Design Overview Document

ARTEMIS

Members
M.Ege Çıklabakkal, Mert Erdemir, Mert Kaan Yılmaz, Ataberk Dönmez

Supervisor
Dr. Pelin Angın

Contents

1 Product Description **2**
- 1.1 Brief Description . . . 2
- 1.2 Problems the Product Aims to Solve . . . 2
- 1.3 How the Product Solves the Problems . . . 3

2 High Level System View **3**
- 2.1 Context Diagram and Description . . . 3
- 2.2 Entities and Interactions . . . 4
- 2.3 System Boundary . . . 4

3 Overall Design **5**
- 3.1 Box and Line Diagram and Explanation of Design . . . 5
- 3.2 Design in Levels . . . 6
  - 3.2.1 Architectural Design . . . 6
  - 3.2.2 Detailed Design . . . 6
- 3.3 Design from Different Viewpoints . . . 7
  - 3.3.1 Customer's Perspective . . . 7
  - 3.3.2 Admin's Perspective . . . 7

4 Alternative Design Options **8**
- 4.1 Signature Based Defense . . . 8
- 4.2 Anomaly Detection on the Gateway . . . 8

1 Product Description

1.1 Brief Description

ARTEMIS, A Near Real-Time Monitoring and Intrusion Detection System for IoT, can be described as an intrusion detection system on Internet of Things devices, that utilizes sniffing network packets and machine learning in order to detect unusual activity in the sensor network and warn the user about the situation.

ARTEMIS mainly inspects the packets generated by IoT sensor devices and also the gateway device, then features are extracted from the packets coupled with network specific data then learning algorithms are used to detect anomalies and outliers. ARTEMIS allows the users to select from different learning algorithms as well as specify some parameters for these algorithms from a simple web interface.

ARTEMIS consists of 3 main components, the first one being the Sniffer device. The Sniffer is a Raspberry Pi that is tasked with sniffing the packets in the network and it is connected to a router so that it can send the collected raw data to the server for proccessing. Which brings us to the second component namely the Anomaly Detection Server. It is responsible from processing the packet data and running learning algorithms. Feature extraction, training the model, detection of the anomalous events and alerts are some of its main tasks. Lastly the Web Server communicates with the Anomaly Detection Server in order to present a visual output to the user about the sensor network, also allows the user to tinker with the learning algorithms so that they can try out different models and see which one fits best to their network.

Simply put, ARTEMIS is an IDS that can discover anomalies in the network consisting of IoT devices using machine learning and presents the user with a simple web interface so that they can monitor the network.

1.2 Problems the Product Aims to Solve

The Internet of Things is now being used increasingly in various fields such as healthcare, agriculture, transportation and smart home systems. IoT devices are expected to reach 50 billion devices all over the world by 2020[1]. Taking into account commercial pressures, security is often lacking in IoT. This exposes many critical points of sensor networks that allows an adversary to attack and disrupt the flow in the network and even view private data of the users of these IoT system. Especially in healthcare systems, this can have irreversible effects.

ARTEMIS aims to patch up these security weaknesses of IoT. It will be able to detect unusual activity in the network and create alerts about the situation. During our literature research we have also observed that available anomaly

detection datasets do not cover IoT. Another important output of this project is that we will generate a dataset for this purpose using IoT sensors.

1.3 How the Product Solves the Problems

The Sniffer Raspberry pi collects huge amount of data from the network that the product aims to secure. It continuously transfers these data to the Anomaly Detection Server where we run streaming data processing programs using Apache Spark Streaming and train models regularly with incoming packet data. Also testing will be done with each packet to check for anomalies. Up to now, we have made use of *Streaming K-means* algorithm as we expect the network data to be clustered. We shall integrate classification algorithms as well. Upon detecting an unusual activity, the server will create alerts which could be displayed in a web interface, as well as other network related data that we can output to the user in some readable/viewable form.

2 High Level System View

2.1 Context Diagram and Description

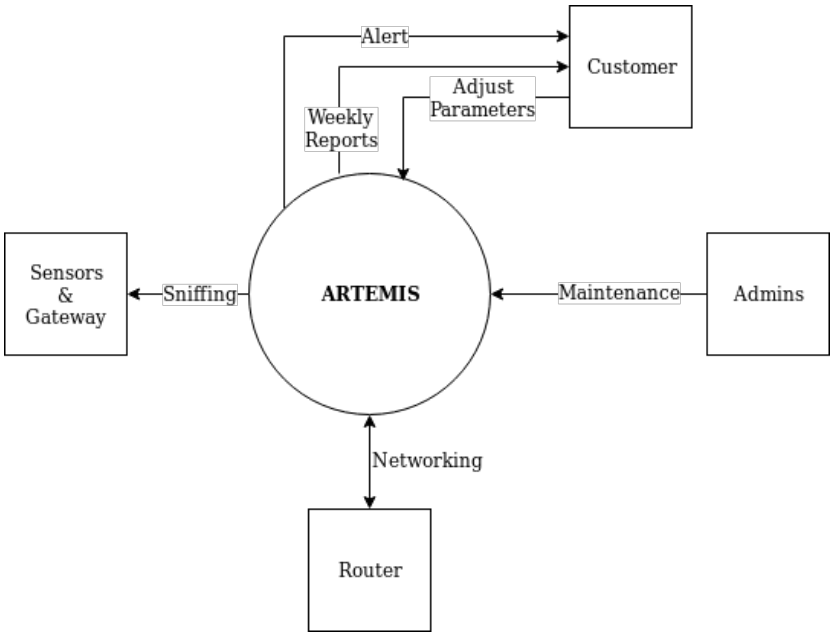

Figure 1: Context Diagram

ARTEMIS consists of 3 main parts, the Sniffer, Anomaly Detection Server and Web Server. If we inspect the context diagram, we see that sniffing is performed

on the network consisting of sensors and gateway. In order to sniff every packet in the network, sniffer must be put on **monitor mode**. However, in monitor mode, it is not able to use any of its Wi-Fi interfaces. The Sniffer has to send the packet data to the server for processing, since it is not able to send via Wi-Fi, we shall connect it to a router and send data over ethernet. Admins are responsible from the development and maintenance. Lastly, customers/users are the main actors of ARTEMIS. They want their sensor network to be secure and we provide them with a web interface that shows various network related properties, graphs and alerts with their reasons if one is to be generated. They are also able to choose the learning algorithm along with their hyperparameters from a set of choices that we have preset. Although the default should be enough and easy to use, we shall provide a modular program so that experienced users can pick the algorithms that suit their networks and systems the best.

2.2 Entities and Interactions

- **Sensors & Gateway:** Sensors generate packets that contain some physical information and send them to the Gateway. In a realistic scenario, many sensors regularly send data to one gateway. For example, a healthcare system consisting of a body area network with a few sensors along with another personal area network can be connected to a single gateway[2]. ARTEMIS and specifically the Sniffer listens the network and gathers the packets.

- **Router:** Router is used in order to solve the problem of the Sniffer not being able to use Wi-Fi interface while sniffing. It provides the networking for it, connected to the Sniffer with ethernet, it transfers the packet data to the Anomaly Detection Server.

- **Admins:** They are responsible from development of the product and maintain it according to the feedback and field experiments.

- **Customer:** Users of the product. Main interactions include reports, alerts of anomalies, visual representation of the learning model, presentation of the info about the network, as well as the specification of the learning algorithm along with some parameter choices.

2.3 System Boundary

ARTEMIS's main goal is to secure the sensor network. For example, anything beyond the Gateway is out of it's scope. Another important point is that the end product will be an intrusion detection system, not intrusion prevention. Therefore it is plug-n-play. ARTEMIS is not directly in communication with the IoT network, instead, it is somewhat like a privileged observer that warns the users if anything suspicious is going on in the network.

3 Overall Design

3.1 Box and Line Diagram and Explanation of Design

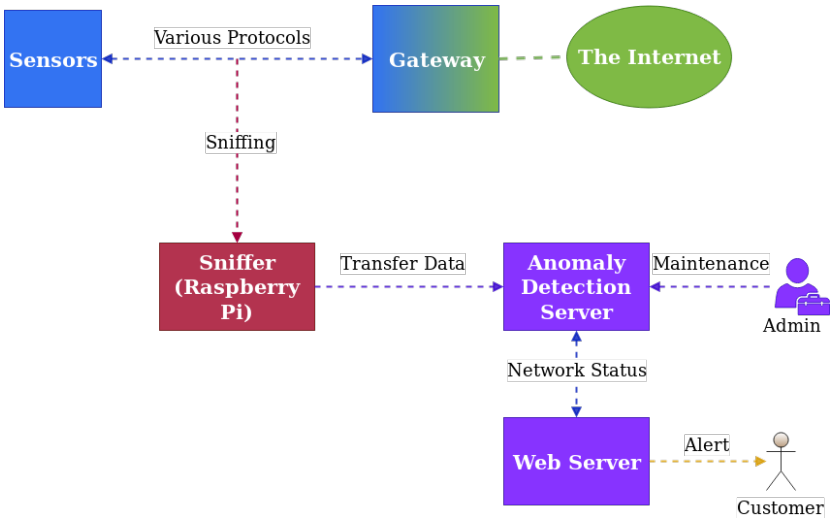

Figure 2: Box and Line Diagram

As the interactions and outer elements to the product are explained in the High Level System View section, this section focuses on the internal components, namely the Sniffer, Central Anomaly Detection Server(CADS) and the Web Server. Only technical detail that needs to be mentioned about the outer components is that there are certain protocols such as MQTT, Bluetooth Low Energy(BLE), Zigbee that are specific to IoT domain. ARTEMIS focuses on these protocols because these were the protocols we have found out when inspecting health systems.

The sensor data transferred to the Gateway is sniffed with a Raspberry Pi device that is amplified with protocol specific hardware such as BLE/Zigbee sniffer and Wi-Fi dongle which supports monitor mode. Since the sniffing is done in monitor mode, the network connection to the Server is provided via ethernet. Sniffed data is not directly sent to CADS by Sniffer itself. Rather, CADS gets all the data with an SSH connection. The Sniffer can perform sniffing with different tools such as Wireshark, tcpdump or Python Scapy according to the needs. There are obviously many different protocols in the network that are being picked up by the Sniffer. Since only a subset of these protocols are targeted, packets are being filtered according to protocols and their types as they are sniffed. The Sniffer is not expected to do any pre-processing on the data and raw data is transferred to CADS. The upside of not performing any processing on the device is that it is able to continuously sniff without missing any packets.

On the Anomaly Detection Server, parsing and pre-processing of the data is performed. Parsing operation is not uniform for all protocols and in order to extract the most useful information, we shall have specialized parsing routines for the aforementioned protocols. Processing of the data requires the Server to extract the features out of the packets, apply dimensionality reduction on the features and send the processed data to the machine learning process running on the same machine. Socket interface is used for this transfer of the data. Machine Learning algorithms are run on Apache Spark Streaming. The incoming data is big in size and continuous (stream). Apache Spark Streaming is very successful in processing large amounts of data, in addition, we used the Spark MLlib with it, in order to easily integrate some of the learning algorithms to the system. From MLlib, the first algorithm we have used is Streaming K-Means. We were able to continuously process the data, train the model and display a figure depicting the packet data features. Being a clustering algorithm, it provides nice visualisation, however a classification algorithm is more of a natural fit to the problem at hand.

The job of the Web Server is simple, it shall serve the users, providing an easy-to-use interface where different learning algorithms along with some parameters can be specified by the user. Python Django shall be used as it follows the model-view-template architectural pattern which is a modern approach for good web server design.

3.2 Design in Levels

3.2.1 Architectural Design

Sniffing routine is being run on the Sniffer continuously. As packets are being collected, the Server connects to the Sniffer via SSH and pulls them. On the Server, data is parsed and features are extracted. These features are used in training of the model. Testing shall be done concurrently and network related data, graphs, alerts shall be transferred to the Web Server. On the Web Server, users can observe the state of their IoT network.

3.2.2 Detailed Design

On the Sniffer, Wireshark(t-shark), tcpdump and Python Scapy are used for sniffing packets of various protocols. For example, along with a BLE sniffer, t-shark is able to collect BLE packets and Scapy is a good option for MQTT packets.

On CADS, there are multiple processes being run, one of which parses the packets and extracts features. Parsing differs from protocol to protocol. Extraction of the features is crucial to the performance of the model that would be trained. So far, we have selected four features from MQTT, they are as follows:

1. Packet Length
2. Packet Payload(Data itself)
3. Packet Arrival Rate
4. Packet Topic(Specific to MQTT)

We shall look to extending this list in the future. After extracting these features, a dimension reduction may be applied. Although the actual learning is performed on high dimensional data, for visualization, reducing it to 2D is needed. Now, over a socket, the processed data is sent to the Spark Streaming process.
On Streaming process, a routine performs the following: receives data, trains a model, writes the outputs to a file. Spark Streaming allows us to receive via the socket interface, which we have used thus far. Spark MLlib provides different algorithms such as SVM, K-Means, linear regression. MLlib makes it easier to integrate these solutions with the system.

Python Django is the framework of the Web Server. The user is provided with a simple UI.

3.3 Design from Different Viewpoints

3.3.1 Customer's Perspective

Customer/User can access the continuous status information about the network through the web server. Web server is provided by the department's virtual servers. Since ARTEMIS aims to detect anomalous events and inform the user according to these events, user/customer will be alerted via this web server. Alerts are sent over e-mail, along with a of visualization on web. Besides anomalies, general near-real time status of the IoT network can be followed by user on this web server. In addition, since some machine learning algorithms may work better in different cases, user will be able to choose different algorithms from the provided ones in our system and tune the parameters to the best while watching the instantaneous behaviour of the system with the adjusted configurations. Therefore, user is not limited to a single specific algorithm or design. Our design is able to shape itself according to the needs of the user and the environment. Also, system visualization feature makes our system to go beyond from a closed box system for the user to open and traceable system.

3.3.2 Admin's Perspective

System admins are responsible from the environment setup and the secure and stable connection between the components. According to feedback coming from the system, admins should take action and, if necessary, stabilize the system if something unusual occurs. Since our system tracks the anomalies in the network and also the anomalies in its internal system components, it is easier

for the system admins to track these problems. There is no need to continuously check the system status. However, in order to provide a robust system, a web visualization of the system status similar to users is given to the admins, as well. In order to make it more comprehensive, more detailed information such as packet details will be provided to admins. Therefore, admins are not only able to see the status but also manually track the system details and take precautions according to these. We tried to minimize the workload for the admins by creating a system that works on its own and track the problems in both network and the environment.

4 Alternative Design Options

4.1 Signature Based Defense

Some products already provide signature based defense mechanism which means that attack have to be recorded in the database to be detected. It does not use machine learning. This kind of method greatly suffers from not being able to detect zero-day attacks.

4.2 Anomaly Detection on the Gateway

One another method could have been detection of anomalies on the gateway itself[3]. This could be the gateway of the internet for the local network. However, the drawback of this system would be the bottleneck it creates. Every packet needs to be controlled and this slows down the internet greatly.

References

[1] D Evans. The internet of things: How the next evolution of the internet is changing everything. *Cisco Internet Business Solutions Group (IBSG)*, 1:1–11, 01 2011.

[2] Hande Özgür Alemdar and Cem Ersoy. Wireless sensor networks for healthcare: A survey. *Computer Networks*, 54:2688–2710, 2010.

[3] Ibbad Hafeez, Markku Antikainen, Aaron Yi Ding, and Sasu Tarkoma. Iot-keeper: Securing iot communications in edge networks. *CoRR*, abs/1810.08415, 2018.