AlertBot: An Automatic Alert Validator

Business Case and Draft Plan

[redacted]
[redacted]

1 Business Case

1.1 Executive summary

Network intrusion detection systems create text alerts as part of their operation to secure networked systems. These alerts may be false positives, so alerts must be validated to determine if they are true. Alert validation is often done manually and is a time-consuming process. Our project will develop a software application that can automatically validate alerts using an artificial intelligence (AI) model. The software will also display network security information to a human operator with a graphical user interface. We believe that faster processing and clear reporting of alert information can improve network security.

1.2 Project motivation

Network security is a growing problem. Networked systems are widely used in important infrastructure, commercial, medical and government services. The high value of these assets make them enticing targets for criminals who wish to disrupt services or steal sensitive data. In 2018 in the USA there were 1241 data breaches affecting over 446 million private records [2]. This trend has been increasing yearly and is forecast to continue. A single data breach costs a company an average $US 3.84 million dollars. Data breaches take on average 97 days to identify, and a further 69 days to fix [3].

Network intrusion detection systems are used to monitor and protect networked systems. Intrusion detection systems monitor activities inside the network such as network traffic, file accesses and user permissions. All activity data is sent to an automated analyser. An intruder in the network may generate a distinct pattern of activity. If the detected activity matches a certain signature, a text-based alert is sent to warn a human operator of the suspicious event [1]. The alert contains information about the type of incident, its severity, the time it occurred, and the originating and destination host. Intrusion detection systems typically generate thousands of alerts a day. Many alerts are false positives. An individual alert may also not contain enough information to fully capture the full extent of an attack. Therefore human operators must validate these alerts to understand the true security context of the system. An operator may validate an alert by correlating it with previous alerts. An alert may also be externally validated by checking whether the alert contains an originating IP address known to be malicious.

To manually validate alerts is a very time-consuming process. Human analysts are often only aware of a successful network intrusion months after it has taken place [3]. By that time the damage is already done. Network security could be improved if the alert validation process was sped up with automation. Automatic validation would allow security operators to more rapidly identify and respond to suspicious behaviour on the network. A faster response time would allow the costs and damage of intrusion events to be reduced or avoided.

1.3 Project activities

Our project this semester will create a software application called AlertBot. AlertBot will automatically validate alerts generated by the open source intrusion detection system known as Snort[8].

AlertBot will receive a stream of alerts as packets from a network source. As alerts are received, AlertBot will parse the text fields of those alerts into features. The extracted features will be converted to a custom data structure and stored. The collected features will be passed through an AI model. The model can determine if a particular attack scenario is likely based on the alert features that have been previously collected. If several different, but associated, alert features occur in a given time period, it can be supporting evidence that a particular type of network attack is happening.

AlertBot can also externally validate a given alert by comparing the origin IP address to a provided database of known malicious IP addresses. If the address is known to be malicious, AlertBot will interpret alert data from that host more suspiciously.

When AlertBot detects a possible security threat, it will display the results in a graphical user interface. The user interface will inform the user of the nature of the threat, the supporting evidence validating the threat and possibly any advice to deal with it. The user will have the ability to mark the event closed in the user interface. Closing events will remove stored alert features associated with it. Removing alert features that no longer affect the system will reduce false diagnosis of threats.

1.4 Benefits

- The main benefit of AlertBot is validating alerts much faster than a human operator could manually. If alerts can be validated alerts more quickly, network intrusions can also be detected more quickly. Detecting intrusions faster will limit their damage.
- The graphical output of alert bot should be easier to understand than reading a flood of alert messages. A user should be able to determine at a glance the current security state of the network.
- The information AlertBot provides can give a security operator information to act decisively to defend the network.
- AlertBot can be extended to recognise more types of attack scenarios. This is particularly necessary in a rapidly changing security landscape with new threats always emerging.

1.5 Goals

We aim to achieve the following goals by the end of the semester:

- To build an artificial intelligence model that can use features extracted from alerts to recognise different types of network attack scenarios.
- To build a software application that uses the above AI model to validate a stream of alerts passed to it from a network connection.
- To externally validate alerts by comparing them to a mock database of malicious IP addresses.
- To create a graphical user interface to display the security context of the system based on the alerts validated.

1.6 Future directions

We only have a semester to complete the project. With more time, we could consider the following:

- AlertBot could be integrated with existing web services that externally validate IP addresses.
- Our system could be extended to read other popular alert formats, such as Bro.
- The number of alert types and scenarios that could be added to AlertBot is vast.
- The application could be further developed into a professional product.

2 Draft Plan

2.1 Schedule

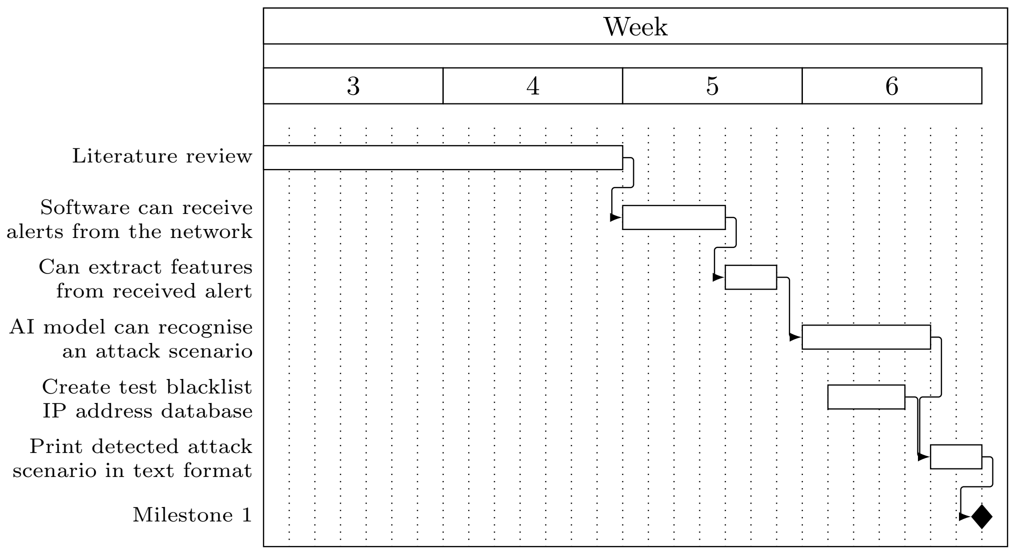

Figure 1: A timetable for milestone 1

Figure 1 shows a schedule of actions and dependencies to achieve milestone 1. These are:

- To complete a literature review. The objective is to understand how alerts are manually validated, what alert features are relevant for validation and which AI model could be used to validate alerts.
- The software must be able to receive alert messages from a network connection. A small test utility will be created to read alerts in a text file and send them through a network connection. The main application will receive incoming alerts from the test application.
- The application will parse alerts it has received from the network and extract the alert's text fields as features. The extracted features will be stored in a custom data structure and stored to be used for processing by the AI model.
- An AI model will be created that detects if a signature pattern of alert features has been seen by the application. This pattern will represent an attack scenario. If a test series of alerts contains this set of features, an attack should be detected. If not, there should be no response.
- A testing database of blacklisted (i.e. malicious) IP addresses will be created. The set of blacklisted addresses will be selected from the test data set. For purposes of milestone 1, an IP address will be randomly assigned as blacklisted. The application should successfully identify if an alert contains a blacklisted IP addresses and report this in the text output. A blacklisted IP address is an alert feature that should influence the output of the AI model.
- If an attack scenario pattern is detected, it will print a text output on the screen.

2.2 Roles

2.2.1 Project manager

[redacted] is responsible for leading the project, making schedules and ensuring the team follows the plan. He is also responsible for organising meetings and any written communication with the supervisors.

2.2.2 Developer

[redacted] are both responsible for writing the software component of this project. Because the team is small, architecture decisions and design choices will be decided through joint discussion.

2.2.3 Testing and Quality Assurance

[redacted] is responsible for writing and running software tests that confirm the produced software satisfies the project requirements.

2.3 Communication Plan

Communicating with	Document	Medium	Frequency
Team member	Text message	WeChat mobile app.	Daily
Team member	Meeting	In person	At least twice weekly
Team member	Task management	Trello	At least weekly
Team member	Project document	Google Drive	As needed
Team member	LaTeX document	Overleaf	As needed
Supervisor	Meeting confirmation	Email	Weekly
Supervisor	Meeting	In person	Weekly
Supervisor	Project document	Github wiki	Weekly

Table 1: How a team member will communicate with another person in the project

The key communication methods for a team member in this project are described in Table 1. Team members can expect receive fast responses from daily text messages. However the team must also meet in person as there are times that text messages lack expressiveness. We anticipate every client meeting will be followed with a team meeting to decide what to do next. Another weekly team meeting will be used to monitor progress and address project issues. Additional meetings may be held depending on need.

The team members collaboratively edit text and presentation documents on Google Drive [4] and LaTeX documents on Overleaf [7]. Copies of all project documents are stored on the Github wiki [6] where they can be viewed by team members and supervisors. These documents include time sheets, meeting minutes, research articles, project requirements, presentations and project source code.

Team members track tasks they are doing using a Trello board [5]. Tasks are assigned to Trello cards along with a due date and the person responsible for it. Team members move cards from the “To do”, “Doing” and “Done” columns as the status of the task changes. At any time the team should be able to see the status of a project task.

References

[1] Stallings, W., Brown, L., *Computer security principles and practice 4th ed.* (2018), Pearson, New York, NY

[2] *2018 end-of-year data breach report* (2019), Identity Theft Resource Center, San Diego, CA

[3] *Cost of a data breach study: global overview* (2018), Ponemon Institute LLC, Michigan, MI

[4] https://drive.google.com

[5] https://www.trello.com

[6] https://github.cs.adelaide.edu.au/MCI-Project2019/team19/wiki

[7] https://www.overleaf.com

[8] https://www.snort.org