Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

School of Electronic Engineering

An Evaluation of Distributed Denial of Service (DDoS) Attacks in IoT Networks

Project Portfolio

Michael Lenehan
ID Number: 15410402

August 2020

MEng in Electronic and Computer Engineering

Supervised by Liam Meany

Meng in Electronic & Computer Engineering

Michael Lenehan

***Abstract*—The abstract goes here.**

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

THIS demo file is intended to serve as a “starter file” for IEEE journal papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

August 31, 2020

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Conclusion

The conclusion goes here.

Liam Meany, Department of Electronic Engineering, Dublin City University, Dublin, Ireland e-mail: liam.meany@dcu.ie.

Dr. Martin Collier, Department of Electroonic Engineering, Dublin City University, Dublin, Ireland e-mail: martin.collier@dcu.ie.

Appendix A:
An Evaluation of Distributed Denial of Service (DDoS) Attacks in IoT Networks - Literature Review

Michael Lenehan

Abstract—**With the increase in the number of Internet of Things (IoT) connected devices in recent years, and the projected growth in the number of connected devices in the coming years, it is evident that there is a need to investigate the security flaws which have made these devices a target for an increasing number of Distributed Denial of Service (DDoS) attacks. Following the Mirai botnet attack in 2016, it has become apparent that there is much work required in securing IoT networks. Due to the relatively low available resources in IoT devices, there is a need to implement either non-resource intensive security features on the devices themselves, or to implement more stringent security for these devices at the level of the network.**

Index Terms—**Internet of Things, Distributed Denial of Service, Security.**

I. Introduction

INTERNET of Things devices have seen a large increase in usage in the past number of years. With estimates of the number of devices exceeding 40 billion[1], and the amount of data produced by these devices in the order of Zettabytes[1] (10^{21}), it is apparent that the recent issues regarding Distributed Denial of Service (DDoS) attacks may be concerning to those utilising the networks.

The motivation of this project is to evaluate the susceptibility of IoT operating systems and devices to these types of attacks. The goal is to develop or recommend a solution for the detection of an attack, and the mitigation of such an attack.

A. Distributed Denial of Service Attacks

Distributed Denial of Service attacks aim to shut down their targets servers through a high volume flood of traffic. This high volume traffic, which can be in the order of Terrabytes of data, serves to overload the available resources of the hosts network, which causes regular user traffic to be rerouted. This re-routing leads to the "denial of service" aspect of the attack.

In order to generate such large volumes of data, an attacker uses multiple devices for transmission of malicious data packets. The "distributed" aspect of the DDoS attack comes from the multiple devices which are infected with malware, placed on the devices by the attacker. These devices are known as bots, and are remotely accessed by an attacker to generate high traffic to drain the networks resources.

B. Internet of Things DDoS Attacks

A DDoS attack relies on the attacker gaining remote access to devices in order to initiate a flood of traffic. Due to their relative abundance, Internet of Things devices have become a target for these types of attacks in recent years. IoT devices can act as an entry point to the network for an attacker due to the "always-on" nature of the devices and their connection to the network, and their lack of security features.

Attacks, such as the Mirai attack, rely on the default security credentials, which tend to remain unchanged when configured by end users, to gain access to the devices for the uploading of malicious software. Once this malware is uploaded to the device, the attacker can begin to flood the desired network with traffic[2].

Since the Mirai attack in 2016, and the subsequent release of the Mirai source code, there has been an increase in the number of IoT related DDoS attacks[3]. As such, it is clear that there is a need to secure these devices, and the networks in which these devices are utilised. As IoT devices become more prevalent in different areas and industries, it becomes increasingly important to not only protect the data being generated, but also to protect the greater network as a whole which is processing this data.

C. IoT Areas

Internet of Things devices have become common place in everyday life, with devices such as internet connected security cameras, smart-home devices, and wearable smart devices being common in households. However, there are many other use cases outside of the home where IoT devices have been adopted.

Industrial Internet of Things (IIoT) includes devices such as cameras and sensors within industrial settings, which, as with IoT, provides the ability to offload intensive processing to higher performance servers[4]. This division of IoT devices is integral for the "Industry 4.0" concept. Devices within this sector can be in control of, for example, safety systems, and as such, their continuous operation is of vital importance.

Internet of Medical Things (IoMT) devices include health-care specific connected devices, such as medical sensors and wearable devices for patient monitoring purposes[5]. These devices allow for an increase in patient comfort, and are an integral component of the "Healthcare v4.0" concept. With an increase in the number of attacks on similar systems, there is the concern that the implementation of such systems could be hindered if more robust security systems are put in place.

II. Review and Analysis of Prior Work

A. Distributed Denial of Service Attacks and its Defenses in IoT: A Survey

Salim et al.[3] have presented a thorough survey of a number of different DDoS attacks which are common in IoT networks. Their work provides classifications for these attacks, along with providing a number of detection, prevention and mitigation solutions.

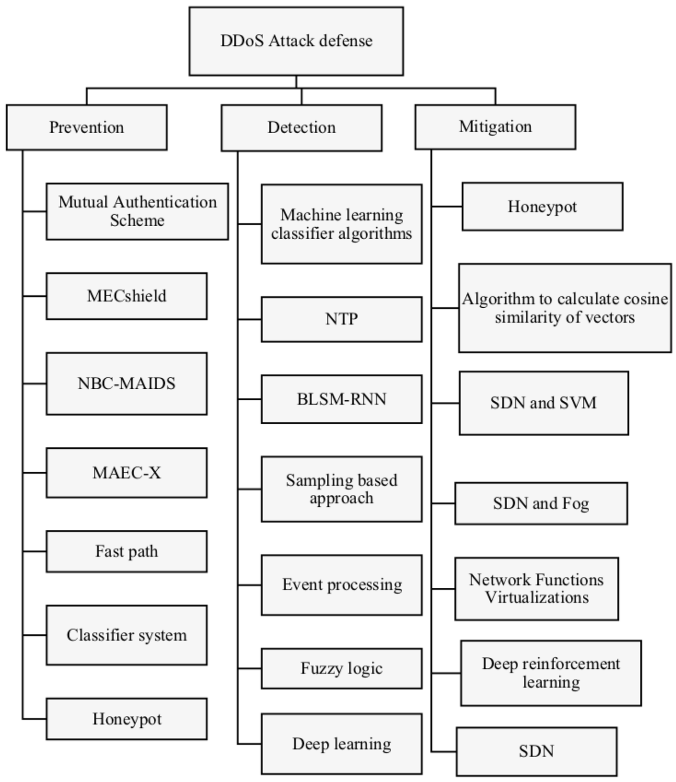

Fig. 1: Salim et al. DDoS attack Defence techniques[3]

As seen in Figure 1, there are 21 proposed solutions; 7 Prevention, 7 Detection, and 7 Mitigation. These solutions act at the device level, edge computing level, and cloud level.

Prevention of attacks deals in stopping traffic flow between nodes flagged for suspicious or "abnormal" traffic. These techniques involve monitoring the traffic between the nodes, and either stopping data flow (Mutual Authentication Scheme), highlighting suspicious traffic to other nodes in the network (MECshield, NBC-MAIDS, Multi-access Edge Computing, Fast Path, Classifier System), or diverting attack traffic (Honeypot).

Detection of attacks deals in correctly detecting that the traffic flow is due to an attack. These techniques utilise machine learning (Machine Learning Classifier Algorithms, BLSM-RNN, Fuzzy Logic, Deep Learning) and event processing (CEPID, Sampling-based Approach) in order to detect attack traffic, in some cases, with very high accuracy. The Machine Learning Classifier Algorithms technique had a recorded detection accuracy in real time IoT devices of 0.99.

Mitigation of attacks deals in stopping a detected DDoS attack. This attack mitigation is implemented by blocking dataflow from malicious nodes within the network (SDN and SVM, ECESID, Deep Reinforcement Learning, and SDN). Honeypot mitigation techniques were shown experimentally to increase attack detection efficiency by up to 60%.

The survey of Salim et al.[3] presents excellent distinctions between types of DDoS attacks, and some of the solutions which are currently being presented. While this survey is useful as an overview of, and introduction to these techniques and solutions, it does not present any new solutions apart from general security information such as "Change passwords", and "Implement a firewall".

B. A Multi-Level DDoS Mitigation Framework for the Industrial Internet of Things

With the increasing number of IoT devices being used within industry, and the growing importance of IIoT within the "Industry 4.0" concept, there is an increasing need to secure the data and data systems being used. Within industrial settings, the data being handled by these IoT systems could be vital in terms of safety systems, or mission-critical systems within production. As such, Yan et al. have presented a Multi-level DDoS mitigation framework (MLDMF) which utilises software defined networking (SDN) for the purposes of managing and securing IoT devices[4].

Yan et al. acknowledge that the defence systems available to IoT devices are "unsubstantial". The MLDMF proposed consists of three levels, the edge, fog, and cloud computing levels. Each of these levels serves its own purpose in securing the IIoT network. The edge computing level provides gateways with higher computational power than the IoT devices used in the network, which can implement more "traditional" security features, such as IP address concealment, and malicious software detection.

The fog computing level analyses data from the edge computing level, and, using a number of techniques, can make decisions on how to handle the data. The three described techniques are collect-detect-mitigate (CDM), honey-pot-detect-react (HDR), and cloud-detect-fog-mitigate (CDFM). CDM allows for modifications to the networks bandwidth based on properties of the traffic. HDR can redirect traffic to a virtual IoT device in order to perform analysis. This technique allows for stopping malware from affecting the network, and allows for modifications to policies based of any new information gained from captured traffic. CDFM shares information between the cloud and fog computing levels in order to inform decision on dropping packets which may be from malicious sources.

The cloud computing level implements machine learning techniques in order to analyse the large amounts of data flowing through the network. This information can be used for the detection and prediction of DDoS attacks, based on the incoming data. Experimentally, the implementation of the proposed MLMDF improves performance within a network undergoing a TCP SYN flood attack by approximately 37.03%.

The use of SDN's for management of IoT devices and the traffic within the network have the disadvantage of adding overhead to the network. There is also the added cost of hardware for the implementation of the proposed system, requiring honeypot servers, and gateways to act as controllers. While the proposed solution may be viable, and these costs may be minimal in the context of an industrial setting, the current scope of the project is more focused on the evaluation of security improvements within the IoT network.

C. Securing Internet of Medical Things Systems: Limitations, Issues and Recommendations

The medical applications of IoT devices have become apparent in recent years. However, with the sensitive nature of the data being generated, privacy and security are of major concern. Yaacoub et al. present an in depth analysis of the dangers of IoMT devices, a number of commonly faced attacks, and a discussion on the possible solutions to these issues[5].

The three relevant recommendations made within this study are the implementation of lightweight cryptographic algorithms for security, alongside a lightweight authentication protocol, and a layered IoT secutrity architecture. The lightweight aspect is key in this assessment, as IoT devices do not have the resources to implement more robust solutions. One suggested security algorithm is the dynamic structure cryptographic algorithm, which is discussed further in Section II-D.

The layered security architecture, much like the MLDMF, separates security responsibilities between different levels. Unlike the MLDMF however, these layers define Quality of Service (QoS) parameters. Due to the nature of the application, IoMT devices must give accurate, real-time feedback. As such, this "zero-tolerance" for error must be represented within the QoS demands of the system.

The proposed use of dynamic structure cryptographic algorithms, and lightweight authentication protocols would greatly improve security within IoT networks. A low latency, non-resource intensive authentication scheme would allow for transmission of data between trusted devices, while allowing real-time processing of data. While focused mainly on IoMT, the insight provided by this study can be translated to IoT in general.

D. One Round Cipher Algorithm for multimedia IoT devices

There is a clear need for a lightweight algorithm for the encoding and decoding of information being transmit between devices in an IoT network. Noura et al. present such an algorithm in the context of Multimedia IoT (MIoT) devices. The proposed solution uses pseudo-random keys for encryption, greatly increasing the difficulty of decoding intercepted transmissions[6].

A number of constraints were considered in the development of this algorithm, including the limited resources available to IoT devices, and the need for an algorithm which can be implemented across devices of varying performance. This leads to a lightweight algorithm which may be used across a number of different devices, regardless of the resources available to that device.

In order to test the robustness of this algorithm, a number of tests were performed, separated under headings which include "Statistical analysis", "Visual degradation", and "Resistance against well-known types of attacks". With regards to the statistical analysis, the algorithm was shown experimentally to have very low correlation between adjacent pixels of the encoded image, with output values of close to zero. With regards to visual degradation, there was a signal-to-noise ratio (SNR) of 8.5894dB, which is regarded within the study to be a "low value", showing that the algorithm has acceptable robustness to degradation during encoding and decoding. Through a number of attack tests, the algorithm was shown to be robust to statistical attacks, and the size of keys used attributed to robustness to brute force attacks.

In terms of performance, when tested on two common embedded Linux platforms, the Raspberry Pi Zero W, and the Raspberry Pi 2, this algorithm gave encryption time gains of up to 29%, and decryption time gains of up to 33% when compared with a similar implementation.

One suggestion made in this study is that the algorithm has been implemented in C, whereas the implementation compared to with regards to performance is implemented in assembly. As such, further performance benefits could be gained from this lower level implementation.

This algorithm proposal demonstrates that a lightweight, "flexible" cryptographic algorithm can be implemented in a robust and highly performing manner. While the proposed algorithm is intended to be used in the context of multimedia transmissions, and is tested in the transmission of an image, it is clear that the uses of the algorithm could be extended beyond this scope.

III. Relation to the Project Problem

The focus of this project will be to evaluate IoT systems in terms of the devices, networks, and operating systems for their susceptibility to DDoS attacks. As such, the survey of Salim et al.[3] provides an excellent background in the types of attacks, and is a starting point for assessing the types of solutions which are available to deal with these attacks. By understanding the types of attacks which commonly occur, more accurate evaluations can be performed between IoT systems.

IIoT is becoming a large source of data, and accounts for a large number of IoT devices. The proposal laid out by Yan et al.[4] gives an insight into how security issues within industry are being approached. This type of system can be investigated as an IoT network as a whole. Simulation of this type of network is a possible solution for the evaluation of a network consisting of SDN gateways and multiple controllers.

The suggestions laid out in the work of Yaacoub et al.[5] can clearly be applied to IoT networks and devices of all types, and are not specific to the field of IoMT. The implementation of a lightweight authentication protocol for IoT networks is an area which could be investigated further within the scope of the project.

The dynamic structure cryptographic algorithm proposed by Noura et al.[6] shows promising results in terms of relative performance and robustness. An evaluation of these algorithms, or potentially an implementation should be further investigated, and are completely within the scope of the project. The proposed algorithm appears to solely be implemented for the purpose of transmitting images, however the extension of this algorithm to other data types could prove useful for security.

It should be noted that these solutions can all potentially be implemented in various combinations. An in depth evaluation should consider any permutation of the aforementioned solutions. For the purposes of testing and evaluating these solutions, either simulation or virtualization will be used. While simulation will require less available computing resources, it is possible that virtualization of the network, as described in [4] will provide a more complete insight to the effects of changes to network level, or operating system/device level parameters.

IV. Conclusion

It is clear from the completion of this literature review that there are a number of proposed solutions for the current issues of security with regards to IoT devices and DDoS attacks. As the scope of the project is to evaluate IoT systems at the operating system (OS), network, and device levels for their susceptibility to attack, the solutions discussed in this review can be used as a baseline for improving upon.

Further work must be done in order to determine the open source tools which will be used for the evaluation of these systems. As this literature review deals solely in the solutions to the security issues in these networks, i.e. the mitigation and detection techniques used, an assessment must follow into the available tools for the purposes of evaluation of these networks.

References

[1] *The growth in connected iot devices is expected to generate 79.4zb of data in 2025, according to a new idc forecast*, Jun. 2019. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS45213219.

[2] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017, ISSN: 1558-0814. DOI: 10.1109/MC.2017.201.

[3] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in iot: A survey," *The Journal of Supercomputing*, Jul. 2019, ISSN: 1573-0484. DOI: 10.1007/s11227-019-02945-z. [Online]. Available: https://doi.org/10.1007/s11227-019-02945-z.

[4] Q. Yan, W. Huang, X. Luo, Q. Gong, and F. R. Yu, *A multi-level ddos mitigation framework for the industrial internet of things*. [Online]. Available: https://ieeexplore.ieee.org/document/8291111/.

[5] J.-P. A. Yaacoub, M. Noura, H. N. Noura, O. Salman, E. Yaacoub, R. Couturier, and A. Chehab, *Securing internet of medical things systems: Limitations, issues and recommendations*, Dec. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19305680.

[6] H. Noura, A. Chehab, L. Sleem, M. Noura, C. Raphaël, and M. M. Mansour, *One round cipher algorithm for multimedia iot devices*. [Online]. Available: https://link-springer-com.dcu.idm.oclc.org/content/pdf/10.1007/s11042-018-5660-y.%20pdf.

Appendix B:

Project Design Plan

1 Research Question

This project aims to critically and technically evaluate the susceptibility of IoT networks to DDoS attacks, focusing on the devices, networks, and operating systems associated with IoT.

2 Project Scope

The following list describes the topics and technologies which will be utilized in the completion of this project.

- Susceptibility to Attack
  - Networks
    - ns-3 Network Simulations
  - Operating Systems
    - OS Level Security Feature Implementations
    - Known
- Mitigation/Detection

3 Design Approach

The proposed approach to this project is to present a detailed technical evaluation of the security features available at the operating systems level. These will include features such as networking security - i.e. firewalls, and software security - i.e. secure boot.

Simulations will be run using ns-3 in order to evaluate the effects of common DDoS attacks on networks consisting of IoT devices. These simulations will utilize the available IoT related protocols in ns-3, including IPv6 and 6LoWPAN. The results of these simulations will be evaluated to determine points of failure within the networks.

Finally, a technical evaluation will be done into the available detection and mitigation techniques. This will include recommendations of techniques which could have specific benefits in IoT networks.

4 Timeline

The Gannt chart below describes the proposed timeline for the project. All items within the Gannt chart are colour coded as follows; green represents documentation - i.e. work on the final report, red represents background research - i.e. background research required for understanding the topic, blue represents testing - i.e. simulation testing, orange represents end of week reviews, during which time the goals of the past week and following week will be assessed.

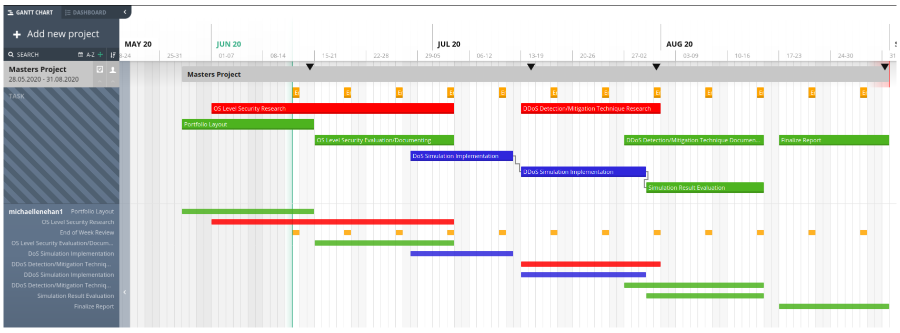

Figure 1: Project Gannt Chart

This project plan can be viewed at the following link: `https://app.agantty.com/#/sharing/d595b7c64d314f5788834d0ed8614ccc/de`

Not included in this design plan are working hours, which, for the duration of the project, will be 8 a.m. to 5 p.m Monday to Friday. All project work must be completed outside of these times.

5 Success Criteria

This project will be considered successful if the following criteria are met:

1. The completion of a literature review
2. The completion of a project project presentation
3. The completion of a Masters project research log
4. The completion of a technical evaluation of OS level security implementations
5. The completion of an investigation into known security vulnerabilities of IoT operating systems
6. The completion of a number of DDoS simulations on simulated IoT networks

7. The completion of a technical evaluation of DDoS detection and mitigation techniques
8. The completion of a list of recommendations of detection/mitigation techniques specifically applicable to IoT networks.

6 Remote Arrangements

It has been confirmed with the project supervisor, Liam Meany, that this project can be successfully completed remotely, i.e. with no access to on-campus resources.

Appendix C:

Research Log

Masters Project Research Log

Masters in Electronic and Computer Engineering 2019/2020

Student Name: Michael Lenehan
Student ID: 15410402
Project Title: An Evaluation of Distributed Denial of Service Attacks in IoT Networks

Please read before making entries in this log

The purpose of this Project Research Log is to capture concise, focused summaries of research materials you read, as you progress through your project. The emphasis is to record (i) how the material you have read will determine or influence your project solution approach and (ii) your assessment of the key strengths and weaknesses of the solutions, methods, technologies, etc. proposed in the material you have read.

In the first stage of your project, the literature review, use the Log to capture this information for the key papers you have read (for example, the three most important papers of your 10 literature review references). As your project progresses into the design and implementation phases, you will need to continue to search the literature so you can review, revise and refine your initial thinking and the details of your approach to a project solution. Use this Research Log to capture your continued research reading and its influence on your project design and implementation.

Be selective about what you record in this log. Do not use it as an informal notebook while you are reading a new paper. Only make an entry after you have read a paper that you consider important to the development of your project solution. It is expected that, by the end of the project, you will have made **between 10 and 20 entries (20 maximum)**.

Your log will be shared with your supervisor for viewing throughout the project and you will submit the final version of the log for grading, at the end of the project implementation period. It will be assessed on the basis of how well you have used your analysis of the literature to inform your project design, implementation and the evaluation of your project results. The Research Log contributes **5%** to the overall project mark.

Note: All entries you make in this log must use the prescribed format. A new entry with the correct format is generated using the "Research Log" menu above (it may take a minute for this menu to appear). Do not make any other entries or change the format of the generated tables. You will maintain other notes as you progress through your project but they should not be recorded here. Do not exceed the maximum word counts indicated.

Statement of project problem / research question (maximum 200 words) *This statement should be periodically reviewed and updated, as necessary, as your project progresses and you gain further insight into the detailed project challenges, requirements and objectives as your project work moves from background reading, literature review, initial project design planning and detailed design and implementation. Initially, start by stating your current understanding of the project objectives. After each meeting with your supervisor, review and refine your project problem statement, as required.*

IoT Device security through dynamic hardware isolation with cloud-Based update
Hategekimana, F., Whitaker, T., Hossain Pantho, M. and Bobda, C., 2020. IoT Device security through dynamic hardware isolation with cloud-Based update. *Journal of Systems Architecture*, 109, p.101827.
https://doi.org/10.1016/j.sysarc.2020.101827
Summary of paper (maximum 100 words)
This paper proposes a security device which can be connected between an IoT device and a network which monitors the traffic between the device and the network. In monitoring the traffic, security breaches can be detected, with reports sent to a centralised server, updating the security policies of other such security devices.
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
This paper examines a method of improving security on IoT devices. As DDoS attacks which make use of IoT botnets take advantage of the relatively poor security implementations on IoT devices, improving the security of these devices could greatly reduce the likelihood of these attacks.
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

Distributed denial of service attacks and its defenses of IoT: a survey
Salim, M., Rathore, S. and Park, J., 2019. Distributed denial of service attacks and its defenses in IoT: a survey. *The Journal of Supercomputing*, 76(7), pp.5320-5363.
https://doi.org/10.1007/s11227-019-02945-z
Summary of paper (maximum 100 words)
Acts as a survey of DDoS attack motivations and methods. Classifies attacks in terms of Iot devices and cloud layer. Presents defense and mitigation strategies. Paper gives a

comprehensive overview of attack tools, various attack methods, defense and mitigation strategies.
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
This paper acts as a good starting point for the project. It presents an excellent baseline of knowledge in terms of DDoS attack types and the tools used by attackers.
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

Dyn DDoS Cyberattack - a case study
Sreekanth, A., Sri, P. and Vartiainen, T., 2020. Dyn DDOS Cyberattack - a case study. [online] Available at: <https://mycourses.aalto.fi/login/index.php>.
https://mycourses.aalto.fi/login/index.php
Summary of paper (maximum 100 words)
This paper presents a case study of the Dyn DDoS attack. This attack took place in 2016, and at the time was the biggest DDoS attack ever launched. This paper delves into the Mirai malware which was used to launch the attack, the timeline of the attack, the DNS exhaustion attack method used, the effects of the attack, and the challenges faced in defending against these types of attack.
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
The Dyn attack is infamous due to the scale of the botnet used to launch the attack. By presenting a case study of this attack, we get a better understanding of the real-world

impact of these large scale attacks. Having a real-world example to refer to, rather than a conceptual, or theoretical threat proves the need for further understanding of the causes of these attacks along with the security measures which can be taken to avoid these attacks.
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
This paper presents an in-depth case study of the Dyn attack, with focus on the attack and the challenges faced in mitigating the attack. However, as this is just a case study, there is not as much of a focus on the current state of mitigation techniques, or
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

Cyber and Physical Security Vulnerability Assessment for IoT-Based Smart Homes
Ali, B. and Awad, A., 2018. Cyber and Physical Security Vulnerability Assessment for IoT-Based Smart Homes. *Sensors*, [online] 18(3), p.817. Available at: <https://www.mdpi.com/1424-8220/18/3/817>.
https://www.mdpi.com/1424-8220/18/3/817
Summary of paper (maximum 100 words)
This paper proposes a security device which can be connected between an IoT device and a network which monitors the traffic between the device and the network. In monitoring the traffic, security breaches can be detected, with reports sent to a centralised server, updating the security policies of other such security devices.
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
This paper examines a method of improving security on IoT devices. As DDoS attacks which make use of IoT botnets take advantage of the relatively poor security implementations on IoT devices, improving the security of these devices could greatly reduce the likelihood of these attacks.
What are the strengths and weaknesses of the solutions/methods/technologies proposed

in this paper? (maximum 100 words)
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

Standaradizing IoT Network Security Policy Enforcement
Barrera, D., Molloy, I. and Huang, H., 2018. Standardizing IoT Network Security Policy Enforcement. *Workshop on Decentralized IoT Security and Standards (DISS) 2018*,.
https://dx.doi.org/10.14722/diss.2018.23007
Summary of paper (maximum 100 words)
This paper looks into the predictability of IoT security implementations, and proposes a secure architecture for protecting networks of IoT devices. This architecture is independent of device manufacturers and aims to protect devices from being compromised by attackers.
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
The “security policy enforcement architecture” aims to protect devices and mitigate the impact of attacks by providing a whitelist of expected network activity, such as metadata - packet size etc. -, content - protocol -, and application - HTTP requests. The paper also provides insight into the need for this type of architecture, looking at attacks and similar architectures.
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

Facing DDoS bandwidth flooding attacks
Furfaro, A., Pace, P. and Parise, A., 2020. Facing DDoS bandwidth flooding attacks. *Simulation Modelling Practice and Theory*, 98, p.101984.
https://doi.org/10.1016/j.simpat.2019.101984
Summary of paper (maximum 100 words)
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

A novel approach for detecting vulnerable IoT devices connected behind a home NAT
Meidan, Y., Sachidananda, V., Peng, H., Sagron, R., Elovici, Y. and Shabtai, A., 2020. A novel approach for detecting vulnerable IoT devices connected behind a home NAT. *Computers & Security*, 97, p.101968.
https://doi.org/10.1016/j.cose.2020.101968
Summary of paper (maximum 100 words)
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)

What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

A Simulation Analysis of Flooding Attack in MANET using NS-3
Bandyopadhyay, A., Vuppala, S. and Choudhury, P., 2011. A Simulation Analysis of Flooding Attack in MANET using NS-3.
https://doi.org/10.1016/j.sysarc.2020.101827
Summary of paper (maximum 100 words)
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

Software Defined Network Defense
Sathyanarayana, S., 2012. Software Defined Network Defense. [online] Available at: <https://www.academia.edu/1833986/Software_defined_network_defense>.

https://www.academia.edu/1833986/Software_defined_network_defense
Summary of paper (maximum 100 words)
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

Progressive researches on IoT security: An exhaustive analysis from the perspective of protocols, vulnerabilities, and preemptive architectonics
Mahbub, M., 2020. Progressive researches on IoT security: An exhaustive analysis from the perspective of protocols, vulnerabilities, and preemptive architectonics. *Journal of Network and Computer Applications*, 168, p.102761.
https://doi.org/10.1016/j.jcna.2020.102761
Summary of paper (maximum 100 words)
This paper presents a survey of the application areas of IoT devices, the architectures these devices use for communications, and a comprehensive review of the types of attacks which these devices are susceptible to. The presented information covers the area of IoT security from the level of infrastructure and protocols, to attack and defense techniques.
How is this paper relevant to solving your project problem or addressing your research question? (maximum 100 words)

The first number of sections of this paper offers a lot of background information about the standards and networking protocols used by IoT devices and the areas in which these devices are used. Section 6, “Layers in IoT architecture”, breaks these networks down, comparing them to the equivalent OSI model layers. The last sections, 7 through 9, offer a lot of relevant information about the threats facing network connected devices.
What are the strengths and weaknesses of the solutions/methods/technologies proposed in this paper? (maximum 100 words)
This paper presents an in-depth look at the vulnerabilities of IoT devices to attack, with information about the network protocols, and common attack types used on these devices. The strengths of this paper lie in the breadth of the information presented. The weaknesses of this paper, for the purpose of addressing my research question, are in the lack of practical demonstration of attacks, or of mitigation techniques.
Log Entry Creation Date: Thu Feb 20 2020 11:05:08 GMT-0000 (GMT)

Appendix D:

IoT Vulnerabilities

1 Introduction

There are an estimated 31 billion Internet of Things devices currently in use. As these devices are being adopted in all aspects of life - from home devices, medical devices, industrial devices - there are valid concerns raised about the security capabilities of these devices. The requirement for improvements to the resilience of these devices to attack is clear following a number of large scale botnet attacks, including the Mirai attack.

The Mirai attack was able to generate a massive 1Tbps of data, and targeted the Dyn DNS service provider. It is estimated that over 600,000 devices were a part of the Mirai botnet. The Mirai code exploits

2 IoT Device Resources

Internet of Things devices are typically low-power devices and sensors. These devices are network connected, and often interact with a users personal information. As an affect of the low power resources used on these devices, the devices critical functions take precedence in terms of memory or processing performance. This leaves little availability in terms of available resources for aspects such as security.

3 OS Security Features

Non-resource constrained, "fully fledged" operating systems which run on consumer PCs and enterprise server hardware have a number of security features which can be implemented to protect against attack. These range from firewall and packet filtering software, to firmware verification software.

3.1 Secure Boot

Secure boot is a security mechanism used to verify that software being run on a system is from a trusted source. The vendors whose software has been granted permission to run on the system are stored in firmware[1]. This allows for any software being loaded onto the device at boot time can be verified by comparing the vendor signature against the keys which are stored in firmware.

3.1.1 UEFI Secure Boot

UEFI found the need to implement the secure boot mechanism as protections for post-boot code injection had become competent enough that attackers were beginning to focus their efforts on pre-boot firmware injection. Pre-boot malware injection includes rootkit and bootkits, which, before secure boot, proved extremely difficult to detect.

There are a number of intended results for pre-boot exploits, ranging from control over the operating system, or user identification credential snooping, to control over ROM and PCIe peripherals.

Vendor signatures are stored in databases in the devices firmware. The two types of keys used for accessing these databases are Platform Keys, and Key Exchange Keys[2]. Platform keys are the vendor defined key, which prevents modifications to the Key Exchange Key. The Key Exchange Key prevents modifications to the signature database. This mechanism allows vendors with valid KEK's to modify the signature database, such as inserting or deleting a signature. By comparing each code module's signature to the database at boot time, a decision can be made on whether to load this module and proceed with the boot process, or not to load the module and terminate the boot process.

The device manufacturer controls the signatures which are included from the factory. Vendors such as Microsoft offer programs for vendors to submit modules for authorization, to have their signatures added to the signatures database. This system has been mirrored by a number of Linux distributions.

Secure boot is compatible with the x64 platform, and is enabled by default on Windows and most Linux distributions[3]. By enabling this mechanism by default, the user must disable the feature to open themselves to vulnerability.

3.1.2 Secure Boot in IoT Systems

While IoT devices typically have a one-time setup, they can be rebooted multiple times over there lifetime, especially in the case of smart home IoT devices, as they are relocated or power cycled. By initiating their boot process, these devices become vulnerable to the injection of malicious code modules. There is a recognition for the need of a secure boot type mechanism for Internet of Things devices, with implementations such as MCUboot in development for a number of microcontroller based operating systems such as RIOT and Zephyr[4]. There are, however, a number of limitations which have to be considered, namely the lack of available storage on these constrained devices. As a workaround for this constraint the boot code can be written to non-modifiable ROM, although this could raise issues in the devices future, as patches required to the devices firmware to mitigate other vulnerabilities could not be implemented[5].

3.2 Device Level Firewalls

Firewalls define rules used for the purpose of allowing or blocking traffic entering a system.

3.2.1 Linux Iptables and Netfilter

Iptables is a software package for filtering traffic at the Network Layer (layer 3 of the OSI model). This package interacts with the "netfilter" kernel hooks, which are five hooks that can be triggered at different stages of packet processing. They work by applying a set of rules to incoming or outgoing packets. These rules can be pre-applied, i.e. defined by the manufacturer, or user-defined.

Iptables are broken down into a number of layers. The first of these is the table. The table is used to separate rules by function. The default tables are the "filter" table, the "NAT" table, the "mangle" table, the "raw" table, and the "security" table. Tables contain a number of chains, which are associated with the five netfilter hooks, and are triggered when a packet invokes a rule.

The built in iptable chains are triggered by each of the five netfilter hooks as follows:

- NF_IP_PRE_ROUTING -> PREROUTING
- NF_IP_LOCAL_IN -> INPUT
- NF_IP_FORWARD -> FORWARD
- NF_IP_LOCAL_OUT -> OUTPUT
- NF_IP_POST_ROUTING -> POSTROUTING

4 IoT Vulnerabilities

Neshenko et al.[6] define 9 classes of IoT vulnerabilities. These are deficient physical security, insufficient energy harvesting, inadequate authentication, improper encryption, unnecessary open ports, insufficient access control, improper patch management capabilities, weak programming practices, and insufficient audit mechanisms. While a number of these are not relevant to the susceptibility of these devices to malicious attack for their future use in a large scale botnet, for example physical security - an attack is not going to physically connect their PC to thousands of IoT devices to build their botnet - issues such as improper and encryption authentication methods, open ports, and an inability to update flaws in firmware certainly leave these devices vulnerable to attack on a large scale.

4.0.1 Common Vulnerabilities and Exposures

CVE, or Common Vulnerabilities and Exposures, is a publicly accessible list of cybersecurity vulnerabilities[7]. This list is comprised of reported vulnerabilities from user and engineers, with details of the vulnerability and level of exposure to the system due to the vulnerability. Each CVE entry contains a CVE ID, details of the vulnerability, and Common Vulnerability Scoring System (CVSS) score. The CVSS score is calculated using a number of metrics, including at its base level the attack vector and complexity, the requirement for user interaction, and the impact of the vulnerability on a users confidentiality or the availability of the service[8]. The score defines a severity rating from Low to High for the listed vulnerability. With the CVSS 3.0 rating system, scores from 0.1 to 3.9 are low severity, 4.0 to 6.9 is medium severity, 7.0 to 8.9 are high severity, and 9.0 to 10 are critical severity.

Listed in Table 1 are the currently listed CVE reports for the IoT operating systems RIOT, Contiki, Amazon FreeRTOS and Zephyr OS.

Table 1: IoT Operating System CVE Logs

Operating System	CVE ID	CVE Score
RIOT-OS	CVE-2019-16754	5.0
	CVE-2019-15702	5.0
	CVE-2019-15134	7.8
Contiki-OS	CVE-2017-7296	4.3
	CVE-2017-7295	7.8
Amazon FreeRTOS	CVE-2018-16603	4.3
	CVE-2018-16602	4.3
	CVE-2018-16601	6.8
	CVE-2018-16600	4.3
	CVE-2018-16599	4.3
	CVE-2018-16598	4.3
	CVE-2018-16527	4.3
	CVE-2018-16526	6.8
	CVE-2018-16525	6.8
	CVE-2018-16524	4.3
	CVE-2018-16523	5.8
ZephyrOS	CVE-2018-1000800	7.5
	CVE-2017-14202	4.6
	CVE-2017-14201	4.6
	CVE-2017-14199	7.5

RIOT OS Vulnerabilities

The listed RIOT OS vulnerabilities range from medium to high severity. The two medium severity vulnerabilities are protocol vulnerabilities. The first of these is an issue with the OS's implementation of the MQTT messaging protocol[9]. Due to the implementation, null pointer dereferencing results in a segmentation fault, creashing the system[10]. The second of these protocol vulnerabilities is a TCP parsing error[11], whereby the system can be places in an infinite loop while attempting to parse a packet with an unknown option specified, and an option length of zero[12].

The highest severity vulnerability listed for RIOT OS is another protocol vulnerability. This is cited as exposing the devices running this operating system to Denial of Service attacks. An error with the implementation of the TCP handshake, in which a memory leak occurs when an ACK is received by the device as the first packet in the handshake instead of a SYN packet[13]. This memory leak can lead to an exhaustion of memory resources on the device.

Contiki-OS

Both the first and second listed CVE reports for Contiki-OS refer to the same Cross-Site Scripting vulnerability in the mqtt.html configuration page. The first of these reports is a medium severity vulnerability[14], while the second is listed as a high severity vulnerability[15]. This difference in severity is due to the level of detail given in the CVE reports. This vulnerability in the operating systems included MQTT configuration implementation causes a null pointer dereference error, which can be initiated via a HTTP POST request, and can cause the host device to crash, causing a denial of service[16].

Amazon FreeRTOS

There are 11 listed CVE reports for Amazons FreeRTOS[17], all of which are based on vulnerabilities in the operating systems TCP/IP implementation. The higher scored medium severity reports are vulnerable to DoS attack and remote code execution. Report CVE-2018-16526[18] outlines how a buffer overflow occuring during the generation of a checksum can give an attacker access to execute code on the device. The vulnerability described in report CVE-2018-16601[19] can lead to a denial of service. All other listed vulnerabilities are either due to buffer overflows or TCP packet parsing error.

ZephyrOS

The listed vulnerabilities for Zephyr OS range from medium to high severity[20]. The first of these allows an attacker connected via telnet to cause a system crash, and therefore a denial of service attack. The crash could be caused by executing the "history" command, causing a buffer overflow**zepCVe1**. The second medium severity report is a duplicate of this issue.

The first of the high severity reports[21] details how an incorrect implementation of type checking within the sys_ring_bug_get() and sys_ring_buf_put() kernel API calls caused the kernel to spin, which again could be used to initiate a denial of service[22].

The final report is of high severity[23], and details how a buffer overflow can be initiated due to the kernel getaddrinfo() call, which is used during DNS resolution[24].

5 Detection and Mitigation Tecniques

5.1 Detection

5.2 Mitigation

6 Conclusion

References

[1] *Secure boot*. [Online]. Available: https://wiki.ubuntu.com/UEFI/SecureBoot.

[2] R. Wilkins and B. Richardson, *UEFI Secure Boot in Modern Computer Security Solutions*, [Online]. Available: https://www.uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf.

[3] C. Hoffman, *How secure boot works on windows 8 and 10, and what it means for linux*, Jul. 2017. [Online]. Available: https://www.howtogeek.com/116569/htg-explains-how-windows-8s-secure-boot-feature-works-what-it-means-for-linux/.

[4] JuulLabs-OSS, *Juullabs-oss/mcuboot*. [Online]. Available: https://github.com/JuulLabs-OSS/mcuboot.

[5] I. S. Foundation, *Device secure boot*. [Online]. Available: https://www.iotsecurityfoundation.org/best-practice-guide-articles/device-secure-boot/.

[6] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019. DOI: 10.1109/comst.2019.2910750.

[7] *Common vulnerabilities and exposures (cve)*. [Online]. Available: https://cve.mitre.org/index.html.

[8] *Common vulnerability scoring system calculator*. [Online]. Available: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator.

[9] *Vulnerability details : Cve-2019-16754*. [Online]. Available: https://www.cvedetails.com/cve/CVE-2019-16754/.

[10] Riot-Os, *Asymcute: Fix null pointer dereference by nmeum · pull request #12293 · riot-os/riot*. [Online]. Available: `https://github.com/RIOT-OS/RIOT/pull/12293`.

[11] *Vulnerability details : Cve-2019-15702*. [Online]. Available: `https://www.cvedetails.com/cve/CVE-2019-15702/`.

[12] ——, $Gnrc_tcp : Optionparsingdoesn'tterminateonallinputs, potentialdosissue\#12086riot - os/riot$. [Online]. Available: `https://github.com/RIOT-OS/RIOT/issues/12086`.

[13] *Vulnerability details : Cve-2019-15134*. [Online]. Available: `https://www.cvedetails.com/cve/CVE-2019-15134/`.

[14] *Vulnerability details : Cve-2017-7296*. [Online]. Available: `https://www.cvedetails.com/cve/CVE-2017-7296/`.

[15] *Vulnerability details : Cve-2017-7295*. [Online]. Available: `https://www.cvedetails.com/cve/CVE-2017-7295/`.

[16] 262588213843476, *Contiki mqtt and xss*. [Online]. Available: `https://gist.github.com/jackmcbride/c9328627f1ee104ce84f3fb7eff42f1e`.

[17] *Amazon " freertos : Security vulnerabilities*. [Online]. Available: `https://www.cvedetails.com/vulnerability-list/vendor_id-12126/product_id-51624/Amazon-Freertos.html`.

[18] *Vulnerability details : Cve-2018-16526*. [Online]. Available: `https://www.cvedetails.com/cve/CVE-2018-16526/`.

[19] *Vulnerability details : Cve-2018-16601*. [Online]. Available: `https://www.cvedetails.com/cve/CVE-2018-16601/`.

[20] *Zephyrproject " zephyr : Security vulnerabilities*. [Online]. Available: `https://www.cvedetails.com/vulnerability-list/vendor_id-19255/product_id-50119/Zephyrproject-Zephyr.html`.

[21] *Vulnerability details : Cve-2018-1000800*. [Online]. Available: `https://www.cvedetails.com/cve/CVE-2018-1000800/`.

[22] Zephyrproject-Rtos, *Get fault when fuzzing* $sys_r ing_b uf_p utandsys_r ing_b ug_g etapisissue\#7638zephyrproject-rtos/z$ [Online]. Available: `https://github.com/zephyrproject-rtos/zephyr/issues/7638`.

[23] *Vulnerability details : Cve-2017-14199*. [Online]. Available: `https://www.cvedetails.com/cve/CVE-2017-14199/`.

[24] ——, *Net: Sockets: Getaddrinfo() buffer overflow, etc. fixes by pfalcon · pull request 6158 · zephyrproject-rtos/zephyr*. [Online]. Available: `https://github.com/zephyrproject-rtos/zephyr/pull/6158`.

Appendix E:

Simulation - Testing and Results

1 Introduction

The aim of a Distributed Denial of Service attack is to disrupt the connection between a service, such as an application server or DNS server, and that services users.

Denial of Service (DoS) attacks typically exploited vulnerabilities with network or application layer protocols, for example, Syn Floods or HTTP Floods. With these types of attacks, spoofed IP addresses were typically used, both to mask the IP address of the attacker, and to take advantages of vulnerabilities of the protocols being used.

As DoS mitigation techniques improved, and attacks using IP spoofing became easier to defend against, attackers began utilising botnets to perform amplification attacks. Distributed Denial of Service (DDoS) attacks utilise large number of hosts to perform attacks. These types of attacks prove more difficult to defend against than typical Denial of Service attacks.

In order to extensively evaluate the impact of Distributed Denial of Service attacks, and the role which IoT devices play in modern DDoS attacks, simulations can be used. Simulations can show the impact to a network, and the disruption caused by these attacks, without having to use real network devices.

1.1 Simulation Software

There were a number of simulation software packages considered for the tests contained in this document. Each of these packages offer their own set of advantages and disadvantages.

1.1.1 ns-3

The most commonly used network simulation software is "ns-3"; a "discrete-event network simulator for Internet systems"[1]. This package is commonly used for research purposes, and numerous examples of D/DoS simulations implemented using ns-3 can be found.

1.1.2 Cooja

Another simulation software package considered was "Cooja". This package simulates Wireless Sensor Networks built on the Contiki IoT operating system[2]. While the underlying system being simulated is an IoT device, there was little information to be found on implementing the types of testing required, and as such this software package was ruled out.

1.1.3 Mininet

The final software package considered was "Mininet". Mininet is a network simulator which uses Linux containers to emulate network devices[3]. As the network devices are emulated using containers, each host uses real Linux Kernel code.

As the Mininet hosts run Linux Kernel code, Linux applications can be easily run. This is advantageous as common attack tools can be installed and executed. Each host can be assigned a unique IP address allowing the attacking host to address it's target as in a real attack situation. As each host is implemented as a container, a terminal can be open for each host for manual command execution.

Mininet implements Software Defined Networking using the OpenFlow protocol. By default, Mininet uses the OpenFlow reference controller, however, it also allows for the use of remote controllers. The "Floodlight" controller will be used for this purpose, as it has a GUI application which can display the network topology.

Mininet virtualizes the network links between each host or switch in the network. These links can be assigned parameters such as a delay time on the link, or a set bandwidth.

The Python API implemented by Mininet allows for the topology and the terminal commands to be created and executed programmatically. The Python API will be used in order to execute repeatable tests with reproducible results.

Due to the advantages offered by the Mininet simulation software package, this will be the simulator of choice for testing. There are however a number of disadvantages which must be noted[4].

1. Containers share the file system of the host, i.e. the desktop or VM on which Mininet is being run
2. A network cannot exceed the bandwidth of a single server
3. Non-Linux-compatible OpenFlow switches and applications are not supported

While these limitations must be kept in mind, they will not prove to be a hindrance for the purposes of these simulations.

2 Attack Types

For testing purposes, a number of different attack types were be simulated. SYN Flood attacks and ICMP Flood attacks were be simulated.

2.1 SYN Flood

SYN floods are an attack which exploit the TCP protocols TCP handshake. The attacking node sends TCP SYN packets to the target using spoofed IP addresses. The target device will reply with a SYN-ACK, and wait for the initiator to reply with a final ACK packet. As the attacking node is using a spoofed IP address, it never responds to the SYN-ACK, and so the target node will wait indefinitely for the ACK packet[5].

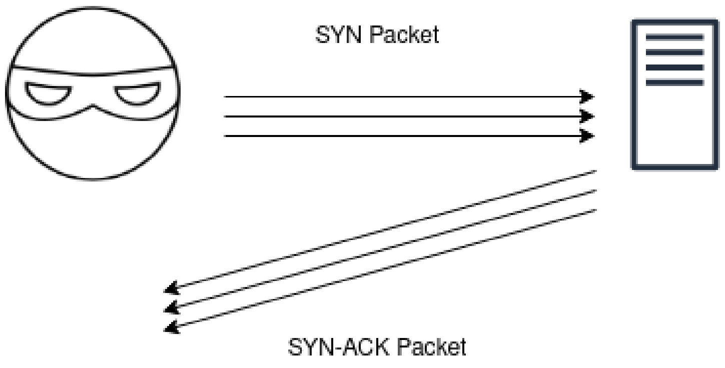

Figure 1: SYN Flood Attack

In order to perform a SYN Flood attack, the "hping" tool is used. Hping is a tool for creating and transmitting TCP, UDP or ICMP packets. This tool has multiple functions, such as port scanning, and firewall mapping, but can also be utilised as a Denial of Service tool. By utilising the tools "rand-source" and "flood" flags, the tool can execute a SYN flood, sending TCP SYN packets to the target address via spoofed IP addresses.

2.2 ICMP Flood

An ICMP flood is a distributed denial of service attack, performed using a botnet[6]. ICMP requests take server resources to process and reply to. By sending large numbers of ICMP requests, the servers resources can be exhausted.

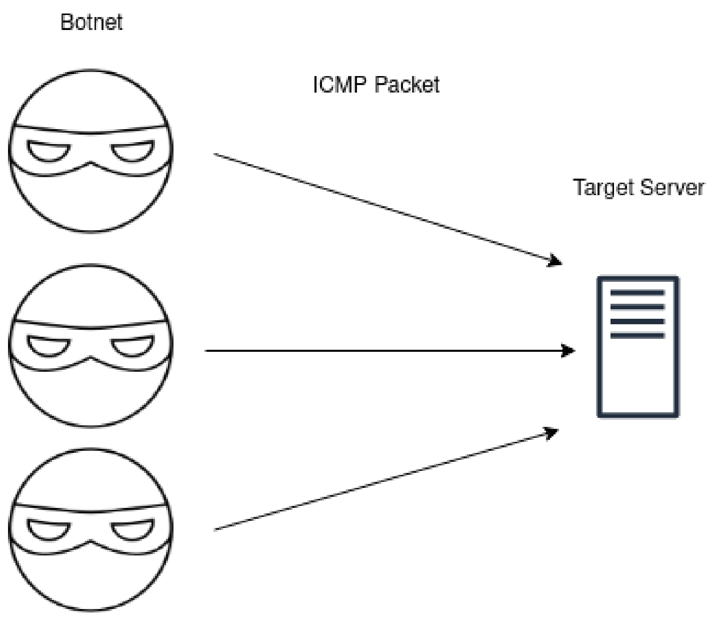

Figure 2: ICMP Flood Attack

The "ping" tool can be used to execute an ICMP flood. By using the "-f" flag, the ping tool can be used to launch an ICMP flood from a single attacking node.

3 Denial of Service Simulation

For the initial Denial of Service simulation, the network topology is as shown in Figure 3. This network consists of an attacking traffic source (shown on the left), a legitimate user traffic source (shown on the right), and a server (shown on the bottom).

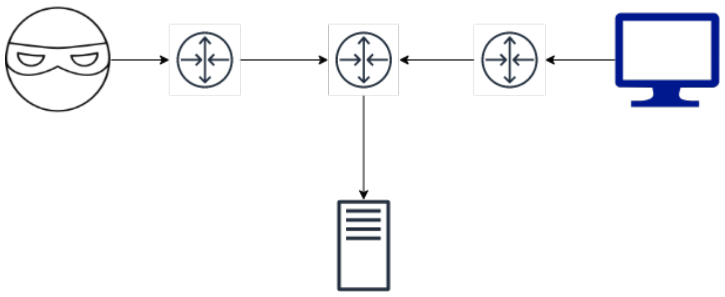

Figure 3: Denial of Service Network Topology

This topology is implemented using the Mininet Python API. Each of the traffic sources is implemented as a "host", and each switch as a "switch". The links between each node in the network must be defined. The completed topology configuration is as follows:

```
class MyTopo( Topo ):
        "Dos Topology"

        def __init__( self ):
                "Create custom topo"

                # Initialize topology
                Topo.__init__(self)

                # Add hosts and switches
                h1 = self.addHost( 'h1' )
                h2 = self.addHost( 'h2' )
                h3 = self.addHost( 'h3' )
                s1 = self.addSwitch( 's1' )
                s2 = self.addSwitch( 's2' )
                s3 = self.addSwitch( 's3' )

                # Add links
```

```
                    self.addLink( h1, s1 )
                    self.addLink( s1, s2 )
                    self.addLink( s2, h3 )
                    self.addLink( s2, s3 )
                    self.addLink( s3, h2 )
```

Listing 1: DoS Simulation Network Topology

3.1 SYN Flood

3.1.1 Attack

For the SYN Flood, the malicious traffic source launches the attack using the following "hping" command:

```
$ hping -S -p <TargetPort> --rand-source --flood <
    TargetIP>
```

Listing 2: SYN Flood DoS Command

This command can be broken down as follows:

- The -S flag denotes setting the SYN flag for the transmission.
- The -p flag sets the destination port number for the transmit packet, in this case the target server.
- The –rand-source flag sets the IP address to spoofed values for each packet that is transmit.
- The "–flood" flag sends packets as fast as is possible.

3.1.2 Mitigation

3.2 ICMP Flood DoS Attack

3.2.1 Attack

For the ICMP Flood, the malicious traffic source launches the attack using the following "ping" command:

```
$ sudo ping -f -l 100000000000000 -s 1500
```

Listing 3: ICMP Flood DoS Command

This command can be broken down as follows:

- The -f flag denotes a flood, as such the tool outputs "ECHO_REQUEST" packets as fast as is possible.
- The -l flag denotes the number of packets to send without waiting for a reply. The value associated with this flag is set to an arbitrarily high number.
- The -s flag denotes the packet size, the value for which can be determined by observing the output of the "ifconfig eth0". For the DoS network, this value was set to 1500 bytes.

3.2.2 Mitigation

4 Distributed Denial of Service Simulation

For the Distributed Denial of Service simulation, an expanded version of the topology used in Figure 3, consisting of a significantly larger number of attacking sources is used. This reflects the increase in attackers, which is common in DDoS attacks involving IoT botnets. It was chosen to use 8 switch nodes, each connected to 8 host nodes, and one switch and host connected as the target server to these networks. The complete topology can be seen in Figure 4.

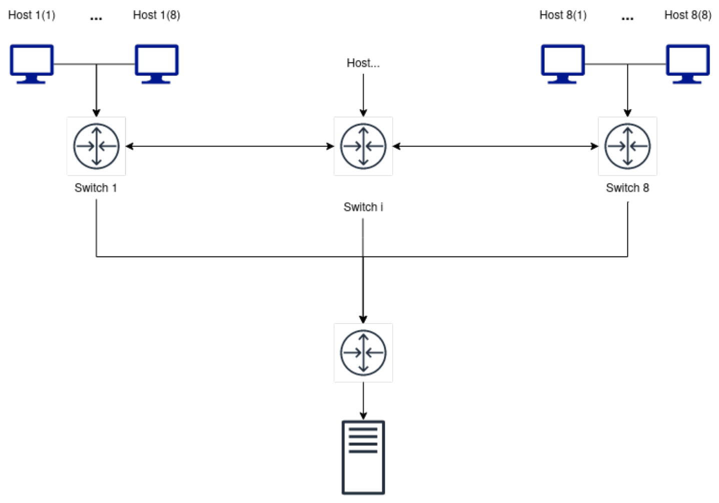

Figure 4: Distributed Denial of Service Network Topology

Within this topology, any number of hosts can be used to launch the distributed denial of service attack. The topology is again implemented using the Mininet Python API as follows:

```
class LineTopo( Topo ):
    "Linear topology example."

    def __init__( self ):
        "Create linear topology"
        super(LineTopo, self).__init__()
        h1 = []
        h2 = []
        h3 = []
        h4 = []
        h5 = []
        h6 = []
        h7 = []
        h8 = []
        s = []          # list of switches
        server = []

        M=9
        N=8
        # add N hosts  h1..hN
        for i in range(1,N+1):
            h1.append(self.addHost('h1' + str(i)))
            h2.append(self.addHost('h2' + str(i)))
            h3.append(self.addHost('h3' + str(i)))
            h4.append(self.addHost('h4' + str(i)))
            h5.append(self.addHost('h5' + str(i)))
            h6.append(self.addHost('h6' + str(i)))
            h7.append(self.addHost('h7' + str(i)))
            h8.append(self.addHost('h8' + str(i)))

        # add N switches s1..sN
        for i in range(1,M+1):
            s.append(self.addSwitch('s' + str(i)))

        # add target server
        server.append(self.addHost('server'))

        # Add links from hi to si
        for i in range(N):
            self.addLink(h1[i], s[0])
            self.addLink(h2[i], s[1])
            self.addLink(h3[i], s[2])
            self.addLink(h4[i], s[3])
            self.addLink(h5[i], s[4])
            self.addLink(h6[i], s[5])
            self.addLink(h7[i], s[6])
            self.addLink(h8[i], s[7])

        # Add links from target server to s8
        self.addLink(server[0], s[8])

        # link switches
        for i in range(M-2):
            self.addLink(s[i],s[i+1])

        # link all switches to target switch
        for i in range(M-2):
             self.addLink(s[i], s[M-1])


topos = { 'mytopo': (lambda: LineTopo() ) }
```

Listing 4: DDoS Simulation Network Topology

5 Results

6 Resources

The following list outlines both the hardware and software resources which were utilised in the execution of these simulations:

6.1 Hardware:

As these simulations utilised Linux containers, only a single PC was required to execute the simulations. The specifications of the laptop PC are as follows:

- HP Envy 17
  - Intel Core i7-6500U (Dual Core)
  - 12GB RAM

6.2 Software:

For these simulations, the Linux operating system was used for it's ease of installation of software packages, and also for the availability of containers. Note that a Linux virtual machine would be a suitable replacement.

6.2.1 Operating System:

- Manjaro Linux
  - Kernel Version 5.4.52-1-MANJARO
  - OS Type: 64-bit

6.2.2 Simulation Software:

- Mininet
  - Version: 2.3.0d6
- Floodlight Controller
  - Version: 1.2

7 Reproduction

The following steps must be performed in order to reproduce the achieved simulations. As the Manjaro Linux distribution was used for these simulations, all package manager instructions will be given for Manjaro.

7.1 Mininet

Mininet must first be installed. This can be done according to the Mininet documentation **??**. For Debian based systems, Mininet can be installed via the "apt" package manager, using the command:

```
$ apt-get install mininet
```

Listing 5: Debian-based Distro Mininet Install

For Arch based distributions, Mininet may be installed using the Arch User Repository using the following command:

```
$ yay -S mininet-git
```

Listing 6: Arch-based Distro Mininet Install

To test the installation, execute the command:

```
$ sudo mn --test pingall
```

Listing 7: Mininet installation test

If the error "ovs-vsctl: unix:/run/openvswitch/db.sock database connection failed" occurs, the Open vSwitch must be started using the command:

```
$ sudo /usr/share/openvswitch/scripts/ovs-ctl start
```

Listing 8: Open vSwitch service start command

7.2 Floodlight OpenFlow Controller

The Floodlight OpenFlow Controller can be installed via GitHub**??**. For version 1.2 (as used for these simulation), the Java 8 development kit must be installed:

```
$ sudo pacman -S openjdk8-src
```

Listing 9: openjdk8 installation

Floodlight has a number of dependencies which can be installed via the following command:

```
$ sudo pacman -S git ant maven python-dev
```

Listing 10: Floodlight Dependencies

To download and build Floodlight, execute the following commands:

```
$ git clone git://github.com/floodlight/floodlight.git
$ cd floodlight
$ git submodule init
$ git submodule update
$ ant
# If the ant build fails, Floodlight can be built using
    the following Maven command
$ sudo mvn package
```

Listing 11: Floodlight installation commands

7.3 Source Code

The source code for the simulations can be found on GitHub, and can be downloaded using the following command:

```
$ git clone https://github.com/mLenehanDCU/MininetCode.
    git
```

Listing 12: Source code download

8 Future Work

For work built upon these simulations, the first recommendation is to use either the Mininet VM image, or a Linux distribution such as Ubuntu. While Manjaro is based on Arch Linux, which has access to a large number of packages, Mininet, along with a number of the packages used for sending and monitoring traffic flow were required to be installed from the Arch User Repositories (AUR). For comparison, Mininet, is available from Ubuntus default "apt" package manager, along with all of the other software packages used.

9 Conclusion

References

[1] Nsnam, 20AD. [Online]. Available: `https://www.nsnam.org/`.

[2] B. Thébaudeau, *An introduction to cooja*, Sep. 2019. [Online]. Available: `https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja`.

[3] 2018. [Online]. Available: `http://mininet.org/`.

[4] *Mininet overview*, 2018. [Online]. Available: `http://mininet.org/overview/`.

[5] 2020. [Online]. Available: `https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/`.

[6] 2020. [Online]. Available: `https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/`.

Appendix F:

Detection & Mitigation Techniques

1 Introduction

There are an estimated 31 billion devices currently in use. As these devices are being adopted in all aspects of life - from home devices, medical devices, industrial devices - there are valid concerns raised about the security capabilities of these devices.

2 IoT Device Resources

Internet of Things devices are typically low-power devices and sensors. These devices are network connected, and often interact with a users personal information. As an affect of the low power resources used on these devices, the devices critical functions take precedence in terms of memory or processing performance. This leaves little availability in terms of available resources for aspects such as security.

3 Conclusion