 in total, \$0.011/follow). The price for likes is slightly higher to that of follows with an average price for 1,000 likes of \$19.54 (\$0.020/like) and of \$158.99 (\$0.016/likes) for 10,000 likes. Instagram comments are also offered but at the much higher price of \$72.25 (\$0.723/comment) for 100 comments. Comments may be more expensive due to the sophisticated nature of having to comment on a picture instead of just liking or following, which can be automated more easily.

We find that the prices for social media fraud on Instagram are unstable, since the standard deviation in the prices of follows, likes and comments is very high. The variation in prices may indicate that the market is young and underdeveloped. If prices are higher on some websites, it is possibly because buyers don't know the worth of the service they are buying and are ready to pay higher prices. Moreover, sellers may also not know the worth of the services they offer; some sellers therefore overprice while others underprice.

### Linux/Moose Bot's Potential Revenue

Assessing the potential profitability of botnets is a difficult task and although it is believed that online criminals make large amounts of money, little empirical data support these statements. With the data gathered, we could assess the potential revenue generated by a bot, based on the average price presented above. We took the average number of follows performed by seven bots over a whole month and weighted this average according to the proportion—found above—of profiles that matched our buyer criterion. Thus, we found that a bot performs, on average, 1,186 follows on Instagram per month. Considering

that the price per follow is USD \$0.011, the potential revenue of a Linux/Moose bot is \$13.05 per month. Below, Figure 14 shows the potential revenue made by the botnet according to the number of bots it has and assuming all follows performed on buyers' profiles are monetized. The grey area delimits 95% confidence intervals based on the mean around the price, due to the high standard deviation found in prices.



**Figure 14** *Linux/Moose Potential Revenue per Month*

Figure 11 shows that the botnet's potential profitability is high. With a large botnet and all follows monetized, the operators of Linux/Moose are sitting on a gold mine. However, as mentioned, the interpretation of the graph needs to be mitigated, because we do not know the proportion of follows that were actually monetized by the botnet operators.

# The Clever Scheme Behind Linux/Moose

The Linux/Moose botnet is the latest example of how innovative online offenders have become, developing new ways to profit from their illicit online activities. This paper shows how the Linux/Moose botnet operators developed a scheme to make money without attracting the attention of law enforcement. Their strategy combines a stealth infrastructure, constant adaptation, the inconspicuous activity of social media fraud with no direct victims, the ability of hiding in plain sight and a large potential profitability.

## Stealthy

Offenders continuously adapt, developing new stealth techniques to conduct their illicit online activities and avoid detection by the authorities. By targeting IoT devices, the Linux/Moose botnet is a good example of such adaptation. The lack of antivirus and security software available for these devices allows the botnet to spread and it is therefore very difficult to detect in the wild. Although most botnets still run on the Windows platform, more and more botnets are taking advantage of IoT devices, which include routers and home appliances. Botnets are also targeting Android phones, which have powerful processors and are always connected to the Internet. Through adaptation, the Linux/Moose botnet operators have likely managed to ensure the survival of their botnet and are representative of forward-looking cyberoffenders. Moreover, the fact that the botnet can run only on embedded systems, with no x86 architecture variant, shows how the operators proactively try to stay under the radar.

## Constantly Adapting

We also found that the operators of the Linux/Moose adapted many features of their botnet. A notable one is how the bot enrollment process was changed. We found that simply mimicking an infected device and contacting the C&C was no longer sufficient to be considered as a bot by the botnet operators. The operators introduced correlation checks between the infecting party and the victim in their C&C server. This behavior highlights the cautiousness of the botnet operators and the fact that they are aware of the monitoring of their activities by the security community. This is probably a result of the ESET research paper which motivated the botnet operators to up their game and develop new defense mechanisms. This suggests that the operators are watching what the industry is publishing or presenting and, as a result, they are more likely to maintain their illicit activities in the long run.

## No Direct Victims

While the botnet is stealthy because of the way its infrastructure was designed, it is also stealthy due to the trivial nature of what it does: social media fraud. Botnets are commonly known for distributed denial-of-service attacks (DDoS), sending spam or contributing to ad click

fraud. We found that the Linux/Moose botnet operators are branching out to new and less risky activities. Social media fraud does not generate victims in need of defending and is thus not an attractive target for law enforcement or the international security industry. The victims of social media fraud are the entities or people that use the number of followers and likes as a measure of worth of a profile, such as record labels or movie studios who offer contracts to prospective stars based on their social media following. Those buying advertisements on social networks based on these metrics are also victims of social media fraud, as they pay for an exposure that is, in reality, much less than what it is supposed to be. A quick investigation of the response generated by each post, as demonstrated in this paper, should help them differentiate the accounts with a real following from those with bought followers. However, as mentioned, instead of requiring law enforcement intervention, these victims are more likely to be more cautious in future transactions, thereby giving a free pass to the botnet operators.

### Hiding in Plain Sight

The possibility of openly advertising online gives the botnet operators involved in social media fraud a large exposure. Instead of selling their botnet services on underground forums or cryptomarkets like Silk Road, the Linux/Moose botnet operators can advertise on the clear Web and reach a large pool of customers that are not criminalized. Indeed, we found that the demand for social media fraud on Instagram consisted of normal people and small businesses looking to increase their fame and credibility through social media. The market in which the Linux/Moose botnet evolves was named EGO market to illustrate how the market is driven by people's greed for online fame. The EGO market benefits large-scale botnets, since the services can be sold online to anyone with a need for attention.

### Large Potential Profitability

Our investigation also shows that providing the service requires several steps. For the Linux/Moose botnet to provide 10,000 follows, it first needs to create 10,000 email accounts and then to register 10,000 accounts on Instagram. In addition, the Instagram social network constantly takes measures to detect the botnet's fake accounts and suspends them. The botnet operators therefore need to actively develop strategies to ensure success in their operations. This may explain the optimization we noticed in the Linux/Moose botnet infrastructure where the botnet undertakes many actions to ensure the follows are successful but little action to ensure its accounts last. The service the botnet provides is ephemeral. Yet, if all follows are monetized, irrespective of whether the follows last or not, Linux/Moose potential for revenue is quite large. The value generated by a bot can go up to \$13.05 per month. To what extent the fraud is profitable for the botnet operators is however unclear. It depends on several factors such as the number of follows monetized, the size of the infrastructure and the number of actors behind it.



# Conclusion

Regardless of the botnet's overall profitability, the Linux/Moose botnet evolves in an inconspicuous EGO market that is driven by normal people and does not attract the attention of law enforcement. This is the ideal situation for illicit online activity: running a stealthy, profitable botnet while advertising the services on the clear Web and selling them to normal people. No connection to the criminal underworld is needed and there are no direct victims, yet the money is generated through illicit activities. The Linux/Moose botnet participates in a clever scheme: it falls into an interstice that allows the botnet operators to continuously commit a computer crime and profit from it in total impunity.

*This research was partially financed by the MITACS Accelerate Program, under Project N. IT06861. We thank them for financing the project and our partners, Université de Montréal and ESET, for helping us throughout the year.*

# Appendix

## Indicators of Compromise (IoC)

The most up-to-date IoCs for Linux/Moose are available here:  
<https://github.com/eset/malware-ioc/tree/master/moose>

### File Hashes

The following are the SHA1 hashes and internal malware versions of the files we've encountered:

```
0x1F
c6edfa2bf916d374e60f1b5444be6dbbee099692
c9ca4820bb7be18f36b7bad8e3044b2d768a5db8
5b444f1ac312b4c24b6bde304f00a5772a6a19a4
f7574b3eb708bd018932511a8a3600d26f5e3be9
0x20
34802456d10efdf211a7d486f7108319e052cd17
0685cb1d72107de63fa1da52930322df04a72dbc
2876cad26d6dabdc0a9679bb8575f88d40ebd960
f94b6cc5aea170cee55a238eaa9339279fba962f
274ef5884cb256fd4edd7000392b0e326ddd2398
c3f0044ffa9d0bc950e9fd0f442c955b71a706b6
f3daea1d06b1313ec061d93c9af12d0fe746839a
0x21
7767c8317fb0bbf91924bddffe6a5e45069b0182
1caac933ae6ca326372f7e5dd9ff82652e22e34
5dea6c0c4300e432896038661db2f046c523ce35
e8dc272954d5889044e92793f0f637fe4d53bb91
0843239b3d0f62ae6c5784ba4589ef85329350fa
1d1d46c312045e17f8f4386adc740c1e7423a24a
d8b45a1114c5e0dbfa13be176723b2288ab12907
0x22
c35d6812913ef31c20404d9bbe96db813a764886
```

### IP Addresses

#### Primary C&C servers

```
192.3.8.218
192.3.8.219
```

#### Whitelist

```
155.133.18.64
178.19.111.181
151.80.8.2
151.80.8.19
151.80.8.30
62.210.6.34
```

## Yara Rules

```
rule linux_moose_v2: mips arm
{
    meta:
        Author = "Thomas Dupuy"
        Date = "2016/10/07"
        Description = "Linux/Moose Malware Version 2"
        License = "BSD 2-Clause"

    strings:
        // - ELF- MIPS
        $header_mips = { 7F 45 4C 46 [14] 00 08 }
        // - ELF- ARM
        $header_arm = { 7F 45 4C 46 [14] 28 00 }
        $xor_key = { 9E AC 89 F7 }
        $xor_key2 = { F7 89 AC 9E }
        $s1 = "echo -n -e "
        $s2 = "password:"
        $s3 = "bad password"
        $s4 = "login error"
        $s5 = "password is wrong"
        $s6 = "authorization failure"
        $s7 = "elan"
        $s8 = "/proc/%s/cmdline"
        $s9 = "kill %s"
        $s10 = "/home/hik/start.sh"
        $s11 = "H3lL0WoRlD"
        $s12 = "Modules are loaded"
        $s13 = "Set-Cookie: PHPSESSID="
        $s14 = "Set-Cookie: LP="
        $s15 = "Set-Cookie: WL="
        $s16 = "Set-Cookie: CP="

    condition:
        $header_mips at 0 and ($xor_key or $xor_key2)
        or
        $header_arm at 0 and ($xor_key or $xor_key2)
        or
        3 of them
}
```

## Snort / Suricata signatures

The fine folks at Emerging Threats designed Snort and Suricata rules to detect Linux/Moose and included them in their open ruleset.

```
#suri 3.1
```

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Moose
CnC Request M1"; flow:to_server,established; urilen:1; content:"GET";
http_method; content:!"Referer|3a 20|"; http_header; content:"PP|3b
20|nhash="; fast_pattern:only; content:"PHPSESSID="; http_cookie;
content:"AAAAAAAAAAAAAAAA"; http_cookie; distance:0; content:"PP|3b
20|nhash="; http_cookie; distance:0; content:"|3b 20|chash="; http_
cookie; distance:0; classtype:trojan-activity; sid:5002576; rev:1;)
```

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET TROJAN Moose
CnC Response"; flow:from_server,established; content:"200"; http_stat_
code; content:"PP|3b 20|expires="; fast_pattern:only; content:"PHP-
SESSID="; http_cookie; content:"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";
http_cookie; distance:0; content:"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";
http_cookie; distance:0; content:"PP|3b 20|expires="; http_cookie;
distance:0; content:"WL="; http_cookie; content:"PP|3b 20|expires=";
http_cookie; distance:0; content:"Content-Type|3a 20|text/html";
http_header; file_data; content:"<html><body><h1>It works!</h1>"; no-
case; depth:30; classtype:trojan-activity; sid:5002577; rev:1;)
```

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Moose
CnC Request M2"; flow:to_server,established; urilen:1; content:"GET";
http_method; content:!"Referer|3a 20|"; http_header; content:"PHPSES-
SID="; http_cookie; content:"|3b 20|nhash="; http_cookie; distance:0;
content:"|3b 20|chash="; http_cookie; distance:0; pcre:"/^Host\x3a\
x20[^\r\n]+\r\nUser-Agent\x3a\x20[^\r\n]+\r\nAccept\x3a\x20[^\r\
n]+\r\nAccept-Language\x3a\x20[^\r\n]+\r\nAccept-Encoding\x3a\x20[^\
r\n]+\r\nConnection\x3a\x20[^\r\n]+[^\r\n]+$/Hmi"; classtype:tro-
jan-activity; sid:5002610; rev:1;)
```

```
#snort 2.9.6
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"ET TROJAN  
Moose CnC Request M1"; flow:to_server,established; urilen:1; content:"-  
GET"; http_method; content:!"Referer|3a 20|"; http_header; content:"P-  
P|3b 20|nhash="; fast_pattern:only; content:"PHPSESSID="; http_cookie;  
content:"AAAAAAAAAAAAAAAA"; http_cookie; distance:0; content:"PP|3b  
20|nhash="; http_cookie; distance:0; content:"|3b 20|chash="; http_  
cookie; distance:0; classtype:trojan-activity; sid:5002576; rev:1;)
```

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"ET TRO-  
JAN Moose CnC Response"; flow:from_server,established; content:"200";  
http_stat_code; content:"PP|3b 20|expires="; fast_pattern:only;  
content:"PHPSESSID="; http_cookie; content:"AAAAAAAAAAAAAAAAAAAAAA  
AAAAAA"; http_cookie; distance:0; content:"AAAAAAAAAAAAAAAAAAAAAA  
AAAAAA"; http_cookie; distance:0; content:"PP|3b 20|expires="; http_  
cookie; distance:0; content:"WL="; http_cookie; content:"PP|3b 20|ex-  
pires="; http_cookie; distance:0; content:"Content-Type|3a 20|text/  
html"; http_header; file_data; content:"<html><body><h1>It works!</  
h1>"; nocase; depth:30; classtype:trojan-activity; sid:5002577;  
rev:1;)
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"ET TRO-  
JAN Moose CnC Request M2"; flow:to_server,established; urilen:1;  
content:"GET"; http_method; content:!"Referer|3a 20|"; http_header;  
content:"PHPSESSID="; http_cookie; content:"|3b 20|nhash="; http_  
cookie; distance:0; content:"|3b 20|chash="; http_cookie; distance:0;  
pcree:"/^Host\x3a\x20[^\r\n]+\r\nUser-Agent\x3a\x20[^\r\n]+\r\nAc-  
cept\x3a\x20[^\r\n]+\r\nAccept-Language\x3a\x20[^\r\n]+\r\nAccept-En-  
coding\x3a\x20[^\r\n]+\r\nConnection\x3a\x20[^\r\n]+[^\r\n]+$/Hmi";  
classtype:trojan-activity; sid:5002610; rev:1;)
```