SAPIENZA
UNIVERSITÀ DI ROMA

IoT Honeypots: State of the Art

Master of Science in Engineering in Computer Science

Seminar in Advanced Topics in Engineering in Computer Science, a.y. 2016-17

Prof. Baldoni Roberto and Dr. Leonardo Aniello

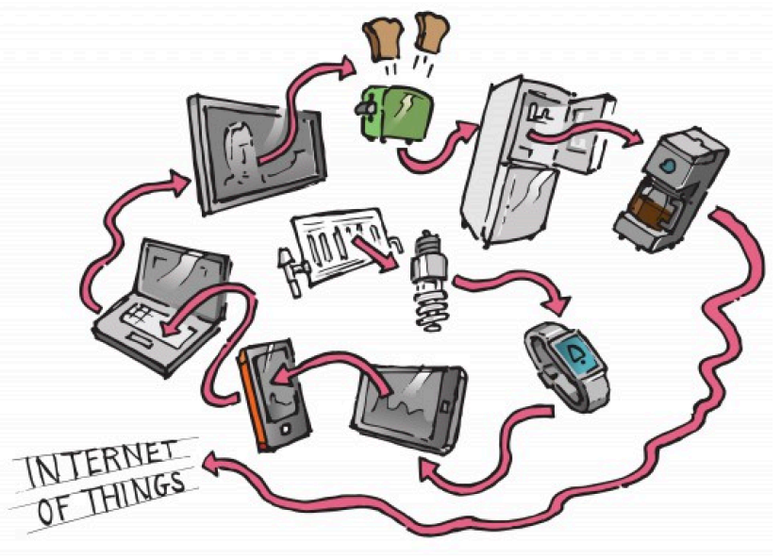
Author: Botticelli Biagio - *botticelli.1212666@studenti.uniroma1.it*

1

IoT:

Introduction

Internet of Things: A Definition



The Internet of Things describes the vision where objects become part of the Internet: where every object is uniquely identified, and accessible to the network, its position and status known, where services and intelligence are added to this expanded Internet, fusing the digital and physical world into a single one.

There is a large variety of smart IoT devices which are being introduced at each layer of IT.

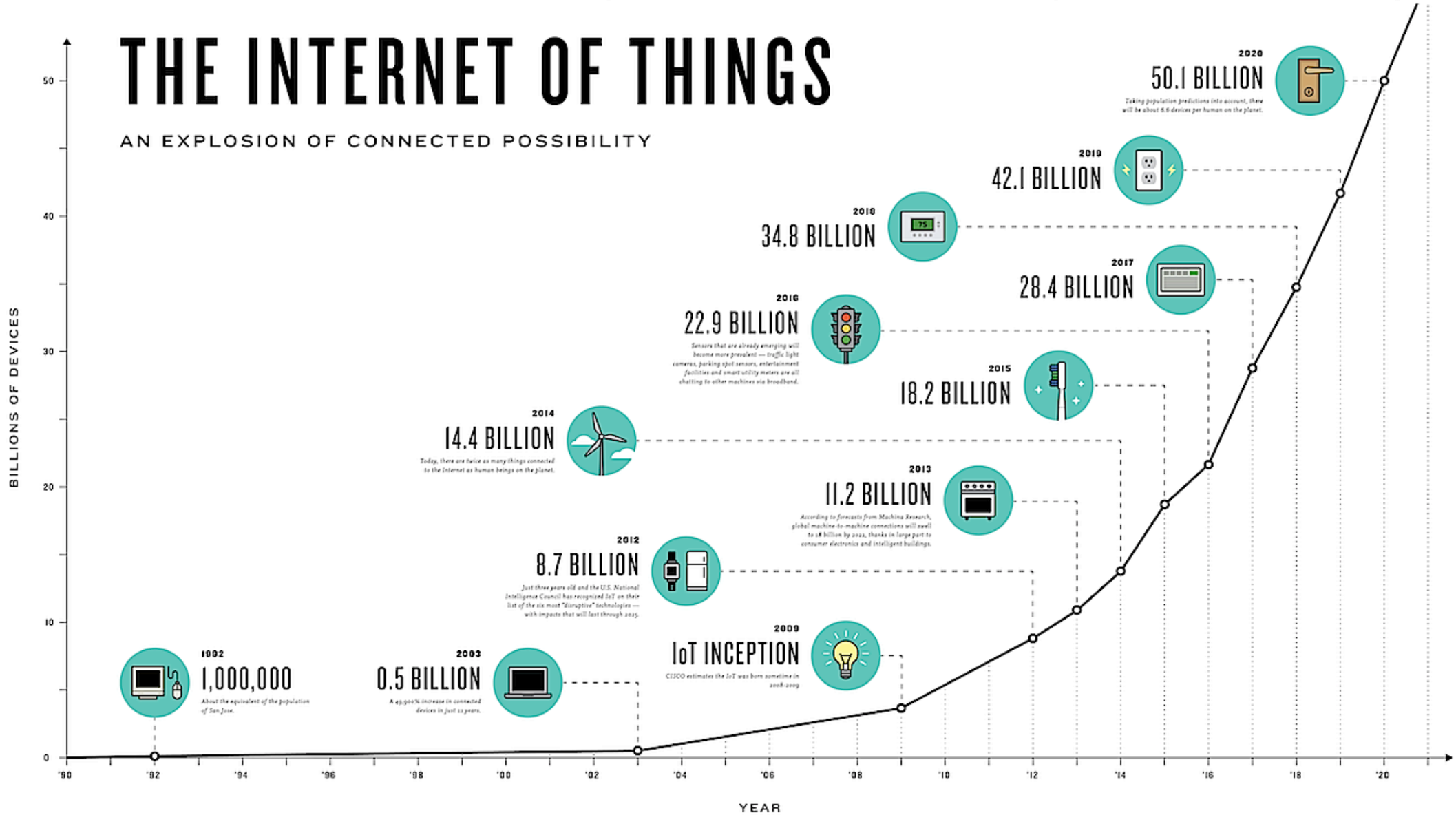
Each device has a precise purpose and specific characteristics.

But there is a common feature:

Human is not the center of the system… but a part of it!

Exponential Growth of IoT

With the rise of IoT, the number and diversity of connected devices is expected to increase **exponentially**.



An increased connectivity leads to an exponential increase in the threat surface: more smart technology we add, more likely it's to be hacked from the point of view of security.

IoT: Security Risks

The majority of IoT devices has lack of even elementary security and presents some features which make things an ideal target for hackers. In fact, wrt. to common systems, IoT systems are at **higher security risk** for several reasons:

- **Fast Connection**: IoT devices are typically connected to fast Internet networks;
- **Always Online**: wrt. to human-controlled devices, IoT ones are typically operating 24 hours a day, 7 days a week;
- **Perimeters**: IoT systems don't have well defined perimeters since they continuously change due to device/user mobility;
- **Heterogeneity**: IoT systems are highly heterogeneous wrt. to communication medium, protocols, platforms and devices;
- **Devices Control**: IoT devices can control other IoT devices and/or they can be controlled by different parties;
- **Expanded Internet**: IoT systems might include objects originally not designed to be connected to the Internet;
- **Physical Protection**: IoT systems (or portions of them) could be physically unprotected;
- **Permissions**: granular permission requests might not be possible in IoT systems;

Main IoT security challenges come from the **heterogeneity** and the **large scale** of IoT devices.

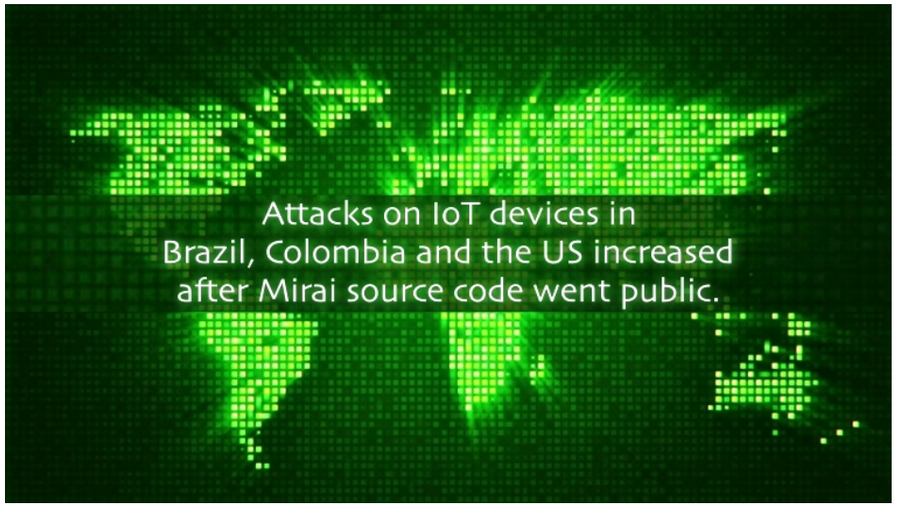
IoT: OWASP Top 10 Vulnerabilities

Vulnerability	Exploitability	Detectability	Security Impact	Examples
Insecure Web Interface	EASY: Attacker uses weak credentials, captures plain-text credentials or enumerates accounts to gain the access to the web interface. Attack could come from external or internal users.	EASY: An insecure web interface can be present when account enumeration, lack of account lockout or weak credentials are present. Issues with the web interface are easy to discover when examining the interface manually along with automated tools to identify other issues such as cross-site scripting.	SEVERE: Insecure web interfaces can result in data loss or corruption, lack of accountability, or denial of access and can lead to complete device takeover.	Inability to change default usernames and passwords; weak passwords; lack of robust password recovery mechanisms; exposed credentials; lack of account lockout; susceptibility to cross-site scripting, cross-site request forgery, and/or SQL injection.
Insufficient Authentication/ Authorization	AVERAGE: Attacker uses weak passwords, insecure password recovery mechanisms, poorly protected credentials or lack of granular access control to access a particular interface. Attack could come from external or internal users.	EASY: Authentication may not be sufficient when weak passwords are used or are poorly protected. Many Issues with authentication/authorization are easy to discover when examining the interface manually and can also be discovered via automated testing.	SEVERE: Insufficient authentication/authorization can result in data loss or corruption, lack of accountability, or denial of access and to complete compromise of the device and/or user accounts.	Privilege escalation; lack of granular access control.
Insecure Network Services	AVERAGE: Attacker uses vulnerable network services to attack the device itself or bounce attacks off the device. Attack could come from external or internal users.	AVERAGE: Insecure network services may be susceptible to buffer overflow attacks or attacks that create a denial of service condition leaving the device inaccessible to the user. Insecure network services can be detected by automated tools such as port scanners/fuzzers.	MODERATE: Insecure network services can result in data loss or corruption, denial of service or facilitation of attacks on other devices.	Vulnerability to denial-of-service, buffer overflow, and fuzzing attacks; network ports or services unnecessarily exposed to the Internet.
Lack of Transport Encryption/Integrity Verification	AVERAGE: Attacker uses the lack of transport encryption to view data being passed over the network. Attack could come from external or internal users.	EASY: Lack of transport encryption allows data to be viewed as it travels over local networks or the internet. Many Issues with transport encryption are easy to discover simply by viewing network traffic and searching for readable data. Automated tools can also look for proper implementation of common transport encryption such as SSL and TLS.	SEVERE: Lack of transport encryption can result in data loss and depending on the data exposed, could lead to complete compromise of the device or user accounts.	Transmission of unencrypted data and credentials.
Privacy Concerns	AVERAGE: Attacker uses insufficient authentication, lack of transport encryption or insecure network services to view personal data which is not being properly protected or is being collected unnecessarily. Attack could come from external or internal users.	EASY: Privacy concerns generated by the collection of personal data in addition to the lack of proper protection of that data is prevalent. Privacy concerns are easy to discover by reviewing the data that is being collected as the user sets up and activates the device. Automated tools can also look for specific patterns of data that may indicate collection of personal data or other sensitive data.	SEVERE:Collection of personal data along with a lack of protection of that data can lead to compromise of a user's personal data.	Collection of unnecessary user data; exposed personal data; insufficient controls on who has access to user data; sensitive data not de-identified or anonymized; lack of data retention limits.
Insecure Cloud Interface	AVERAGE: Attacker uses insufficient authentication, lack of transport encryption and account enumeration to access data or controls via the cloud website. Attack will most likely come from the internet.	EASY: An insecure cloud interface is present when easy to guess credentials are used or account enumeration is possible. Insecure cloud interfaces are easy to discover by simply reviewing the connection to the cloud interface and identifying if SSL is in use or by using the password reset mechanism to identify valid accounts which can lead to account enumeration.	SEVERE: An insecure cloud interface could lead to compromise of user data and control over the device.	Inability to change default usernames and passwords; weak passwords; lack of robust password recovery mechanisms; exposed credentials; lack of account lockout; susceptibility to cross-site scripting, cross-site request forgery, and/or SQL injection.
Insecure Mobile Interface	AVERAGE: Attacker uses insufficient authentication, lack of transport encryption and account enumeration to access data or controls via the mobile interface. Attack will most likely come from anyone who has access to the mobile application.	EASY: An insecure mobile interface is present when easy to guess credentials are used or account enumeration is possible. Insecure mobile interfaces are easy to discover by reviewing the connection to the wireless networks and identifying if SSL is in use or by using the password reset mechanism to identify valid accounts which can lead to account enumeration.	SEVERE: An insecure mobile interface could lead to compromise of user data and control over the device.	Inability to change default usernames and passwords; weak passwords; lack of robust password recovery mechanisms; exposed credentials; lack of account lockout; susceptibility to cross-site scripting, cross-site request forgery, and/or SQL injection.
Insufficient Security Configurability	AVERAGE: Attacker uses the lack of granular permissions to access data or controls on the device. The attacker could also use the lack of encryption options and lack of password options to perform other attacks which lead to compromise of the device and/or data. Attack could potentially come from any user of the device whether intentional or accidental.	EASY: Insufficient security configurability is present when users of the device have limited or no ability to alter its security controls. Insufficient security configurability is apparent when the web interface of the device has no options for creating granular user permissions. Manual review of the web interface will reveal these deficiencies.	MODERATE: Insufficient security configurability could lead to compromise of the device whether intentional or accidental and/or data loss.	Lack of granular permissions model; inability to separate administrators from users; weak password policies; no security logging; lack of data encryption options; no user notification of security events.
Insecure Software/ Firmware	DIFFICULT: Attacker uses capturing update files via unencrypted connections, the update file itself is not encrypted or they are able to perform their own malicious update via DNS hijacking. Attack could come from the local network or the internet.	EASY: The lack of ability for a device to be updated presents a security weakness on its own. Devices should have the ability to be updated when vulnerabilities are discovered and software/firmware updates can be insecure when the updated files themselves and the network connection they are delivered on are not protected. Software/firmware issues are easy to discover by simply inspecting the network traffic during the update to check for encryption or using a hex editor to inspect the update file itself for interesting information.	SEVERE: Insecure software/firmware could lead to compromise of user data, control over the device and attacks against other devices.	Lack of secure update mechanism; update files not encrypted; update files not verified before upload; insecure update server; hardcoded credentials.
Poor Physical Security	AVERAGE: Attacker uses vectors such as USB ports, SD cards or other storage means to access the Operating System and potentially any data stored on the device. Attack could come from anyone who has physical access.	AVERAGE: Physical security weaknesses are present when an attacker can disassemble a device to easily access the storage medium and any data stored on that medium. Weaknesses are also present when USB ports or other external ports can be used to access the device using features intended for configuration or maintenance.	SEVERE: Insufficient physical security could lead to compromise of the device itself and any data stored on that device.	Device easy to disassemble; access to software via USB ports; removable storage media.

Mirai Botnet: Example of IoT as a weapon

20th September 2016 : KrebsOnSecurity.com was targeted by an extremely large and unusual Distributed Denial-of-Service (DDoS) attack of over 660 Gbps of traffic.

The most interesting aspect of this attack is that it was <u>not</u> performed by using traditional reflection/amplification DDoS, but with direct traffic generated by a botnet (or zombie network) of hacked IoT devices.



The IoT devices were infected by a malware called **Mirai** using Telnet connection and very simple dictionary attacks.

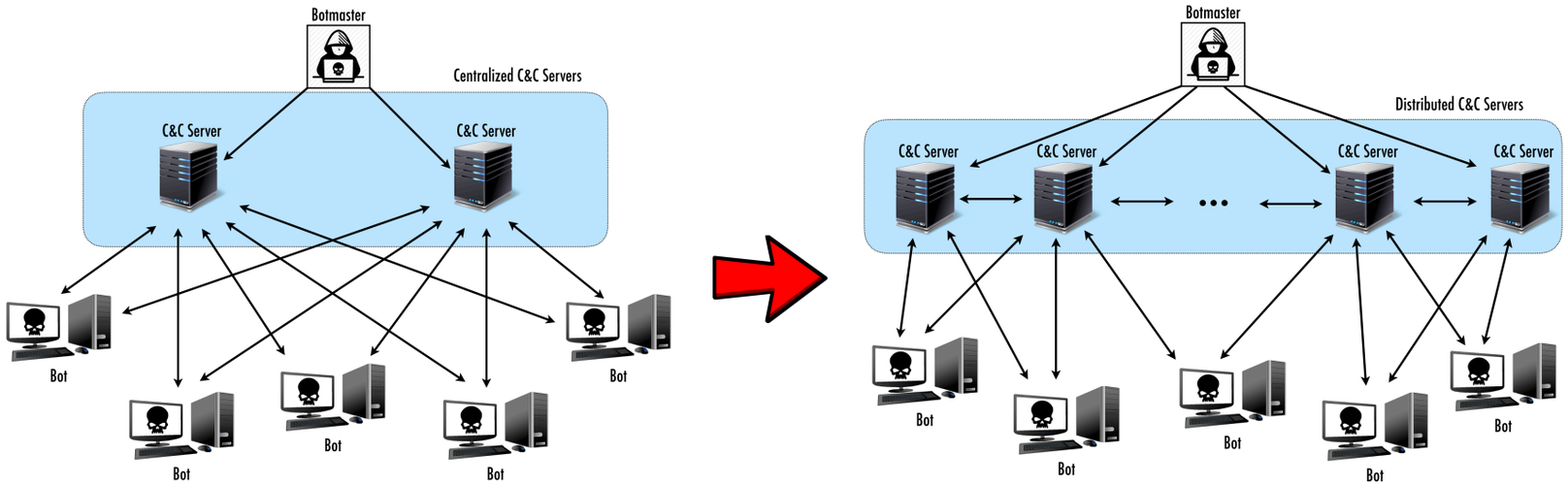
Once the device has become infected, it's made part of the Mirai botnet that will be used to make the DDoS attack.

2

Botnets & DDoS Attacks

Botnets

- A botnet is a robot network of hacked machines, or bots (usually infected by trojan, backdoors or worms), which run malicious code under the remote command and control (C&C) of a botmaster.
- Botnets have a wide range of malicious purposes including email spam delivery, denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks, permanent denial of service (PDoS), password cracking, key logging and crypto currency mining.
- Early botnets was organized in a hierarchical centralized architecture, in which the bots relied on the Internet Relay Chat (IRC) for communicating with their botmaster.
- As detecting strategies evolved to monitor or block IRC, botnets shifted their communication channels previous to HTTP and then to an alternative distributed Peer-to-Peer (P2P) architecture.
- Early botnet-based DDoS attacks were manual and attacker had to implement many steps before the launch of final attack.
- Nowadays, sophisticated automated DoS or DDoS attack tools been developed to assist attackers in implementing some (or all) steps automatically, with minimal human effort. Some of these tools are: TFN, Trinoo, TFN2K, Shaft, Stacheldraht, Knight and Trinity.

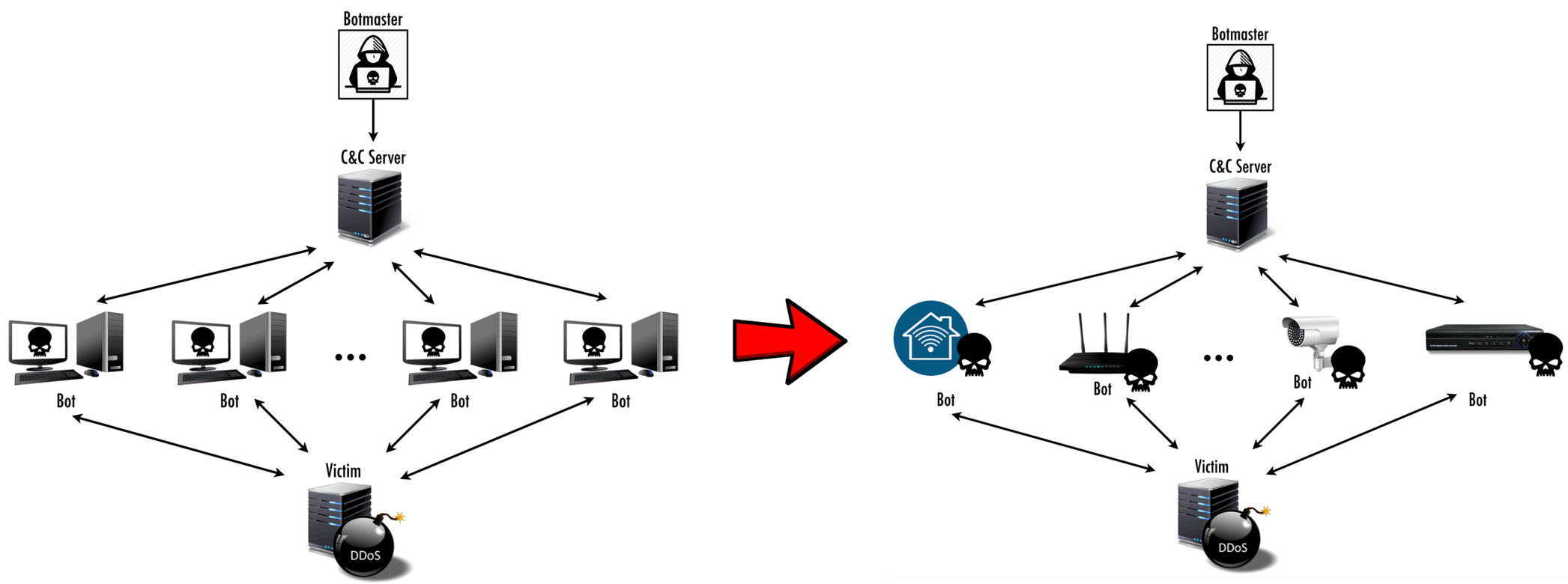


Centralized Botnet Architecture

P2P Botnet Architecture

From botnets to thingsbots

- A conventional botnet is composed by computers that have been remotely accessed without the owners' knowledge and set up to exploit attacks against other hosts on the Internet.
- The Internet of Things (IoT) is made up of sensors and devices equipped with IP addresses and the ability to transmit data over a network.
- With the rise of the IoT, things are in danger of become part of thingsbots.
- The term Botnet of Things (BoT), or thingsbot, refers to a botnet of compromised IoT devices which are infected and controlled to exploit cyber-crime activities (such as DDoS attacks).

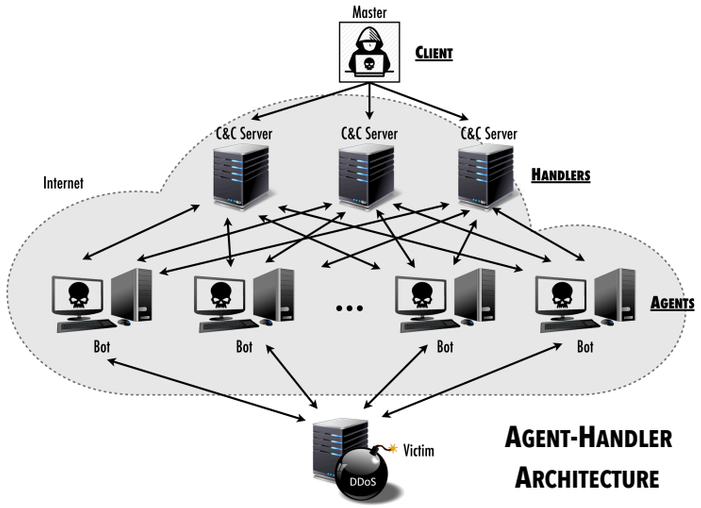


Botnets: DoS and DDoS attacks

Availability is one of the main components of IT security, along with **confidentiality** and **integrity**.

Denial of Service (DoS) and Distributed Denial of Service (DDoS) are considered serious threats that exhaust the computing and communication resources of a victim system within a short period of time, making these resources unavailable for legitimate users.

Denial of Service: a non-authorized user is said to deny a service to another authorized user if the former user makes the specified service unavailable to the latter one, for a period of time which exceeds the intended waiting time. The use of single computer to carry out these attacks is termed as **Denial Of Service (DoS),** whereas the use of multiple computers (called zombies) to carry out a coordinated attack is called **Distributed Denial of Service (DDoS)** attack.

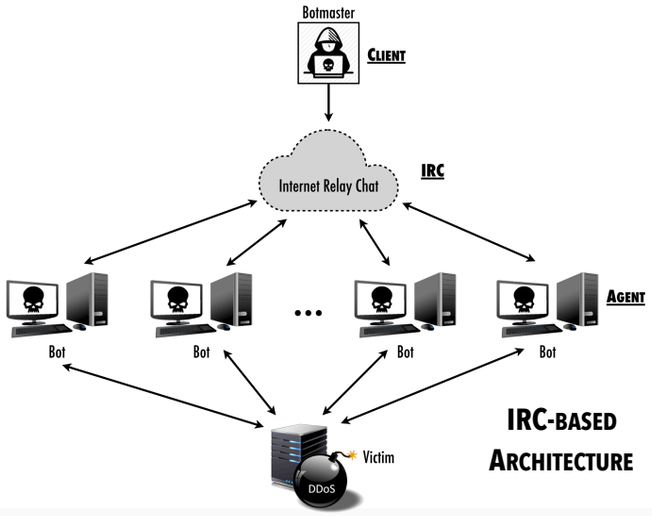


A DDoS attack is launched in two phases:

1. Attacker builds an attack network which is distributed and consists of different infected devices called zombies or bots while the main controller machine is called master.
2. Attacker exploits the attack against target system by flooding a tremendous volume of traffic either manually or automatically.

The master-zombie architecture is an Agent-Handler model that consists of client (attacker), handlers (C&C servers), and agents (zombies).

Another type of architecture is the IRC-based model: an Internet Relay Chat (IRC) communication channel is used to connect the client to the zombies.



Types of DDoS: Bandwidth Depletion Attacks (1)

There are two main classes of DDoS attacks: bandwidth and resource depletion attacks.

1. **Bandwidth Depletion**: attack is designed to flood the victim network with unwanted traffic that prevents legitimate traffic from reaching the victim system by exhausting all the available network bandwidth.
2. **Resource Depletion**: attack is designed to bind all the available resources of the target system (such as CPU, memory, I/O bandwidth, disk space, or any combination of them), making the victim unable to process legitimate requests for service.

1. <u>**Bandwidth Depletion Attacks**</u>

   A. **Flood Attack:** it involves zombie systems sending large volumes of traffic to a victim system to congest the victim's available network bandwidth with IP traffic. The victim system slows down or crashes, preventing access by legitimate users.

   - **ICMP Flood Attack:** it involves zombie systems flooding the victim's network with **ICMP ping** packets for a reply from the victim resulting in the saturation of the victim's network bandwidth (both outgoing and incoming).
   - **UDP Flood Attack:** it involves zombie systems flooding the victim's network with **UDP packets** on random/specified ports of the victim system. The victim system tries to determine which application have requested the incoming data. If the victim system is not running any application on the targeted port, it will respond with an **ICMP packet** to the sending system indicating a **port unreachable** message. If the number of packets being delivered to ports of the victim is large, the host resources will be depleted and it will lead to unavailability.

Types of DDoS: Bandwidth Depletion Attacks (2)

B. Amplification Attack: it involves the attacker sending messages to a broadcast IP address, causing all systems within the broadcast network sending a reply to the victim system. In this kind of attacks, the broadcast IP address is used to amplify and reflect the attack traffic, and thus reducing the victim's bandwidth.

- **Smurf Attack:** the attack uses a network amplifier and IP broadcasting in which, when a packet is sent to an IP broadcast address the packet is delivered to all machines on that subnet. The attacking hosts forge ICMP echo requests (similar to ping) having the spoofed victim's address as source address and the broadcast address of these remote networks as the destination address. Since the packets are sent at broadcast address, the packets are received by all the nodes within the broadcast address range in the network. Then, each node responds back to the victim machine (since the source IP address is spoofed). This creates a large amount of echo response packets sent to the target making the victim's network unstable and causing network congestion.

- **Fraggle attack:** the attacker sends packets to a network amplifier, sending UDP echo packets to the port that supports character generation (*chargen*, port 19 in Unix systems), with the return address spoofed to the victim's echo service (*echo*, port 7 in Unix systems) creating an infinite loop. The UDP Fraggle packet will target the character generator in the systems reached by the broadcast address. These systems generate a character to send to the echo service in the victim system, which will send an echo packet back to the character generator, and the process repeats. This attack can generate more bad traffic and cause more damage than a Smurf attack.

Types of DDoS: Resource Depletion Attacks

2. <u>Resource Depletion Attacks</u>

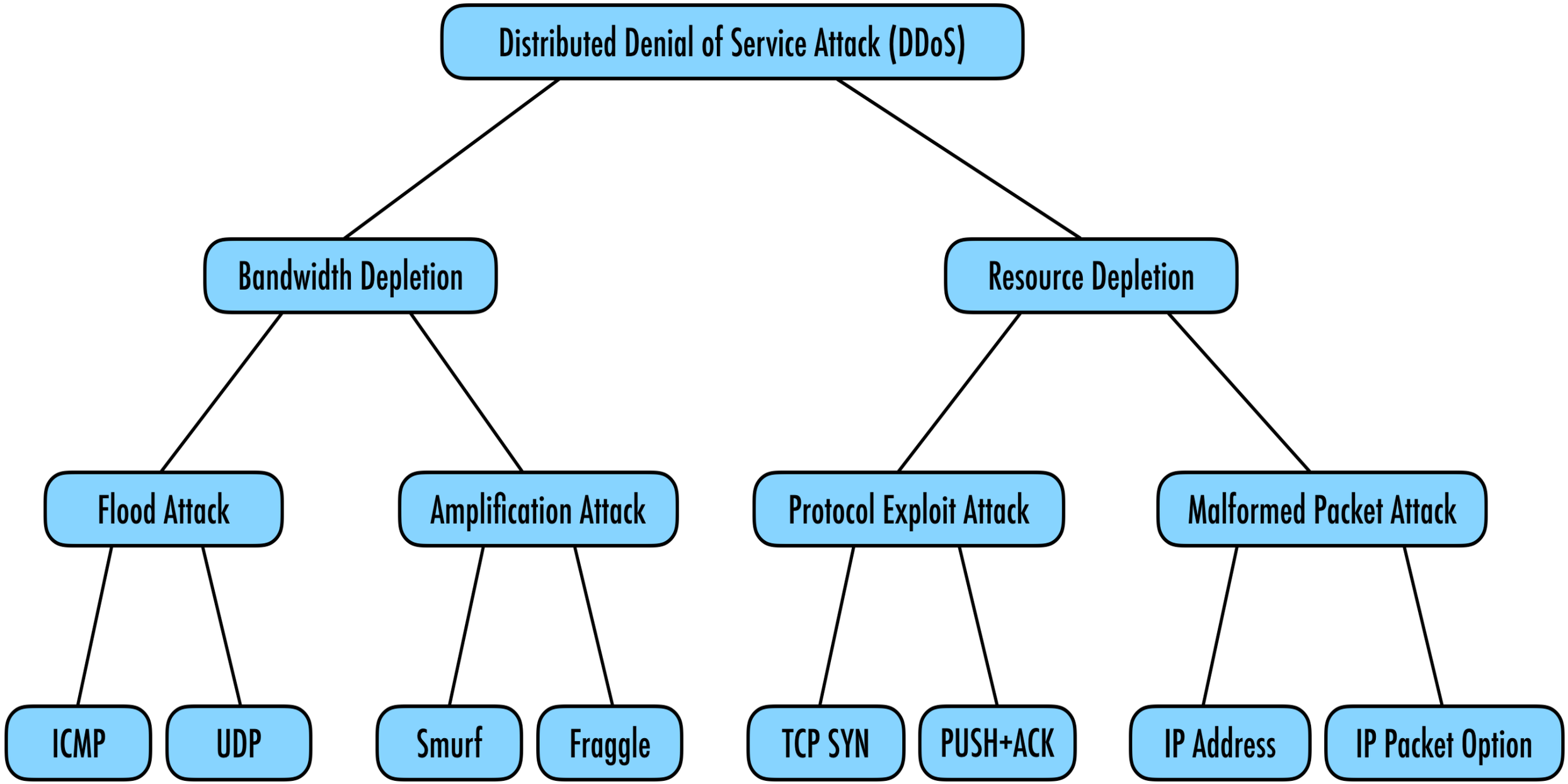
   A. **Protocol Exploit Attack:** There are two main attacks which exploit protocol vulnerabilities. one misusing the TCP SYN (Transfer Control Protocol Synchronize), and the other misusing the PUSH+ACK protocol.

   - **TCP SYN Attack**: the attacker sends fake **TCP SYN requests** to a victim in order to tie up its resources, hence preventing the target host from responding to legitimate requests.
     The TCP SYN attack floods the victim with half open connections: attacker sends TCP SYN packets with spoofed source IP addresses, so the victim responds to a non-requesting system with an ACK+SYN response and it waits for an ACK that will never come. When a large volume of SYN requests are processed and none of the ACK+SYN responses are returned, the victim runs out of resources and is unable to respond to legitimate users.
   - **PUSH+ACK Attack**: the attacker sends TCP packets with the **PUSH** and **ACK** bits set to one. These instruct the victim system to unload all data in the TCP buffer and send an acknowledgement when complete. If this process is repeated with multiple agents, the receiving system cannot process the large volume of incoming packets and it will crash.

   B. **Malformed Packet attacks:** the attacker sends malformed IP packets to the victim system in order to crash it.

   - **IP Address Attack:** the IP packet contains the same source and destination IP addresses. This can confuse the operating system of the victim system and can cause the victim system to crash.
   - **IP Packet Options Attack:** a malformed packet may randomize the optional fields within an IP packet and set all QoS bits to 1 so that the victim must use additional processing time to analyze the traffic. If this attack is multiplied, it can exhaust the processing ability of the victim system.

DDoS Taxonomy



3 IoT Malware

Malware targeting IoT (1)

The definition of IoT Malware directly follows by the kind of devices targeted: what makes IoT malware different from any other type of malware is that it targets a specific class of devices, Internet of Things devices.

Malware samples properly designed to infect IoT devices are born to create thingsbots exploiting IoT-based DDoS attacks.

- **Hydra**: it is a router malware considered as the earliest known malware targeting IoT devices (appeared in 2008). It was managed by IRC and its target were routers exploited by brute force methods for the purposes of performing DDoS attacks.
- **Psyb0t**: it's an IRC botnet targeting routers and DSL modems based on C&C servers. Psyb0t is self-proliferating and has commands for DDoS attacks, brute-force access, commands injection for services like MySQL, PHP MyAdmin, FTP, etc.
- **Chuck Norris**: it's a IRC bot targeting routers and DSL modems that reminds in many ways Psyb0t.
- **Tsunami**: very similar to Chuck Norris, with an innovating exploit to change DNS settings and to block the range of ports from 22 through to 80. It's a DDoS however, but it can also grant the botmaster the full control of the infected system.
- **LightAidra/Aidra:** it's an IRC-based mass scanning TCP flood DDoS tool supported on several IoT architectures (MIPS, MIPSEL, ARM, PPC, x86/86-64 and SuperH).
- **Carna**: it's a botnet created to measure the extent of the Internet and to get an estimation of the IP address usage in what the creator called the "Internet Census of 2012".
- **Linux.Darlloz:** it's a worm which attacks IoT devices by exploiting an old PHP vulnerability (CVE-2012-1823) to access the system, make a privilege escalation through default and common credential lists, propagate in the network and establish a backdoor on the infected system.

Malware targeting IoT (2)

- **Qbot:** it's a network-aware worm capable of discovering credentials and creating backdoors on IoT systems. It also leverages Rig Exploit Kit against vulnerable websites to gain write access on the backend and to inject malicious JavaScript onto the site.
- **Wifatch:** it's a malware which infects IoT devices with embedded Linux (ARM, MIPS and SH4) having weak or default credentials. It's a vigilante: once the device is infected, it does not make malicious actions; instead, it removes other malwares and disables Telnet access while logging a warning message.
- **TheMoon:** it's an IoT P2P worm specifically targeting Linksys and Asus routers and exploiting a command execution vulnerability on their hard-coded iptables rules.
- **Spike/Dofloo:** is a backdoor/DDoS malware targeting quite all device architectures (32-bit/64-bit Windows, Linux PCs as well as MIPS and ARM IoT devices). It has various DDoS payloads including SYN floods, UDP floods, DNS query floods, and GET floods.
- **BASHLITE** (also known as Gafgyt, Lizkebab, Torlus, LizardStresser, Bash0day and Bashdoor): it's a malware which infects Linux systems in order to launch DDoS attacks, but it's also designed to easily cross-compile to various computer architectures. It exploits a flaw in the bash shell (the Shellshock software bug) to infect devices running BusyBox.
- **Remaiten:** it's a IoT Malware which combines the features of Tsunami and BASHLITE. It infects embedded Linux IoT devices by brute force, it discovers the victim's architecture and then download the appropriate malicious code; so, in practice, it can adapt itself based on the device and the type of attacks (file/malware download, command execution, DDoS, etc.) it wants to launch.

Malware targeting IoT (3)

- **Mirai:** is one of the most predominant DDoS IoT botnet in recent times derived by BASHLITE. It brute-forces the target by Telnet, it exploits the infection and then it self-propagates over the network. Moreover it can exploit all main DDoS attacks.
- **IRCTelnet** (also known as newAidra): it's the first successor to Mirai and it's a IRC botnet ELF malware aimed at IoT devices with IPv6 capabilities. It borrows some key concepts from the most important botnets like Tsunami (IRC protocol), BASHLITE (infection techniques, i.e. telnet brute force access and code injection) and Mirai (botnet's IoT credential list).
- **Hajime**: it's an IoT worm that uses a three-stage infection mechanism (check, exploit and propagation) and self-propagates. It presents a new feature: it employs the BitTorrent protocol to communicate with C&C; so, there is no explicit harcoded C&C address in Hajime code, as it employs the P2P network to receive commands. It's a vigilante fighting Mirai: it infects the same devices of Mirai without performing malicious activities (not yet) and log a warning message every 10 minutes.
- **Amnesia**: it's a botnet targeting an unpatched remote code execution vulnerability that was publicly disclosed in DVR devices made by TVT Digital (approx. 227.000 devices). Amnesia is the first malware to adopt virtual machine evasion techniques to defeat malware analysis sandboxes.
- **BrickerBot**: IoT botnet whose sole purpose is to compromise IoT devices and corrupt their storage. It perform Permanent Denial of Service attack (also known as phlashing) against infected devices: this attack damages a system so badly that it requires replacement or reinstallation of hardware. By exploiting security flaws or misconfigurations, this type of cyber attack can destroy the firmware and/or basic functionalities of a system.

IoT Malware Summary (1)

Malware	Target	Platform	Access Method	Purpose	First Appearance
Hydra	Routers	Linux	Brute-Force (Default Passwords on Telnet) or Authentication Bypass	DDoS Attacks	2008
Psyb0t	Routers and DSL Modems	Linux MIPSel	Brute Force (Default Passwords on Telnet/SSH) or Authentication Bypass	DDoS Attacks	2009
Chuck Norris	Routers and DSL Modems	Linux MIPSel	Brute Force (Default Passwords on Telnet/SSH) or Authentication Bypass	DDoS Attacks	2010
Tsunami	Routers and DSL Modems	Linux MIPSel	Brute Force (Default Passwords on Telnet) or Authentication Bypass	DDoS Attacks, DNS Redirection, Command Execution	2010
LightAidra	IoT Devices with Embedded Linux	MIPS, MIPSEL, ARM, PPC, x86/86-64, SuperH	Brute Force (Default/No Passwords on Telnet) or Bug Exploitation on D-Link devices	DDoS Attack, Command Execution	2013
Carna	IoT Devices	Linux	Brute Force (Default/No Passwords)	"Internet Census"	2012
Darlloz	IoT Devices with Embedded Linux	Intel x86 (1st version) x86, ARM, MIPS, MIPSEL, PPC (Later)	PHP Vulnerability (CVE-2012-1823)	Crypto-Currency Mining, Command Execution	2013
QBot	Vulnerable Websites	Windows	Rig Exploitation Kit	Command Injection Command Execution	2009
Wifatch	IoT Devices	ARM, MIPS, SH4	Brute Force (Default/No Passwords on Telnet)	Security Improvement	2014

IoT Malware Summary (2)

Malware	Target	Platform	Access Method	Purpose	First Appearance
TheMoon	Linksys and Asus Routers	Firmware iptables	Command Execution Vulnerability	Command Execution	2014
Spike/Dofloo	Windows, Linux PCs, IoT devices	32 and 64-bit Windows, Linux, MIPS and ARM	Brute Force (Default/No Passwords)	DDoS Attacks, Command Execution	2014
BASHLITE	IoT Devices, Routers, Linux Servers	Designed to be compiled for all Embedded Linux architectures	Brute Force (Default/No Passwords) Shellshock Vulnerability (BusyBox), Bashdoor	DDoS Attacks, Command Execution	2014
Remaiten	IoT Devices with Embedded Linux	Embedded Linux (tipically MIPS and ARM)	Brute Force (Default/No Passwords on Telnet)	DDoS Attacks, Command Execution	2016
Mirai	IoT Devices with Embedded Linux, Windows	Embedded Linux, Windows	Brute Force (Default/No Passwords on Telnet)	DDoS Attacks	2016
IRCTelnet	IoT Devices with IPv6 Capabilities	Embedded Linux	Brute Force (Default/No Passwords on Telnet) Code Injection	DDoS Attacks	2016
Hajime	IoT Devices with Embedded Linux	Embedded Linux, ARM, Intel x86-64, MIPS	Brute Force (Default Passwords on Telnet)	Security Improvement	2016
Amnesia	TVT Digital Digital Video Recorders	Embedded Linux	RCE Vulnerability, Remote Code Execution	DDoS Attacks Command Execution	2016
BrickerBot	Linux/Busybox based IoT Devices	Embedded Linux	Brute Force (Default Passwords on Telnet)	PDoS	2017

4 IoT Honeypots

IoT Honeypots: Introduction

Honeypots are the most promising technology to counter the problem of finding a methodology which can provide a coverage on detecting IoT botnets exploiting DDoS attacks.

Honeypot: trap system that appears to be part of the network (but typically isolated and protected) containing information or functionalities that would be of value for attackers. Its aim is to attract hackers to fall into it, trying to monitor the interaction between the attackers and the infected devices once they think they've exploited a vulnerable system and trying to capture the downloaded malware samples. The primary value given by the honeypot is the information about malware samples and attack patterns which is possible to obtain.

An honeypot can be seen as a fake system that looks like a real one and which main functions are:

- Mislead the attention of the attacker from the real network, in a way that real resources could be not compromised.
- Capture new viruses or worms samples for future analysis.
- Create attacker profiles in order to identify their preferred attack methods.
- Detect new vulnerabilities and threats of systems which are not clearly identified at the moment.

IoT Honeypots: Pros & Cons

Advantages

+ Honeypots capture attack into logs and log analysis allows to find details about exploits and attack patterns.
+ Honeypots are designed to capture anything that interacts with them, including tools or tactics never seen before (0-Days).
+ Honeypots only deal with incoming malicious traffic; so, the collected information is smaller and higher-value.
+ Fewer false positives with respect to other security solutions, since only attack traffic is detected (no legitimate traffic).
+ Honeypots require minimal resources, not costing additional budget to companies.
+ Honeypots are simple to understand, to configure and to install.
+ Honeypots do not require known attack signatures (unlike IDS).
+ Honeypots will detect an IPv6-based attack the same way it does with an IPv4 attack.

Disadvantages

- Honeypots suffer of fingerprinting: it is easy for an experienced hacker to differentiate a fake system from a real one.
- Honeypots lead to a risk: an hijacked honeypot system (if not properly designed) may be used to attack other systems.
- Honeypots have a limited vision: an honeypot can only capture data of directly interacting hacking activity.

Types of IoT Honeypots

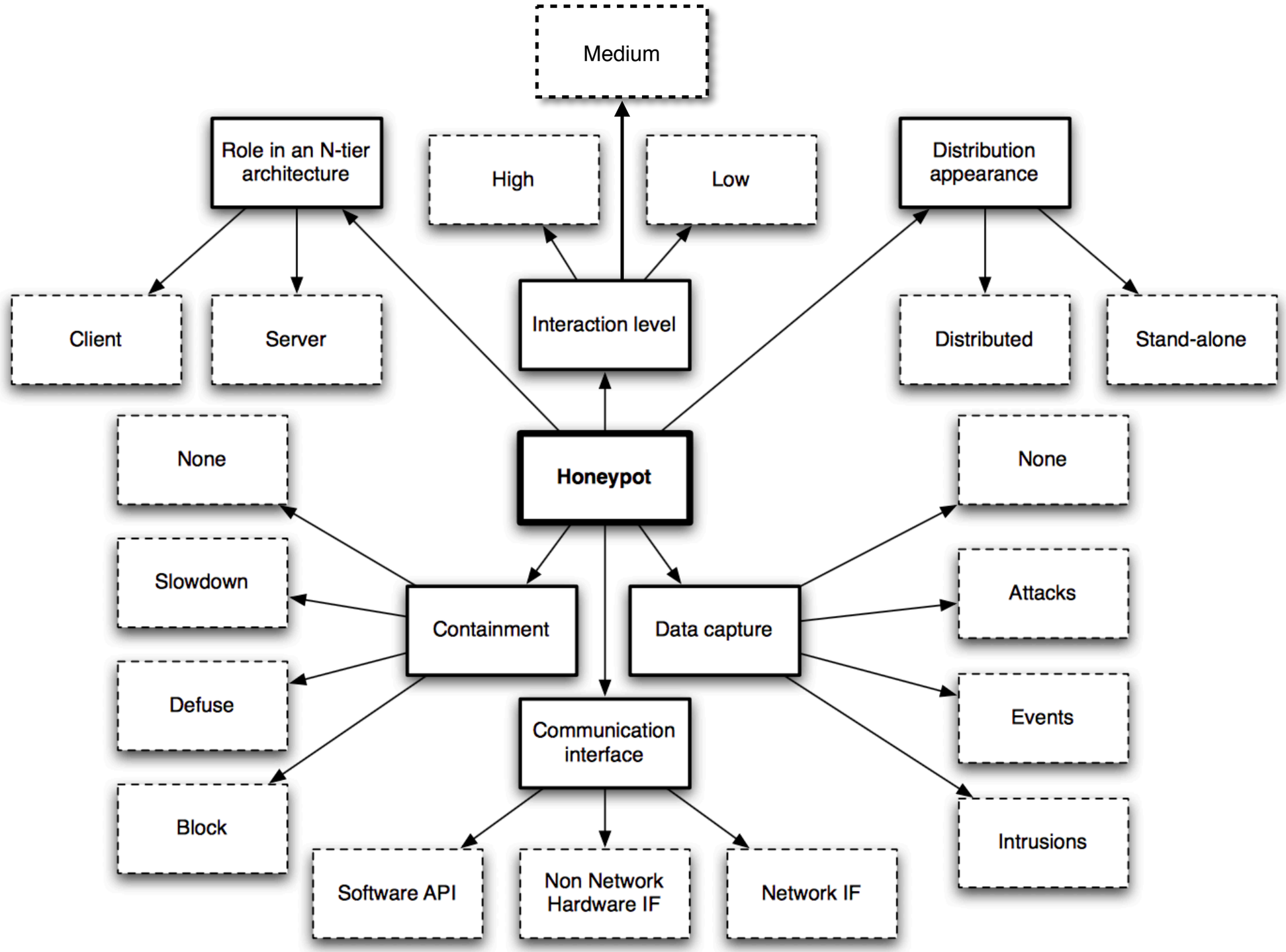
Different honeypot system can be differentiated according to:

- <u>Aim</u>
  - A. **Research Honeypots:** their aim is to discover and learn how to better protect a system from all attack patterns and threats, not bringing any direct value to the security of an organization, since they are not designed to protect networks.
  - B. **Production Honeypots:** they are used to protect companies in real production environments, to protect from attacks. They are implemented inside the network (parallel to IT Infrastructures) to improve the overall security.

- <u>Level of Interaction</u>
  - A. **Low:** they present to attackers emulated services (no OS) with a limited subset of the functionality they would expect, with the intent of detecting sources of unauthorized activity. Using them, the analyst can get the ***least*** amount of data.
  - B. **Medium:** they provide an expanded interaction from low interaction honeypots but less than high interaction ones; they provide partial implementation of services and do not allow full interaction with the system. In medium interaction honeypots still OS doesn't exist; but this time, more information can be obtained from the attack.
  - C. **High:** they are the most advanced honeypots and they let the hacker interact with a real OS, allowing the attacker to perform quite anything with the goal of capturing the ***maximum*** amount of information. Any command or end-user application which is expected to be installed, it's available on the system (enriched with monitoring functions).

- <u>Role</u>
  - A. **Server:** systems or devices that expose services acting like servers that ***passively*** wait to be attacked.
  - B. **Client:** systems that ***actively*** poses as clients and interact with servers to examine whether an attack has occurred.

IoT Honeypot Taxonomy



Low-Interaction IoT Honeypots (1)

- **Nepenthes:** it's a low interaction honeypot that presents the scalability of low- interaction honeypots but at the same time offers a high degree of expressiveness based upon a flexible modular architecture. It's designed to emulate vulnerabilities by vulnerability modules offering common services like FTP, HTTP, TFTP and MSSQL that are typically exploited by botnets during an attack and to detect and capture the related worms.
- **Dionaea:** successor of Nepenthes whose main purpose is gathering worm samples for Microsoft Windows systems. It implements a full Windows Remote Procedure Call (RPC) stack used by various Windows network services. Additionally, system can simulate other protocols, such us MongoDB, MSSQL, MySQL, SIP, SMB, FTP, and HTTP/HTTPS (suitable to emulate IoT devices). Dionaea doesn't emulate single network vulnerabilities, but the whole underlying network protocol stack.
- **Glastopf:** it's an easy to deploy low-interaction honeypot web server that collects information about web application-based attacks such as local and remote file inclusion or SQL injections. Furthermore, Glastopf downloads files from links that are included in incoming requests. An important aspect of Glastopf is that it's developed by implementing a vulnerability type instead of vulnerability emulation. Once a vulnerability type is emulated, unknown attacks of the same type can be handled.
- **Honeyd:** it's one of the best-known low interaction honeypots concerning the virtualization of hosts in Unix systems. HoneyD is a small daemon that allows to simulate thousands of virtual hosts at the same time on a single system. The hosts can be configured to run arbitrary services which mimic the basic behavior of the service on specified open ports and their fingerprint can be adapted so that they appear to be running certain operating systems (also IoT architectures).
- **HoneyBOT:** it's a low-interaction honeypot for Windows which opens a range of roughly 1.000 UDP and TCP listening sockets emulating vulnerable services. When attackers connect to these services, the honeypot safely captures all communications and logs results for future analysis. If the attacker uploads a malware to the server, then the honeypot will safely store the files on the system for analysis and submission to antivirus vendors.

Low-Interaction IoT Honeypots (2)

- **Honeytrap:** it's a low-interaction honeypot daemon which distinguishes itself from other honeypots by implementing a dynamic server concept: in contrast to other honeypots, which often focus on malware collection, honeytrap aims for catching the initial exploit, collecting and further processing attack traces. This concept targets completely unknown attacks which might occur on random ports and unknown protocols (no predefined emulation is required).
- **Bluepot:** it's designed to capture attacks on bluetooth devices by accepting and storing any malware sent to it and to interact with common bluetooth attacks such as Blue Bugging (remotely taking over control of the target device) and Blue Snarfing (remotely reading files from another device).
- **Shockpot:** it's designed around a single, critical vulnerability called ***Shellshock*** (CVE- 2014-6271) for the bash shell. ShellShock is a virus which can exploit a vulnerability in one of the operating computer's ports and then enters unauthorized commands. Then, the infection will auto-propagate to other networks by creating viral botnets. Once built, botnets can be used to drive huge amount of traffic to a specific network, making a DDoS attack.
- **MTPot:** it's a low interaction honeypot specifically designed for the detection of the Mirai botnet. It's a simple and very specific tool built to match what the Mirai malware expects, directly based on its source code. Using MTPot it's possible to collect verified mirai Indicators of Compromise (IoCs), in particular IP addresses and malware samples which are observed during the infection exploited in order to build the botnet that will be used for the DDoS attack.
- **Telnet-IoT-Honeypot:** it's a low interaction honeypot developed to catch IoT botnet binaries after the Mirai source code release. The project implements a Python Telnet server trying to act as an honeypot for IoT Malware which simulate the very insecure default passwords of telnet services of IoT devices on the internet. It's fast and easy to use since it does not provide a full simulated behavior of an Embedded Linux system; it's designed with the goal of simply collect statistics about IoT botnets and their patterns.

Low-Interaction Client IoT Honeypots

- **HoneyC:** it's developed to introduce a low-interaction client honeypot that emulates only the essential features of high interaction client honeypots, applying signature matching to allow fast static analysis of server responses.
- **Monkey-Spider:** it's a crawler based low-interaction client honeypot utilizing anti-virus solutions to detect malware. Its main idea is to first crawl the contents of a website and then analyze the crawled content for maliciousness to expose threats to Web clients. It is claimed to be fast and expandable with respect to other detection mechanisms.
- **YALIH (Yet Another Low Interaction Honeyclient):** it's a client honeypot designed to detect malicious websites by integrating a combination of multiple antivirus engines and pattern matching using string or regular expressions for detection. It aims to provide both a high rate of detection and a very low scanning time using multiple analysis engine and low resources. It deliberately does not address the problem of 0-day attacks. It tries to avoid the high false negative detection rate of other client honeypots that increases significantly with obfuscated attacks that exploit obfuscation techniques that scripting languages (as JavaScripts) provide.

Medium-Interaction IoT Honeypots

- **Kippo:** it's a medium interaction honeypot designed mainly to be used to log the SSH brute force attacks and, most importantly, the entire shell interaction performed by the attacker against a target system. It provides a fake file system, emulating a Debian Linux server with the ability to add and remove files. The honeypot allows the attacker to attempt a login to the system, believing it is entering into a legitimate SSH session with the real server. Upon successful guessing of the password (deliberately trivial), the attacker is then moved into a fake system in which the attacker can navigate around, interact with (no real damage can be done) and use some integrated basic tools (such as cat, wget, etc.). All interactions with the shell are monitored and recorded with session logs stored in an UML compatible format, so that they can be replayed with the original timings of the prompt. By analyzing the logs, it's possible to understand for what purpose the system was hacked, which data the attacker was trying to get from it and what techniques were used. The samples dowloaded during the attack are stored in a specific directory.

- **Cowrie:** Kippo successor, it's itself a low interaction honeypot that emulates the SSH service, with an additional introduction of the emulation for the Telnet protocol. Cowrie inherited all the main properties of Kippo, and it has also introduced the support for SCP and SFTP protocols for the file upload; so, also malicious samples uploaded using these services are placed in the same download folder. Other additional functionalities are introduced by Cowrie over Kippo such as the support for SSH exec commands, TCP/IP tunneling, SSH proxying, SMTP connections, JSON Logging, additional linux commands (sleep, sync, uname -r, dir, halt, users, killall, etc.) and ELK stack integration.

- **IoTPOT:** it's medium interaction honeypot designed to emulate Telnet services of IoT devices running on different CPU architectures to analyze the threats constituted by ongoing attacks. It consists of a front-end low-interaction responder cooperating with backend high-interaction virtual environments (called IoTBOX). IoTBOX operates various emulated systems commonly used by embedded systems for different CPU architectures (MIPS, MIPSEL, PPC, SPARC, ARM, MIPS64, SH4 and x86); moreover, it is a malware analysis environment for the samples obtained during attacks against IoT devices.

High-Interaction IoT Honeypots

- **HoneyPHARM:** is an high interaction honeypot not directly designed for the world of IoT; however it presents an innovative idea which could be a starting point for new IoT honeypots: since almost any sample captured by honeypots is malicious by definition (since they do not run any real services), then the more honeypots are present in different networks, the higher would be the chances to capture new malwares. HoneyPHARM is a distributed client/server system of Nepenthes nodes which gives the ability to collect malware from the distributed network in a central repository. Findings can be analyzed and actions can be taken accordingly in order to identify new threats and vulnerabilities.

- **SIPHON:** it's a prototype of scalable and reliable high interaction honeypot which goal is to learn about existing and novel attack vectors (both human and automatic) targeting IoT devices. It has a very innovative architecture: real IoT devices (instead of emulated ones) are exposed on entry-points (wormholes) in a wide range of geographic locations, exposing one or more services on open ports (e.g., HTTP, Telnet, SSH, etc.) configured with weak/defaults passwords. Once that devices are attacked, the wormholes collect (.pcap by Tcpdump) and send the data to the Storage & Analysis Unit (SAU) which perform off-line analysis (network gateway). The exposed devices are ***real*** and it could be possible for the attacker to fully compromise a device during an attack. The target of SIPHON is to understand how many TCP connections are received per each wormhole, what kind of services are consulted, whether interactions are performed using known scripts or new ones, which is the number of different ports accessed by the same wormhole, as well as the user agent used in HTTP sessions. Moreover, since to have access to the admin interface of IoT devices the attacker needs to authenticate, the honeypot also counts the number of brute-force login attempts (successful or not), to further investigate what happens after a successful login performed not only by automatic botnets but mainly from human attackers.

IoT Honeypots Summary

Type	Honeypot	Year	Services	Feature
LOW	**Nepenthes**	2005-2009	FTP, HTTP, TFTP, MSSQL	Capture worm payload, Flexibility
	Dionaea	2009-2013	SMB, FTP, SIP, MYSQL, Telnet, HTTP/HTTPS	Capture worm payload, Nepenthes successor
	Glastopf	2011-2015	HTML, PHP, SQL	Web applications, Vulnerability types
	HoneyD	2003-2008	HTTP, POP3, SMTP, FTP	Heterogeneous Networks Emulation
	HoneyBOT	2002-2007	FTP, SSH, Telnet, SMTP, POP3, HTML, HTTP	Windows vulnerabilities and GUI
	Honeytrap	2005-2015	HTML, FTP, Dynamic Server Emulation	Attacks via TCP/UDP, Dynamic emulation
	Bluepot	2010-2015	Bluetooth	Bluetooth Malware
	Shockpot	2014-2015	Apache, Bash	Shellshock Vulnerability
	HoneyC	2004-2007	HTTP Client	Identify malicious servers by Snort IDS signatures
	Monkey-Spider	2007-2009	HTTP Client, JavaScript, Crawling	Threat DB creation, Anti Virus support
	YALIH	2014-2015	HTTP Client, JavaScript, IMAP	Advanced Combination of Analysis Methods
	MTPot	2016-2017	Telnet, Mirai Fingerprinting	Mirai Detection
	Telnet IoT Honeypot	2016-2017	Telnet	Telnet Emulation, Malware Collection
MEDIUM	**Kippo**	2009-2014	SSH, XML, MySQL, XMPP	Emulate shell interaction, Brute-Force on SSH
	Cowrie	2014-2017	Same of Kippo + Telnet, SFTP, SCP	Kippo's successor (Bug fixes, Better Logging)
	IoTPOT	2014-2015	Telnet	Low-Intereraction Front-End + High-Interation Virtual Environments
HIGH	**HoneyPHARM**	2016-2017	HoneyNet Implemenation	Network of Distributed Nepenthes honeypots
	SIPHON	2016-2017	Real Devices over a Distributed and Scalable Architecture	0-day Identification, Distributed HoneyNet, Attack Detection (human/automated)

References

- Brian Krebs Website: https://krebsonsecurity.com/
- OWASP Top 10 IoT Vulnerabilities: https://www.owasp.org/index.php/Top_IoT_Vulnerabilities
- CERT (Computer Emergency Response Team) Nazionale Italia: https://www.certnazionale.it
- Paloalto Networks Research Center: https://researchcenter.paloaltonetworks.com
- NJ Cybersecurity & Communications Integration Cell: https://www.cyber.nj.gov
- Malware Must Die Blog: http://blog.malwaremustdie.org
- Softpedia - Security News: http://news.softpedia.com/cat/Security/
- TrendLabs Security Intelligence Blog: http://blog.trendmicro.com/trendlabs-security-intelligence/
- E. Peter and T. Schiller.
  A Practical Guide to Honeypots.
- L.Spitzner.
  Honeypots: Tracking Hackers.
- N. Provos, T. Holz.
  Virtual Honeypots: From Botnet Tracking to Intrusion Detection.
- K. Cabaj, P. Gawkowski.
  HoneyPot systems in practice.
- N. Weiler.
  Honeypots for Distributed Denial of Service Attacks.
- D. Akkaya and F. Thalgott.
  Honeypots in Network Security.
- S. M. Specht, R. B. Lee.
  Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures.
- K. M. Prasad, A. R. M. Reddy, K. V. Rao.
  DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms - A Survey.
- S. R. Oh, Y. G. Kim.
  Smart World of Internet of Things (IoT) and it's Security Concerns.
- S. Farraposo, L. Gallon, P. Owezarski.
  Network Security and DoS Attacks.
- T. J. Charity, H. J. Hua.
  Security Requirements Analysis for the IoT.
- M. Nawir, A. Amir, N. Yaakob, O. B. Lynn.
  Internet of Things (IoT): Taxonomy of Security Attacks.
- F. A. Alabaa, M. Othmana, I. A. T. Hashema, F. Alotaibi.
  Internet of Things security: A survey.
- P. Baecher, M. Koetter, T. Holz, M. Dornseif and F. Freiling.
  The Nepenthes Platform: An Efficient Approach to Collect Malware.
- L. Rist.
  Glastopf: a dynamic, low-interaction web application honeypot.
- N. Provos.
  Honeyd: A Virtual Honeypot Daemon.
- A. L. Podhradsky, C. Casey and P. Ceretti.
  The Bluetooth Honeypot Project.
- C. Seifert, I. Welch and P. Komisarczuk.
  HoneyC - The Low-Interaction Client Honeypot.
- A. Ikinci, T. Holz and F. Freiling.
  Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients.
- M. Mansoori, I. Welch and Qiang Fu.
  YALIH, Yet Another Low Interaction Honeyclient.
- K. Koltys.
  An in-depth look at Kippo: an integration perspective.
- Y.M. Pa Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama and C. Rossow.
  - IoTPOT: Analysing the Rise of IoT Compromises.
- A. Hassan and M. A. Ali.
  Collecting malware from ditributed honrypots: HoneyPHARM.
- J. Guarnizo, A. Tambe, S.Bhunia, M. Ochoa, N. Tippenhauer, A. Shabtai and Y. Elovici.
  SIPHON: Towards Scalable High-Interaction Physical Honeypots.

Thank You!

botticelli.1212666@studenti.uniroma.it

https://www.linkedin.com/in/biagio-botticelli-444b87105/