IoDDoS — The Internet of Distributed Denial of Service Attacks
A Case Study of the Mirai Malware and IoT-Based Botnets

Roger Hallman, Josiah Bryan, Geancarlo Palavicini, Joseph Divita and Jose Romero-Mariona
US Department of Defense, SPAWAR Systems Center Pacific, San Diego, California, U.S.A.

Keywords: Internet of Things, Cybersecurity, Botnets, Mirai Malware, Emerging Threats.

Abstract: The Internet of Things (IoT), a platform and phenomenon allowing everything to process information and communicate data, is populated by 'things' which are introducing a multitude of new security vulnerabilities to the cyber-ecosystem. These vulnerable 'things' typically lack the ability to support security technologies due to the required lightweightness and a rush to market. There have recently been several high-profile Distributed Denial of Service (DDoS) attacks which utilized a botnet army of IoT devices. We first discuss challenges to cybersecurity in the IoT environment. We then examine the use of IoT botnets, the characteristics of the IoT cyber ecosystem that make it vulnerable to botnets, and make a deep dive into the recently discovered IoT-based Mirai botnet malware. Finally, we consider options to mitigate the risk of IoT devices being conscripted into a botnet army.

1 INTRODUCTION

On 20 September, 2016, the cybersecurity blog, KrebsOnSecurity[1], was the target of a massive Distributed Denial of Service (DDoS) attack, exceeding 620 gigabits per second (GBps) (US-CERT, 2016). The attack was initially unsuccessful, but nearly double the size of the largest previously seen DDoS attack, according to the internet security firm Akamai[2], and the decision was made to suspend KrebsOnSecurity (Krebs, 2016b; Ragan, 2016). Less than a month later, a DDoS attack was launched on the Domain Name System (DNS) provider Dyn[3] that was clocked at 1.2 terabytes per second (TBps), the largest ever recorded (Woolf, 2016). Both attacks were believed to have been perpetrated by a conscripted botnet army, where devices are infected by malware and connected to an outside command and control (C&C) server, of Internet of Things (IoT) devices (US-CERT, 2016; Newman, 2016). There were additional reports of the same family of IoT-based botnets launching DDoS attacks against French webhost OVH (Bertino and Islam, 2017) and Liberian mobile infrastructure (Krebs, 2016a). There have been growing concerns about cybersecurity vulnerabilities in IoT systems that are rushed to market, and the risk of IoT-based botnets has been theorized for several years (Holm, 2016; Lin et al., 2014). The large DDoS attacks against KrebsOnSecurity and Dyn demonstrate the magnitude of IoT security vulnerabilities and how unsecured 'things' on the internet may be conscripted into an army capable of massive damage and disruption (Symantec, 2016).

In this paper, we discuss the characteristics of the IoT ecosystem that make its 'things' so vulnerable to conscription for a massive botnet army and steps that may be taken to mitigate these vulnerabilities. The rest of this paper is organized as follows: Section 2 discusses IoT and gives examples of IoT environments as well as an overview of their cybersecurity vulnerabilities. Section 3 gives a more in-depth analysis of botnets and DDoS attacks. Section 4 gives an overview of the IoT-based botnets as well as an in-depth analysis of the IoT-based botnet malware used in the KrebsOnSecurity and Dyn DDoS attacks. Vulnerability mitigations will be dealt with in Section 5. Finally, Section 6 will have concluding remarks.

2 THE IoT ECOSYSTEM

IoT is a platform and a phenomenon that allows everything to process information, communicate data, and analyze context collaboratively or in the service of individuals, organizations and businesses. IoT,

[1] http://krebsonsecurity.com/
[2] https://www.akamai.com/
[3] https://dyn.com/

Hallman R., Bryan J., Palavicini G., Divita J. and Romero-Mariona J.
IoDDoS âĂŤ The Internet of Distributed Denial of Sevice Attacks - A Case Study of the Mirai Malware and IoT-Based Botnets.
DOI: 10.5220/0006246600470058
In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017)*, pages 47-58
ISBN: 978-989-758-245-5

with its 'things' connected to the Internet, is an instantiation of the "Network of Things" concept (Voas, 2016). There are considerable benefits of having numerous devices connected to larger infrastructures, with utility to be gained in multiple contexts (e.g., home, industrial environments) (Gubbi et al., 2013). IoT is an explosive market, with estimates that the number of Internet-connected smart devices will more than triple to around 40 billion in use by 2019 (Thierer and Castillo, 2015).

2.1 The Home IoT Environment

There has been an explosive growth in the number of Internet-connected household devices which allow for monitoring and controlling a "smart home" (Sivaraman et al., 2015). These devices include smart plugs, lights, and electrical switches which give insight into power consumption. Appliances may be remotely monitored and controlled such as a garage door opener, thermostat, or oven. Children or pets may be monitored remotely through Internet-connected cameras and microphone systems.

Internet-connected smart homes typically make use of a home automation system, with multiple commercially available systems to choose from. These systems tend to have a central C&C unit and a diverse array of lightweight, sensor-enabled devices that monitor one or more aspects of an area (e.g., the number of people who have entered a room). An interesting aspect of many home automation systems is that they can be controlled with little or no authentication credentials; a great deal of their system security comes from a "firewall" feature of the home router's network address translation between public and private Internet Protocol (IP) addresses (Sivaraman et al., 2016).

2.2 The Industrial IoT Environment

Industrial processes have been automated for decades, with machine-to-machine communication protocols (e.g., Modbus/TCP) for Supervisory Control and Data Acquisition (SCADA) and other industrial control Systems (Boyer, 2009). SCADA and other legacy machine-to-machine communication systems are still widely used, however the descendant of these older systems is the Industrial Internet. The Industrial Internet is a form of IoT in which the things are objects in manufacturing plants, utilities, dispatch centers, process control industries, etc. (Stankovic, 2014). These operational technology (OT) systems were traditionally kept separate from their information technology (IT) counterparts, but with the ubiquity of IP this separation is vanishing (Beel and Prasad, 2016).

2.3 An Overview of Iot Cybersecurity Vulnerabilities

IoT cybersecurity vulnerabilities are well documented. Some of these include (Bertino and Islam, 2017; Romero-Mariona et al., 2016; Cruz et al., 2016):

- Systems without well-defined parameters which are continuously changing due to device and user mobility.
- Devices in an IoT system may be autonomous themselves, functioning as a control for other IoT devices.
- IoT systems may include components which were never intended to be connected to the Internet, or communicate by standardized protocols.
- IoT systems may be controlled by different parties.
- Granular permission requests, common in smartphone applications, are problematic in IoT systems due to the diversity and number of devices.
- Firmware updates and system patching are difficult and often completely lacking.
- Many IoT devices possess only the memory and processing capabilities to accomplish their assigned task and consequently cannot support standard security solutions.
- Devices often operate under unique constraints (e.g., timing).
- Device lifespans may continue for many years after vendor support has ceased (e.g., patching).
- Vulnerable devices are deeply embedded within networks.

An additional, detailed list of IoT cybersecurity vulnerabilities and attack vectors are described in Table 1.

3 BOTNETS AND DDoS ATTACKS

A botnet is an organized network of software robots, or 'bots', that run autonomously and automatically on zombie machines that have been infected by malware (Hachem et al., 2011; Zhu et al., 2008). Botnets are used to conduct DDoS attacks, distributed computing (e.g., mining bitcoins), spread electronic spam and malware, conduct cyberwarfare, conduct click-fraud

Table 1: Internet of Things Cybersecurity Vulnerabilities and Attack Vectors. Adapted from the Open Web Application Security Project Top 10 IoT Vulnerabilities (www.owasp.org/index.php/Top_IoT_Vulnerabilities).

Vulnerability	**Attack Vectors**
Insecure Interfaces	Weak credentials, capture of plain-text credentials, insecure password recovery systems, or enumerated accounts, and lack of transport encryption may be used to access data or controls.
Insufficient Authentication and Authorization	Weak passwords, insecure password recovery mechanisms, poorly protected credentials, and lack of granular access control may enable an attacker to access a particular interface.
Insecure Network Services	Vulnerable networks services may be used to attack a device or bounce an attack off of a device.
Lack of Transport Encryption/Integrity Verification	The lack of transport encryption allows an attacker to view data being passed over the network.
Privacy Concerns	Insecure interfaces, insufficient authentication, lack of transport encryption, and insecure network services all allow an attacker to access data which is improperly protected and may have been collected unnecessarily.
Insufficient Security Configurability	A lack of granular permissions, lack of encryption or password options may allow an attacker to access device data and controls. An attack (malicious or inadvertent but benign) could come from any device in an IoT system.
Insecure Software/Firmware	Update files captured through unencrypted connections may be corrupted, or an attacker may distribute a malicious update by hijacking a DNS server.
Poor Physical Security	USB ports, SD cards, and other storage means allow attackers access to device data and operating systems.

scams, and steal personal user information. They are arguably the most potent threat to the security of Internet-connected systems and users (Khattak et al., 2014). A "botmaster" will seize control of a network of infected computers and through a C&C system direct the infected computers to conduct malicious activities. More sophisticated botnets decentralize their C&C system by structuring their network in a peer-to-peer (P2P) manner such that any one infected computer will only communicate with a small number of 'locally' infected computers. This makes the detection of all the infected computers, the bots that comprise the botnet, difficult. Botnets, like virtually all cyber-threats, have typically been targeted against IT systems, but there are several recent incidences of their use in attacks on OT systems.

The Internet was designed with functionality as a primary concern with security often as an afterthought, and consequently there are very limited resources available for the purpose. The first well-known DDoS attack was against the University of Minnesota in 1999, attacking university servers with User Datagram Protocol (UDP) packets from thousands of machines for two days (Boyle, 2000). DDoS attacks have increased in frequency and size, commonly being used in conjunction with conventional warfare methods during times of conflict (Nazario, 2009). Because of the ease of conducting an attack there are DDoS-for-hire services run by cybercriminals (Santanna et al., 2015). Figure 1 illustrates a botnet-conducted DDoS attack.

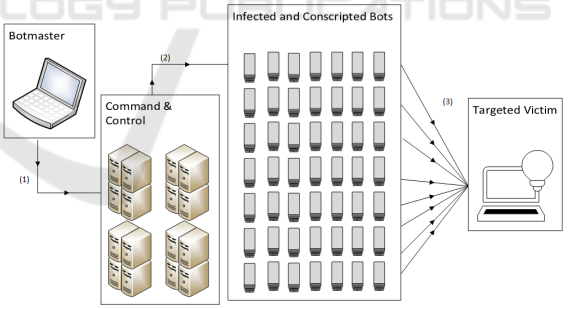

Figure 1: A simplified model of a botnet conducting a DDoS attack. Once an army of infected and conscripted bots has been assembled, the botmaster issues attack instructions via proxy C&C servers in (1). The C&C servers disseminate their attack instructions to their conscripted bots in (2). In (3), the conscripted botnet army proceeds to overwhelm the targeted victim's system with a series of short but continual attacks.

A recent high profile botnet malware incident was the use of BlackEnergy in the attack which took down the Ukrainian power grid in December, 2015 (Khan et al., 2016). The BlackEnergy malware was first described in 2007 (Nazario, 2007) as a simple Hypertext Transfer Protocol (HTTP) based botnet which used encryption at runtime to avoid detection. The vic-

tims of this early BlackEnergy malware were the distributed compromised systems and the end targeted systems of the DDoS attack. Arbor Networks located 27 active DDoS networks in Russia and Malaysia with targets in Russian IP space (Khan et al., 2016). There is wide speculation that Russia used the BlackEnergy malware to launch a DDoS attack during the Russian-Georgian Conflict in 2008, although this has never been conclusively proven (Nazario, 2009). The software controlling several national critical infrastructures including electric grids, water filtration systems, and oil and gas pipelines have been compromised by BlackEnergy since 2011 (Pultarova, 2016a). NATO Headquarters was also the victim of a BlackEnergy attack (Khan et al., 2016).

The Ukranian incident involved a coordinated attack on three power distribution companies, utilizing a new variant of BlackEnergy to illegally enter the companies' IT and OT systems. Opening the breakers to seven substations resulted in a power outage to more than 225,000 people. This variant of BlackEnergy included a KillDisk function which erased several systems and corrupted the master boot records in all three companies, extending the power outage. While the BlackEnergy malware was not utilized as a botnet in the classical sense, its use during the reconnaisance phase of the attack shows the dangerous versatility of botnet malware in coordinated cyber attacks (Lee et al., 2016). Moreover, custom firmware was deployed for serial-to-Ethernet converters which disabled devices and prevented technicians from restoring power until they bypassed those converters (Khan et al., 2016).

3.1 A Taxonomy for DDoS Attacks

Specht and Lee (Specht and Lee, 2004) propose a taxonomy for DDoS attacks consisting, at the highest level, of *Bandwidth Depletion Attacks* and *Resource Depletion Attacks*. They describe Resource Depletion Attacks as flooding the victim network with unwanted traffic that prevents legitimate traffic from reaching the victim. A Resource Depletion Attack ties up the victim system's resources, making the processing of legitimate requests for service impossible.

3.1.1 Bandwidth Depletion DDoS Attacks

Bandwidth Depletion Attacks are characterized as either flood or amplification attacks (Specht and Lee, 2004). Specifically, these types of attacks are distinguished by adversary-controlled botnets which flood their victims with IP traffic and ultimately saturate their network and resources (Okafor et al., 2016). These attacks are usually aimed at making the victims unreachable or disabling their services (Gasti et al., 2012).

3.1.1.1 Flood Attacks

Flood attacks involve a conscripted botnet army sending extraordinarily large volumes of traffic to a victim network with the intent of congesting it to the point of service degradation, or a crash. Some examples of Flood attacks include (Singh and Gyanchandani, 2010):

- *UDP flooding* is a host-based attack which send a large number of UDP packets to a random port on the target system, causing it to fail.
- *Ping of Death*, or Internet Control Message Protocol (ICMP) flood attacks use a ping utility to create IP packets exceeding the maximum 65,536 bytes allowed by IP specification. The over-sized packets are sent to the target system causing it to crash, hang, or reboot.

3.1.1.2 Amplification Attacks

Amplification attacks involve the conscripted botnet army sending messages to a broadcast IP address, a feature found on many routers. All systems in the subnet reached by the broadcast address send a reply to the targeted victim system. The broadcast IP address amplifies and reflects attack traffic to reduce the victim system's bandwidth. Examples of amplification attacks include (Specht and Lee, 2004; Singh and Gyanchandani, 2010):

- *Smurf attacks* generate large amounts of traffic on the targeted victim's computer. Packets are sent as a direct broadcast to a subnet of a network to which all of the subnet's systems reply. However, the packets' source IP address is spoofed to be the targeted victim, which is then flooded.
- *Fraggle attacks* send packets to a network amplifier using UDP ECHO packets which target the character generator in the systems reached by the broadcast address. Each system then generates a character to send to the echo service in the victim system, which causes an echo packet to be sent back to the character generator.

3.1.2 Resource Depletion DDoS Attacks

Resource depletion attacks are characterized by an attacker sending packets that are malformed or misuse communication protocols, using network resources to such an extent that there are none left for legitimate users (Specht and Lee, 2004). The main difference between a resource depletion attack and a bandwidth depletion attack is that the former aims at exhausting the resources as opposed to the latter which attempts to deny critical services (Amiri and Soltanian, 2015). Furthermore, resource depletion attacks can allow adversaries to consume more energy in a network than an honest node, thus affecting the actual physical resources of certain IoT components (Gayatri and Naidu, 2015).

3.1.2.1 Protocol Exploit/Misuse Attacks

Protocol exploit attacks misuse common Internet communication protocols, such as Transfer Control Protocol Synchronize (TCP SYN) and Acknowledge (ACK). Examples of Protocol exploit attacks include (Specht and Lee, 2004; Singh and Gyanchandani, 2010):

- *TCP SYN flooding* uses a conscripted botnet army to initiate an overwhelming number of three-way handshakes to initialize new TCP connections. The source IP address is spoofed, causing the victim system to send ACK+SYN responses to a non-requesting system. The victim system eventually runs out of memory and processor resources after none of the ACK+SYN responses are returned.
- *PUSH + ACK attacks* involve sending the targeted victim TCP packets with PUSH and ACK bits set to 1, causing the victim to unload all data in the TCP buffer and send an acknowledgement once this is completed. The targeted system cannot process the large volume of packets from multiple senders and crashes.

3.1.2.2 Malformed Packet Attacks

Malformed packet attacks involve a botnet army sending the targeted victim incorrectly formed IP packets, which cause the victim system to crash. There are two types of malformed packet attacks (Specht and Lee, 2004):

- *IP address attacks* where the packet contains the same source and destination IP addresses. This confuses and crashes the targeted victim's operating system.
- *IP packet options attacks* involve randomizing an optional field within IP packets sent from the botnet army, setting all quality of service bits to 1. The targeted victim must use additional processing resources to analyze the traffic, exhausting its processing ability.

4 IoT-BASED BOTNETS AND THE DDoS ATTACKS OF 2016

The number of Internet-connected devices in the IoT environment is expected to triple by 2019 (Thierer and Castillo, 2015). The consequence of this rush to put products on the market is insufficient security designed into systems (Romero-Mariona et al., 2016). This poor security makes IoT devices soft targets, and attacks on IoT devices have been long predicted. While there has been concern about the hijacking of various devices in the home or industrial ecosystems to exfiltrate information on individual victims, IoT devices are also easily conscripted into botnet armies for multi-platform DDoS attacks (Symantec, 2016).

There were eight families of IoT malware families discovered in 2015 (Symantec, 2016):

- The Zollard worm exploits the Hypertext Preprocessor (PHP) 'php-cgi' Information disclosure vulnerability that was patched in 2012. Once infected, a backdoor on a TCP port opens to allow remote command execution.
- Linux.Aidra spreads through TCP on port 23 and looks for common username and password combination in order to login into devices. A backdoor is opened on the device and the infected device is conscripted into a botnet army that uses floods of TCP packets, UDP packets, or DNS requests.
- XOR.DDos opens a backdoor in its device and uses COR encryption in both the malware code and in C&C server communication and its main function is to conduct DDoS attacks.
- Bashlite-infected devices become part of a botnet army for launching DDoS attacks of EDP and TCP floods. It also contains a function to bruteforce routers with common username and password combinations and can collect CPU information from infected devices.
- LizardStresser was reputedly created by the Lizard Squad hacker group and has the ability to launch DDoS attacks. It is distributed by scanning public IP addresses for Telnet services and will attempt a variety of common username and password combinations. After a successful logon, it reports back to its C&C server and awaits further instruction.
- AES.DDoS uses the AES algorithm to encrypt communication with its C&C server. It carries out DDoS attacks, but also collects information about the compromised device and sends it to the C&C server.
- PNScan is a Trojan that scans a network segment for devices with an open port 22 and attempts a bruteforce login with common username and password combinations. It does not have botnet functionalities, but can be used to download botnet malware like Tsunami.
- The Tsunami Trojan is used to launch DDoS attacks. It modifies files in such a way that it gets run each time a user logs in or a device boots up. Moreover, Tsunami is known to kill processes, download and execute various files, and spoof IP addresses of the infected device.

4.1 The Mirai Malware Family

The Mirai botnet malware, which was utilized for the DDoS attacks against KrebsOnSecurity, Dyn, OVH, and others, is part of a Trojan IoT malware family that was first discovered in May 2016 (Dr.Web, 2016). The initial malware was named Linux.DDoS.87, a later variant named Linux.DDoS.89 and finally Linux.Mirai (referred to simply as 'Mirai') were discovered in August. We highlight Linux.DDoS.87 and Linux.DDoS.89 before an in-depth examination of the Mirai Malware.

4.1.1 Linux.DDoS.87

Linux.DDoS.87 is a malicious program that uses the uClibc[4] C Library for embedded systems and carries out DDoS attacks. Prior to receiving and executing commands, the malware makes the following calls: `fillConfig`, `init_random`, `runkiller`, and `fillCmdHandlers`.

The `fillConfig` function uses a memory sector to store configuration information and some strings may be stored in encrypted ELF files and decrypted prior to recording. The generation of pseudo-random sequences is achieved by the `init_random` function. Linux.DDoS.87 displays a certain "territorial" behavior with a `runkiller` function that searches for the running processes of other Trojans and terminates them. `fillCmdHandlers` is a function that fills the structure that stores command handlers. Once these commands are run, Linux.DDoS.87 removes its name to hide itself and child processes are subsequently launched with simplified code and containing no requests to the configuration. The Linux.DDoS.87 Trojan has a maximum uptime of one week on an infected machine before it terminates its operation (Dr.Web, 2016).

Linux.DDoS.87 then attempts to connect to a C&C server. The IP address of the used interface is

[4]https://uclibc.org/

saved and a string containing information on the architecture of the infected device is sent to the C&C server. The `process` function parses command arguments and fills respective structures (e.g., `target_ip`, `sockaddr`, etc.) and a `run_command` function is called after parsing is completed. This function receives a time value, command number, quantities and arrays of targets and parameters. Linux.DDoS.87 is capable of the following attacks: HTTP flood, UDP flood, TCP flood, Domain Name System (DNS) flood, and TSource Engine Query (TSource) flood among others (Dr.Web, 2016).

4.1.2 Linux.DDoS.89

Discovered in early August 2016, Linux.DDoS.89 is a modified version of Linux.DDoS.87 (Dr.Web, 2016). The command format is unchanged from Linux.DDoS.87, but fields in some of the Trojan's structures have been moved. Rather than reallocating memory, a statically allocated area of memory is used. A `decode` function decrypts stored configuration values by an XOR operation and then re-encrypts the value after use. Linux.DDoS.89's pseudo-random number generator as well as its order of performed operations are different from Linux.DDoS.87.

It first starts operating with signals, installing signal handlers but ignoring SIGINT. Once the IP addresses of the network interface are received, Linux.DDoS.89 connects to the Internet via a Google DNS server and a local server is launched. Once the local server is launched, the C&C server information is added to the `sockaddr_in` structure. A SIGTRAP signal is used for changing C&C servers. Finally, a `runkiller` function uses PID. It will not terminate a process if its PID is the same as the current or parental process.

Linux.DDoS.89 uses a `run_scanner` function that is copied from the Linux.BackDoor.Fgt Trojan family. It is capable of the following attacks: UDP flood, TSource flood, DNS flood, TCP flood, UDP over Generic Routing Encapsulation (GRE) flood, and Thread Environment Block (TEB) over GRE flood.

4.2 The Mirai Malware

The DDoS attacks on KrebsOnSecurity and Dyn were the largest known attacks to date (Woolf, 2016). The attacks were perpetrated using the Mirai malware which scans the Internet for unsecured IoT devices and has been active since at least August 2016, usually conscripting armies of CCTV cameras (Zeifman et al., 2016; Pultarova, 2016b). Indeed, since the source code was made publicly available, there have been several low-volume application layer HTTP floods which were characterized by relatively low requests per second counts and small numbers of source IPs and were likely new users of the malware making test runs (Zeifman et al., 2016).

Mirai infects IoT devices and uses them as a launch platform for DDoS attacks, as shown in Figure 2. Mirai's C&C module is coded in Go[5], its bots are coded in C, and communication betwen infected devices and the C&C server often occurs over port 48101 (US-CERT, 2016). Port 48101 is also used for control purposes, coordinating bot instances (Mirai, 2016). Mirai performs wide-ranging scans of IP addresses in order to locate IoT devices that could be remotely accessed via easily guessable login credentials (Moffitt, 2016). Specifically, Mirai uses a dictionary attack of at least 62 common default usernames and passwords to gain access to networked cameras, digital video recorders, and home routers (Bertino and Islam, 2017). The attack function enables Mirai to launch HTTP floods and various network DDoS attacks. Mirai is capable of launching Generic Routing Encapsulation (GRE) IP and GRE Ethernet encapsulation (ETH) floods, as well as SYN and ACK floods, STOMP (Simple Text Oriented Message Protocol) floods, DNS floods and UDP flood attacks (Loshin, 2016). Interestingly, Mirai bots are programmed to avoid certain IPs when performing their IP scans, including the US Postal Service, the Department of Defense, the Internet Assigned Numbers Authority (IANA) and IP ranges belonging to Hewlett-Packard and General Electric (Zeifman et al., 2016). Mirai also has a segmented C&C feature which allows its botnet to launch multiple DDoS attacks against multiple, unrelated targets (Loshin, 2016). Much like its predecessors, the Mirai malware displays territorial characteristics, for instance it's `runkiller` hunting for and destroying competing malware such as 'Anime'[6], and closing all processes that use SSH, Telnet, or HTTP ports. Analysis of the Mirai malware code shows that, in spite of the English C&C interface, there are Russian strings in the code and this has led to speculation that the malware originates from Russian hackers (Zeifman et al., 2016).

The Mirai attacks against KrebsOnSecurity and Dyn were so large that early estimates suggested that more than 10,000,000 devices had been conscripted for the botnet army (Zeifman et al., 2016), however later forensics determined that the attack on Dyn may have involved only 100,000 Mirai-infected devices (Woolf, 2016). Thus far, Mirai malware has been found to have infected 500,000 or more devices in 164 countries (Zeifman et al., 2016; Loshin, 2016).

[5]https://golang.org/

[6]https://evosec.eu/new-iot-malware/

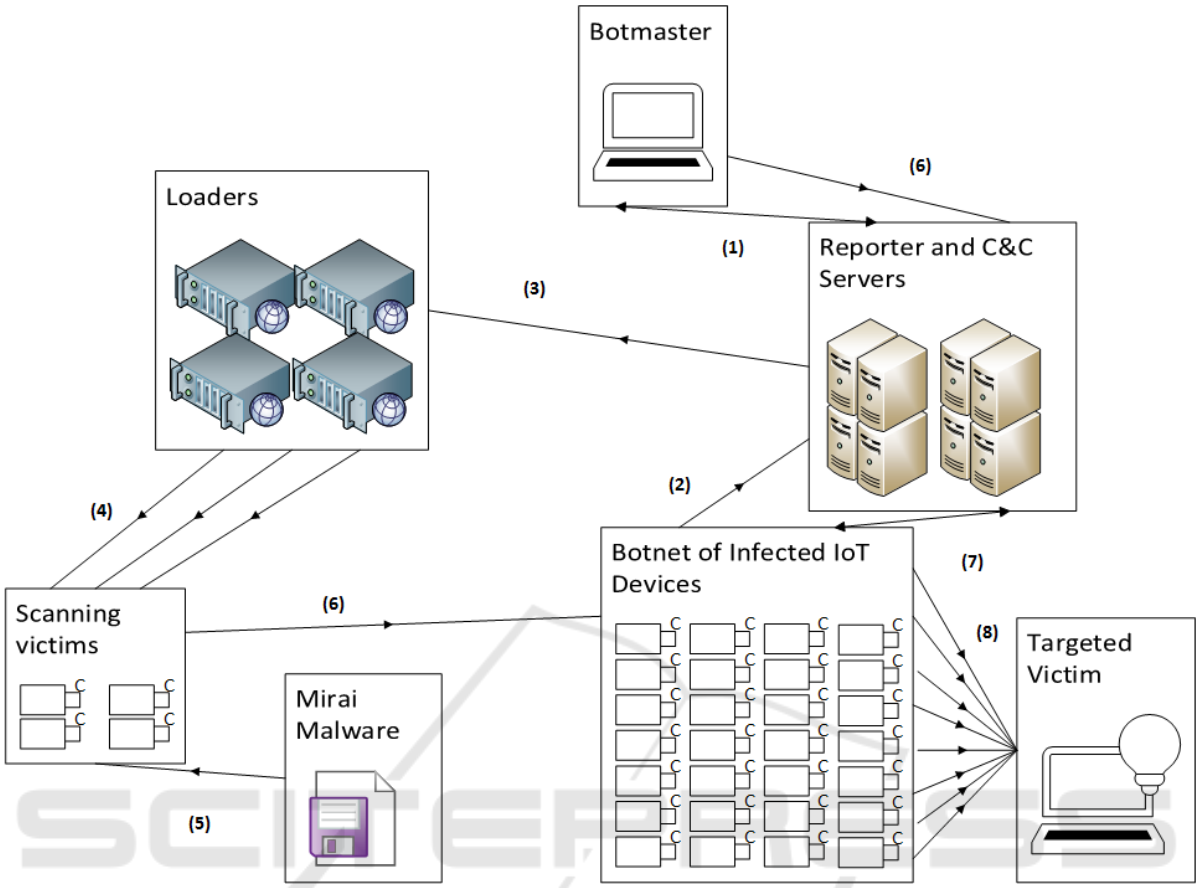

Figure 2: The Mirai Botnet Malware DDoS workflow, adapted from Level 3 Threat Research (http://netformation.com/level-3-pov/how-the-grinch-stole-iot). In (1), the botmaster maintains connection to the reporter server via a TOR connection. In (2), scan results are sent to the reporter servers. IP addresses of vulnerable IoT devices are sent to loaders in (3). In (4), loaders log into devices and instruct them to download the Mirai botnet malware. As in structed, the vulnerable IoT devices download and run the Mirai botnet malware (5) and are conscripted into a Mirai botnet (6). The botnet maintains communication with the C&C servers in (7) which constantly change their IP addresses. Finally, the Mirai botnet army conducts a DDoS attack, primarily with TCP and UDP floods in (8).

4.2.1 Analysis of Mirai Botnet C&C Behavior

The C&C portion of the Mirai malware (Mirai, 2016) contains code for handling connections, adding new clients (bots) to the conscripted botnet army, connecting admin consoles, accepting API calls, and crafting and delivering attacks for bot execution. Its database is called `mirai` and server listens for connections on the loop-back address only, with a default authentication of `root` and `password`. The `main.go` file manages initial communication, setup, and handling of incoming requests:

1. The C&C server opens up listening ports on port 23 for telnet connections and on port 101 for remote API calls on all IPv4 addresses.
2. A lightweight thread to handle API connections on port 101 is created.
3. The C&C server goes into an infinite loop waiting for telnet connections.
4. The botmaster is alerted if the C&C server fails.

4.2.1.1 `apiHandler`

The `apiHandler` function (in `main.go`) uses functionality from `api.go` to first set a timeout for accepting API commands. Once it has retrieved the command, the format for an API call embeds the API user's key with the command to verify that the user is allowed access to the API and can request attacks. Among other processes, it also checks the users' key against the database, creates a new attacks, and queues attack command for delivery to bots. In `api.go`, lines 24-30:

```
this.conn.SetDeadline(time.Now().Add(60 *
time.Second))
```

```
cmd, err := this.ReadLine()
   if err != nil {
   this.conn.Write([]byte("ERR|Failed
   reading line\r\n"))
   return
   }
passwordSplit := strings.SplitN(cmd, "|",
2)
```

Line 52:

```
atk, err := NewAttack(cmd,
userInfo.admin)
```

Line 57:

```
buf, err := atk.Build()
```

And line 71:

```
clientList.QueueBuf(buf, botCount, "")
```

4.2.1.2 `NewAttack`

`NewAttack` is a function in the `attack.go` file which is used to parse attack commands received, set attack type and duration, as well as network targets. `NewAttack` also has the ability to modify attack behavior. From `attack.go`, we highlight excerpts from lines 197-316:

```
func NewAttack(str string, admin int)
(*Attack, error) {
...
   var atkInfo AttackInfo
   if len(args) == 0 {
     return nil, errors.New("Must specify
     an attack name")
   } else {
     if args[0] == "?" {
     validCmdList := "\033[37;1mAvailable
     attack list\r\n\033[36;1m"
     for cmdName, atkInfo := range
     attackInfoLookup {
     validCmdList += cmdName + ":  " +
     atkInfo.attackDescription + "\r\n"
     }
      return nil, errors.New(validCmdList)
}
   var exists bool
   atkInfo, exists =
   attackInfoLookup[args[0]]
   if !exists {
   return nil, errors.New(fmt.Sprintf("\033
   [33;1m%s\033[31mis not a valid
   attack!",
   args[0]))
   }
   atk.Type = atkInfo.attackID
   ...
}
```

4.2.1.3 `admin.go`

`admin.go` manages authentication and welcomes users to Mirai's C&C administration console. This console provides functionality for the following commands: `adduser`, `botcount`, `quit`, and `exit`. It carries traces of potential Russian origin within the management interface, for example its prompt text,

я люблю куриные наггетсы

which translates to 'I love chicken nuggets'. More useful Russian strings in the administration console include,

произошла неизвестная ошибка

which translates to 'An unknown error occurred', and

нажмите любую клавишу для выхода

which translates to 'Press any key to exit'.

4.2.2 Analysis of Mirai Botnet Client Behavior

An analysis of the Mirai source code (Mirai, 2016) illustrates the client's behavior and workflow as follows:

1. The Trojan is launched, it immediately removes its own executable from the disk and blocks SIGINT signals.
2. The Mirai malware attempts to open the `watchdog` process, specifically `/dev/watchdog` and `/dev/misc/watchdog` for reading/writing and disables it.
3. The Mirai Trojan then attempts to obtain the IP address of its network by sending a request to Google's Public DNS server at the IP address 8.8.8.8 on port 53.
4. Mirai next calculates a function from `argv[0]` which will return an index in an array of commands to run (an integer from 0-8), and will subsequently open a local socket on the device.
5. Mirai then renames itself (to a random string), forks a child process, and executes the functions `attack_init`, `killer_init`, and `scanner_init`, which are described in detail below.

4.2.2.1 `attack_init`

The function `attack_init` contains structures that consist of attack handlers. In `attack.c`, lines 26-41:

```
add_attack(ATK_VEC_UDP,
(ATTACK_FUNC)attack_udp_generic);
add_attack(ATK_VEC_VSE,
(ATTACK_FUNC)attack_udp_vse);
add_attack(ATK_VEC_DNS,
(ATTACK_FUNC)attack_udp_dns);
```

```
add_attack(ATK_VEC_UDP_PLAIN,
(ATTACK_FUNC)attack_udp_plain);
add_attack(ATK_VEC_SYN,
(ATTACK_FUNC)attack_tcp_syn);
add_attack(ATK_VEC_ACK,
(ATTACK_FUNC)attack_tcp_ack);
add_attack(ATK_VEC_STOMP,
(ATTACK_FUNC)attack_tcp_stomp);
add_attack(ATK_VEC_GREIP,
(ATTACK_FUNC)attack_gre_ip);
add_attack(ATK_VEC_GREETH,
(ATTACK_FUNC)attack_gre_eth);
//add_attack(ATK_VEC_PROXY,
(ATTACK_FUNC)attack_app_proxy);
add_attack(ATK_VEC_HTTP,
(ATTACK_FUNC)attack_app_http);
return TRUE;
```

4.2.2.2 killer_init

The next function killer_init, located in killer.c, initializes the killer which terminates Secure Shell (SSH), HTTP, and telnet services and prevents them from restarting. This same file also contains a function called memory_scan_match that searches memory for other malware residing on the system. In killer.c, lines 39-49:

```
#ifdef KILLER_REBIND_TELNET
#ifdef DEBUG
printf("[killer] Trying to kill port
23\n");
#endif
if (killer_kill_by_port(htons(23)))
   {
   #ifdef DEBUG
   printf("[killer] Killed tcp\23
   (telnet)\n");
   #endif
   }
```

And lines 215-221:

```
if (memory_scan_match(exe_path))
   {
   #ifdef DEBUG
   printf("[killer] Memory scan match
   for binary %s\n", exe_path);
   #endif
   kill(pid, 9);
   }
```

4.2.2.3 scanner_init

scanner_init initializes the scanner scanner.c, which generates random IP addresses to scan (with the exception of those listed in Section 4.2), and contains 62 default username and password sets used to brute force other devices (scanner.c lines 124-185). Examples of these (username, password) sets include:

- (admin, admin), line 127.
- (admin, password), line 136.
- (admin, 1111), line 145.
- (admin1, password), line 156.
- (ubnt, ubnt), line 160.
- (root, user), line 172.
- (admin, pass), line 182.
- (tech, tech), line 184.

4.3 A Potential Vulnerability in the Mirai Botnet Malware

Interestingly, after reviewing the code for the Mirai C&C server, there appear to be SQL injection vulnerabilities within the server code. There is a lack of input validation on the C&C server code which could lead to a specially crafted API call to be used to leak information or gain unauthorized access to the botnet.

5 DEFENDING AGAINST CONSCRIPTION INTO A MIRAI IoT-BASED BOTNET

The Mirai malware is designed to conduct wide-ranging IP scans as it searches for IoT devices. Since the Mirai malware infects IoT devices that have common factory default usernames and passwords, the most obvious method for securing IoT devices is to change the passwords and/or usernames (Zeifman et al., 2016). Indeed, if a device is infected with Mirai then simply shutting it down and restarting it will disinfect it, however unless the login credentials are modified it is almost guaranteed to be reinfected within a few minutes (Moffitt, 2016; Zeifman et al., 2016). The United States Computer Emergency Response Team recommends taking the following steps to remove Mirai malware from infected devices (US-CERT, 2016):

1. Disconnect device from the network.
2. Perform a reboot of the device while disconnected to clear the malware from the dynamic memory.
3. Ensure that the device password has been changed from the factory default.
4. Reconnect the device to the network only after rebooting and ensuring that its password has been changed.

There are many preventative measures that can be taken to strengthen an IoT system's resistance to Mirai and other malware, assuming that its devices are not currently infected (US-CERT, 2016):

- Change device passwords before deployment on a network.
- Update IoT devices with security patches as they become available.
- Disable Universal Plug and Play on routers unless necessary.
- Monitor IP ports 2323/TCP and 23/TCP for attempts to gain unauthorized control of devices using network terminal protocol.
- Monitor port 48101 for suspicious traffic.

Additionally, IoT devices and systems should be purchased from manufacturers and vendors with a reputation for providing secure products. Users and system administrators should be aware of the capabilities of devices and systems that they are bringing into their homes and businesses; this may include medical devices or vehicles. It must be understood that if a device uses or transmits data then it may be vulnerable to infection. If a device comes with a default password or an open Wi-Fi connection, these must be secured by changing the password and only allowing it to operate on wireless networks with a secure router.

It is speculated that the DDoS attacks—extraordinarily effective thus far—have been perpetrated by non-state actors. As more and more devices are connected to the Internet, the security vulnerabilities of the IoT ecosystem will multiply with ever-more devastating consequences. The Mirai malware has taken advantage of IoT security vulnerabilities to spectacular effect, but it will certainly not be not the last malware to do so.

6 CONCLUSION

The Mirai malware is responsible for conscripting at least 500,000 IoT devices in 164 countries into a botnet army and launching DDoS attacks on a cybersecurity blog, a DNS provider in the United States, and numerous targets internationally. These IoT-botnet armies are capable of launching DDoS attacks that are larger than ever seen before, with rates exceeding 1 TBps. IoT offers revolutionary advances in many contexts, from the home to almost every type of business, but internet-connected 'things' bring security vulnerabilities as they enter the cyber ecosystem.

This paper discussed the IoT ecosystem and some of the benefits and vulnerabilities that are unique to its various environments (e.g., home use versus industrial use IoT). It discussed botnets and DDoS attacks, giving a taxonomy for DDoS attacks as well as examples, and using the BlackEnergy malware as an illustrative example of botnet malware being utilized in a coordinated campaign, before discussing the DDoS attacks by Mirai malware-infected IoT botnet armies. We reviewed the Mirai malware family, inspecting the source code for various processes of both the C&C and client modules. During our analysis of the Mirai malware code, we discovered a vulnerability to SQL injection attacks which render the botnet susceptible to a counterattack. We are currently researching the extent of this vulnerability. The Mirai malware is a very recently released malware that scans IP addresses searching for IoT devices with factory default usernames and passwords. Because the vulnerabilities that it seeks out are easily remedied, IoT devices can be strengthened against it with relative ease. (However, like BlackEnergy, future variants of the Mirai malware will likely prove much more difficult to defend against.)

The IoT ecosystem is so heavily populated with unsecured devices, and the number of these devices is expected to grow exponentially over the coming years, it is therefore reasonable to expect IoT-based DDoS attacks to become much more commonplace. Indeed, there is a Twitter feed[7] which continually monitors for Mirai botnet attacks worldwide. The Mirai malware attacks targeted a cybersecurity blog and a DNS provider, with consequences that were financial in nature. It is not unreasonable to anticipate and prepare for large-scale IoT-based botnet DDoS attacks on critical infrastructure, which would certainly have more serious and disastrous results.

ACKNOWLEDGEMENTS

This research was funded by the Office of Naval Research's Energy System Technology Evaluation Program[8]. The authors would also like to convey their gratitude to the Civilian and Uniform leadership at SPAWAR Systems Center Pacific and the United States Department of the Navy for their support of our cybersecurity research efforts.

[7] https://twitter.com/miraiattacks

[8] http://www.aptep.net/partners/estep/

REFERENCES

Amiri, I. S. and Soltanian, M. R. K. (2015). *Theoretical and Experimental Methods for Defending Against DDoS Attacks*. Syngress.

Beel, J. and Prasad, R. (2016). Internet of everything (ioe): Information technology (it) and operational technology (ot). In *Proceedings of the 2016 Advanced Machinery Technology Symposium*. American Society of Naval Engineers.

Bertino, E. and Islam, N. (2017). Botnets and internet of things security. *Computer*, 50(2):76–79.

Boyer, S. A. (2009). *SCADA: supervisory control and data acquisition*. International Society of Automation.

Boyle, P. (2000). Idfaq: Distributed denial of service attack tools: trinoo and wintrinoo. Available at https://www.sans.org/security-resources/idfaq/distributed-denial-of-service-attack-t ools-trinoo-and-wintrinoo/9/10 (2016/11/01).

Cruz, T., Queiroz, R., Simões, P., and Monteiro, E. (2016). Security implications of scada ics virtualization: survey and future trends. In *ECCWS2016-Proceedings fo the 15th European Conference on Cyber Warfare and Security*, page 74. Academic Conferences and publishing limited.

Dr.Web (2016). Investigation of linux.mirai trojan family. Available at https://st.drweb.com/static/new-www/news/2016/september/Investigation_of_Linux.Mirai_Trojan_family_en.pdf (2017/02/21).

Gasti, P., Tsudik, G., Uzun, E., and Zhang, L. (2012). Dos & ddos in named-data networking. arxive-prints. Technical report, Tech Rep 1208.0952 v2.

Gayatri, V. and Naidu, K. K. (2015). Resource depletion attacks in wireless ad-hoc sensor networks. *International Journal of Innovative Research in Computer and Communication Engineering*, 3(6):81–86.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

Hachem, N., Mustapha, Y. B., Granadillo, G. G., and Debar, H. (2011). Botnets: lifecycle and taxonomy. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–8. IEEE.

Holm, E. (2016). The role of the refrigerator in identity crime? *Cyber-Security and Digital Forensics*, page 1.

Khan, R., Maynard, P., McLaughlin, K., Laverty, D., and Sezer, S. (2016). Threat analysis of blackenergy malware for synchrophasor based real-time control and monitoring in smart grid.

Khattak, S., Ramay, N. R., Khan, K. R., Syed, A. A., and Khayam, S. A. (2014). A taxonomy of botnet behavior, detection, and defense. *IEEE communications surveys & tutorials*, 16(2):898–924.

Krebs, B. (2016a). Did the mirai botnet really take liberia offline? Available at https://krebsonsecurity.com/2016/11/did-th e-mirai-botnet-really-take-liberia-offline / (2017/02/21).

Krebs, B. (2016b). Krebsonsecurity hit with record ddos. Available at https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/ (2016/09/22).

Lee, R. M., Assante, M. J., and Conway, T. (2016). Analysis of the cyber attack on the ukrainian power grid. *SANS Industrial Control Systems*.

Lin, K.-C., Chen, S.-Y., and Hung, J. C. (2014). Botnet detection using support vector machines with artificial fish swarm algorithm. *Journal of Applied Mathematics*, 2014.

Loshin, P. (2016). Details emerging on dyn dns ddos attack, mirai iot botnet. Available at http://searchsecurity.techtarget.com/news/450401962/Details-emerging-on-Dyn-DNS-DDoS-attack-Mirai-IoT-botnet (2016/11/01).

Mirai (2016). Github: Mirai source code. Available at https://github.com/jgamblin/Mirai-Source-Code (2017/02/22).

Moffitt, T. (2016). Source code for mirai iot malware released. Available at https://www.webroot.com/blog/2016/10/10/source-code-mirai-iot-malware-released/ (2016/11/01).

Nazario, J. (2007). Blackenergy ddos bot analysis. *Arbor*.

Nazario, J. (2009). Politically motivated denial of service attacks. *The Virtual Battlefield: Perspectives on Cyber Warfare*, pages 163–181.

Newman, L. H. (2016). What we know about friday's massive east coast internet outage. Available at https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/ (2016/10/21).

Okafor, K., Okoye, J. A., and Ononiwu, G. (2016). Vulnerability bandwidth depletion attack on distributed cloud computing network: A qos perspective. *International Journal of Computer Applications*, 138(7):18–30.

Pultarova, T. (2016a). Cyber security-ukraine grid hack is wake-up call for network operators [news briefing]. *Engineering & Technology*, 11(1):12–13.

Pultarova, T. (2016b). Webcam hack shows vulnerability of connected devices. *Engineering & Technology*, 11(11):10–10.

Ragan, S. (2016). Some thoughts on the krebs situation: Akamai made a painful business call. Available at http://www.csoonline.com/article/3123797/security/so me-thoughts-on-the-krebs-situat ion-akamai-made-a-painful-business-call.ht ml (2016/11/07).

Romero-Mariona, J., Hallman, R., Kline, M., Miguel, J. S., Major, M., and Kerr, L. (2016). Security in the industrial internet of things - the c-sec approach. In *Proceedings of the International Conference on Internet of Things and Big Data - Volume 1: IoTBD,*, pages 421–428.

Santanna, J. J., van Rijswijk-Deij, R., Hofstede, R., Sperotto, A., Wierbosch, M., Granville, L. Z., and Pras, A. (2015). Booters—an analysis of ddos-as-a-service attacks. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 243–251. IEEE.

Singh, S. and Gyanchandani, M. (2010). Analysis of botnet behavior using queuing theory. *International Journal*

of Computer Science & Communication, 1(2):239–241.

Sivaraman, V., Chan, D., Earl, D., and Boreli, R. (2016). Smart-phones attacking smart-homes. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 195–200. ACM.

Sivaraman, V., Gharakheili, H. H., Vishwanath, A., Boreli, R., and Mehani, O. (2015). Network-level security and privacy control for smart-home iot devices. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on*, pages 163–167. IEEE.

Specht, S. M. and Lee, R. B. (2004). Distributed denial of service: Taxonomies of attacks, tools, and countermeasures. In *ISCA PDCS*, pages 543–550.

Stankovic, J. A. (2014). Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1):3–9.

Symantec (2016). Symantec security response: Iot devices being increasingly used for ddos attacks. Available at https://www.symantec.com/connect/blogs/iot-devices-being-increasingly-used-ddos-atta cks (2016/11/07).

Thierer, A. and Castillo, A. (2015). Projecting the growth and economic impact of the internet of things. *George Mason University, Mercatus Center, June*, 15.

US-CERT (2016). Alert (ta16-288a): Heightened ddos threat posed by mirai and other botnets. Available at https://www.us-cert.gov/ncas/alerts/TA16-2 88A (2016/10/17).

Voas, J. (2016). Networks of 'things'. *NIST Special Publication*, 800:183.

Woolf, N. (2016). Ddos attack that disrupted internet was largest of its kind in history, experts say. Available at https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet (2016/10/22).

Zeifman, I., Bekerman, D., and Herzberg, B. (2016). Breaking down mirai: An iot ddos botnet analysis. Available at https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html (2016/11/01).

Zhu, Z., Lu, G., Chen, Y., Fu, Z. J., Roberts, P., and Han, K. (2008). Botnet research survey. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 967–972. IEEE.