How to footprint, report and remotely secure compromised IoT devices

Filippo Lauria - filippo.lauria@iit.cnr.it
Institute for Informatics and Telematics, Italian National Research Council
via G. Moruzzi, 1 - 56124 Pisa, Italy

Abstract
The increase of IoT consumer devices connected to the Internet, along with the lack in cybersecurity awareness among their owners, has established a breeding ground for new malware families targeting IoT devices. An example of IoT malware is the MIRAI botnet, responsible for the biggest DDoS ever detected in the history of the Internet, occurred on October 2016. Since then, many MIRAI variants have appeared, competing among themselves in order to infect as many as possible IoT devices. This article describes a software infrastructure, called Fooresec, which aims to remotely footprint, report and secure those devices. It also presents experimental results, produced from the data collected so far by using a working prototype of the proposed infrastructure, which indicate the validity of the approach.

Overview
The number of IoT consumer devices (e.g. security cameras, projectors, refrigerators, etc.) connected to the Internet is constantly increasing. It has been estimated that 3.9 billion IoT consumer devices were in use in 2016 and that, by 2020, up to 12.8 billion devices will be deployed. [1] This trend, combined with the fact that most of the owners still lack in cybersecurity awareness, has established a breeding ground for new malware families targeting IoT devices. An example of IoT malware is the MIRAI botnet that was responsible for the biggest DDoS attack ever detected in the history of the Internet, occurred on October 2016. [2]

Since then, many MIRAI variants have appeared, racing among themselves and infecting as many IoT devices as possible. Each of these variants is responsible for the participation of a single IoT device in a specific malicious botnet. This race leads to an extremely dynamic behaviour that makes mapping and classifying compromised IoT nodes a challenging task. Once compromised IoT devices have been mapped and classified - i.e. footprinted - what could be done to slow down the spread of the malware? This article answers to the latter question, describing Fooresec: a software infrastructure, which aims to remotely footprint, report and (if possible) secure compromised IoT devices. Implementation details of a working prototype are also illustrated, along with experimental results produced from the data collected so far.

Key terms and concepts

It is necessary to explain a few key terms and concepts that have been introduced in the preceding paragraph:

- Footprint. This is the process that involves Intelligence Gathering techniques [3] for the purpose of:

    - mapping IP addresses by using External Footprinting techniques, e.g. retrieving the Autonomous System number/country and an abuse email contact(s), geolocating the address, etc;
    - classifying nodes by using Active Footprinting techniques, e.g. TCP port scan, Operative System (OS) family detection, etc.
- Report. This is the process that notifies the information, gathered by the preceding footprinting process, to the interested parties (e.g. to the abuse email contact(s) or to some third-party online blacklisting services).
- Secure. This is the process that involves Intrusion Techniques to gain remote access to a compromised IoT device, with the aim of preventing further unauthorized Telnet accesses (e.g. changing device password, terminating suspicious processes, terminating the Telnet daemon, etc.). Of course, this process introduces not only ethical, but also legal challenges especially in those jurisdictions where computer intrusions are criminalized by laws, statutes and regulations. [4]
  Just to be clear: this article illustrates implementation details of a Fooresec prototype capable of securing only IoT devices belonging to a subnet under our control.
- IP Feeder. Fooresec relies on an external process (e.g. a honeypot, a IDS/IPS, a firewall, etc.) to obtain public IP addresses needed to be footprinted, reported and secured. We named it IP Feeder. In this article, it is taken for granted that the IP Feeder is able to record each infection attempt led by any malicious node, including attempted login credentials, in the database shared with a Fooresec implementation. The implementation of the IP Feeder is beyond the scope of this article.
- Infection. For the purpose of this article, an infection is the process that includes all the activities executed by an attacker on a victim node, in order to remotely command & control it. Typically, the aim of an infection consists of installing a hidden malware on the victim node.

Related works

Both formal and non-scientific research has been conducted on this field:

on June 2015, Yin Minn Pa Pa et al. [5] introduced a sandboxed IoT honeypot (called IoTPot), capable of emulating various IoT devices and analyzing Telnet-based attacks against them;

on February 2017, James Jerkins [4] used the MIRAI malware as a case study to motivate a market or a regulatory solution for IoT insecurity.

on February 2017, Ivo van der Elzen and Jeroen van Heugten [6] have studied different ISP-Level techniques for detecting compromised IoT devices.

A different approach has been followed by Shodan[1] and the threat intelligence company Recorder Future, that have investigated ways to identify botnet C&C servers. They have

[1] https://www.shodan.io/ - the search engine able to find specific devices connected to the Internet

come up with a new service called Malware Hunter (http://malware-hunter.shodan.io) in May 2017. [7]

Fooresec architecture

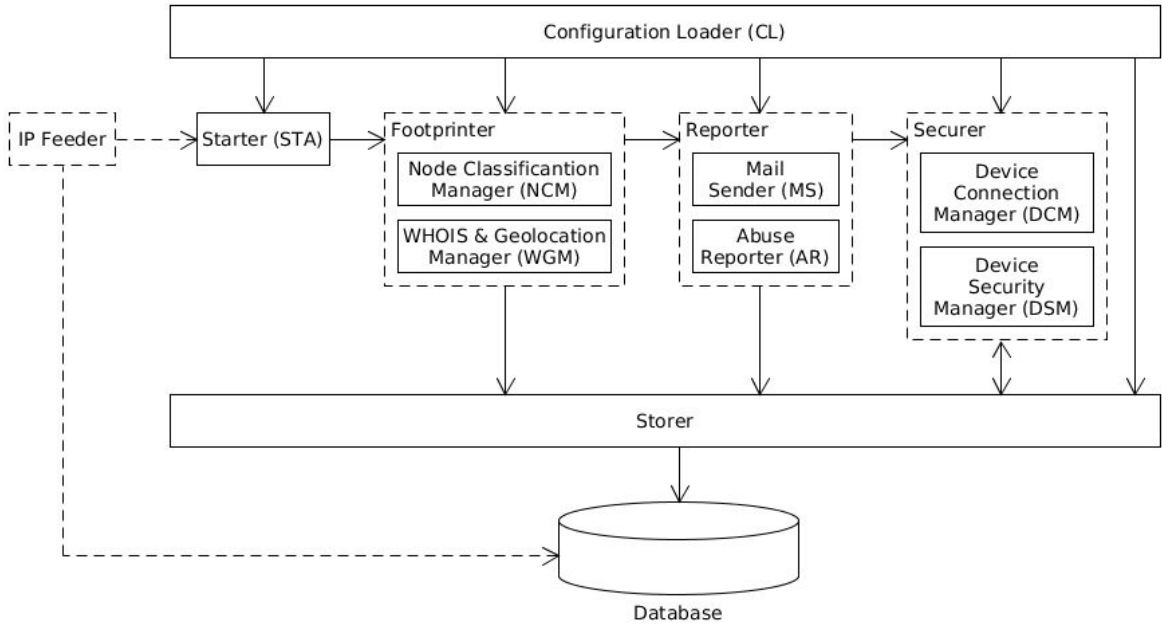

Figure 1: Fooresec architecture

As depicted in Figure 1, Fooresec architecture may be split in six distinct logical modules:

- the Configuration Loader (CL) is the module used to pass configuration parameters to the underlying modules. The parameters are stored in an external configuration source (e.g. a file or a database);
- the Starter (STA) is the module that passively waits for IP addresses from the IP Feeder. When a new IP address is received, it is then passed to the Footprinter;
- the Footprinter is the module that maps and classifies remote nodes. It is composed of:
    - the Node Classification Manager (NCM), which is the submodule responsible for the TCP port scan and the OS family detection of the processed IP address;
    - the WHOIS & Geolocation Manager (WGM), which is the submodule that geolocates IP addresses and performs WHOIS lookups to obtain Autonomous System number/country, abuse contact(s) and much more;
- the Reporter is the module that notifies nodes proven to be malicious, to the interested parties. It is composed of:
    - the Mail Sender (MS), which is a submodule that aims to periodically send notification emails to the abuse contact(s) retrieved from the overlying module;
    - the Abuse Reporter (AR), which is a submodule that aims to report IPs to third-party online blacklisting services;
- the Securer is the module that ensures the security of remote nodes. It is composed of:

    - the Device Connection Manager (DCM), which is a submodule that attempts to connect to the remote IoT device by using a set of known username/password pairs (typically factory default credentials of IoT consumer devices) retrieved from the Storer, as described in the next paragraph;
    - the Device Security Manager (DSM), which is a submodule for ensuring the security of the remote IoT device, once a remote connection has been established by the DCM;
- the Storer, as shown in Figure 1, is used as communication interface between each module and the underlying database. All modules, except the Starter, write collected data to the database. In addition, the Securer also reads from the database, for the purpose of retrieving the credentials required by the DCM.

Fooresec behaviour

In Figure 2 an activity diagram shows a simplified behavioural description of Fooresec: once configuration parameters have been passed to all modules (the very first activity), the STA starts waiting for incoming IP addresses to be processed. When a new IP address is received by the STA, the footprinting activities (i.e “Classify Node”, “Perform WHOIS Lookup” and “Geolocate”) are sequentially executed. The result of the footprinting activities determines if the classified node is a IoT device or not. Of course, the classification process is not deterministic and false positive results may occur.

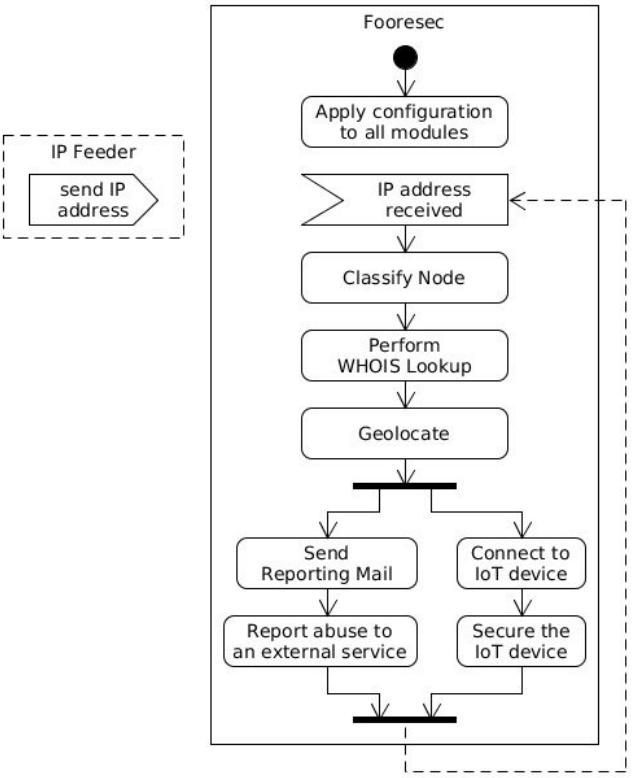

Figure 2: diagram of the activities performed by Fooresec

When the footprinting stage ends, report (i.e “Send Reporting Mail” and “Report abuse to an external service”) and secure (i.e. “Connect to IoT device” and “Secure the IoT device”) activities are simultaneously executed.

The approach used by Fooresec to remotely ensure the security of compromised IoT devices mimics the procedure adopted by MIRAI to infect victim devices and it has been extracted from the source code of MIRAI. [6, 8] The code analysis has also revealed that no particular software vulnerability is actually used during the MIRAI infection process. In fact, it merely consists of trying different usernames and passwords, chosen among a hard coded set of 61 pairs, in order to gain remote access via Telnet to the vulnerable IoT device. In practice, the aforementioned set includes factory default credentials of several IoT consumer devices produced by different vendors. [9]

Furthermore, it could be assumed that other variants of MIRAI will modify the initial set of factory default credentials with new username/password pairs. For this reason, the IP Feeder, as shown in Figure 1, is capable of extending the initial set of default credentials with additional username/password pairs. These can be then used by DCM to attempt connections to the remote IoT devices. If the connection is successful, the DSM can try to secure the IoT device.

It is important to consider that, once MIRAI malware manages to execute on the compromised node with superuser privileges, it tries to kill the Telnet daemon and rebind the TCP port 23 and 2323 to itself, with the aim of preventing further remote Telnet accesses. If this happens, the DCM is unable to connect to the remote IoT device, thus it is unable to ensure the security of the device.

Third-party elements

Some part of the prototype relies on third-party software and online services:

- The honeypot Cowrie[2] has been used as IP Feeder. It is a *medium interaction SSH and Telnet honeypot*, which records each infection attempt led by any malicious node in the database shared with Fooresec. From our perspective, the honeypot behaves as a vulnerable IoT device ready to accept Telnet connections from an attacker. Once logged in, the attacker interacts with a full fake filesystem (honeyfs), resembling a Debian 5.0, which is capable of adding, removing, reading, etc. files. Every single interaction with the shell is logged in a database instance shared with the prototype for the uses described in the preceding paragraphs.
- The NCM relies on Nmap[3], the free and open source utility for network discovery, which uses raw IP packets in order to determine if a node is available on the network, the TCP ports it is exposing, the OS (family and version) it is running and much more.

[2] http://www.micheloosterhof.com/cowrie/

[3] https://nmap.org/

- The AR relies on AbuseIPDB[4], which is a project dedicated to reduce the spread of malicious activity on the Internet. It provides a free REST API for reporting up to 1000 IP addresses per day from a single registered user.
- The Storer interacts with InfluxDB[5], a time-series database (TSDB) built to handle high read and write loads.

Prototype implementation

The prototype has been developed using the Python programming language, version 2.7.

The CL is responsible for passing configuration parameters to all the underlying modules on Fooresec initialization. It is built on top of the Python ConfigParser library, which makes it easy to parse configuration files. The content of the configuration file consists of sections, led by a *[section]* header and followed by *name = value* entries. These configuration parameters are stored in a file named *fooresec.cfg*.

The STA is responsible for interacting with the IP Feeder. The interaction between Cowrie and the STA has been implemented exploiting Unix Domain (Stream) Sockets, which is one of the possible ways of performing client/server communication on a single node.

In this prototype, data sent from Cowrie to Fooresec (more precisely, to the STA) consists of serialized JSON objects. The serialization is obtained using pickle, which is a module for serializing/de-serializing Python object structures. Each JSON object contains the following entries:

- *'addr': <IP address>* - the IP address of the attacking node;
- *'session_id': <eight-digit-string>* - an eight-digit string to uniquely identify the detected intrusion attempt;
- *'sensor_id': <honeypot_name>* - in the case of multiple Cowrie instances sharing the same TSDB, the sensor ID uniquely identifies a single Cowrie instance.

The Footprinter relies on the Python threading library to simultaneously footprint different IP addresses. Each Footprinter thread receives a copy of the JSON object described above, then it executes the activities related to the following submodules:

- the NCM, which relies on the libnmap Python library to manipulate Nmap process. Its activities consist of performing the TCP port scan and the OS family detection of a remote node;
- the WGM, which relies on the ipwhois and geoip2 Python libraries. Its activities consist of:
    - retrieving the Autonomous System number and the abuse contact(s) related to the processed node, from the WHOIS lookup service;
    - geo locating the processed node.

The Reporter waits for JSON objects, incoming from the Footprinter, which contain:

[4] https://www.abuseipdb.com/

[5] https://www.influxdata.com/

- ‘tcp_ports’: [port1, …, portN] - a list of open TCP ports on the attacking node;
- ‘os’ : { ‘family’: <os_family>, ‘accuracy’: <percentage> } - the guessed OS family of the attacking node, along with a percentage indicating the guess accuracy;
- ‘abuse_contacts’: [‘email1’, …, ‘emailN’] - a list of the abuse email contact(s) gathered by the WGM.

The IP address of the compromised node is reported to AbuseIPDB, along with the footprinted information. If the Mail Sender (MS) is enabled and correctly configured, the emails containing the discovered information are sent to the retrieved abuse contact(s). If the same IP address is detected several times per day, only the first attempt is reported to AbuseIPDB, for that specific day.

The proposed prototype is purposely configured to secure only IoT devices belonging to a network under our own control. By querying the Storer, the Securer retrieves a list of credentials to iterate through, in order to obtain a Telnet shell on the compromised IoT device. Therefore, if a remote shell is obtained, a valid pair of username and password has been used. After that, the DSM secures the device by setting as current password the one specified in the configuration file. Currently, setting a new password represents the only available option to secure a device. Our aim is to introduce more possible actions to better secure IoT devices, such as terminate suspicious processes or add specific iptables rules, etc.

All the relevant events occurring within the processes described so far (Footprint, Report and Secure) are logged in an InfluxDB instance, where each session (uniquely identified by a session ID) is a time-series. A new session is created by Cowrie whenever a new infection attempt is detected. All related events detected by the prototype are stored in the time-series database for that specific session. Table 1 shows the list of events currently handled by the prototype.

Fooresec Event Name	**Fooresec Event Description**
fooresec.footprinter.open_port_found	a new TCP port is detected on an attacking node
fooresec.footprinter.os_detected	the OS of an attacking node has been detected
fooresec.reporter.wgm	WGM activities have been performed
fooresec.reporter.reported	IP address of an attacking node has been reported
fooresec.securer.shell_obtained	a remote shell on an attacking node has been obtained
fooresec.securer.device_secured	an attacking node has been secured

Table 1: events handled by Fooresec

Our Fooresec prototype installation has been set up as shown in Figure 3. The purpose of the installation is to prove the validity of the architecture by collecting data related to infection attempts and producing experimental results (described in the next section)

based on the collected data. As illustrated in Figure 3, the installed prototype executes on a single network node, directly accessible from the Internet and placed in the DMZ of our firewall, configured to allow in/out communications on TCP port 23.

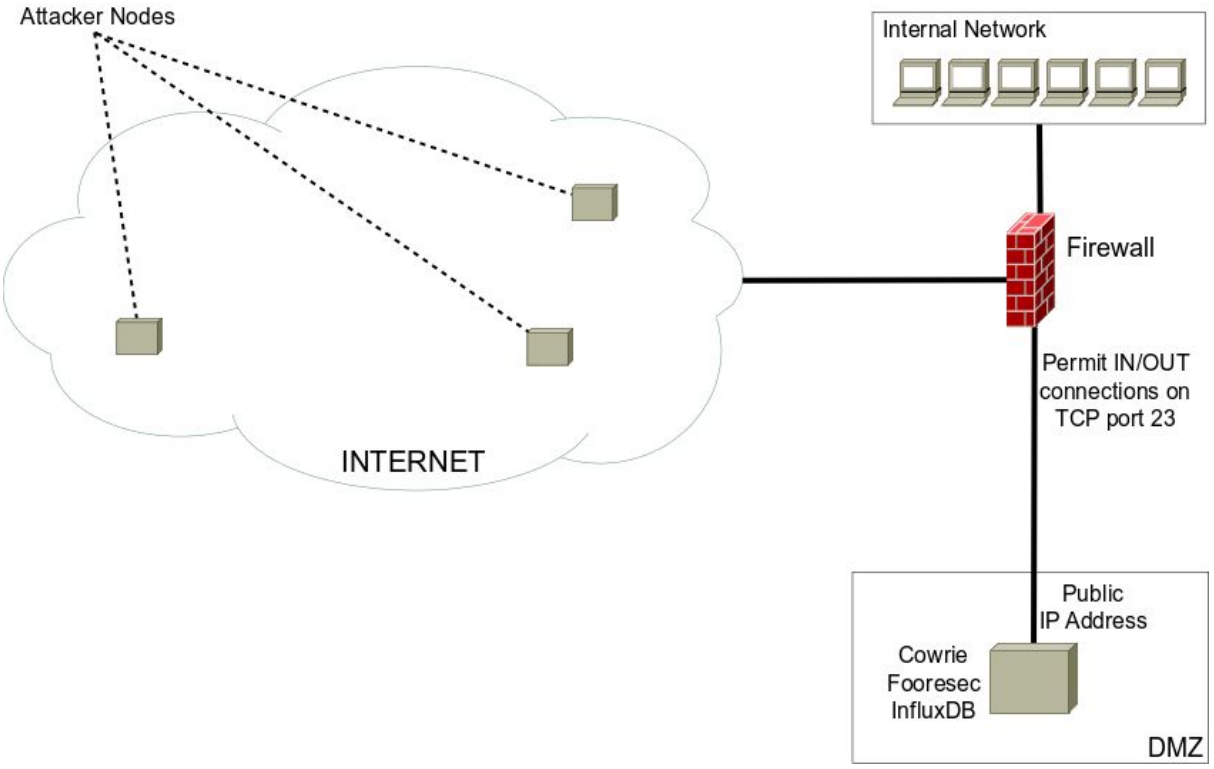

Figure 3: an overview of the implemented prototype

Results

This section presents experimental results to prove the validity of the adopted approach. The presented results have been produced by analyzing either events logged by the IP Feeder and data collected by each module (Footprinter, Reporter and Securer).

The following graphs and tables have been produced with data gathered by the IP Feeder, consisting of a sample of 15251 observations collected in the short period from April 20, 2017 to May 5, 2017 (16 days). Chart 1 shows the number of infection attempts per day occurred during the observation period. On the first and the last day, it has not been possible to detect infection attempts all through the 24 hours, therefore in these two days a lower rate of attempts has been logged.

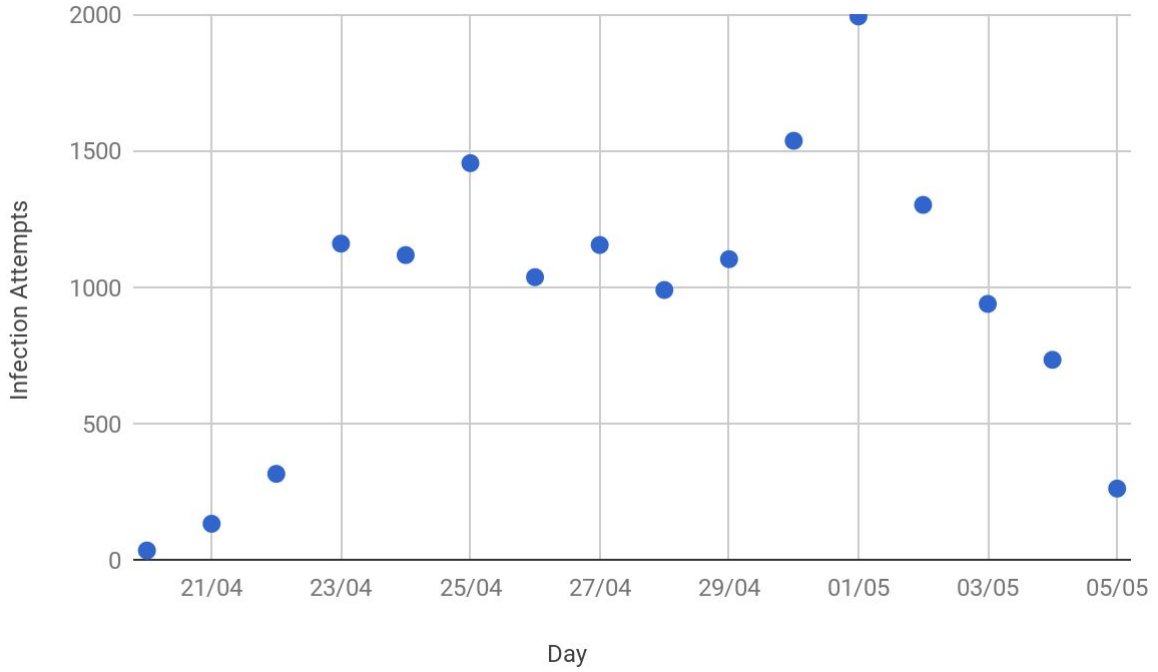

Chart 1: overall infection attempts per day

Table 2 shows the top 10 credentials used by the attacking nodes (username/password pairs) ordered by the number of related login attempts.

Username	Password	Login attempts
root	alpine	1720
support	support	1653
root	default	1466
root	jvbzd	1202
root	1234qwer	889
root	hi3518	872
root	pa55w0rd	719
admin	1234567890	696
admin	12345	512
admin	password	377

Table 2: top 10 credentials used by attacking nodes

From Table 2, it can be noticed that some of the collected credentials are factory default username/password pairs of IoT consumer devices. [9] The first pair of reported credentials is also worthy of note. In fact, these are known to be the default credentials for the Telnet/SSH server on iOS devices ("alpine" was actually the codename for iOS 1.0). [10]

Chart 2 shows the number of infection attempts per country. It can be noticed that, during the observation period, most of the infection attempts have come from Ukraine (~56%).

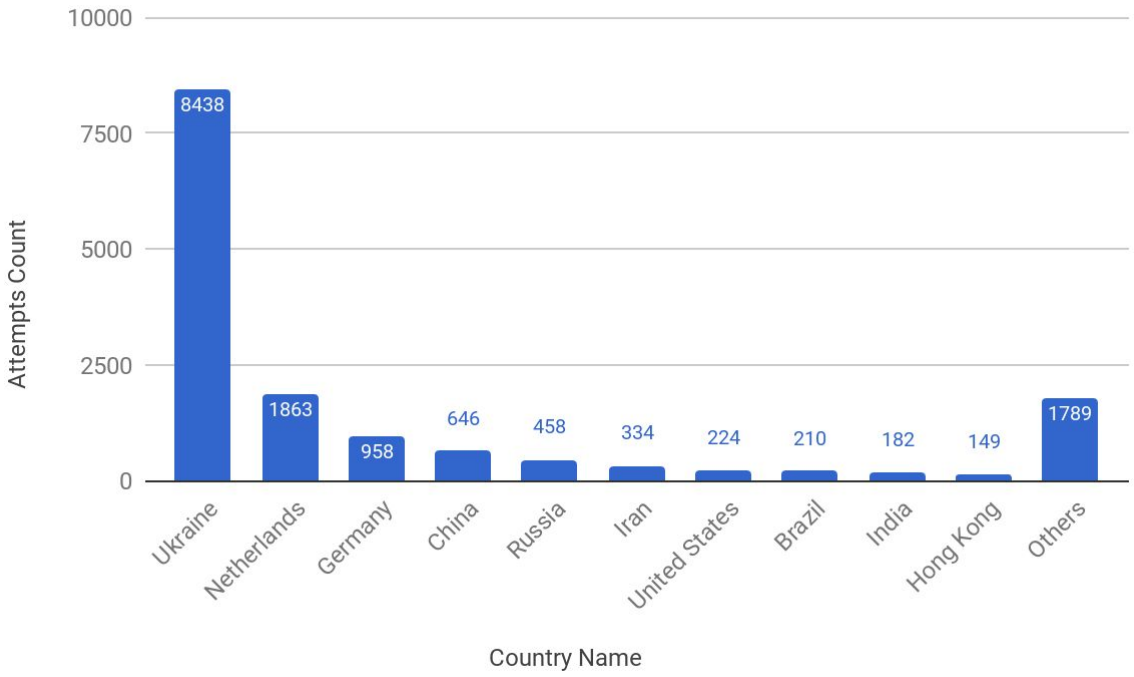

Chart 2: number of infection attempts per country

The total number of observed infection attempts has originated from 2324 distinct nodes. Chart 3 shows the top 10 countries from which a higher number of attacking nodes (1317 observations out of 2324) have been detected. In addition, Chart 3 clearly shows that the largest number of detected nodes is from China (~13%).

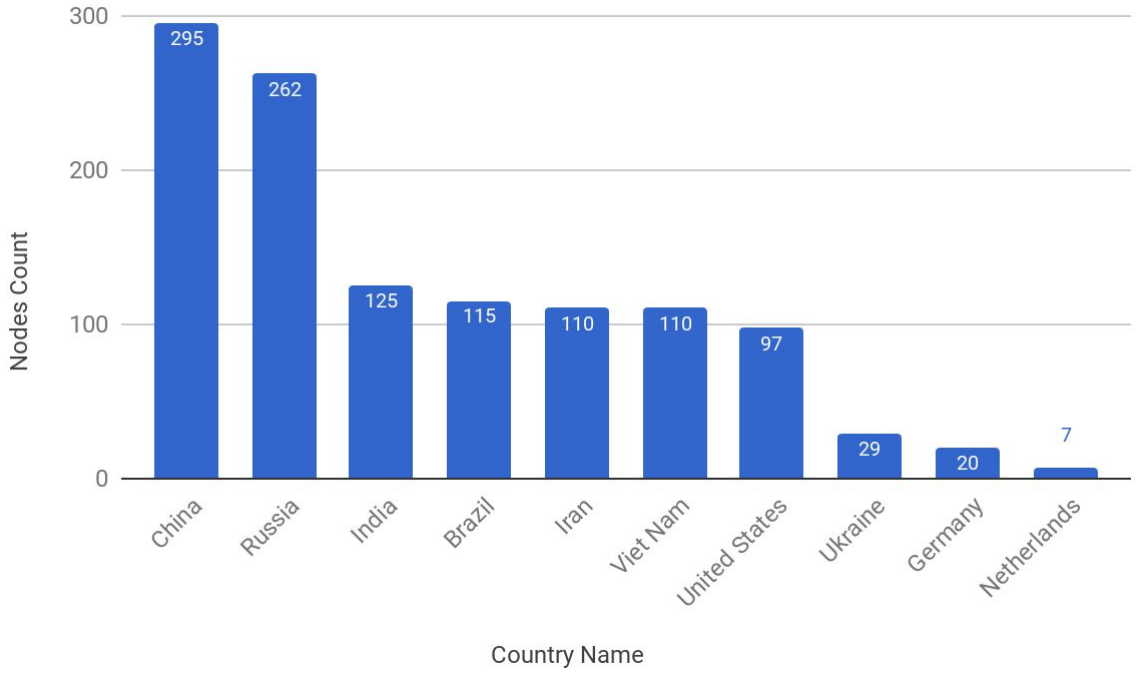

Chart 3: number of compromised nodes, per country

As mentioned above, the classification of remote nodes is a challenging task, in fact, the implemented NCM has been able to detect the OS family for only 765 (~33%) nodes out of 2324. However, Chart 4 reports the number of nodes per OS detected. Of course, the unclassified nodes are not shown in the graph. Having said that, the chart clearly illustrates that 576 (~75%) nodes are Linux IoT devices and that the majority of them (377, ~49%) runs Linux 2.6.x. Only ~6% of the footprinted nodes has been classified as non Linux IoT device, e.g. FreeBSD, MS Windows, etc. Finally, we want to underline that during the OS detection process only those nodes with a guess accuracy greater than 80% have been considered as successfully detected.

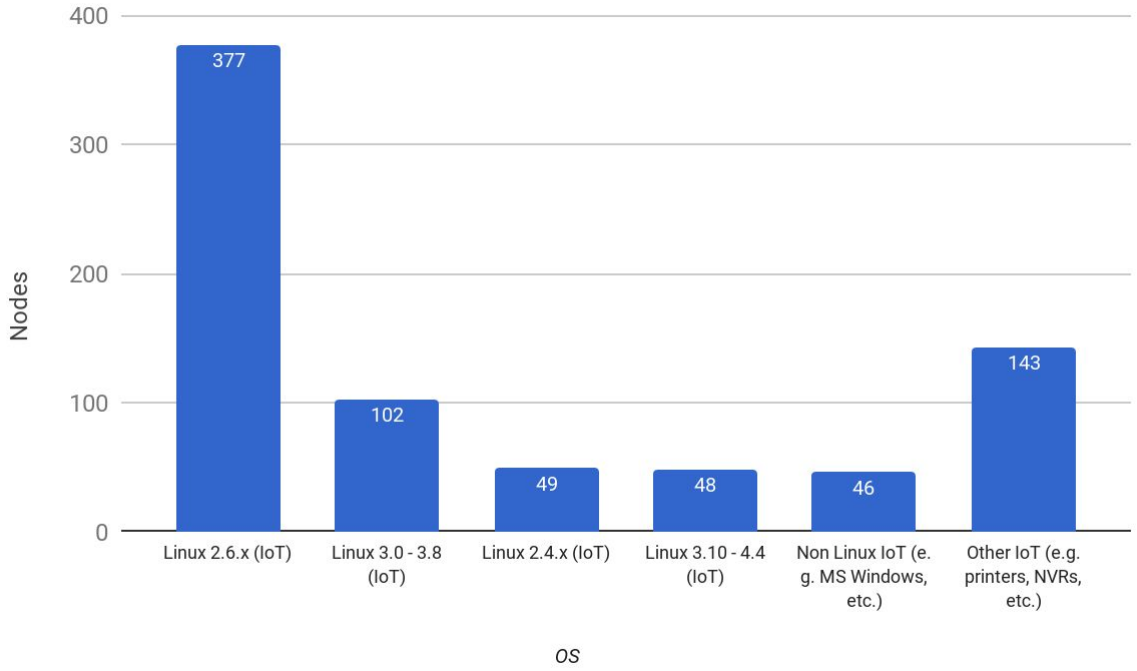

Chart 4: number of nodes per OS detected

In our prototype we decided to enable the Reporter for only a short observation period (3 days, from April 21, 2017 to April 23, 2017), with the AR enabled and the MS disabled.

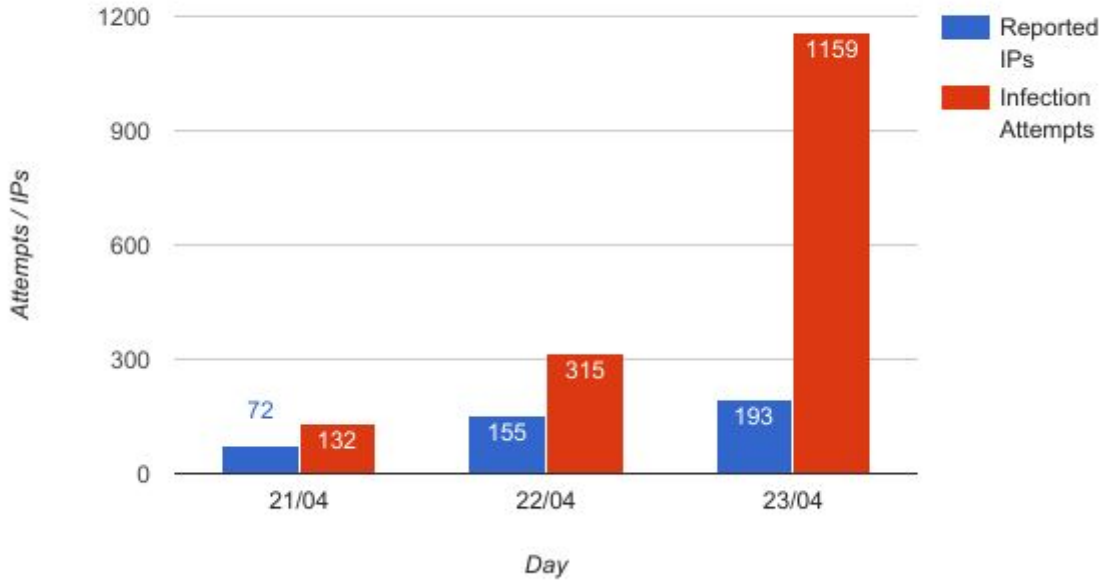

Chart 5: the number of reported IP addresses (420) compared with the number of infection attempts (1606), per day

Chart 5 has been produced with data collected by the Reporter and shows how many IP addresses have been reported during each day (in blue) compared with the number of

infection attempts. The number of reported IP addresses matches the total number of nodes detected by the honeypot.

As the last experiment, we decided to estimate the most probable maximum time needed to infect a new IoT device in a real case scenario. Few concepts have been depicted in Figure 4 to better explain the approach adopted for the estimation. The picture shows the occurred events and the elapsed time during a single infection attempt, from the perspective of Cowrie. In particular, it highlights the time taken by a malicious node to attempt one single successful login on Cowrie (we named it “Login Time”) and the time taken by the same malicious node to complete the infection process (we named it “Infection Time”).

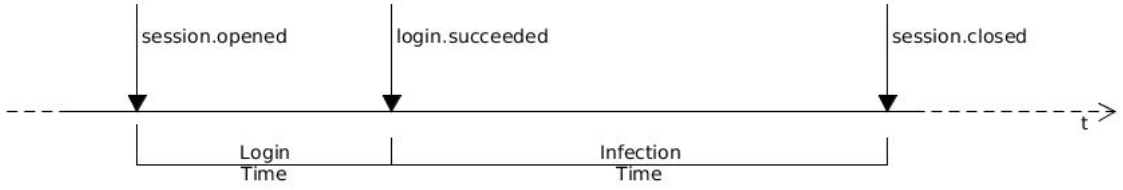

Figure 4: events occurring during a single infection attempt

In our prototype, it has been possible to measure “Login Time” since the honeypot allowed to login using any pair of credentials. In a real case scenario, the attacker needs most likely to try several pair of credentials, before successfully logging in to the IoT device. It also has to be noticed that “Login Time” and “Infection Time” observations include extra time due to both network overhead (for “Login Time” it is mainly due to TCP connection opening) and computational overhead. For the purpose of our estimation, time due to these overheads has been considered negligible.

For the sake of simplicity, we based our estimation only on those nodes detected from April 21, 2017 to April 23, 2017 (a sample of 420 observations, as shown in Chart 5). We decided to compute the empirical probability distribution function (EPDF) for both “Login Time” observations (shown in Chart 6) and “Infection Time” (shown in Chart 7). Further information on the computation method used, can be found in [11]. In addition, some sample statistics have been reported in Table 3.

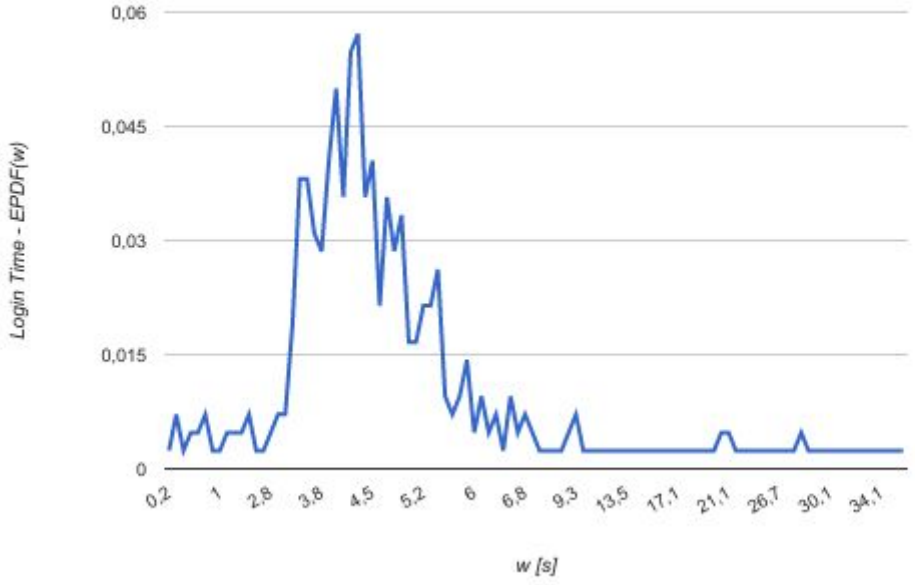

Chart 6: “Login Time” EPDF

In Chart 6 it could be noticed that the most probable “Login Time” is 4.3 seconds. Furthermore, the probability of having a “Login Time” in the range 3.5-6.8 seconds is 76.2%.

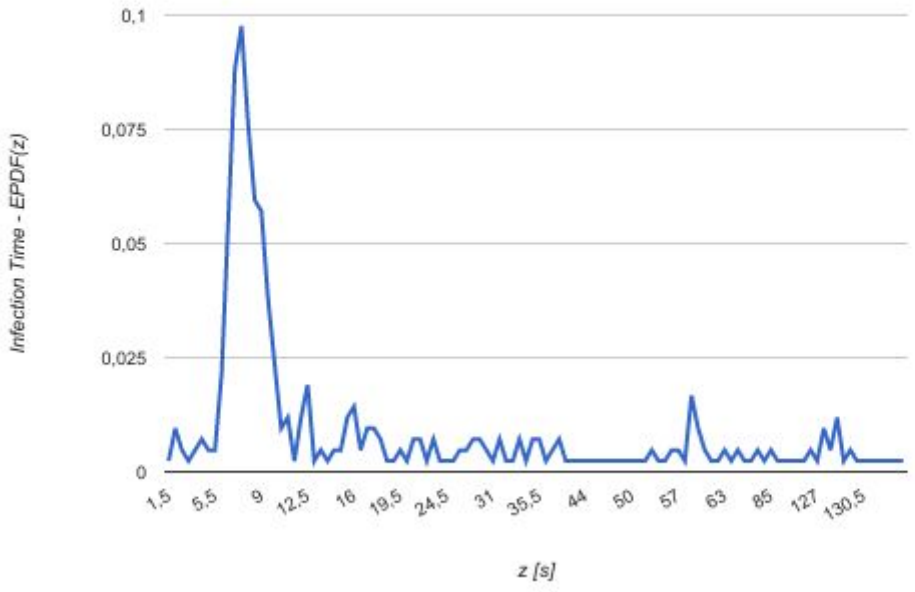

Chart 7: “Infection Time” EPDF

In Chart 7 it could be noticed that the most probable “Infection Time” is 7.6 seconds. Furthermore, the probability of having an “Infection Time” in the range 6-10 seconds is 51.7%.

	Login Time (single attempt)	**Infection Time**
Sample Mean (S_M)	~6.59 s	~26 s
Sample Std Deviation (S_{StD})	~6.97 s	~34.28 s

Table 3: mean and standard deviation values of “Login Time” and “Infection Time” samples

For the purpose of this article, it has not been necessary to refine the obtained EPDFs into specific mathematical forms. Hence, “Login Time” and “Infection Time” values have been extracted directly from their respective EPDFs.

Before we proceed with the estimation, it is necessary to introduce the following simplification: all the login attempts can be considered as login attempts coming from nodes infected by the original MIRAI malware. The latter assumption allows to make a second simplification: since the process of iterating through credentials used by the original MIRAI malware stops after 10 unsuccessful attempts, [8] hence the maximum “Login Time” is obtained when the credentials of the victim device are guessed on the 10th login attempt.

That being so, we computed the most probable maximum time to fully infect a device, by choosing the most probable "Login Time" and "Infection Time" values and maximizing the number of failed login attempts: 4.3 seconds * 10 attempts + 7.6 seconds = 50.6 seconds.

In conclusion, we decided to extend to the Securer all the assumptions and results previously discussed, with the aim of estimating the most probable time taken by the DCM to successfully login to a device known to be compromised. This could be achieved if we further assume that:

- the time taken by the DCM to successfully login on the remote node is comparable with the "Login Time". We named it "DCM Login Time".

We have chosen this approach because, for the reasons indicated above, it has not been possible to connect the Securer directly to the Internet. For what has been said so far, in Table 4 we reported the most probable minimum and maximum "DCM Login Time", that is the most probable minimum/maximum time taken by the DCM to successfully login to a device.

	DCM Login Time / attempts
Best Case (minimum)	4.30 seconds / 1
Worst Case (maximum)	~4.37 minutes / 61

Table 4: most probable minimum and maximum time taken by the DCM to successfully login to a device

To estimate the most probable time taken by the DCM to successfully login to a device, we also assumed that the username/password pairs, contained in our set of known credentials, are the same used by the original MIRAI malware. Given that, in Table 4, it is reported that the most probable minimum "DCM Login Time" is 4.30 seconds, corresponding to a successful username/password guess on the first login attempt (Best Case). On the contrary, the most probable maximum "DCM Login Time" is 4.37 minutes (Worst Case).

We are aware that all the considered simplifications have introduced estimation errors, but nevertheless we think that, the obtained results can be used as reliable "upper bounds" for comparisons with data from similar scenarios.

Conclusion

This article has proposed a new software infrastructure, called Fooresec, for analyzing and mitigating the spread of IoT botnets over the Internet. Beside describing the design of the architecture, some implementation details of a working prototype have been illustrated and, in order to prove the viability of the approach, experimental results have been also presented.

The produced results confirmed that remotely footprint IoT devices is a challenging task. Despite this, it can be deduced that most of the successfully footprinted IoT devices targeted by the malware are based on Linux (~75%). It is also clear that reporting compromised IoT devices can help ISPs in taking immediate actions (e.g. traffic filtering, shaping, etc.). Furthermore, the estimation of "DCM Login Time" values, considering all the

assumptions and simplifications taken into account, represents an encouraging starting point for the possibility of a full implementation of Fooresec.

Hence, we think that a full implementation of the proposed architecture, placed on the ISP side, can both reduce the number of already compromised IoT devices and slow down the spread of IoT botnets, in those scenarios where the remote access to customer IoT devices for the only purpose of fighting IoT botnets, is granted by a legal agreement between customer and ISP.

References

1. 'Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016'. Gartner, Inc. February 7, 2017. Accessed May 2017. www.gartner.com/newsroom/id/3165317.
2. 'Akamai's [state of the Internet] / security - Q4 2016 report'. Akamai Technologies, Inc. Accessed May 2017.
3. 'The Penetration Testing Execution Standard - Intelligence Gathering'. Accessed May 2017. www.pentest-standard.org/index.php/PTES_Technical_Guidelines#Intelligence_Gathering.
4. Jerkins, James. 'Motivating a Market or Regulatory Solution to IoT Insecurity with the Mirai Botnet Code'. University of North Alabama. March 2017. Accessed May 2017
5. Pa Pa, Yin Minn et al. 'IoTPOT: Analysing the Rise of IoT Compromises'. Yokohama National University, Japan et al. June 2015. Accessed April 2017.
6. van der Elzen, Ivo; van Heugten, Jeroen. 'Techniques for detecting compromised IoT devices'. University of Amsterdam. February 2017. Accessed April 2017.
7. Gundert, Levi. 'Proactive Threat Identification Neutralizes Remote Access Trojan Efficacy'. Recorded Future, Inc. May 2017. Accessed May 2017.
8. 'Leaked Mirai Source Code for Research/IoC Development Purposes'. Accessed March 2017. www.github.com/jgamblin/Mirai-Source-Code.
9. 'Who Makes the IoT Things Under Attack?'. KrebsOnSecurity. October 3, 2016. Accessed May 2017. www.krebsonsecurity.com/2016/10/who-makes-the-iot-things-under-attack.
10. 'Hacking iOS Applications'. Security Innovation, Inc. 2017. Accessed May 2017, https://web.securityinnovation.com/hubfs/iOS Hacking Guide.pdf.
11. Ross, Sheldon. 'Introduction to probability and statistics for engineers and scientists'.