Network Wide Enabled Packet Sniffer Using A Raspberry-Pi

Implementation Project

Matthieu Dubray & Emiliya Jones

University of Washington

CSS 537 – Network & Systems Security

Winter 2024

Abstract
The growth of the internet has led to an increase in quality and quantity of cyberthreats, and yet most people don't have adequate knowledge nor security measures in place in their home networks to reliably defend against a targeted attack. Our goal is to create a simple yet effective packet analysis tool, that anyone from no knowledge to someone with a technological background can deploy and use to monitor their network for any suspicious activity. The packet sniffer alone would likely not be effective in this as it requires the user to manually review each packet and be able to spot suspicious activity, and to that an IDS would be more effective in allowing a user to automatically see when suspicious activity was generated. To that, this project is a simple start to what would become a fully functional and portable network analysis solution of a packet analyzer, IDS, and IPS solution, offering users a good starting point into securing their networks.

Introduction
With the development and continuous growth of the world-wide web, or internet, emerged more than simply the ability and resources to transfer data and access systems and data, but introduced the concept of cybercrime and cybersecurity. With the developments of technology and expanded access worldwide, discipline was required in order to appropriately ensure data and information system security. Cybersecurity has become increasingly necessary in recent years as attackers have created and leveraged significantly advanced methods to attack networks, gain unauthorized access, and steal or modify data. The primary focus of this particular research project is to explore conceptually and ultimately build a packet sniffer as a way to analyze data coming and going between the internet and the LAN. Furthermore, the intention of this project is to leverage the development and results within a larger scoped intrusion detection and prevention system. Many existing organizations have created tools that are very effective in analyzing traffic and displaying that information to the user, alerting suspicious traffic, and potentially even preventing said traffic from crossing between the internet and LAN network. However, this project focuses on the primary goal of accessibility for users of varying knowledge and experience levels, including simple installation and instructions.

The packet sniffing tool that can be plugged into any physical connection to a network and can be leveraged to view inbound and outbound traffic. The tool is beneficial as it provides visibility and allows users to view all incoming and outgoing packets and the payloads being sent within the packets. With this knowledge, a user could potentially view any suspicious or malicious behaviour in their network that would cause harm to the user. As previously noted, the most distinguishing and key feature of this project is the attempt to ensure a user-friendly and user-centric focus from installation to analysis. The achievement of accessibility is evident through the minimal installation and configuration that a user must complete on their machine or network. This will allow users with little to no technical or networking knowledge to deploy this tool. Accessibility is also shown through the dashboard displayed within the application. As a final component of this tool, it is entirely open source, allowing users to modify the underlying code to fit their specific needs if needed.

The project group is composed of two members and contributors, Matthieu D., and Emiliya J. Each of these contributors has contributed to different parts of the project based on their skills

and knowledge, including exploring skills or knowledge that they may have not known prior to the course or project. Matthieu D. has worked on locating various architectures used in previous research of similar tools. He has formulated a working architecture to enable the tool to complete the necessary functions. Matthieu D. has also worked on the back-end portion of the code that manages packet sniffing and the APIs used to send data to the front end. Finally, Matthieu D. worked on testing the tool to ensure it could handle a specific workload level and looked into future implementations to see how the tool could potentially scale up for higher workloads. Emiliya J. has worked on looking at the ethical implications of this tool to see if it can be used maliciously, how an attacker might go about using this tool for a malicious purpose, and how to mitigate the ability of someone to do so. Emiliya J. has also worked on the front-end portion of the code that offers the user a simple dashboard to allow the user to control the packet sniffer and also receive the packets sniffed from the sniffer and display them correctly on the dashboard so the user can view relevant data about the traffic.

Both contributors worked on the final project report and provided equal contributions throughout the project. For all work related to the project, including source code, please refer to the GitHub link below.

GitHub – CSS 537

Related Work

Similar research previously completed has proposed varying perspectives and use cases, ranging from a single user's needs to a possible SMB, and potentially to a full enterprise-level solution. Previous research and projects were leveraged prior to the finalization of the tool to see if this would be feasible, to receive inspiration for the architecture necessary for development, and lastly, as a base for building.

The work completed by S. Tripathi, and R. Kumar offered excellent insight on using Raspberry Pi in various forms of network security. The authors proposed using a Raspberry Pi in three network security devices: a honey pot, an IDS, and a packet analyzer. While each of these solutions can be implemented individually, all three of these solutions can be implemented in tandem to increase the level of network security [1].

The first proposed solution, the IDS, uses three essential tools to perform packet inspection and alerts on any suspicious activity that triggers specific rules. The first tool, snort, critical in an IDS, is used to sniff and analyze data against a set of rules. These rules are set to alert administrators of any traffic that triggers it and offer an accompanying message, so the administrators know what was triggered and what to look for. The second tool, barnyard, is used to process alerts triggered for snort into a database format, allowing analysts to quickly query all of the alerts to see for specific or persistent threats. Barnyard uses the unified2 binary files that snort generates and uses input processors to read the data, which are then sent to output plugins to format the

data into a database. The final tool, pulledpork, is used as a rule management tool. This tool can be used to download and set snort rules chosen for the network's needs [1].

The second proposed solution is a honeypot, which is used to trick attackers into thinking they are accessing significant network resources but, in reality, are accessing a fake server that can capture data on the attacker. This is a simple solution that only implements one tool, Cowrie Honeypot. Cowrie is known as a medium interaction honeypot; the level of interaction is used to show how detailed a honeypot is based on what services or protocols might be enabled in the honeypot. For a medium interaction, we see the computer emulate a few services so that an attacker would think they are accessing an actual host on the network. Cowrie enables a few services, the important one being SSH, which will fool the attacker into thinking they are entering a natural host. Once the attacker SSHs into the honeypot, the honeypot will mimic a natural system and log all actions the attacker performs [1].

The third solution, the packet sniffer, aligns best with this project as it shows a prominent tool used in packet sniffing, Tshark. Tshark is a command line-based adoption of Wireshark, which offers extensive packet sniffing support, real-time packet sniffing, filtering for specific traffic, data formatting to be more readable by users, and the ability to store packet dumps for later analysis [1].

In the research proposed by Sumanth R. and Bhanu K. N., they proposed a way of reducing the analysis portion of an IDS by using a K-means clustering algorithm. During high intensity traffic, it is possible that an IDS would have an increased detection rate, some if not most of which could be false positives. Using the K-means clustering algorithm to interpret and group the data into similar detection triggers, it enables larger scale networks to effectively analyze all data and prepare it for administrators and SOC analysts to better interpret any suspicious activity [2].

Garand V. Y. and Rini W. W. offer a great IDS solution using snort and a popular sniffer protocol TaZmen. TaZmen is a protocol that wraps around wireless packets to add support for IDS, and wireless tracking, which is used to encapsulate other transmission protocols over UDP to allow the capture of packets. TaZmen was chosen because it adds support for port mirroring which can help to ensure that traffic sent to different hosts over the network can be captured by a singular device [3]. This is something that came up in our project as we needed to find a way to have a similar functionality to port mirroring or some other way to ensure packets destined for any host would also be routed to the sniffer. In the end we chose to use port mirroring as it was simple enough to set up, but would require the user have access to a console of the router to setup.

The work provided by Carl N. and Martin L. was used to create a lightweight portable IDS using a raspberry pi. Their solution is an amazing and terrifying example as to what a raspberry pi can do in terms of network reconnaissance. Their implementation when started will look for an AP and try to connect, if it could not connect to the current AP it would scan for another, until finally it was connected to a wifi, once connected the IDS would start capturing all packets that were inbound for the AP and the raspberry pi would receive a copy of the packets. The IDS portion would then search for traffic matching the rules set into place and begin alerting once found [4].

June J. offers a simple solution using a raspberry pi and the kali linux OS, with plenty of built in tools to create a IDS/honeypot that can be deployed onto any network. Using a tool known as Pentbox, a user is able to quickly setup a low-medium level interaction honeypot, which can be used to fool an attacker into executing commands which would then be reported back to the network administrators. June J. decided on this approach as it was a relatively cheap, easy to deploy, and autonomous solution that allows low knowledge people to effectively use this device to secure their network [5].

Kasperi T. uses a raspberry pi, along with python and the low level socket API provided to create a simple network monitoring tool. The decision to use the basic socket API rather than something like Scapy, was likely to have a more fine grained control over the packet sniffing functionality, however with the down side of missing out on plenty of quality of life solutions offered by something like scapy, which can end up taking more development time [6].

Milad G., Hussein J., Zahraa D., and Oussama T., display a solution that can be used for wifi monitoring rather than requiring a physical connection to the network. This solution can be helpful for case when a user would not have physical access to the network but would still like to be able to monitor for suspicious activity, such as on public networks. To be able to do this, the user would require a wifi enabled usb device to enable the pi to connect wirelessly, finally they use Tshark as a means of packet capturing and data formatting to enable the user to view packets [7].

Methodology

The implementation of this project leveraged several popular open-source code libraries to allow packet sniffing, an API to control and request information from the sniffer, and a front-end to allow a better user experience. We have created a docker container which can be set up to simplify the installation even further. The identifies components are subsequently implemented into a Raspberry Pi to allow the user to load the code, run it, and begin use.

The first portion of the implementation is the packet sniffing aspect. The packet sniffer is built upon Python and uses a popular packet-sniffing framework referred to as Scapy. Scapy offers a full packet sniffing implementation similar to Wireshark, such as real-time sniffing, filtering, and data formatting, which the users can choose to format even further. Please note that the identified portion only implements the previous mechanisms discussed.

Another component of this implementation is a web API. Similarly to the packet sniffer, this component is built upon Python and uses Flask, a simple and lightweight framework. Flask was chosen because it is easy to implement, does not use many resources, and offers an easy way to create APIs for the project objectives. For the APIs themselves, there are three different ones: one which allows the user to set a filter for specific traffic and then start the sniffer, one which allows the user to stop the sniffer, and finally, one that is used to retrieve information on the sniffed packets including the source and destination IP and ports and the payload of the packet.

The third portion of this implementation is the front end. The front end leverages a popular JavaScript framework, React, to offer users a seamless experience that allows users to use all the APIs. It also readily displays the data so the user can analyze the packets. In addition to React, we used Bootstrap to add simple styling to improve the UI of the dashboard and align with the objective of improving user experience and accessibility. We also created a docker container that can be used to quickly deploy this application within seconds and allow the user to plug and play with any device. Docker is a tool used to set up containers, which are individual virtual machines that contain pre-configured services, code, etc., which offers users a quick way to deploy some services.

Finally, we decided to use a Raspberry Pi as the physical hardware to run the project. However, a Raspberry Pi is **not** required. The tool can be run on any hardware with minimal resources and with an Ethernet port to connect to the network physically. However, a Raspberry Pi would be a good choice as it is inexpensive, small, has low power requirements, and is portable. The Raspberry Pi operating system also comes pre-loaded with Python as it is the core of our application code base, which offers less setup requirement for the end-user.

Once fully implemented, we tested how the Raspberry Pi was under different workload levels and tested how users interacted with the tool. For the first test, we created a simple tool using Scapy to generate ICMP traffic at varying levels to see how the Raspberry Pi performs and if it could handle the traffic. For the second test, we tested different users with varying levels of technical knowledge, ranging from none to those with a previous software engineering background or a computer science degree.

Results

This project resulted in a fully functioning application that can perform the basic functionality of a packet sniffer. Overall, the results aligned with the initial objectives. However, due to time constraints, the full functionality of a packet sniffer has yet to be implemented, and that was the overall goal. Although the full implementation of this project could not be reached within the time limit given, this project was meant to span multiple stages to reach the final goal of creating an open-source IDS that can be implemented to any network for minimal cost and resources based on the volume of traffic of a network.

We conducted a few stress tests to explore how successful the deployment and performance of the application would be. With a minimal user test size of ten people, each with varying levels of technical knowledge from using a computer for essential functions such as web browsing, document writing and education to users with computer science and technological backgrounds or degrees. We saw that seven out of ten participants could successfully deploy and boot up the device and application and continue to use the web dashboard to sniff and understand the sniffed packet information effectively. We see that the remaining three out of ten participants need help deploying the application and need to provide instructions on how to deploy the application on the device. Once the device was successfully deployed, the remaining users could easily navigate and interact with the dashboard. Please note the sample size of the test was minimal due to time constraints. For future research and expansion of the existing project, further testing must be completed.

The other test performed in the research was a performance stress test of the application to see how much traffic it could handle on its own. This test was conducted on a network capable of 1 gigabit per second transfer rates and can vary based on the network bandwidth of each user. We found that the Raspberry Pi and the application could handle at least a full gigabit per second worth of packets, which was generated using another script using Scapy to send traffic over the network. While we cannot test further due to personal network bandwidth constraints, within the technology spec of the Raspberry Pi 4, the Ethernet port used on the Pi has a bandwidth of 1 gigabit. It is possible to set up a load balancer and connect multiple devices running the application to scale up the bandwidth capacity of the packet sniffer.

Conclusion

This project has shown to provide a deeper understanding of networking and the underlying protocols that are used to communicate between systems across a LAN or across the internet. Users with little to no knowledge were able to quickly learn how to use the device and application, and being to learn about what packets are and what information packets contain. This device offers ease of use, accessibility, minimal power requirements, and minimal prerequisite materials.

References

[1] S. Tripathi and R. Kumar, "Raspberry Pi as an Intrusion Detection System, a Honeypot and a Packet Analyzer," in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, Dec. 2018, pp. 80–85. doi: 10.1109/CTEMS.2018.8769135.

[2] R. Sumanth and K. N. Bhanu, "Raspberry Pi Based Intrusion Detection System Using K-Means Clustering Algorithm," in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India: IEEE, Jul. 2020, pp. 221–229. doi: 10.1109/ICIRCA48905.2020.9183177.

[3] G. Vira Yudha and R. Wisnu Wardhani, "Design of a Snort-based IDS on the Raspberry Pi 3 Model B+ Applying TaZmen Sniffer Protocol and Log Alert Integrity Assurance with SHA-3," in *2021 9th International Conference on Information and Communication Technology (ICoICT)*, Yogyakarta, Indonesia: IEEE, Aug. 2021, pp. 556–561. doi: 10.1109/ICoICT52021.2021.9527511.

[4] C. Nykvist and M. Larsson, "– Implementation and evaluation of a lightweight portable in- trusion detection system using Raspberry Pi and Wi-Fi Pineapple".

[5] J. Jeremiah, "Intrusion Detection System to Enhance Network Security Using Raspberry PI Honeypot in Kali Linux," in *2019 International Conference on Cybersecurity (ICoCSec)*, Negeri Sembilan, Malaysia: IEEE, Sep. 2019, pp. 91–95. doi: 10.1109/ICoCSec47621.2019.8971117.

[6] K. Tuovinen, "Network monitoring with Raspberry Pi".

[7] M. Ghantous, H. Jaber, Z. Darwish, and O. Tahan, "PiMonitor: A Wi-Fi Monitoring Device Based on Raspberry-Pi," in *2018 International Conference on Computer and Applications (ICCA)*, Beirut: IEEE, Aug. 2018, pp. 1–26. doi: 10.1109/COMAPP.2018.8460248.

[8] F. D. De Mello Silva, A. K. Mishra, A. C. Viana, N. Achir, A. Fladenmuller, and L. H. M. K. Costa, "Performance Analysis of a Privacy-Preserving Frame Sniffer on a Raspberry Pi," in *2022 6th Cyber Security in Networking Conference (CSNet)*, Rio de Janeiro, Brazil: IEEE, Oct. 2022, pp. 1–7. doi: 10.1109/CSNet56116.2022.9955615.

[9] P. Poonia and V. Kumar, "Performance Evaluation of Network based Intrusion Detection Techniques with Raspberry Pi - a Comparative Analysis," *Int. J. Eng. Res.*, vol. 5, no. 10, 2017.

[10] C. Bousaba, T. Kazar, and W. C. Pizio, "Wireless Network Security Using Raspberry Pi," presented at the 2016 ASEE Annual Conference & Exposition, Jun. 2016. Accessed: Mar. 11, 2024. [Online]. Available: https://peer.asee.org/wireless-network-security-using-raspberry-pi