Applying the PASTA Framework for Security Threat and Response Analysis in Disaster Relief System Architecture

Alex Lappin
Department of Cyber Security Engineering
George Mason University
Fairfax, VA, USA
alappin@gmu.edu

Matt Young
Department of Cyber Security Engineering
George Mason University
Fairfax, VA, USA
myoung45@gmu.edu

Sumayah Alomari
Department of Cyber Security Engineering
George Mason University
Fairfax, VA, USA
salomar3@gmu.edu

Mahi Khan
Department of Cyber Security Engineering
George Mason University
Fairfax, VA, USA
mkhan209@gmu.edu

Sherok Neamaalla
Department of Cyber Security Engineering
George Mason University
Fairfax, VA, USA
sneamaal@gmu.edu

***Abstract*—Every country deals consistently with natural disasters and tragedies that cost trillions of dollars in damage and millions of lives lost. During these disasters it is crucial to establish communication during a time when people are at most in need without internet, food, shelter, and basic resources. Due to the lack of telecommunication links between Crisis Management Centers (CMC) and those affected, there requires a unique and innovative solution using drones which bring communications using an ad-hoc network style to relay messages. This solution currently focuses on functionality whereas the paper aims to leverage a PASTA framework to accurately analyze the possible threats and vulnerabilities within the system and then provide valuable mitigation techniques with detailed threat modeling encompassing the scope of the solutions. In doing so, we can realize that this protocol requires a multi layered zero trust architecture (ZTA) within the boundaries of disaster relief.**

Keywords—Cyber-Security, natural disasters, disaster relief, Crisis Management Centers (CMC), Process for Attack Simulation and Threat Analysis (PASTA), zero trust architecture (ZTA)

I. Introduction

Natural disasters will never be a problem that can be solved, only mitigated through cooperation in the form of technology and communication. In many cases of disaster, traditional technological infrastructure proves to be insufficient due to the extremities in place. This calls for an additional solution that will take effect when everything fails.

Disasters have caused irreparable damage to many nations including Central and West Africa where 7.2 million people are currently affected by flooding and need disaster relief solutions [1]. Climate change is said to play a significant role in this disaster with an increase of global temperatures of just 2 degrees Celsius being a risk factor which could cause regular torrential downpours of this magnitude annually. In 2024, the flooding was enough to completely overwhelm all local disaster resources at hand and cause people in charge like Governor Babagana Zulum of Borno State to advocate for global assistance due to the humanitarian crisis causing loss of thousands of lives [2]. This becomes a problem of technology when local resources are used up and the internet becomes unavailable after the flooding.

A smaller scale example of this same exact occurrence would be in Hunga Tonga-Hunga Ha'apai, an island in the South Pacific, where an underwater volcano erupted on January 15, 2022, and was so intense that the atmosphere was changed globally. These people were then impacted by a resulting tsunami and complete severance of internet due to the damage [4].

The solution is to provide internet in the case of a disaster causing terrestrial internet implementations such as fiber optic cables, cell towers, and local routers to become unavailable. Rescue teams will direly need this internet to know victim's whereabouts and situational status. Flooding will guarantee that this is the case since it has the most impact on ground electronics versus other natural disasters. Using drones to relay local traffic to responders using a lightweight protocol will provide this operational need of the internet.

A. Current Technical Limitations

Clearly current technical infrastructure is not capable of handling the extent of these disasters as seen in the case of Central and West Africa as well as the island of Tonga. In Africa, flooding caused damage to crucial terrestrial infrastructure points connecting submarine cables to onshore landing stations, which led to physical fiber optic disconnections. Additionally, there was water infiltration into cable landing stations, combined with debris and sediment movement from intense flooding which destroyed the integrity of the fiber optic links themselves. Countries including Nigeria, Chad, Niger, Mali, and Cameroon were all impacted by the severing of critical emergency communication channels and humanitarian efforts [3]. Although the area had some infrastructure already in place for this scenario, the sheer amount of flooding ultimately proved the terrestrial optic cable to be inadequate regarding redundancy and resilience.

This case is also reflected in our second example where Hunga Tonga–Hunga Ha'apai caused a huge tsunami after a volcanic eruption. The fiber optic cable line connecting Tonga to Fiji was destroyed, which was the main form of communication. A 55-kilometer segment of this cable was either buried beneath thick volcanic sediment or completely severed due to the eruption's intense seismic activity and resulting underwater landslides. The result was an inaccessible cable that was completely unfunctional. Repair efforts became challenging when specific underwater vehicles were required (ROVs) in relaying the groundwork causing the time to recover (TTR) to be much longer than anticipated [4]. Starlink was involved in deploying specialized ground stations which had to be sourced internationally. Transport constraints, damaged infrastructure, customs clearance processes, and the need for specialized personnel to oversee installation and operations also delayed the amount of time that this temporary solution was able to be implemented.

The proposed solution of a drone-implemented ad-hoc network will bridge the gap between the affected ground circumstances, the specialized personnel needing to be called in, as well as the victims themselves as they will use a personal device. Drones will provide reliability and scalability due to them being easily replaceable as well as rapid deployment. MQTT will also conserve bandwidth and power in this IoT context without sacrificing reliability of transmitted messages. However, this solution currently does not support much security and privacy due to the lightweight protocol being used to link the victims' phone with the Crisis Management Center (CMC) [5]. This reliability is only the case when there is security and privacy, thus the need for PASTA.

B. *PASTA Framework for Threat Modeling*

Due to the security and privacy requirements of the disaster response system utilizing MQTT-based drone communications in IoT, we need a structured approach, where the Process for Attack Simulation and Threat Analysis (PASTA) is particularly suited to this scenario, since it integrates the necessary operational objectives into cybersecurity planning, creating a relationship between the technical solution and practical disaster-response needs.

Pasta provides a very detailed seven phase methodology of starting with clear system objectives about the underlying technical infrastructure and ties them to associated threats and weaknesses that are uncovered. By modeling threat scenarios, it creates clear guidelines for engineers of the system so that vulnerabilities and proactively discovered and dealt with early into the design process. Ultimately this should improve the detection and response rate of any cybersecurity issues which may impact on the functionality of the disaster recovery and response system.

C. *Objectives and Purpose*

Regarding this solution, we will be utilizing PASTA to perform a systematic and detailed threat analysis which will assess the security posture of the MQTT based drone communication system for disaster response. These objectives specifically include:

1. Mapping and identifying vulnerabilities specific to the disaster response system architecture in its entirety. This will include the users and their associated devices, the drones acting as publishers and responders, and the CMC cloud as an endpoint and hub for all communications.
2. Proposing and simulating realistic cyber-attack scenarios to evaluate potential impacts on communication integrity, confidentiality, and availability. This will also include authentication and authorization of proper individuals [5].
3. Recommending security improvements based on identified risks to effectively mitigate potential cyber related attacks down the line. These will be accessed and procured from standard practice and implementations within the industry [5] [6].

The conjunction of these security standard methods aims to ensure that the designed communication system remains secure, reliable, and trustworthy, which will create a more effective environment for disaster response teams to then respond to the crisis at hand without worry of cyber related problems on top of the already given problem. This will minimize financial damage, loss of life, and further improve infrastructure in place for disaster affected areas. This is a basic standard in IoT since there is usually a lack of secure focus early in the design process and IoT can cause damage beyond the scope of normal software due to its interaction with the physical environment [5].

II. System Architecture

The system is an innovative solution to ensure reliable and lightweight transmission within the event of a disaster where traditional forms of communication are unavailable. It uses sole air and elevated traffic transmission where terrestrial events that occur do not play a role in the reliability of the messages being sent. This IoT drone-based solution employs an ad-hoc aerial network of drones to facilitate communication. Victims that are being affected by the disaster, or in this case the flood, will utilize a smartphone application to send distress signals which include location and the severity of their situation via MQTT protocol. The messages are sent to drones safely nearby and then dynamically forwarded to the Crisis Management Center. At the CMC these messages are visualized on a platform called Grafana which enhanced the situational awareness of the response team based on the victims' needs.

A. *System Purpose*

The core objective of the project is to develop a relationship between reliable, secure, and private between the victims in a flooding scenario and rescue operations which need vital information. Terrestrial communication often fails within this context so there are additional measures in order to subvert this flaw. Traditionally there are things underground or underwater which have countermeasures against flooding, however, the failing rate of this is too high to consider usable. There are also efforts such as Starlink which require large teams to input groundwork first and cannot cover an area as large since the ground isn't necessarily available. This solution fills the need for crisis teams to be able to view information that will help the victim depending on their own personal situation without interference from disaster or technological flaws getting in the way.

B. *Key Components and Architecture*

This system operates as a lightweight traffic forwarding and management architecture which has some key technical components [6].

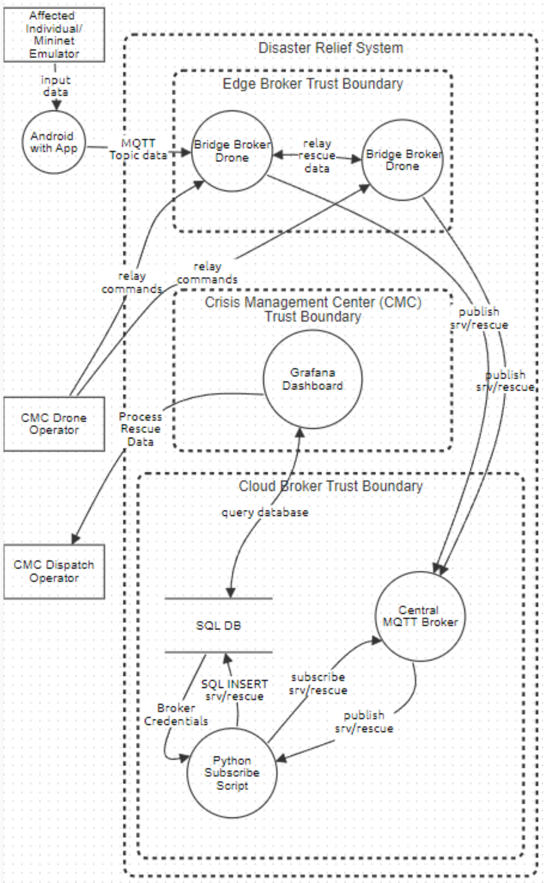


Figure 1: Entire System Architecture DFD

Figure 1 illustrates the entire system's key components working in unison with every stakeholder, boundary, process, data flow, and database included.

1. Stakeholders: These are all on the outside of the boundaries since they are considered external entities. They can be affected by outside influence and control and are usually considered the most vulnerable regarding security and privacy. These will also be the ones getting impacted if things malfunction.

   The victims here are the affected users in the disaster scenario who are non-technical users and carry a smartphone with them. These are the people in need of rescue and will be preoccupied by the gravity of the disaster [2]. They will be publishing their own distress information which includes the levels of severity, low, medium, or high as well as the description of the condition they are in and the location of themselves in the form of GPS coordinates. They will also update their situation according to their needs and wants [6].

   The emulated victim is less of a stakeholder itself but will mimic the stakeholder in a testing scenario for development purposes so they will be regarded in the same category as the victims themselves. These are simply coming from python scripts using Mininet WIFI which establishes false victims and drone nodes to forward similar information to the real scenario.

   Finally, there is the CMC Operators and Dispatchers which are the people who are committed to helping the victims, whether volunteer or for a career. This includes the drone operators, who will be flying the drone bridge brokers around remotely from a command center and will be responsible for the correct positioning of these drones to properly capture the victims' traffic within an Ad-Hoc mesh network, as well as avoid the natural disasters at hand. There are also the operators responsible for the crisis management center aggregation and processing of data. These operators will be focusing on rescue planning and management based on the data flowing in from the Grafana dashboard. They will be looking specifically for trends or individuals in the most need and may also coordinate with drone operators and rescue teams to further assist them.

2. Processes: These are the ongoing software elements within the system that provide key roles in connecting stakeholders with one another based on the flow of data given. They are reprogrammable and must be resilient.

   The drone bridge broker is a lightweight MQTT (Message Queuing Telemetry Transport) broker running on Eclipse Mosquitto and Ubuntu within a drone that is able to take controls from an operator and cover aerial support. This is responsible for receiving the published messages from the victims and then forwarding them to the central broker, which is the subscriber. They may also be responsible for relaying rescue data to one another to then better reach the central broker. This will be using a topic, which is a virtual channel used to filter and route messages; thus, publishers and subscribers do not need to know each other directly. The MQTT protocol that it is running on is a machine-to-machine protocol designed to be lightweight and for IoT specifically [5]. This causes low bandwidth and power consumption. This broker specifically uses the bridging feature, which requires a hostname and credentials, to send it to the central broker. Once this topic has been republished or forwarded, the drone's duty has been completed by subverting any need for a cell tower.

   The Central broker is using the exact same MQTT technology, however it is in a much more stable technology of being in the cloud environment. This makes it easily reachable for all incoming messages from the drone bridge brokers. There will be multiple drones so there must be bandwidth and processing power available on this cloud broker. This requires high availability and scalability. It will be looking for authentication within its incoming bridge messages as well as the same hostname and credentials when the python subscriber reaches out to it. Once the subscriber has been confirmed to receive a topic, the central broker will then forward the MQTT data over to the python subscriber function.

The python subscriber will have a mqtt_subscribe function which will be responsible for subscribing to the target topic in the central broker. Then to authenticate with the main broker. After a proper handshake, the central broker in the cloud will then be forwarding this MQTT data over to the python cloud lambda function. This will be a function called mqtt_publish where it accepts the data and then parses the significant fields such as the IDs, severity, location, and timestamp. Once this data is extracted it will persist into the SQL database by using a SQL insert expression. It will also be responsible for error handling within any missing logs or fields that the central broker has received [6].

Finally, the Grafana dashboard is an open-source data visualization and monitoring tool. This will be connected to the SQL database and be presenting real-time data in accessible formats for operators to read in the form of the ID, severity, geo-location, and timestamp. There will be a dynamic map where you identify the risk situation colors the node positions by severity, a bar chart shows the number of nodes classified by risk, and a bar chart shows the number of nodes classified by severity. It will be running on HTTPS/HTTP to display on operators' screens. This will provide operators with the necessary information to perform their duties to full efficiency [6].

3. Database: The database is a persistent storage of all the details forwarded by the python subscriber including IDs, severity, location, and timestamp. This allows for historical records of any published data as well as repudiation in the event of tampering. Operators may also use this to analyze the trends in an area over time. It is also responsible for real-time SELECT queries from Grafana where any amount of information may be taken out or modified. This is stored in the cloud somewhere to prevent any blackouts where even if local internet is down, the cloud is still active. There can also be calculations of KPI involved within this technology.

4. Boundaries: These are the zones that transition one domain of security to another in a readable format and encompass everything aside from the stakeholders. This looks specifically at risks and authentication.

   The edge broker trust boundary is the ad-hoc mesh network of aerial drones that occupy the middle point between the victims and the cloud environment. It includes drone bridge brokers. This boundary focuses on authentication and authorization of smartphones, emulated and real, as well as accessibility to the cloud. This environment is the most physically dangerous since drones will be flying near disaster events and may receive physical harm out in the field. Data integrity is critical in this area since it can easily be tampered with or destroyed given any interference. Local untrusted networks will be involved out in the field as well as attackers that are far away from management, so physical compromising is a concern here.

   The cloud broker trust boundary utilizes Software as a Service (SaaS) as well as Infrastructure as a Service (IaaS). This means that the resources are provided on the cloud without management of any hardware. It will be highly scalable and include plenty of redundancies, especially with a budget. It includes the central broker, python subscriber, and SQL database. All these processes will be looking for credentials from one another hence the longer process of receiving information without having the hardware private on site. There will also be a third party managing these so there must be trust in place regarding security and privacy. There will also be access controls in place regarding what can and cannot be subscribed to further prevent any excess load of messages or corrupted data.

   The crisis management trust boundary is the physical location boundary of the CMC headquarters. This will be occupied by CMC operators of all types as well as anyone else involved in rescue operations. This boundary only includes the Grafana dashboard process which will be locally hosted and connected to the cloud environment. There will be computers and hardware servers involved at this physical location as well as physical restrictions such as badges and guards to prevent unauthorized users from accessing the area. There may also be compliance with the local government here or any HIPAA since workers are involved.

5. Data flows: These are the actual streams of bits going over some sort of connection and are the heart of the operations' integrity. These connect processes to users and other processes as well as the SQL database.

   The MQTT data is originally sent from the victim to the central broker, and then to the python subscriber. MQTT transmits binary data in small packets. This technology makes it very lightweight since there are fewer bytes associated [5]. This TCP stack also allows it to run on something like a raspberry pi in cases such as drones. The packets have fixed headers and may contain an associated payload. This operates on a subscriber-publisher model where instead of HTTP, it uses a topic channel publishers send a message and subscribers receive a message. The topic is a string which serves as a routing key. The brokers will then use these topics to determine which subscriber will receive the message. This then works perfectly in an ad-hoc environment. In the case of this scenario, it will be carrying the data of IDs, severity, locations, and timestamps.

   The python subscriber will create an MQTT client in python using the python library Paho [6]. This will then prompt an authentication to be sent to the central broker. Once it has been accepted, there are callbacks set for on_connect and on_message. The script then attempts to connect to srv/temperature or srv/rescue which will be the topic in this case of the central broker [6]. It then takes the messages in on_message and decodes them and inserts them into the SQL database. The script will also be pulling these credentials from the SQL database itself.

Grafana supports SQL queries by default and after configuration of connection information of the host, database name, credentials, and TLS settings, Grafana will be able to access the data from the SQL Server [6]. Once this is done it queries the data in real-time from the database itself. This is then ready for operators to view and interpret live. They may categorize the data or make trends as the situation evolves. They will also probably forward the data to other rescue operators based on their interpretations.

Finally, drone operators will be using a local interface at the CMC. Since the drones are running Ubuntu, they can run a lightweight SSH command session which allows them to control the drone remotely after giving a proper authentication [6]. They will be navigating through flight paths and waypoints. The operators will also be modifying the Mosquitto file to set bridge parameters such as the topic, credentials, and bridge target.

C. *Current limitations and Areas for Development*

Despite the tight knit efficiency and lightweight nature of the system as a whole, its biggest flaw revolves around the assumption that there will not be any attempts from attackers to modify the system flow. Strong TLS encryption and robust certification management will be an entirely difficult operation on its own and may even ruin the simple lightweight nature of MQTT. Bridge rules will also be equally frustrating to effectively manage since they are error prone and may not be used by trained operators. Each broker is also prone to being a bottle neck since under high volume, lightweight nature will not help. This framework is known to be insecure to many types of attacks, which will be further discussed later [5].

The drones themselves are also an extremely fragile link within the system. Drones have limited battery life and flight time and act as a single point of failure. Without an excess of extra drones and skilled operators they will completely shut down the system if the signal is somehow lost. The weather is a huge variable that the drones need to operate properly as well as sunlight giving heavy restrictions. Data will be completely lost in this case with no recovery or redundancy. Keeping a proper ad-hoc multi hop connection between these drones is also not an easy task as this will be flakey and should be regarded going into the design process as such. Additionally, these drones will be out in the field and subject to anybody flying a counter-offensive drone or measure to capture and ruin the drone. Finding the proper number of operators in order to use these things also may prove challenging.

Finally, using Mininet as a substitute for drone operation and victim messaging will not prepare the operators for real world scenarios of signal loss or flight variables. Additionally, it is notably complex to set up properly in a large-scale topology.

III. Threat Modeling Overview

To secure critical infrastructure like a disaster relief system it is vital to adopt a threat modeling methodology that supports both business goals and attacker-oriented perspectives. For this project, we will be using the Process for Attack Simulation and Threat Analysis (PASTA) framework because of its ability to connect these two aspects.

PASTA is a risk-centric, seven-stage technique that offers a structured, iterative approach to system threat analysis. PASTA stresses a dynamic simulation of threats based on actual attack vectors, technological assets, and business objectives, in contrast to STRIDE, which is more checklist-based, or OCTAVE, which strongly leans toward organizational maturity and risk posture. Thus, PASTA is suitable for disaster relief systems, where public safety and human lives are directly impacted by system uptime and data reliability. PASTA's attacker-centric strategy, based on actionable intelligence, is its core component. Every step builds on the one before it, gradually raising situational awareness and educating stakeholders about the vulnerabilities, threat surfaces, and risk impacts unique to a given system. This is an explanation of its seven phases in relation to our use case for disaster relief.

A. *Stage 1: Define Objectives*

The first step in the process is to establish the disaster relief system's primary business goals. These business goals are then aligned with security objectives, including ensuring the integrity of situational data, authenticating legitimate response actors, and maintaining system resilience against outages or cyberattacks. After a broad mission has been set, the application boundaries can be adjusted to satisfy business requirements. It is essential to suitably scope the functional needs of the system. Although an overly broad scope may appear comprehensive at first, it can greatly increase the attack surface of the system and introduce additional possible vulnerabilities. Businesses may guarantee that operational goals are reached while reducing needless exposure to cyber threats by closely tying business and security objectives together throughout the scoping process [7][8].

B. *Stage 2: Define Technical Scope*

The technical components of the system and the deployment environment are clearly defined at this level. Cloud-based dashboards, field equipment connected via MQTT (via brokers like Mosquitto), API endpoints, mobile apps, and data stores are all part of our disaster relief system. The system's surface area and complexity are increased by external stakeholders such as volunteers, emergency services, and cloud service providers. This scope lays the groundwork for threat identification and simulation by specifying the function, entry points, communication protocol, and data flow of each component [7][8].

C. *Stage 3: Decompose the Application*

The system is broken down into functional elements that an attacker could target those elements are then corresponded to technical modules, such as user identity management and MQTT communication pipelines. To clearly visualize how information moves between components, Data Flow Diagrams (DFDs) are created by clarifying data input, processing, storage, and output [8]. To appropriately depict the structure and relationships of the system, Data Flow Diagrams (DFDs) are updated and improved. They guarantee that access control is properly documented by assisting in the identification of crucial components including users, roles, and permissions. Along with identifying data entry points and establishing component trust levels, DFDs also list important assets across the data, hardware, and software layers. DFD components include external entities, processes, data flows, data store, and trust boundaries which are defined in table 1.

Table 1: DFD Components and Descriptions

DFD Component	Description
External Entity	An external system or user that communicates with the system under diagramming by sending and receiving data.
Process	A function that converts input data into output, whether by logic, computation, or data flow management in accordance with business standards.
Data Flow	The path or route that information takes between data stores, processes, and external entities.
Data Store	A database, file, or other type of storage space where data is kept for later use.
Trust Boundary	A boundary inside the system that divides regions with varying degrees of trust is used to determine where security measures might need to be implemented.

D. *Stage 4: Threat Analysis*

This stage investigates potential attack scenarios, understanding various ways adversaries could damage the disaster relief system [7][8]. The PASTA framework offers a methodical way to evaluate high-impact and realistic threats. Targeted attack patterns and known vulnerabilities can be found by utilizing external intelligence sources like MITRE ATT&CK and CAPEC. These insights help model real-world threats such denial-of-service (DoS) attacks against centralized services, injection attacks, unauthorized access to coordination dashboards, and man-in-the-middle (MitM) assaults on MQTT communications. Thus, this stage strengthens the system's threat awareness and response planning by simulating how these attacks might disrupt operations through attacker narratives and misuse scenarios.

E. *Stage 5: Vulnerability & Weakness Analysis*

The primary goal of this stage is to map identified threats to specific vulnerabilities across system assets, providing a clear understanding of risk exposure [8]. Vulnerabilities can stem from various sources such as insecure code, misconfigured services (e.g. Mosquitto brokers), or insufficient MQTT encryption. Evaluating components like authentication, authorization, session management, data storage, and web applications is essential. Threat intelligence sources such as MITRE CWE, CVE, and scoring systems like CVSS and CWSS, help in assessing the severity of each vulnerability. These identifiers link weaknesses to system components and support risk prioritization. Additionally, misuse cases are used to simulate potential system abuse by adversaries, providing insight into how vulnerabilities might result in actual compromise [7][8].

F. *Stage 6: Attack Modeling*

In this stage, identified threats and vulnerabilities are mapped into structured attack models, such as attack trees or attack flows, which provide proof of concept for how adversaries could exploit the system. These models aid in identifying crucial weaknesses and chokepoints by visualizing the attack progression patterns, from initial access to final impact [8]. The creation of these trees is aided by tools such as CVE, CWE, CAPEC, and CVSS, which provide comprehensive information on known attack patterns, vulnerabilities, and misconfigurations. CVSS offers risk grading, CAPEC describes typical attack techniques, and CVE and CWE assist in identifying exploitable vulnerabilities. Integrating this intelligence into attack models, threat simulations become more accurate and useful, allowing for a more proactive and knowledgeable security approach.

G. *Stage 7: Risk and Impact Analysis*

This is the final step of the PASTA framework, which includes using a risk matrix that assesses threat probability, impact severity, and the efficacy of current or potential controls [7]. The goal is to lower risks by putting countermeasures in place that target the most serious dangers. The ability of each mitigation to reduce residual risk is evaluated. Through increased visibility into system vulnerabilities, simulated attacks help stakeholders comprehend the practical repercussions of unresolved threats. This step facilitates well-informed, fact-based decisions that maximize security investments and minimize organizational exposure by estimating residual risk and evaluating the efficacy of controls.

IV. THREAT MODELING APPLIED

A. *Stage 1: Define Objectives*

In a crisis scenario, such as floods, earthquakes, or other natural disasters, the Disaster Relief System must be designed in such a way that the populations of affected areas are effectively provided aid. To that end, system objectives must be clearly defined in a way that covers functional, security, and regulatory/compliance requirements. Defined objectives should also be categorized from a tactical, operational, and strategic perspective.

Strategic Objectives:

- Provide reliable means for affected individuals to send locational data and status to the CMC personnel in an area of operation with limited to no communications infrastructure
- Ensure confidentiality, integrity, and availability of communication channels from affected user to bridge broker drone, bridge broker drone to bridge broker drone, and bridge broker drone to central MQTT broker
- Ensure confidentiality, integrity, and availability of backend cloud services

Beginning at the highest level of objective definition, strategic objectives describe the overall purpose and requirements of our system in a crisis scenario. From a functionality standpoint, the system needs to provide a reliable means for affected individuals in operations to communicate with Crisis Management Center (CMC) personnel via a pre-installed Android application. It is expected that the area of operation will have limited to no communication infrastructure due to the disaster, so the communication system needs to be easily deployable, redeployable, and must utilize lightweight communication protocols to compensate for limited resources. The system needs to have a method of processing and storing data that CMC personnel can reliably access so that they can relay information to search and rescue units.

In addition to providing functionality, the system needs to possess security features that provide resilience and reliability in the event of malicious activity. This means ensuring the confidentiality, integrity, and availability of communication channels throughout the system. Namely the communication channels from affected user to bridge broker drone, bridge broker drone to bridge broker drone, and bridge broker drone to central MQTT broker. Data stored in the system must retain confidentiality, integrity, and availability to ensure operational effectiveness and public trust. Both internal and external threat actors must be considered in security design.

Operational Objectives:

- Ensure reliable means of distributing app to affected individuals in area of operation
- Process and store data acquired from central MQTT broker in an SQL database
- Maintain queryable database to store field data
- Ensure data processing and storage actions are resilient to tampering disruptions
- Ensure CMC dispatch operators can effectively and securely receive and relay situational data to first responder units
- Ensure CMC drone operators have secure and reliable means to operate bridge broker drone field assets
- Ensure asset manipulation tracking is implemented

Next, operational objectives are defined. These state the system level functions that support the strategic objectives. From a functionality standpoint, this includes utilizing MQTT communication protocol due to its low overhead and low bandwidth and power requirements. The system must relay data–which will need to be processed via cloud-based services–from bridge broker drones to the central MQTT broker. All data is stored and queryable via an SQL database. The system must utilize a dashboard interface that presents a visualization of situational data for CMC dispatch operators to process. Furthermore, CMC dispatch operators must also coordinate search and rescue units. CMC drone operators must be able to reliably communicate to fielded drone assets for redeployment based on changes to operational needs.

From a security standpoint, the system must secure the MQTT protocol to ensure the integrity and confidentiality of messages from affected users. Communication channels from the CMC drone operators to fielded drone assets must also be secured to ensure resilience to external threats. The SQL database must have methods in place to mitigate attack via external incoming malicious data and internal CMC personnel manipulation. The system must also have methods in place to identify and track internal and external anomalous behavior.

Tactical Objectives:

- Ensure data processing and storage actions are resilient to tampering disruptions
- Ensure CMC operators cannot affect integrity of the database
- Utilize Python subscribe client in backend cloud services
- Implement hardening mechanisms in MQTT communication channels
- Develop security policies that define permissions, roles, and response plans

Finally, tactical objectives define the needed functionalities that support real-time operations. These objectives ensure the system remains constantly operable to meet a larger goal. The system must utilize a publish/subscribe Python process that persists the messages received from the central MQTT broker to an SQL database. The system must then use Grafana as the dashboard service that visualizes the data for CMC dispatch operators.

From a security standpoint, the system must integrate methods of end-to-end encryption and authentication into MQTT to ensure confidentiality and integrity of the data. The system must ensure the central MQTT broker is resilient to a denial-of-service scenario which would negatively impact availability. The data coming from affected individuals must be checked for malicious content to prevent SQL database compromise. Methods of communication used between CMC personnel and field assets must implement encryption and authentication. A log system and intrusion detection system must be utilized to monitor anomalous activity and identify external attacks and insider threats in real-time. Security policies and settings must be put in place that follow industry standards.

B. Stage 2: Define Technical Scope

In this section, the technical scope of the system is defined by identifying components involved in data transmission, processing, and visualization in the IoT architecture. First it is important to establish trust boundaries within the system. There are three main trust boundaries (not including the external boundary) within the Disaster Relief System, those being the Edge Broker, the Cloud Broker, and the CMC. The initial component in the system is the Android device with the Disaster Relief System application installed. This component is technically outside the system's external trust boundary and acts as the initial data sender. Data flows from this external source into the Edge Broker trust boundary. Within the Edge Broker trust boundary there are bridge broker drone components. They act as sensors in the system, collecting data from the external Android devices in operation. There are multiple bridge broker drones within this boundary since they communicate with each other to forward MQTT messages to a central MQTT broker located in the Cloud Broker. During this process data flows from the Edge broker trust boundary into the Cloud broker trust boundary. It is important to note that throughout the process of the affected user's Android sending data and the central MQTT broker receiving it, the MQTT protocol is used. Eclipse

Mosquitto is the message broker that implements the MQTT protocol. From this point on, however, the MQTT protocol is not used.

Within the Cloud Broker Boundary there are three components: the central MQTT broker, the Python subscribe script, and the SQL database. When the central MQTT broker receives data, it is then forwarded to the Python subscribe script through a publish/subscribe process. The nature of the publish/subscribe relationship means that data is flowing bidirectionally between the central MQTT broker, and the Python subscribe script. The data then persists within the SQL database. From there the data flows from the SQL database trust boundary to the Grafana dashboard located within the CMC trust boundary. This is done via SQL queries.

Data flows from the Cloud Broker trust boundary into the CMC trust boundary when Grafana dashboard queries the SQL database. Grafana dashboard helps CMC personnel visualize, and process data related to location and status of affected individuals through an interface. The Grafana dashboard is the only component within the CMC trust boundary. As previously mentioned, CMC personnel are the ones accessing the Grafana dashboard, so the data flow ends when it exits the CMC and external trust boundary and flows to the CMC operator who is considered an external actor. Additionally, a CMC drone operator is sending commands to fielded drone assets. This CMC drone operator is classified as an external actor to the system.

To summarize, the components of the system that are identified in the technical scope are listed in the below table with their respective boundaries. This helps us to better understand the components that may have security concerns and how their place is within the system.

Table 2: Technical Scope Components and Description

COMPONENT	BOUNDARY	DESCRIPTION
Android Device	External	Sends affected user data to the nearest drone
Bridge Broker Drones	Edge Broker	Collects data and sends the information to the MQTT broker
Central MQTT Broker	Cloud Broker	Receives information from the drones and sends it to the database
Python Subscribe Script	Cloud Broker	Service within the cloud that receives data from the central broker and forwards to SQL database
SQL Database	Cloud Broker	Database where all the data from MQTT broker gets stored
Grafana Dashboard	Crisis Management Center (CMC)	Dashboard that visualizes data from SQL database

As for the technologies that are being used with their appropriate dependencies, they can be categorized into five layers. The first layer is the front-end portion, where react and the Android app reacts throughout the entire operation. JavaScript ecosystem and mobile OS API are the dependencies for this layer, since they are directly dependent on React and the Android app. Second layer is the backend layer, where we are using Python as our main language. To use Python to operate the entire backbone of the operation, the Python libraries and runtime environments become dependencies of the technology in the second layer. Communication is a crucial part of the operation, which is our third layer. Technologies like MQTT Protocol and Eclipse Mosquitto are being used in our system, since these are network components that are being used, network libraries can be considered as dependencies of these components. Fourth layer is the Data Storage layer, where MQTT is uploading data in SQL databases in our system and since SQL databases are being used, our key technology here is MySQL. To operate SQL databases using MySQL, SQL components like SQL connectors and cloud storage are being used as the dependencies of MySQL. This is because the database for our system is in the cloud and would require SQL connectors to connect to the database and storage to store all the data. Our last layer is the dashboard, where Grafana is being used as a Graphical User Interface for users to read and interact with the data from the database. The Data Source Plugins allow the user to connect and interact with the data source, since they act as bridges which allows Grafana to retrieve and display data from the database. Therefore, data source plugins are the dependencies of Grafana. Table 3 showcases the layers, technologies that are being used within the layers and their dependencies. Figure 2 further explains these relationships in regard to their respective data flows within the form of a system-wide DFD.

Table 3: Technical Scope layers

LAYER	TECHNOLOGIES	DEPENDENCIES
Frontend	React, Android App	- JavaScript ecosystem - Mobile OS API
Backend	Python	- Python libraries - Python runtime environment
Communication	MQTT Protocol, Eclipse Mosquitto	- Network libraries
Data Storage	MySQL	- SQL connectors - Cloud storage
Dashboard	Grafana	- Data source plugins

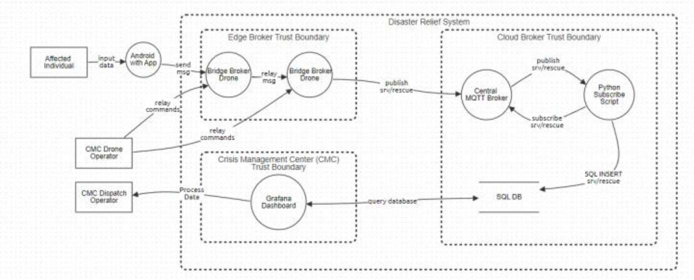


Figure 2: System DFD

C. *Stage 3: Decompose the Application*

In this section, the system is broken down into assets that are identified and categorized. This mainly consists of going into more detail of how components in Stage II work, how the system views them from a trust perspective, and how different users will interact with them. Users are also identified, trust levels are defined, and use cases are established.

The Android app is used by affected individuals to input a severity level and their geographical location which can later be reviewed by CMC personnel. It is assumed that affected individuals in operation will have this app installed before the disaster. To connect to a local fielded drone asset acting as a bridge broker, the app uses the bridge broker's IP address and port to connect. This will most likely be done either automatically as the drone gets within range or there will be another method relaying that information to the affected individual for manual input. The user can then enter positional data latitude and longitude format. The user can then select a severity level ranging from either low, medium, or high. The Android device then publishes a message containing the user's device ID, severity, and location via MQTT protocol (TCP/IP) every 10 seconds. The user can also update information as needed.

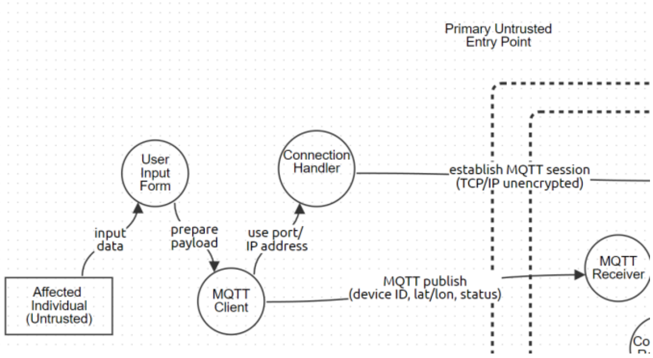


Figure 3.1: DFD of System App

The app is considered an external untrusted component to the system as seen in Figure 3.1 and represents a primary entry point into the system's data flow. In addition, the data contained within the message sent to the bridge broker drone can be classified as PII and thus is considered sensitive information. Currently there is no encryption or authentication mechanism in place. These factors make the component a compelling target for malicious actors who wish to disrupt system integrity and confidentiality. This constitutes a major weakness in the system and may serve as a platform for various attacks on the system.

The bridge broker drones are deployed in groups over an area of operation and serve as relays for incoming messages from affected individuals. They utilize MQTT protocol for communications and implement this through Eclipse Mosquitto, acting in bridge mode. This means they receive messages from a user node, the Android devices, and forward them to a central MQTT broker. In an operation, the bridge broker drones are remotely given instructions by a CMC drone operator who positions them over affected individuals. The method of communication between drone operator and drone is not specified.

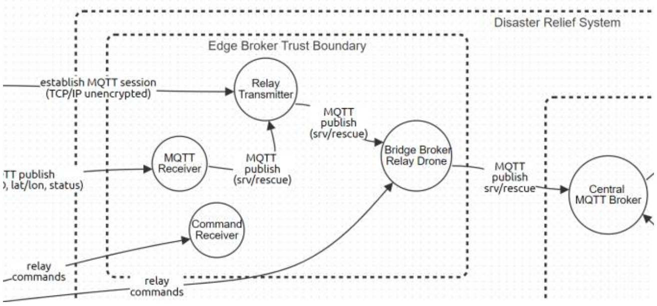


Figure 3.2: DFD of Edge Broker Boundry

Bridge broker drones are considered an internal trusted system component located within the Edge Broker trust boundary, which is represented in Figure 3.2. They are forwarding messages that contain user PII and possess a direct link to the Cloud Broker trust boundary. They also utilize the same MQTT protocol that lacks any encryption or authentication mechanisms. These factors make the component a compelling target for malicious actors who wish to breach system confidentiality and integrity. This constitutes a major weakness to the system.

The central MQTT broker is the primary node that receives all data from the field via the bridge broker drone network. It is located within the Cloud Broker trust boundary and marks the entry point to the boundary. The central MQTT broker publishes data to clients within the cloud with a subscribe/publish and topic approach. Essentially the clients receiving data subscribe to a topic string which filters what information the central broker will publish to them. In the case of the system, this topic would be srv/rescue. Through this architecture the central MQTT broker acts as a message router for the Python subscriber script.

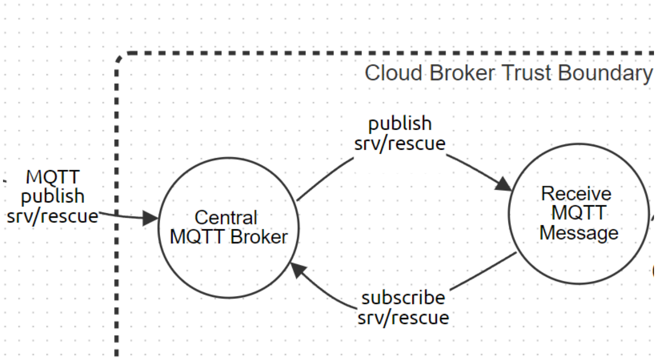


Figure 3.3: DFD of Central MQTT Broker

In Figure 3.3, the central MQTT broker is considered an internal trusted system component. The contents of the data being received is user PII and the component itself is the only entry point for data from the field to the Cloud Broker boundary. This makes the component a high priority target for malicious actors since it acts as a major entry point that can either be bypassed or blocked. Adversaries may attempt to use this attack surface to compromise system confidentiality, integrity, and availability. It can be expected that many attacks will be directed at this component to either access stored data or disrupt the flow of data. As previously stated, the MQTT protocol used to communicate with the field drone assets possesses no forms of encryption or authentication. The central MQTT broker also currently has no security controls in place. This constitutes a major weakness to the system.

The Python subscriber script is a trusted component within the Cloud Broker trust boundary that processes data from the central MQTT broker which the script is subscribed to. The received message is parsed for device ID, coordinates, and severity level. Data is then sent to the SQL database using an SQL insert operation. The Python subscriber script itself does not represent a major attack surface for adversary actions. It can be seen more as an obstacle that must be bypassed to reach the SQL database. The component does not have a public facing interface and is not an entry point but processes unvalidated input data from external sources. At the current moment there are no such security features to validate external inputs. This component does not constitute a high priority target for adversaries but serves as a critical processing point. It may be used as an attack vector to compromise the SQL database's integrity and confidentiality. This constitutes a moderate weakness in the system.

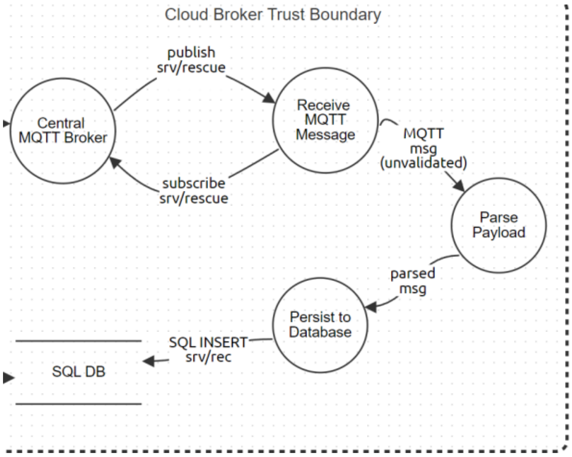


Figure 3.4: DFD of Cloud Trust Boundary

In Figure 3.4, the SQL database is a trusted component within the Cloud Broker trust boundary that serves as a data storage unit and is implemented via MYSQL. It receives the processed data from the central MQTT broker and can be queried by the Grafana dashboard by CMC dispatch operators. The database stores the device IDs, positional data, and severity level of all affected users in the area of operation. Because of the queryable nature of the component, it can be considered a controlled entry/exit point from the Cloud Broker trust boundary to the CMC trust boundary.

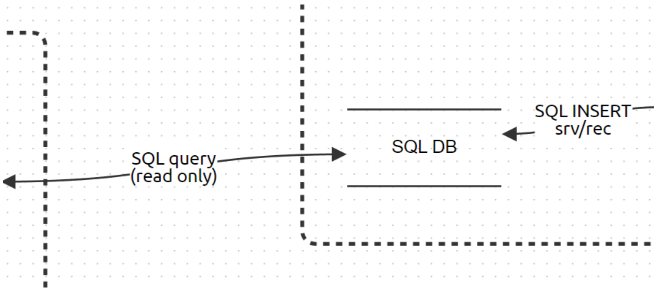


Figure 3.5: DFD of SQL Database

Similar to the Python subscriber script, in Figure 3.5 the SQL database is not a public facing component and is not directly exposed to external users. Due to the high value data being stored it represents a critical asset in the system's operation. Adversaries may attempt to compromise confidentiality and integrity of the component which would degrade operational reliability. Currently there are no security controls in place to mitigate against attacks from data being persisted by the Python subscribe script or from insider threats operating the Grafana dashboard. This constitutes a major weakness to the system.

The Grafana dashboard is the final major component of the system. It is located within the CMC trust boundary and acts as the interface between the CMC dispatch operator and the SQL database. The dashboard visualizes data to assist the dispatch operator in processing data and make decisions on where to allocate search and rescue units. Grafana utilizes a plugin that allows it to query MYSQL data sources, such as the SQL database in the cloud. The Grafana dashboard can be considered an entry point into the system and the CMC boundary and is considered trusted. The only external users able to interface with the dashboard are CMC personnel who are currently considered external but trusted users.

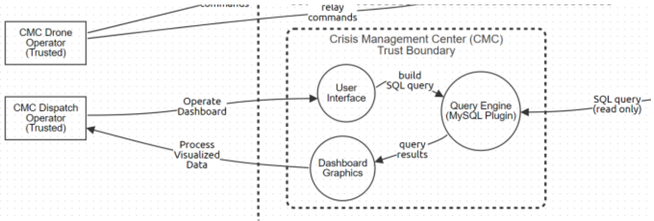


Figure 3.6: DFD of CMC Trust Boundary

In Figure 3.6, the Grafana dashboard is not exposed to public external networks but interfaces with sensitive backend systems. As such it is a possible attack surface for the system. For the system design, it is assumed that the only users who can access the dashboard will be onsite CMC staff. There is no mention of remote workers accessing the dashboard. Due to the interaction with the SQL database, it is possible for insider threat actors to breach confidentiality and integrity of the system. There are no security controls in place to detect or defend against insider threats. As such, this constitutes a moderate weakness to the system. Combining these flows together, we achieve the results presented in Figure 3.7.

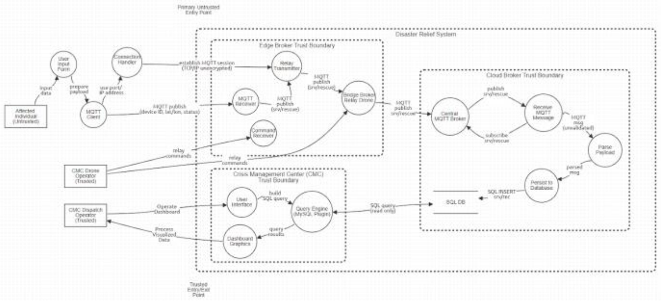


Figure 3.7: Final DFD of System

In addition to identifying components and entry points that potentially face threats, it is important to identify system users. This means identifying roles and permissions of the users and demonstrating use cases of the system. This helps map expected interactions and trust relations. The users of the system are currently the CMC dispatch operator, CMC drone operator, and affected individuals.

The CMC dispatch operator is considered an external but trusted user whose role is to process data and allocate search and rescue units. Their current permissions are to interface with the Grafana dashboard and query the SQL database. It is important to note that they are expected to read from the database and not modify it, but there are no access controls in place to enforce this. A typical use case represented in Figure

4 would have the dispatch operator access the dashboard service onsite, query the database for collected information, analyze data displayed on the dashboard, and make a call where to send search and rescue units.

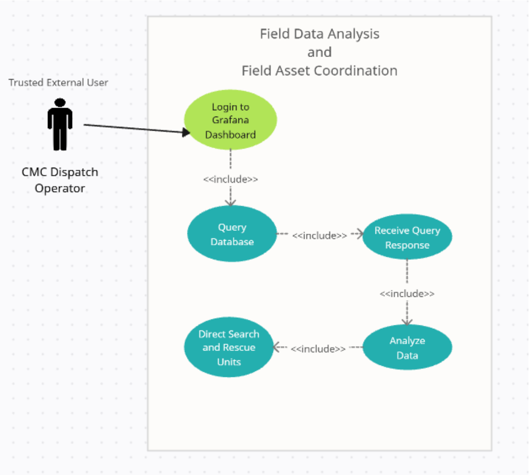


Figure 4: Use Case 1

In Figure 5, the CMC drone operator is also considered an external but trusted user whose role is to operate fielded drone assets. Their current permissions are to remotely operate drones. The method of operation will depend on the type of drone used which is not defined in the system. It is expected that they do not have access to the Grafana dashboard. A typical use case would have the drone operator arrive onsite, receive instruction on where to deploy drone assets, and redeploy if required by operation.

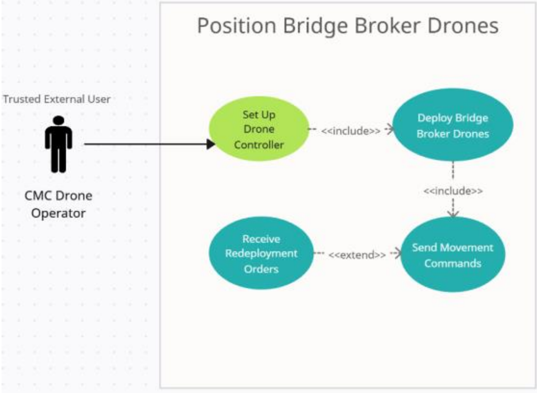


Figure 5: Use Case 2

In Figure 6, the affected individual is considered an external user whose role is to send data to the bridge broker drone network via an Android app. In the current implementation, their inputs are considered trusted by the system, which presents a major security risk. Their current permissions are to interface with bridge broker drones through input data which is processed by backend cloud services. A typical use case would have the affected individual download the app before a disaster event, connect to a local fielded drone asset, input situational data, and await search and rescue units.

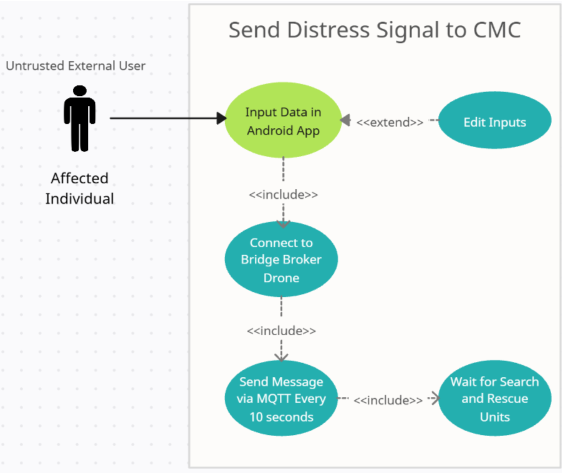

Send Distress Signal to CMC
Untrusted External User
Affected Individual
Input Data in Android App
<<extend>>
Edit Inputs
<<include>>
Connect to Bridge Broker Drone
<<include>>
Send Message via MQTT Every 10 seconds
<<include>>
Wait for Search and Rescue Units

Figure 6: Use Case 3

D. Stage 4: Threat Analysis

With the system components identified in detail, users identified, and boundaries established, threat identification is the next stage of the threat model. The chosen framework will be STRIDE. Using STRIDE, we will be identifying every single possible threat that could take place, given our system's architect. STRIDE is a threat modeling framework that is designed to identify and categorize security threats in an application or system. Each letter of the word represents a category of threat, Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. We will be taking all the components that are being used in the system, actors who are part of the system, and the data flow within our system into consideration when listing out all the possible threats that apply to our system within Table 4. After the list has been created, we will be identifying which category each of the listed contents align with. In table 4, a description will be provided of how each of the listed items affects our system according to the STRIDE classification category.

Table 4: STRIDE classification

COMPONENT	STRIDE CLASSIFICATION	THREAT	DESCRIPTION
Android App	Spoofing	Device Identity Spoofing	An attacker impersonates a legitimate user's Android device to send false distress signals, misleading drone responses and potentially redirecting rescue resources inefficiently.
Android App	Spoofing	False App	An attacker manipulates the user into downloading malware disguised as legitimate on their Android device. The attacker then malicious actions while effectively cutting off the user from the system, negatively impacting system availability.

Bridge Broker Drones	Information Disclosure	Interception and Extraction	A hacker with access to a network traffic monitoring tool can sniff network traffic being sent from affected user to bridge broker or from bridge broker to central broker, which leads to the information regarding affected user status and location being disclosed, negatively impacting the confidentiality of the data.
Bridge Broker Drones	Spoofing	Identity Spoofing	A hacker with abilities to craft messages that appear to come from a survivor within the area of operation can send false requests, which leads to dispatch operators directing resources to a false report, negatively impacting the reliability and effectiveness of the system.
Bridge Broker Drones	Tampering	Man in the Middle Tampering	A hacker with the ability to place themselves in between the communication channel of the survivor and bridge broker drone can intercept and alter the data, which leads to inaccurate reports being received by dispatch operators, negatively impacting the integrity of the system.
Central MQTT Broker	Spoofing	MQTT Spoofing	A hacker with access to the MQTT broker and knowledge of legitimate drone client IDs can impersonate a rescue drone, injecting false telemetry data or deceptive commands into the system, which leads to incorrect information being processed by the command center, negatively impacting the accuracy and reliability of critical decision-making during crisis response.
Central MQTT Broker	Denial of Service	Configuration Backdoors	A hacker who exploits the MQTT broker's default or weak credentials can gain administrative access, allowing them to modify the broker's configuration, create persistent backdoors, and restrict topic access for legitimate backend clients, which disrupts the communication flow and negatively impacts the availability of the central messaging system.
Central MQTT Broker	Denial of service	DoS on MQTT Broker	An unauthorized device repeatedly floods the MQTT broker with distorted or excessive messages, overwhelming system resources and delaying the delivery of legitimate rescue data. This disruption in communication between drones and the command center can critically hinder real-time decision-making and reduce the overall effectiveness of emergency response operations.
SQL Database	Tampering	SQL Injection	A hacker with access to the channel can inject malicious SQL statements into the app's input fields, which leads to the SQL database being open to manipulation, negatively impacting the integrity and confidentiality of the SQL database.
Python Subscribe Script	Elevation of Privilege	Unauthorized DB Access	An attacker who gains control over the data processing script can exploit it to perform unauthorized write operations to the database, leading to the alteration of sensitive records and compromising the accuracy and reliability of stored mission data.
SQL Database	Repudiation	Insider Threat	A malicious operator with trusted access to the system can manipulate the database and remove entries, which leads to some affected individuals not receiving assistance, negatively impacting

			the integrity of the SQL database.
Grafana Dashboard	Elevation of Privilege	Unauthorized Access via Stolen Credentials	A malicious operator or infiltrator with a set of stolen credentials can operate the Grafana dashboard and access the SQL database, which leads to sensitive information being leaked, negatively impacting confidentiality of the SQL database.

While not being nearly as critical as other components such as bridge brokers, the Android app faces a number of threats from adversaries attempting to target the point of initial data flow. One such threat is device identity spoofing, in which the adversary utilizes identity spoofing (CAPEC-151) to take the identity of a legitimate user and supply false data. The tactic utilized here is persistence (TA0003) and the technique is valid accounts (T1078) which involves obtaining legitimate user accounts to interface with the system. Adversaries may utilize a VPN to spoof location to match the location of the area of operation. The impact is that the attacker degrades system availability since resources can get redirected to false alerts. Another possible threat would be the creation of a false app by an adversary. The adversary would create a false app that would appear to be legitimate but would actually be loaded with malware (CAPEC-185). The tactic used would be defense evasion (TA0030) and in order to bypass Play store static security checks the adversary would use the download new code at runtime technique (T1407). The impact of this is that the availability of the service would be gone for affected individuals who fall for the manipulation since they would be effectively cut off from the service [9][10].

There are three threats that can take place within the Bridge Broker Drone’s component, they are information disclosure, spoofing and tampering. The interception and extraction threat falls under information disclosure of STRIDE. It is where an adversary with access to network traffic monitoring tools can monitor the network and gain sensitive information which the adversary should not have access to. This threat impacts confidentiality in a negative manner and puts the entire system in a vulnerable state. The following tactics can be referenced for this threat, Collection (TA0009), Discovery (TA0007) and Exfiltration (TA0010). The techniques are Adversary-in-the-Middle (T1557), Network Service Discovery (T1046), Exfiltration Over Alternative Protocol (T1048) and Exfiltration Over C2 Channel (T1041). As for CAPEC it is CAPEC-117: Interception, T1040: Network Sniffing. The second threat is identity spoofing, which falls under spoofing, it is where an adversary may act as a survivor and send false requests to the drones, leading to false reports and dispatching rescue teams in the wrong areas. Tactics like Initial Access (TA0001), Credential Access (TA0006), Lateral Movement (TA0008) and Persistence (TA0003) can be referenced for this specific threat within the scenario. Techniques like Valid Accounts (T1078), Network Sniffing (T1040), Exploitation of Remote Services (T1210), Remote Services (T1021) and Traffic Signaling (T1205). As for CAPEC, it is CAPEC-151: Identity Spoofing. The final threat from this component is Man-in-the-Middle, which falls under the Tampering aspect of STRIDE. It is where an adversary is able to position themselves between users and the communication bridge, which is being used by the drones. Defense Evasion (TA0005), Impact (TA0040), Collection (TA0009) and Credential Access (TA0006) are the tactics that can be aligned with this threat. Impersonation (T1656), Indicator Removal (T1070), Masquerading (T1036), Data Manipulation (T1565), Adversary in the Middle (T1557). CAPEC-94: Adversary in the middle, is the attack method for this threat [9][10].

The Central MQTT Broker is the backbone of the disaster relief system's communication, due to its exposure to data sources, it is an important target for enemies. One major threat is MQTT spoofing, in which a hacker poses as a genuine drone by using identity spoofing (CAPEC-151). An attacker can introduce incorrect telemetry into the system by taking advantage of legitimate accounts or weak identity verification, which could cause confusion and imprecise decision-making at the command center. Configuration backdoor is another serious vulnerability, where attackers exploit unsecured credentials (T1552) to gain persistent access and manipulate broker settings, such as topic permissions or logging functions. A Denial-of-Service (DoS) attack directed at the broker itself poses a more disruptive threat. An attacker can flood the broker with traffic using techniques like flooding (CAPEC-125), forced deadlocks (CAPEC-25), or network denial of service (T1498), which stops or delays data transmissions. This can degrade response coordination and system availability significantly in high-pressure disaster situations [9][10].

The Python Subscribe Script, responsible for receiving MQTT data and forwarding it to the SQL database, plays a vital role in the backend data pipeline of the disaster relief system. A significant threat is unauthorized database access, where attackers take advantage of weaknesses in authentication or credential storage to gain access. Using brute force techniques (T1110) or extracting credentials from password stores (T1555), attackers can bypass login protections and gain access to sensitive data. Attackers can also use known operating system credentials (CAPEC-653) acquired through reconnaissance or system misconfigurations, or they may rely on credential exploitation (T1212) due to that attackers can elevate privileges. Attackers may also exploit incorrectly configured access control levels (CAPEC-180), where poorly scoped permissions for example allow them more extensive database access than needed. In more advanced situations, attackers can manipulate the way database-connected scripts or processes run to enter unauthorized commands by hijacking execution flow (T1574) [9][10].

The possible threat that can come out of SQL Database is Insider Threat; it falls under the Repudiation portion of STRIDE. It is when a malicious actor with trusted access to the system is able to modify data within the database according to their malicious objectives. This can lead to several users in the field not getting the proper assistance they need, ultimately impacting the integrity of the database. The following tactics can be aligned with this specific threat, Insider Threat (T1078), Collection (TA0009), Exfiltration (TA0010), Privilege Escalation (TA0004) and Credential Access (TA0006). Valid Accounts (T1078), Automated Collection (T1119), Exfiltration Over C2 Channel (T1041), Abuse Elevation Control Mechanism (T1548), Exploitation for Credential Access (T1212), OS Credential Dumping (T1003) and Unsecured Credentials (T1552) are the techniques that can be referenced for this threat. CAPEC-180: Exploiting Incorrectly Configured Access Control Security Levels, CAPEC-653: Use of Known OS Credentials and CAPEC-600: Credential Stuffing are attack methods that can be referenced with this threat. The SQL database may also be vulnerable to SQL injection attacks, such as SQL injection (CAPEC-66), content injection (T1659), blind SQL injection (CAPEC-7), or command-line execution (CAPEC-108), if they interface with backend databases. These attacks impact systems integrity and confidentiality by taking advantage of insufficient input validation and can result in full database compromise or unauthorized data alteration [9][10].

The Grafana Dashboard is used to visualize mission-critical data, and there is one threat that unauthorized access via stolen credentials can take place, especially if authentication mechanisms are not properly secured. Attackers can exploit these using techniques like OS credential dumping (T1003) or credential access exploitation (T1212), or by harvesting unsecured credentials (T1552) from configuration files. They can also use tactics such as exploitation of trusted identifiers (CAPEC-21) or reuse known OS credentials (CAPEC-653) to act as authorized users. If those tactics are successful then an attacker could gain access to sensitive incident dashboards, monitor ongoing operations, or alter data [9][10].

E. *Stage 5: Vulnerability & Weakness Analysis*

In this phase of the PASTA technique, we find vulnerabilities linked to threats that have already been listed and analyze how these weaknesses appear in system components and technologies. Using the Common Vulnerabilities and Exposures (CVE) database and the Common Weakness Enumeration (CWE), we expand on the threat surface and match each threat with technical weaknesses. We also use DREAD (Discoverability, Damage, Reproducibility, Exploitability, and Affected Users) modeling to prioritize risks and help with evidence-based analysis.

Device identification spoofing, when an attacker acts as a valid mobile app that is being used by volunteers or civilians, in order to submit false reports, which is one of the system's main mobile weaknesses. CWE-290: Authentication Bypass by Spoofing, which exposes weaknesses in verification procedures. Attackers can guess or reuse valid device IDs due to weak binding between the app and its client identity. One real-world example is CVE-2020-8913, where unauthorized apps were able to take advantage of exported components due to insufficient permission checks in Android. Due to this vulnerability, false distress signals may be interpreted as legitimate in the context of disaster response, which could result in an improper allocation of resources which then delays the help that needs to be sent out to real victims. This also ruins the credibility of the field reports [11][12].

The integrity of the system and user safety are severely compromised by the potential for a user to install a fake disaster relief app that looks like the real one. CWE-353: Missing Support for Integrity Check describes the absence of runtime verification to determine the legitimacy of application. The well-known StrandHogg vulnerability and CVE-2023-21398 both show how Android apps can be imitated to obtain user input or get around permission limitations. A fake app on your system can steal user information or secretly stop valid communications, making it impossible for survivors to share their situations [11][12].

Insecure channels can allow sensitive telemetry and user-submitted data in the disaster relief scenario to be intercepted, especially when MQTT interactions are taking place between the central server, field devices, and drones. This is aligned with CWE-319: Cleartext Transmission of Sensitive Information, which happens when protocols send private information without proper encryption. A vulnerability in an unencrypted MQTT library, CVE-2017-7650, shows the danger of this type of communication. Attackers could use MQTT traffic to spy on our system to learn about victim locations. This could lead to data breaches, targeted misinformation, or coordinated interference with field teams [11][12].

Attackers can introduce fake telemetry or control messages into the system when MQTT brokers exploit client authentication and authorization. This falls under CWE-290, and is shown by CVE-2021-34431, in which the broker did not accurately verify the identities of the clients. This might be used by an attacker to appear as a real drone and modify readings. Operational insecurity during time-sensitive events could happen from this type of tampering which might confuse crisis responders and cause unnecessary rescue operations [11][12].

Misconfigured MQTT can leave communication channels vulnerable to MitM attacks, especially if TLS is not used. When data paths are not encrypted or verified end-to-end, they fall under CWE-300: Channel Accessible by Non-Endpoint. As shown in CVE-2018-12546, Mosquito's weak listener configurations exposed users to possible listening and message manipulation. The accuracy of decisions made at the crisis management center may be compromised if an attacker positioned themselves between a bridge drone and the central broker. This could result in fake alerts, delayed critical telemetry, or modified field reports [11][12].

MQTT spoofing happens when the integrity of data is not checked at the broker level, same as identity spoofing. This corresponds to CWE-345: Insufficient Verification of Data Authenticity. CVE-2020-13849 showed how unauthenticated clients might get around Mosquito's topic access restrictions, allowing attackers to secretly add or edit MQTT messages. This could include modified responder instructions or false alerts, which might cause confusion and block coordinated actions in the field [11][12].

Another major vulnerability is poorly secured broker configurations with default credentials or admin interfaces that are not defined. This is aligned with CWE-912: Hidden Functionality, which focuses on the possibility of backdoors that avoid standard security measures. CVE-2021-41039 found cases in Mosquito where persistent accessibility might be maintained by taking advantage of weak setups. In a real-world attack, adversaries might secretly generate illegal topic access or plant hidden subscriptions [11][12].

If resource limitations are not properly maintained, the MQTT broker that serves as the communication center of your system is vulnerable to DoS attacks. In CWE-401: Missing Release of Memory after Effective Lifetime, memory pools may be consumed by improperly handled MQTT packets. CVE-2021-34431, describes how malicious MQTT data could be used by attackers to bring down Mosquito [11][12].

Since python is being used as the backend system to communicate with the database, it can be vulnerable to SQL injection attacks, if it is not properly cleaned up. CWE-89, Improper Neutralization of Special Elements in SQL

Commands, explains this basic web application vulnerability. Through unsafe API calls, attackers were able to successfully inject SQL payloads in CVE-2021-27965. Data integrity might be compromised if an attacker were to take advantage of this vulnerability in the system and get access to sensitive data, tamper with incident reports, or disable parts of the database [11][12].

Access to databases that depend on hardcoded credentials are vulnerable to unauthorized intrusion. CWE-798: Use of Hard-coded Credentials, with CVE-2018-1000525, showcases real-world examples, where sensitive credentials are embedded in public-facing code and bad actors could take advantage of this and can get access to disaster system's backend, allowing them to make unauthorized changes to operational data [11][12].

Insider risks can come from trusted employees with excessive or inaccurate authorizations for access. Systems that do not strictly enforce role-based access control (RBAC) are represented by CWE-732: Incorrect Permission Assignment for Critical Resource. This is shown in CVE-2020-36326, where unauthorized access was made possible through MySQL roles. A malicious insider could manipulate data and disrupt the coordination relationship between the dispatcher and rescue team, without triggering security alarms [11][12].

Grafana dashboards can become targets for attackers using stolen or reused credentials. This aligns with CWE-522: Insufficiently Protected Credentials, and CVE-2021-43798. This showcases how unsafe configuration and session handling led to administrative interfaces being exposed to unauthorized users. Unauthorized access from an insider to the dashboard could lead to modified visualizations and data, which might damage the accuracy of decision making or making judgements when it comes to giving the appropriate data to the responders [11][12].

We will now create a DREAD model for the threats in Table 5 by calculating the DREAD score for each threat. DREAD is a structured security risk assessment model used to evaluate and prioritize threats based on their potential impact on a system. Each letter of the word stands for different factors that are taken into consideration while doing the calculation, Damage Potential, Reproducibility, Exploitability, Affected Users and Discoverability. The DREAD score is calculated by this equation, DREAD Score = (D+R+E+A+D)/5.

Table 5: DREAD Model

Threat	D	R	E	A	D	DREAD Score (Avg)	Risk Level
Device Identity Spoofing	7	8	7	8	7	7.4	High
False App Deployment	8	6	6	9	7	7.2	High
Interception & Extraction	7	7	8	7	8	7.4	High
MQTT Identity Spoofing	8	7	7	9	7	7.6	High
Man-in-the-Middle Tampering	8	7	7	8	6	7.2	High
MQTT Spoofing	8	7	6	9	6	7.2	High
Configuration Backdoors	9	7	7	9	6	7.6	High
DoS on MQTT Broker	9	8	9	10	7	8.6	Critical
SQL Injection	8	7	7	9	7	7.6	High
Unauthorized Database Access	8	7	7	9	7	7.6	High
Insider Threat	9	6	6	8	6	7.0	High
Stolen Credentials (Grafana Access)	8	7	7	9	7	7.6	High

F. *Stage 6: Attack Modeling*

In stage six, potential attack paths throughout the disaster relief system architecture are highlighted by building attacker-centric models that link threats to system vulnerabilities. This stage illustrates how attackers might use configuration errors, weak authentication, and exposed interfaces to gain access to system components using a variety of threat intelligence sources, including MITRE, ATT&CK, and CAPEC. As attackers move from initial access through the system to critical impact, these attack flows allow for the accurate identification of control failures, privilege escalation paths, and exploitation obstacles.

Table 6 maps out the threat along with the vulnerabilities, possible attack paths, and impacts. Each row shows how an attacker could compromise the different system components by taking advantage of a particular weakness identified using CWE references. This mapping makes it easier to see how technological errors in communication security, data validation, access control, and authentication can result in serious operational risks like inaccurate information, misallocation of resources, or DoS.

Table 6: Threats mapped to vulnerabilities

Threat	Vulnerability	Attack Path	Impact
Device Identity Spoofing	CWE-290: Authentication Bypass	Spoof device ID → Send false MQTT message → Broker accepts → Command center misled	Improper allocation of resources, reduced trust in system reports
False App Deployment	CWE-353: Missing Integrity Check	Bypass Playstore security checks → Mimic legitimate app → Install Malware on Android → Harvest Data	Users cut off from reporting, privacy breached

Interception & Extraction	CWE-319: Cleartext Transmission	Deploy sniffer tool → Exploit Unencrypted MQTT Comms Channel → Access to Sensitive Data	Data confidentiality compromised, targeted attacks possible
MQTT Identity Spoofing	CWE-290: Authentication Bypass	Gain access to valid drone ID → Spoof identity of drone → Exploit lack of authentication → Send false data to system	Misleading situational data, resource misdirection
MITM Tampering	CWE-300: Channel Accessible by Non-Endpoint	Intercept MQTT message → Modify payload → Broker accepts tampered data	Misinformation, command center confusion
MQTT Spoofing	CWE-345: Insufficient Data Verification	No authorization validation → Inject fake MQTT input → Altered data in database → Inaccurate operation calls	Operational delay, false data accepted
Configuration Backdoors	CWE-912: Hidden Functionality	Bypass weak/default credentials → Gain admin access → Manipulate security settings → Deny service to system	Persistent control over broker, difficult to detect
DoS on MQTT Broker	CWE-401: Memory Exhaustion	Malformed messages → Exhausting resources → Broker crash	Complete communication disruption during emergencies
SQL Injection	CWE-89: Improper Input Neutralization	Malicious input → SQL executed → Data exposed or altered	Integrity breach, data leakage
Unauthorized Database Access	CWE-798: Hard-coded Credentials	Locate credentials → Direct Database access	Data exfiltration, backend compromised
Insider Threat	CWE-732: Excessive Privileges	Internal misuse of elevated access	Operational disruption, confidential data leaked
Stolen Credentials (Grafana)	CWE-522: Insufficient Credential Protection	Steal login info → Access and alter live dashboards	Live operational data exposed or modified, leading to wrong decisions

Since Table 6 with threat, vulnerability, attack path and impact are instantiated, we can now create a prioritized attack path list that needs to be mitigated, based on the DREAD risk assessment model. Each threat in the DREAD model is given a score from 1 to 10 in each of its categories, giving us our average DREAD score which determines the risk level for each threat. This implies that if a threat has a high score, then that threat should be mitigated first and then the priority should move to the one with lower score. Even though DREAD directly determines the rank order in the list, there are other factors that are also taken into consideration when it comes to prioritizing mitigation for threats. One of them is severity of impact, for example the DoS MQTT broker threat is ranked higher than Configuration backdoor threat because if a DoS attack is successful then it will shut down the entire system but backdoor on the other hand allows attackers to stay in the system long term and have access to it. Given how heavily dependent our system is on the MQTT broker to function properly, any high-level tampering with it can cause the entire system to shut down and stop our operations. Another factor that plays a part is the complexity of mitigation. This is when a mitigation process of a threat is easier to fix compared to another threat of the same risk level or score, then that specific threat might be higher in the mitigation priority list. After all of these factors are taken into consideration, here is the list of prioritized attack paths for mitigation.

1. DoS on MQTT Broker:

- DREAD Score: 8.6 (Critical)
- Attack Path: Malformed messages → Exhausting resources → Broker crash
- Impact: Complete communication disruption during emergencies
- Priority: Highest Mitigation Focus

2. Configuration Backdoors:

- DREAD Score: 7.6
- Attack Path: Bypass weak/default credentials → Gain admin access → Manipulate settings → Deny service
- Impact: Persistent control over broker, difficult to detect
- Priority: High Mitigation Focus

3. SQL Injection:

- DREAD Score: 7.6
- Attack Path: Malicious input → SQL executed → Data exposed/altered
- Impact: Integrity breach, data leakage
- Priority: High Mitigation Focus

4. Unauthorized Database Access:

- DREAD Score: 7.6
- Attack Path: Locate credentials → Direct database access
- Impact: Data exfiltration, backend compromised
- Priority: High Mitigation Focus

5. Stolen Credentials (Grafana)

- DREAD Score: 7.6
- Attack Path: Steal login info → Access/alter live dashboards
- Impact: Operational data exposed/altered
- Priority: High Mitigation Focus

6. MQTT Identity Spoofing

- DREAD Score: 7.6
- Attack Path: Spoof identity of drone → Exploit

lack of auth → Send false data

- Impact: Situational awareness loss
- Priority: High Mitigation Focus

7. Device Identity Spoofing

- DREAD Score: 7.4
- Attack Path: Spoof device ID → Send false MQTT → Broker accepts
- Impact: Resource misallocation
- Priority: High Mitigation Focus

8. Interception & Extraction

- DREAD Score: 7.4
- Attack Path: Deploy sniffer → Exploit unencrypted MQTT channel
- Impact: Data confidentiality loss
- Priority: High Mitigation Focus

9. False App Deployment

- DREAD Score: 7.2
- Attack Path: Bypass Play store → Install malware
- Impact: Privacy breached
- Priority: Medium Mitigation Focus

10. MITM Tampering

- DREAD Score: 7.2
- Attack Path: Intercept MQTT → Modify payload → Broker accepts
- Impact: Command redirection
- Priority: Medium Mitigation Focus

11. MQTT Spoofing

- DREAD Score: 7.2
- Attack Path: Inject fake MQTT input → Alter data
- Impact: Delay, acceptance of wrong data
- Priority: Medium Mitigation Focus

12. Insider Threat

- DREAD Score: 7.0
- Attack Path: Misuse of privileged access
- Impact: Operational disruption, data leak
- Priority: Medium Mitigation Focus

Since we created the list with a prioritized attack path, now we are going to showcase the attack flow through each component of the system. There are five components in our system, and the attacks that could take place are taking place through or within a component of our system. Figure 7.1 is the diagram of the attack flow that is taking place in the entirety of our system.

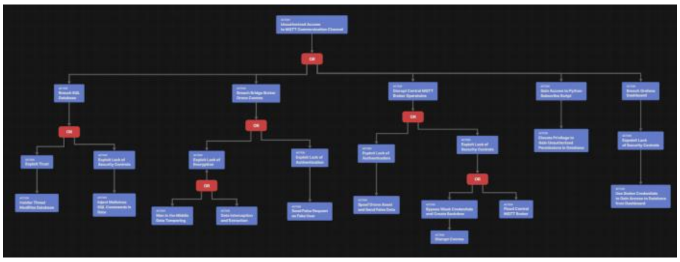


Figure 7.1: Attack Flow

As mentioned previously, there are attacks taking place through each component of our system. First, the component through which attacks will take place is the breach of the SQL database. The entire attack flow is represented in Figure 7.1 and will be further broken down.

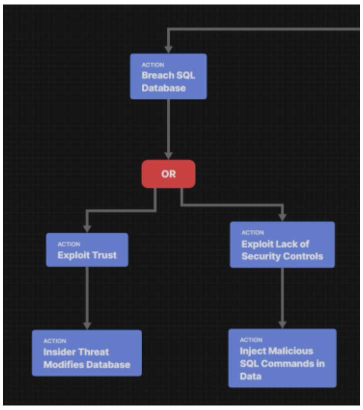

ACTION
Breach SQL Database
OR
ACTION
Exploit Trust
ACTION
Exploit Lack of Security Controls
ACTION
Insider Threat Modifies Database
ACTION
Inject Malicious SQL Commands in Data

Figure 7.2: Breach of SQL Database

In Figure 7.2, the attack flow starts at unauthorized access to the MQTT communication channel. This leads adversaries to breach the SQL database and this action can cause two possible actions, one leading to exploiting trust and then eventually modifying database by being an insider threat or another action leading to injecting malicious SQL command by exploiting the lack of security control that is being placed in the system. The second component is breaching bridge broker drone communications.

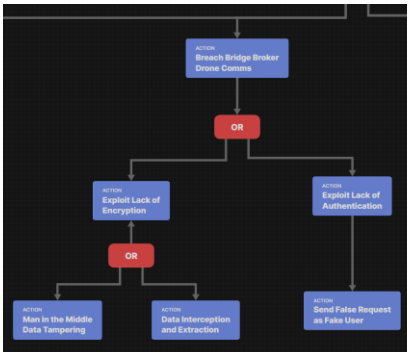

ACTION
Breach Bridge Broker Drone Comms
OR
ACTION
Exploit Lack of Encryption
ACTION
Exploit Lack of Authentication
OR
ACTION
Man in the Middle Data Tampering
ACTION
Data Interception and Extraction
ACTION
Send False Request as Fake User

Figure 7.3: Breach of Bridge Broker Drone Communication

In Figure 7.3, the drone communication channel is getting affected. This is because of the lack of encryption and authentication within the system. This allows the adversary to exploit the lack of authentication and send false requests imitating an actual user. This also allows the adversary to intercept and extract any data that might be flowing within the drone communication channel and tamper with data by becoming an entity in the middle who is overseeing what is being sent and received within the channel, this is done by exploiting the lack of encryption.

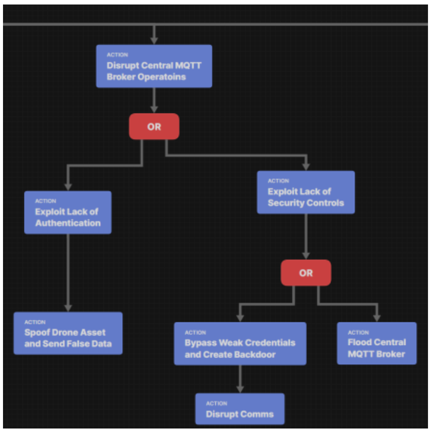

ACTION
Disrupt Central MQTT Broker Operatoins
OR
ACTION
Exploit Lack of Authentication
ACTION
Exploit Lack of Security Controls
OR
ACTION
Spoof Drone Asset and Send False Data
ACTION
Bypass Weak Credentials and Create Backdoor
ACTION
Flood Central MQTT Broker
ACTION
Disrupt Comms

Figure 7.4: Disrupt of Central MQTT Broker Operations

In Figure 7.4 we see that since the central MQTT broker is the gateway that connects our organization with affected users who are outside the organization, disrupting it could cause some major damage to the entire system and organization. Exploiting the lack of security controls in central MQTT broker operations can lead to flooding the broker, making the system not usable at all. It can also lead to adversaries bypassing the weak credentials and create backdoors in the system, which can eventually lead to disrupting communication channels. Tampering and sending false data by exploiting the lack of authentication is also another possible action that can be done by disrupting central MQTT broker operations.

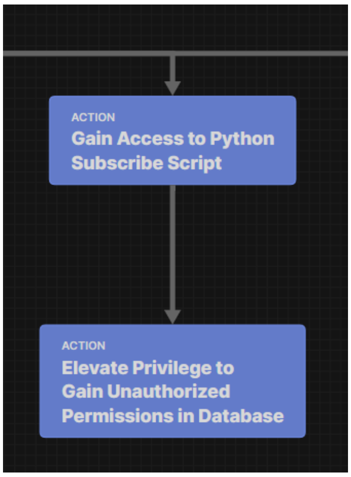

ACTION
Gain Access to Python Subscribe Script
ACTION
Elevate Privilege to Gain Unauthorized Permissions in Database

Figure 7.5: Gaining Access to Python Subscribing Scripts

Gaining access to python subscribing scripts and elevating privilege to gain unauthorized permission within the database is the fourth possible attack flow represented in Figure 7.5 that could happen within our system. This makes python subscribe script our fourth component since adversaries will have direct access to our database, if they are successfully able to execute this attack by gaining unauthorized access to MQTT communication channel.

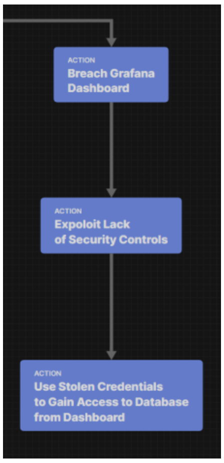
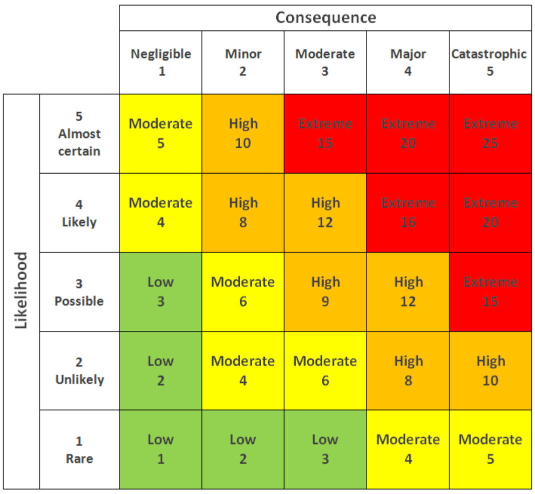

ACTION
Breach Grafana Dashboard
ACTION
Expoloit Lack of Security Controls
ACTION
Use Stolen Credentials to Gain Access to Database from Dashboard

Figure 7.6: Grafana Board Breach

The final component represented in Figure 7.6 is the Grafana dashboard, it is a dashboard which is used to access and modify data from our database by authorized users. However, if an adversary or inside personnel were to gain unauthorized access to the dashboard then they can exploit the lack of security controls that exists in the system and use the stolen credentials to gain access to the database. The only difference is that through this attack flow, the database will think an authorized person is requesting access or modifying data in the database.

G. Stage 7: Risk and Impact Analysis

This is the final stage, stage 7, which lists the potential threats, their likelihood and the proper mitigations for each threat. The likelihood and impact of threats can be used to determine the severity of their risk, which is known as heat map. A heat map is a risk assessment tool that visually represents the severity of risks based on their likelihood of occurrence and the consequence of their impact. Each cell in the matrix showcases a risk level. It goes from low, moderate, high and extreme. To determine which cell a risk falls under, the multiplication of likelihood and consequences score must be calculated. After the calculation is done, the number that comes out as a result will directly correspond to one of the cells from the matrix, determining the risk level. The cells are color coded according to the severity of the risk level, which quickly helps to identify the risk. For example, Red is critical risk and requires immediate action, Orange is high risk and needs to be dealt with as soon as it can be dealt with, Yellow is less severe risk and can be dealt with by using the planned mitigations and Green is low risk and can be monitored, in case the severity changes. This is represented in Figure 8.



		Consequence				
		Negligible 1	Minor 2	Moderate 3	Major 4	Catastrophic 5
Likelihood	5 Almost certain	Moderate 5	High 10	Extreme 15	Extreme 20	Extreme 25
	4 Likely	Moderate 4	High 8	High 12	Extreme 16	Extreme 20
	3 Possible	Low 3	Moderate 6	High 9	High 12	Extreme 15
	2 Unlikely	Low 2	Moderate 4	Moderate 6	High 8	High 10
	1 Rare	Low 1	Low 2	Low 3	Moderate 4	Moderate 5

Figure 8: Heat Map

Since we went over the heat map, now we can dive into proper mitigations for each threat. Mitigations for each threat can be listed using the mitigation pyramid, which determines the hierarchy of hazard controls by dividing mitigation into a layered approach. There are five layers in the mitigation pyramid, and it is designed as an upside-down pyramid, since the width of the pyramid showcases effectiveness for each layer. Even though the bottom of the pyramid is the most effective mitigation, and the tip of the pyramid is least effective, all the layers have their own role to fulfill. The five layers are elimination, substitution, engineering controls, administrative controls and PPE. Elimination being the most effective one and then it goes down to PPE which is the least effective. Starting from least effective, PPE is protecting the workers with personal protective equipment, administrative controls is changes in the work environment as training or working practices, engineering controls is isolating people from the hazard, substitution is replacing whatever produces the hazard and elimination is physical removal of the hazard. Tables 7 through 12 demonstrate mitigations in the mitigation pyramid layer format, for each of the threats that we have, and they are categorized by each component, where the threats are taking place.

Table 7: Android Device Mitigations

Threat **STRIDE Type + Priority**	**Description**	**Mitigations**
Device Identity Spoofing -Spoofing -High	An attacker impersonates a legitimate user's Android device to send false distress signals, misleading drone responses and potentially redirecting rescue resources inefficiently.	- Eliminate unauthorized channels that are used for distress signaling Limit Software Installation (M1033) Application Isolation (D3-AI) - Substitute Android client app with a solid version, which has a built-in identity verifier (e.g., using Google SafetyNet or hardware-backed security modules) Application Developer Guidance (M1013) Application Hardening (D3-AH) - Substitute string distress signals to verified multi-sensor alerts (e.g., GPS) Encrypt Network Traffic (M1009) User Geolocation Logon Pattern Analysis (D3-UGLPA) - Match the location origin with the pattern of user behavior and with already known risk zones (Geofence validation) User Guidance (M1011) User Geolocation Logon Pattern Analysis (D3-UGLPA) - Train CMC dispatcher to recognize and determine unusual or high impact distress calls with abnormal patterns User Training (M1017) User Behavior Analysis (D3-UBA)
False App -Tampering -Medium	An attacker manipulates the user into downloading malware disguised as legitimate on their Android device. The attacker then malicious actions while effectively cutting off the user from the system, negatively impacting system availability.	- Eliminate the support for third-party app installations Limit Software Installation (M1033) Application Configuration Hardening (D3-ACH) - Substitute traditional distribution of apps to secure integrations of app store (e.g., enterprise-managed Play Store with vetted apps)

		Application Developer Guidance (M1013) Application Configuration Hardening (D3-ACH) - Apply code signing verification for all applications that interact with our system Code Signing (M1045) Message Authentication (D3-MAN) - Make a security incident reporting channel, that users can use to report suspicious device behavior or apps User Training (M1017) User Behavior Analysis (D3-UBA) - Provide checklist or security guides to users, to help them determine suspicious apps or links User Guidance (M1011) User Training (M1017)

Table 8: Bridge Broker Drone Mitigations

Threat STRIDE Type + Priority	Description	Mitigations
Interception & Extraction -Information Disclosure -High	A hacker with access to a network traffic monitoring tool can sniff network traffic being sent from affected user to bridge broker or from bridge broker to central broker, which leads to the information regarding affected user status and location being disclosed, negatively impacting the confidentiality of the data.	- Eliminate the usage of unencrypted channels across the entire system completely (clients, bridge brokers, and central brokers) Network Segmentation (M1030) Outbound Traffic Filtering (D3-OTF) - Substitute current unencrypted protocol (unencrypted MQTT) with secure protocol like MQTTS (MQTT over TLS) or HTTPS-based APIs for data transfer Encrypted Channel (T1573) Message Encryption (D3-MENCR) - Usage of certificate pinning and mutual TLS (mTLS) to verify both sides and of communication and prevent any interception by bad actors Encrypt Sensitive Information (M1041) Certificate-based Authentication (D3-CBAN) - Monitor the network logs for sniffing attempts or any suspicious flow of traffic Audit (M1047) Network Traffic Signature Analysis (D3-NTSA) Network Traffic Analysis (D3-NTA)
MITM Tampering -Tampering -Medium	A hacker with the ability to place themselves in between the communication channel of the survivor and bridge broker drone can intercept and alter the data, which leads to inaccurate reports being received by dispatch operators, negatively impacting the integrity of the system.	- Eliminate all unverified or unauthorized communication channels between the user device and bridge broker Network Segmentation (M1030) Network Traffic Filtering (D3-NTF) - Substitute public communication protocols (MQTT or TCP) to encrypted, authenticated ones (MQTT over TLS) or CoAP with DTLS Encrypted Channel (M1040) Message Authentication (D3-MAN)

		- Substitute unstructured, direct communication between each component in the system (user app and drones) to a centralized and secure communication model which will be managed by a trusted gateway User Account Management (M1018) Message Authentication (D3-MAN) Client-server Payload Profiling (D3-CSPP) - Use end to end encryption and sequence verification to avoid undetected message replay or injection Encrypt Sensitive Information (M1041) File Encryption (D3-FE) Message Authentication (D3-MAN) - Requiring security testing and inspection of communication paths Application Developer Guidance (M1013)
Identity Spoofing -Spoofing -High	A hacker with abilities to craft messages that appear to come from a survivor within the area of operation can send false requests, which leads to dispatch operators directing resources to a false report, negatively impacting the reliability and effectiveness of the system.	- Eliminate unauthorized device communication with the system and brokers by terminating the support for unverified message inputs Multi-Factor Authentication (M1032) Agent Authentication (D3-AA) Password Authentication (D3-PWA) - Substitute the usage of ad hoc message formatting to strongly typed, schema validated protocols (eg protobuf with signed payloads) Client-server Payload Profiling (D3-CSPP) - Check geolocation consistently and use temporal anomaly detection model to pinpoint suspicious requests from similar zones DNS Traffic Analysis (D3-DNSTA) User Geolocation Logon Pattern Analysis (D3-UGLPA) - Train dispatchers to have the ability to recognize suspicious repetitive distress patterns User Training (M1017) Job Function Access Pattern Analysis (D3-JFAPA)

Table 9: Central MQTT Broker

Threat **STRIDE Type + Priority**	**Description**	**Mitigations**
DoS on MQTT Broker -Denial of Service -Critical	An unauthorized device repeatedly floods the MQTT broker with distorted or excessive messages, overwhelming system resources and delaying the delivery of legitimate rescue data. This disruption in communication between drones and the command center can critically hinder real-time decision-making and reduce the overall effectiveness of emergency response operations.	- Eliminate public facing interface and non-essential broker endpoints Network Segmentation (M1030) Network Traffic Filtering (D3-NTF) - Substitute standard MQTT with MQTTS (MQTT over TLS) with the addition of token-based access control, which would require signed certificates or tokens for message publishing Encrypt Sensitive Information (M1041)

		Message Encryption (D3-MENCR) - Implement firewalls and intrusion prevention system (IPS) to detect and stop flooding patterns and malformed packets Network Intrusion Prevention (M1031) Inbound Traffic Filtering (D3-ITF) - Implement broker hardening policies (e.g, QoS usage) Software Configuration (M1054) Application Configuration Hardening (D3-ACH) - Create incident response protocols which directly deal with DoS events by including dynamic traffic rerouting and broker failover User Account Management (M1018) Access Policy Administration (D3-APA) - Perform stress and penetration testing to determine the durability MQTT architecture Application Developer Guidance (M1013) Application Configuration Hardening (D3-ACH) - Train administrators to recognize early warning signs of a DoS attack within the system logs User Training (M1017)

Table 10: Python Subscribe Script

Threat STRIDE Type + Priority	Description	Mitigations
Unauthorized DB Access -Elevation of Privilege -High	An attacker who gains control over the data processing script can exploit it to perform unauthorized write operations to the database, leading to the alteration of sensitive records and compromising the accuracy and reliability of stored mission data.	- Eliminate permissions that allow users to directly write from the script to the database and make read only basis, wherever it is possible Restrict File and Directory Permissions (M1022) User Account Permissions (D3-UAP) - Substitute general scripting environments with sandbox environments (e.g., using Docker with limited privileges or AWS Lambda with strict IAM roles) Software Configuration (M1054) Application-based Process Isolation (D3-ABPI) - Implementing role-based access control (RBAC) for database operations, making sure the script can execute specific actions which are allowed for the user User Account Management (M1018) User Account Permissions (D3-UAP) - Implement the practice of requiring peer review or approval from higher users before a script is being sent to the database Application Developer Guidance (M1013) Job Function Access Pattern Analysis (D3-JFAPA) - Make users aware of the risks of improper usage of privilege and

		consequences the user would have to face from the organization User Training (M1017) Job Function Access Pattern Analysis (D3-JFAPA)

Table 11: SQL Database Mitigations

Threat **STRIDE Type + Priority**	**Description**	**Mitigations**
SQL Injection -Tampering -High	A hacker with access to the channel can inject malicious SQL statements into the app's input fields, which leads to the SQL database being open to manipulation, negatively impacting the integrity and confidentiality of the SQL database.	- Eliminate writing SQL queries which directly adds user input within the query Database Query String Analysis (D3-DQSA) - Substitute raw SQL queries practices with prepared statements practice, which separates code from data Database Query String Analysis (D3-DQSA) - Implement server-side input verification and sanitization for user input with the help of allowlists (e.g, regex-based validation) Database Query String Analysis (D3-DQSA) - Place Web Application Firewalls (WAFs) to monitor and block malicious SQL injection Database Query String Analysis (D3-DQSA) - Allow SQL query logging and anomaly detection to recognize patterns of unauthorized modification Audit (M1047) Database Query String Analysis (D3-DQSA) Application Configuration Hardening (D3-ACH) - Perform penetration testing and regular security assessments that focus mainly on injection vulnerabilities Database Query String Analysis (D3-DQSA) - Train users on secure coding principles, give input sanitization cheat sheets and require users to attend security awareness programs for injection detection and mitigation
Insider Threat -Information Disclosure -Medium	A malicious operator with trusted access to the system can manipulate the database and remove entries, which leads to some affected individuals not receiving assistance, negatively impacting the integrity of the SQL database.	- Eliminate shared admin credentials by requiring every user to have their own trackable accounts User Account Management (M1018) User Account Permissions (D3-UAP) - Substitute manual database interaction with role specific dashboards which will allow users to see only what they need to see User Account Management (M1018) User Account Permissions (D3-UAP) - Implement Role Based Access Control (RBAC) to limit who can modify data in the database User Account Management (M1018) User Account Permissions (D3-UAP)

		- Monitor user roles and access patterns to identify any form of anomalies User Account Management (M1018) User Account Permissions (D3-UAP) - Train operators on legal consequences of violating on data integrity Audit (M1047)

Table 12: Grafana Dashboard Mitigations

Threat STRIDE Type + Priority	Description	Mitigations
Unauthorized Access via Stolen Credentials -Elevation of Privilege -High	A malicious operator or infiltrator with a set of stolen credentials can operate the Grafana dashboard and access the SQL database, which leads to sensitive information being leaked, negatively impacting confidentiality of the SQL database.	- Eliminate default admin credentials and make credential rotation on first use a requirement Password Policies (M1027) Credential Rotation (D3-CRO) Strong Password Policy (D3-SPP) - Substitute password only access to Multi Factor Authentication (MFA) for all users Multi-Factor Authentication (M1032) Authentication Event Thresholding (D3-AET) - Implement log monitoring for login attempts, data queries and dashboard access Audit (M1047) User Behavior Analysis (D3-UBA) - Implement Role Based Access Control (RBAC) with Grafana to set appropriate boundaries for each user, limiting the capabilities of dashboard users with respect to their job roles User Account Management (M1018) User Account Permissions (D3-UAP) - Regularly monitor the activity of dashboard permissions and user roles Audit (M1047) User Behavior Analysis (D3-UBA) Operating System Monitoring - D3-OSM - An approval process, where higher roles and new users are approved by multiple reviewers User Account Management (M1018) Access Policy Administration (D3-APA) - Train users on how to use Grafana in a secure manner, report suspicious activity and phishing awareness M1017 – User Training

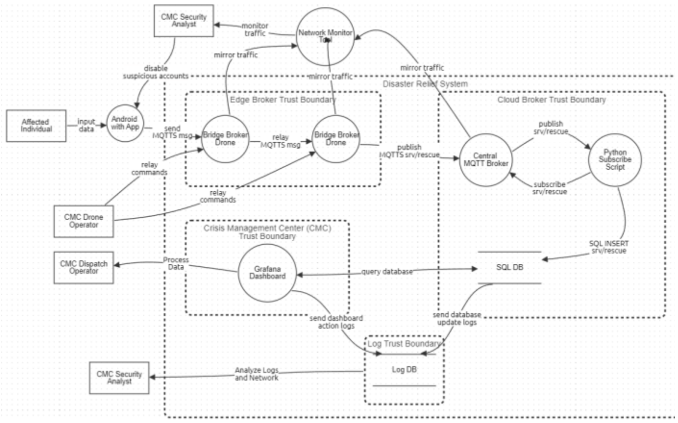


Figure 9: DFD with Mitigations

Following application of the above proposed mitigations, the DFD has been updated to reflect some implementation of the previously mentioned mitigations shown in Figure 9. While not all mitigations can be visualized in a DFD, employee training or access controls for instance, the revised DFD has implemented log and network monitoring mechanisms. Logging CMC operator actions taken on Grafana dashboard and tracking SQL database changes is crucial to maintain accountability within the system. Logging actions taken within the SQL database ensures that integrity and confidentiality are maintained in the face of both internal and external threat actors. A critical attack vector identified in the proposed system is the communications channel running between the affected individual's Android and the central MQTT broker. MQTT broker drones receiving messages from untrusted external entities within the area of operation should be considered open to hostile contact. Subsequently the central MQTT broker also has a line of connection to any untrusted external entities. Due to this, the DFD reflects the addition of a network monitoring tool provided by a third party, such as Splunk or Wireshark. They would be considered a third party trusted vendor and would assist in detecting anomalous activity. As a result of these mitigations, CMC security analysts would have to be hired or contracted to perform network monitoring, log monitoring, and account administration.

The mitigations that could not be visualized in the DFD are user training, user guidance, user account management, and user account permissions. Although they could not be explicitly represented within the DFD, these mitigations are essential for enhancing the system's overall security posture. To mitigate social engineering attacks, it is vital to train users to avoid such attacks. To improve human-layer defenses, dispatch operators and system administrators are trained to spot spoofing irregular telemetry data, or early warning signs of DoS attacks. As for user guidance, it is crucial for users to differentiate between trustworthy and suspicious behaviors, particularly in high-stress emergency scenarios. RBAC, implemented by user account management and user account permissions, is another non-DFD control. It restricts user rights based on their operational tasks and guarantees that they have minimal access. To stop insider threats and privilege elevation, access to vital interfaces such as the database, Grafana dashboard, and MQTT broker is strictly limited, audited, and tracked. Thus, although not being shown in the DFD, these controls are crucial for maintaining operational integrity and lowering residual risk in both the business and human domains.

The technique of identifying the amount of risk that a system still has after safety mechanisms are put in place is called residual risk analysis. It helps organizations to decide if additional action is required or if the remaining risk is acceptable. Residual Risk = (Vulnerability{p1} × Threat Likelihood{p2} × Impact) / Countermeasure Effectiveness, is the formula used to calculate residual risk in the analysis. Threat likelihood is the possibility that a threat actor will take advantage of the vulnerability, impact is the possible harm that could result from the threat developing, vulnerability is the likelihood that a weakness exists and can be taken advantage of, and countermeasure effectiveness is the strength of the security measures that have been put in place (such as TLS encryption, authentication protocols, or anomaly detection).

Using the Common Vulnerability Scoring System (CVSS) calculator by National Vulnerability Database (NIST), we were able to get the vulnerability and threat likelihood value. We achieved this by taking our attack scenarios and putting in the appropriate factors from the scenarios to the calculator to find p1 and p2 and look into the scoring system that exists within each vulnerability. After we get out p1 and p2, we then get our impact value and put it in the equation, (Vulnerability{p1} × Threat Likelihood{p2} × Impact) / Countermeasure Effectiveness, which then provides us with the residual risk scores, respectively. This is iterated on within Table 13.

Table 13: Residual Risk Analysis

Threat	**Vulnerability (p1)**	**Threat Likelihood (p2)**	**Impact**	**Residual Risk Score**	**Risk Level**	**Likelihood**
Device Identity Spoofing (Android App)	0.8	0.7	5	5.49	High	High
False App (Android App)	0.75	0.7	4	5	Medium	High
Interception and Extraction (Bridge Drones)	0.9	0.65	5	4.05	Medium	High
MQTT Spoofing (Central Broker)	0.85	0.75	5	7.14	High	High

SQL Injection (SQL DB)	0.9	0.6	4	2.82	Medium	High
Unauthorized DB Access (Python Script)	0.75	0.7	4	3.09	Medium	High

V. FINAL REMARKS

This project set out to build a clear, practical understanding of the security risks facing a disaster relief communication system, designed to keep working even during crisis situations. The system spans everything from Android devices used by people in need, all the way to a central hub made up of drone-based data relays, a message broker (MQTT), back-end services written in Python, an SQL database, and a Grafana dashboard to give emergency teams a real-time view of the situation.

Using industry-standard threat modeling techniques such as STRIDE and PASTA, we carefully mapped out potential threats at every level, from big-picture strategic risks to more hands-on operational and technical ones. We then assessed those risks using DREAD scoring, visualized the results with heat maps, and found that many parts of the system are exposed to serious threats. The Denial-of-Service (DoS) attack on the central MQTT broker stood out as especially critical. However, every layer of this system needs strong security measures to ensure it can be relied on during a real emergency. Diving deeper into the system's components revealed specific weak spots. For example, the Android devices that people use to send distress signals are vulnerable to identity spoofing and fake apps. If an attacker exploited these, they could send out false alerts, possibly wasting rescue efforts. The drones acting as communication bridges are also at risk—they could be intercepted or impersonated, potentially feeding the system with false data and throwing off the big-picture view of what's happening on the ground. The MQTT broker proved to be the most vulnerable. If it's knocked offline or manipulated through misconfigurations, it could cripple the entire communication network.

Even the parts of the system that operate behind internal firewalls, like the back-end scripts and database, aren't immune. They face threats like SQL injection or unauthorized access, which could compromise data accuracy. And if the Grafana dashboard is breached, it could expose sensitive data or mislead response teams with inaccurate insights.

Given the severity of these findings, the next steps are clear. Securing communications is priority number one. That means replacing any unencrypted channels with secure alternatives like MQTTS, using mutual TLS and certificates to protect data. Stronger credential management, rotating passwords regularly and using multi-factor authentication across all critical components can drastically reduce the risk of unauthorized access. We also recommended system hardening removing default settings, eliminating insecure protocols, and watching network traffic closely for anything out of the ordinary. In short, defense-in-depth needs to be more than a buzzword, it has to be built into every part of this system. Of course, it's important to acknowledge the limitations of this study. The system is complex and spread out, so the threat model can't capture every real-world scenario. In a crisis, implementing secure protocols might introduce delays or extra steps, creating trade-offs between security and speed. Also, the risk scores we used rely on estimated values that will need to be fine-tuned with real-world testing and updated threat intelligence.

This analysis provides a solid foundation for improving the system's security posture. Prioritizing encrypted communications, better credential handling, and continuous monitoring will go a long way in making the system more resilient. Just as importantly, we suggest creating training programs for everyone involved - from drone operators to crisis managers - to reduce human-related vulnerabilities like social engineering. And security needs to be an ongoing process. Regular updates to threat models and risk assessments are critical for keeping up with new attack methods.

In summary, this work offers a realistic and actionable security roadmap for disaster response systems. It blends strong technical defenses with practical steps for improving system resilience over time. While there are still challenges to overcome, the framework we've outlined shows that by continuously adapting to new threats, it's possible to maintain secure, reliable communication.

VI. REFERENCES

[1] OCHA, "Relief Web Topics," ReliefWeb, 2024. [Online]. Available: https://reliefweb.int/.

[2] AP News, "Floods devastate Nigeria, Niger, Chad, Mali amid intense rains and climate concerns," AP News, Mar. 2024. [Online]. Available: https://apnews.com/article/floods-nigeria-niger-chad-mali-rains-climate-change-397b303cdae17ef2a0076192c8c908ac.

[3] The Guardian, "Much of West without internet after undersea cable failures," The Guardian, Mar. 14, 2024. [Online]. Available: https://www.theguardian.com/technology/2024/mar/14/much-of-west-and-central-africa-without-internet-after-undersea-cable-failures.

[4] "Tonga volcano: Internet restored five weeks after eruption," BBC News, Feb. 22, 2022. [Online]. Available: https://www.bbc.com/news/world-asia-60474362.

[5] Rochester Institute of Technology, "Cybersecurity of MQTT-based drone communications in disaster scenarios," Thesis, Rochester Institute of Technology, Rochester, NY, USA, 2023. [Online]. Available: https://repository.rit.edu/cgi/viewcontent.cgi?article=12719&context=theses.

[6] Bevywise Networks, "MQTT Security Best Practices," Bevywise Networks, 2023. [Online]. Available: https://www.bevywise.com/blog/mqtt-security-best-practices/.

[7] Tony UcedaVelez; Marco M. Morana, "Intro to Pasta," in Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis, Wiley, 2015, pp.317-342, doi: 10.1002/9781118988374.ch6.

[8] Nick. (2025, February 9). PASTA Threat Modeling - Threat-Modeling.com. Threat-Modeling.com. https://threat-modeling.com/pasta-threat-modeling/

[9] MITRE ATT&CK Framework. MITRE Corporation. Available: https://attack.mitre.org

[10] MITRE CAPEC – Common Attack Pattern Enumeration and

Classification. MITRE. https://capec.mitre.org

[11] MITRE CWE – Common Weakness Enumeration. MITRE. https://cwe.mitre.org

[12] MITRE CVE – Common Vulnerabilities and Exposures. MITRE. https://www.cve.org/