segment

Introduction to Threat Modeling

Updated March 2021

1

Welcome to our introductory course into Threat Modeling!

The Threat Modeling training is a requirement for everyone within Engineering. Please note that our Engineering teams lead the threat modeling efforts at Segment and there is an expectation that all engineers can actively participate in the threat modeling sessions.

There are three different workshops that we have prepared

- The current workshop is to introduce the concept of threat modeling
- The second workshop adds some more theory, but it will be considerably more hands on
- The final session will be a live threat model of a feature that your team owns or is familiar with

Agenda

- Introduction
- Why Threat Model?
- Process overview
- Deep dive of threats (STRIDE)
- Takeaway

segment

Today we are going to introduce you to the Threat Modeling workflow and do a deeper dive into how Security discovers risks.

Important notes about the workshop:

- The workshop is a collaborative session, the more you participate, the more all of us learn
- There are lots of questions and places to collaborate in this workshop, it is important that you pay attention, you may get selected at random to answer a question
- It is also important that the attendees speak up if there are unfamiliar terms used, we Security folks have tend to use academic terms or jargon, please call us out on that

Goals

- Get familiar with Threat Modeling
- Understand Threat Modeling workflow
- Learn about different types of vulnerabilities in systems
- ~~Become a Threat Modeling legend~~

What are the goals for this training?

Goals:

- Get familiar with the concept of threat modeling, what is it, how can we use it?
- Get used to the threat modeling workflow, how do we actually do it?
- Understand the different classification of vulnerabilities and how security engineers discover them

We are not trying to make you a threat modeling legend, that will come over time. The goal is for you to have a much better understanding and to have you lead the threat modeling sessions, we want to make sure that you are comfortable with discovering threats. Each journey begins with one step and today we are taking that first step.

What is Threat Modeling?

The main goal of threat modeling is to **determine the assets** in the system and how we are going to **protect them**.

segment

Why does Security Engineering threat model?

We use Threat Modeling as a tool to uncover the assets in the system and figure out the ways that an adversary might try to attack those assets. Once we understand that, we think of different ways that we can protect our assets.

Security Engineering believes that Threat Modeling should be simple, it should be transparent and the process should be democratized, everyone needs to have a voice because they see the system differently. To the last point, we cover why diversity is extremely important in Threat Modeling more in part two of the training.

Surprise, you already Threat Model today!

Most people call Threat Modeling “personal safety”, but it is the same thing. Threat Modeling is personal safety for software.

Let’s go through a scenario, how have you changed your grocery shopping habits due to COVID-19?

Side note:

- The point of asking the above question(s) is to have the attendees converse and provide their thoughts, this is a light icebreaker, but more importantly, you need them to start thinking about risks
- When folks provide their answers, please follow up with questions and dig into why they changed their behaviors, try to get them to talk about their behavior changes with respect to risks

Threat Modeling Benefits

- Address risks before they are created in system
- Cheapest way fix vulnerabilities
- Great document for assumptions and risks
- Great tool for security researchers testing our application for vulnerabilities

segment

We talked about what is Threat Modeling, but we haven't addressed why it is beneficial to the organization.

Threat Modeling is important because we can embed security right into the design of the system. We are adding security defences and addressing security vulnerabilities before a single line of code is written. This is important because we can do security for "cheap", it is way more expensive to fix security vulnerabilities when a security researcher discovers them in production, especially if we have to re-architect the solution or if the original engineer is no longer here at the organization.

Many times we don't document our assumptions or risks without Threat Modeling. Having a document with all of our assumptions is great because aim to one day provide our threat modeling documents and artifacts to security researchers to validate our assumptions. It will make the penetration testing engagements even more beneficial.

Threat Model Steps

- Breaking down the feature
- Find threats
- Prioritize threats
- Mitigate threats

We break down Threat Modeling into four phases:

- Break down the feature - basically this is our Software Defined Document (SDD), written documentation on what the system is, there are diagrams in there, all of the details we need to discover threats
- Find threats - in this phase, we review the documentation and work as a team to discover the different threats to the system
- Prioritize threats - As a team, we will review the threats and prioritize them
- Mitigate threats - Finally we figure out which of the threats we are going to address and go ahead to address them

Scenario

A long lost relative passed away and left everything to you. With that money you bought this beautiful home. We don't know what the previous owners' threat model and how they think about safety, so let's threat model the safety of your new house.

- What is the asset we are trying to protect?
- Do you have any immediate concerns?

Let's go through a scenario and better understand the different Threat Modeling phases. In this scenario, we are going to shop for a home, but before we purchase the home we want to Threat Model it for our safety.

A long lost relative passed away and left everything to you. With that money you bought this beautiful home. We don't know what the previous owners' threat model and how they think about safety, so let's threat model the safety of your new house.

Side notes:

- Relate the people and items within the home back to software data within a feature, also relate the home itself to the infrastructure - get your attendees to relate Threat Modeling to personal safety
- The second question is great to remind folks how important diversity is for Threat Modeling, hopefully there are many different immediate concerns about the home.

What is the asset we are trying to protect? (have attendees answer this)

- People in the home
- Items within the home
- The home itself

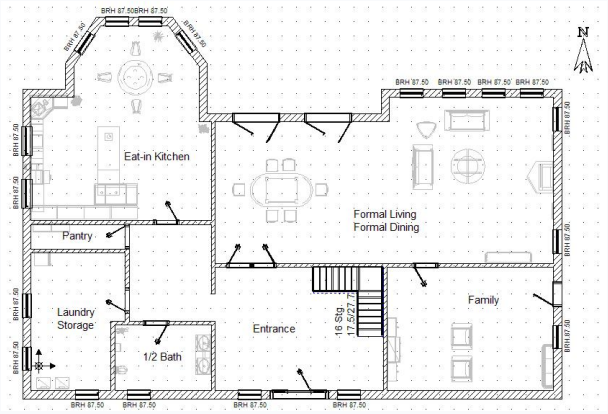
What concerns do you have? (have attendees answer this)

- Accessible windows on the ground floor
- Garage access

- Woods in the back, adversary might come from there
- Woods in the back, one might fall on the home
- Woods in the back, forest fires
- Lack of lights and fencing around the home
- Possible for natural disaster area (tornado, earthquake, floods, etc)
- etc

1. Breaking Down the Feature

- What was built?
- How can I enter/exit the system?
- Where can it be more clear?



Breaking down the feature means that we are taking in the requirements and creating a technical design for the requirements. Whenever we build a Software Defined Document (SDD), it is important that the readers of the document fully understand what is being built, similar to the blueprint of a house. We need this information to understand threats to the design.

These are good questions to ask the attendees:

How can someone enter or exit this home (reasonably)? Relate this question back to data coming in and out of your system.

- Windows, the doors leading outside

Where can this diagram be more clear? Relate this question back to making sure everyone's questions are answered in your SDD.

- Where do those stairs lead to?
- Is there an easy way for someone to gain access at a different house level?
- There looks like a door leading out of the family room, etc
- What protections are there at the windows, doors, etc?

2. Finding Threats

We want to protect the house and the items inside of it. What are all of the risks associated with this home? Let's find the threats.

- Who are we concerned about?
- How can they gain access to the asset?
- What can go wrong?

Once we have a strong understanding of the feature (or this home), we can start to look for threats.

This is a good place to talk a bit about the different types of attackers that can harm your home. There can be racoons, bears, rats or even humans that attack your home. With respect to humans, there are various types of human attackers, kidnappers, burglars, arsonists, thieves, etc.

For this exercise, we are going to stick with thieves. Thieves are folks that want to steal your things, but don't want to hurt anyone.

Understanding the attacker can help determine what can go wrong. If we are worried about a burglar maybe they will be brazen enough to steal things while we are in the home and we will need a different set of protections. If we are worried about a thief, perhaps we need to worry about our things rather than the people in the home.

This is similar to Threat Modeling our application. If we are concerned with script kiddies, we will consider certain security defenses versus if we are concerned with various nation states attacking us.

We are going to choose the four threats for the group

- Breaking in through our unlocked windows
- We hired a security guard to be at our home when we go on vacation, concern is that we only pay them minimum wage and they might be bribed to let

- someone in
- We have a cheap lock for our doors
- We have a simple garage door opener

3. Prioritizing Threats

Prioritize your threats.

- Going through unlocked window
- Bribing a Security Guard
- Picking a lock
- Breaking into garage

Let's prioritize the four threats. Which would be your highest priority and what is your lowest priority?

Side note:

- This is another slide in which we will collaborate with the attendees and ask one attendee to prioritize the above risks
- Do not provide them any guidance, the goal is to talk about how they decided to rank these risks
- What are the different ways that they prioritize risks? Severity, ease of fix, ease of exploit, etc

At Segment, we prioritize threats by looking at the weakness of the system.

4. Mitigating Threats

So now we know our asset is our home and the threat actor is a thief, how do we want to protect the home?

- What can we do to protect the home?
- Which controls will be implement?
- Do we have any constraints?

Threats

- Going through unlocked window
- Picking a lock
- Breaking into garage
- Bribing a Security Guard

Let's talk about mitigating threats and sometimes that challenges we face with mitigating threats.

We have a mitigation constraint, we only have $1000 dollars for initially investing in mitigation techniques

- Constraints happen all the time in the software world
- Time constraints - we might not have enough time to implement controls in our software system
- Money constraints - perhaps we don't have enough money to hire a vendor or we can't add more resources to a project
- Requirements constraints - perhaps we have the money and time, but there may be another type of constraint that doesn't allow us to implement the appropriate the security defenses

There is also a balance that we need to implement when securing a system. If we wanted perfect security for Segment's application, we would pull it off the internet and drop it at the bottom of the sea with a lot of sharks that have lasers. Unfortunately that doesn't work for business because we still need clients to access our system. When mitigating threats to our system we have to better understand balance.

Question for attendees: Does it make sense to spend $100,000 on security for Fort Knox?
Question for attendees: Does it make sense to spend $30M on security for the home you are purchasing?

We need to budget how we do security appropriately, we might not be exactly happy about spending $1000, but the great thing about Threat Modeling is that we can always revisit the system in the future and implement more controls. Think about software that way as well, we can't always afford to implement all of the security controls and we need to find that right balance for the organization. We can revisit the system again later.

For our home, we have $1000 to spend on upgrading the home security, so how are we going to secure the home?

Ordered threats

- We lock our windows when we leave the home - family can do this for free
- Picking a lock - we found locks that we love for $150 each, four doors means it is going to cost us $600
- We only have $400 to secure the garage, but the ideal garage securing mechanism will cost us $1200

We have expensive tools that we want to protect from within the garage, what can we do to reduce the risk?

Side note:

- The goal of the last question is to have someone suggest moving the tools from the garage into the home
- We can relate that back to attack surface and reducing the attack surface of our system and we re-used some controls (strong locks on the doors)
- Someone might be able to get into your garage (or system), but there isn't anything important in there for them to take

We are done

You have walked through the Threat Modeling process. Let's add everything back to your SDD. SDD should now include:

- Description of system
- Architectural diagrams
- Assumptions made
- Threats discovered
- Prioritization of threats
- Mitigations for risks to address
- Comments for risks not addressed

The Security team expects that each Software Defined Document (SDD) to be robust and have all of the information needed in order to do a proper Threat Model. After the Threat Model is completed, there should be artifacts for everyone to review.

Switching gears now, we talked about the Threat Modeling process, but how do we actually find threats?

In order to Threat Model, you need to understand the different threats.

S Spoofing

T Tampering

R Repudiation

I Information Disclosure

D Denial of Service

E Elevation of Privilege

segment

In late 1990s and early 2000s, things were changing at Microsoft, employees started to understand Security better, but it was in 2002 that Bill Gates wrote his Trustworthy Computing memo and changed Microsoft and the software industry.

2001 was a very difficult year, 9/11 happened and there were several worms Windows XP that were embarrassing for Microsoft. Bill Gates understood that people were nervous about their safety and that everyone wanted strong security of critical infrastructure. In his trustworthy memo, Bill Gates talked about Microsoft needing to change and show everyone that they are trustworthy as a platform.

Microsoft's impact on the Security world is still being felt today, there were very brilliant security folks out of Microsoft that literally changed how we do Security. There were folks that built out the Threat Modeling workflow that we discussed and there were folks that came up with STRIDE. Even after all of these years, these systems and processes are still in use.

What is STRIDE? It is an acronym used to help us uncover threats to our system. Let's go through them individually.

Spoofing

A situation in which a person or program successfully identifies as another by falsifying data, to gain an illegitimate advantage.

What is Spoofing? It is a situation in which a person or program successfully identifies as another by falsifying data, to gain an illegitimate advantage.

How can spoofing be used in real life? If we think about the housing example we had earlier, perhaps a bad actor found your keys and used that to get into the house or they were able to scan your garage door code to get in.

Is there a way to relate these sorts of attacks back to software? There are many different ways that someone can use a spoofing attack on our software. Some examples include:

- Brute force attack, the attacker can be aware of a user in the system and they try one credential after another to try to gain access into Segment
- Credential stuffing attack, a little bit more sophisticated than a brute force, an attacker uses a list of username/passwords that were disclosed in a different breach to try to gain access
- It is possible that if an attacker can gain access to our codebase, we may have mistakenly left secrets in the codebase and they can use them to gain access to the application or even bypass authentication controls
- May the attacker used default credentials that were never updated in the system

Side note:
This is a great time to tell a story about a spoofing vulnerability, one such example is MongoDB default credentials vulnerability

Spoofing at <Organization>

Security Tickets

- <Organization ticket related to spoofing>
- <Organization ticket related to spoofing>
- <Organization ticket related to spoofing>

Mitigations

- Passwords
- Multi-factor authentication
- Digital signatures
- <Solution specific to Organization>

segment

Organizations are not immune to spoofing vulnerabilities.

<talk about specific organizational vulnerabilities, ideally give an example that you have encountered in your career>

There are many mitigations for spoofing vulnerabilities. Ensuring that you have strong authentication, passwords or multi-factor authentication are good ways to prevent or reduce spoofing vulnerabilities. Reactive controls are also important, make sure that you have strong application logs around authentication.

Scenario: Add User API

One of our clients asked if they can add a user into the application via API. Since this was a very big client, we said yes and the Engineering team is going to build out the functionality.

Engineering team did the right thing and reached out to you to help evaluate the security posture of the functionality and to see if they needed to do anything to make the functionality more secure.

Let's look to see if there could be any problems.

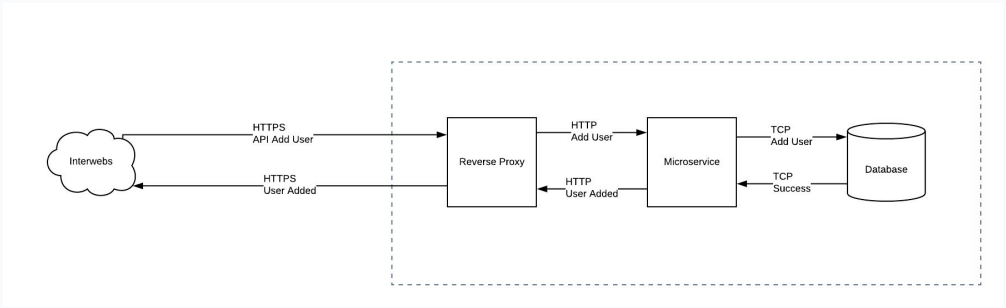
Let's go through an example and try to find some spoofing vulnerabilities.

<read the slide>

Spoofing — Add User API

A situation in which a person or program successfully identifies as another by falsifying data, to gain an illegitimate advantage.

Where can things be spoofed? Where can we use a lost/stolen house key?



Let's look for spoofing vulnerabilities in this system.

<walk through the diagram>

What type of spoofing or authentication-related questions do you have? Please note that this system is purposefully vague because we want to spur conversations.

Tampering

Interfere with something in order to cause damage or make unauthorized alterations.

What is Tampering? Interfere with something in order to cause damage or make unauthorized alterations.

Is there a way to relate tampering attacks back to software? There are many different ways that someone can using tampering to attack on our software. Some examples include:

- Modifying requests using tools like Burp Suite to bypass client side controls
- Perhaps there is a SQL injection vulnerability that an attacker can use to modify data in the database
- Editing authentication application logs to cover your tracks

Tampering at <Organization>

Security Tickets

- <Organization ticket related to tampering>
- <Organization ticket related to tampering>
- <Organization ticket related to tampering>

Mitigations

- Do not trust information coming from the client
- Client validation is great for UX, but have server-side checks to validate data
- Ensure that there are strong role permissions (least privilege)

Organizations are not immune to tampering vulnerabilities.

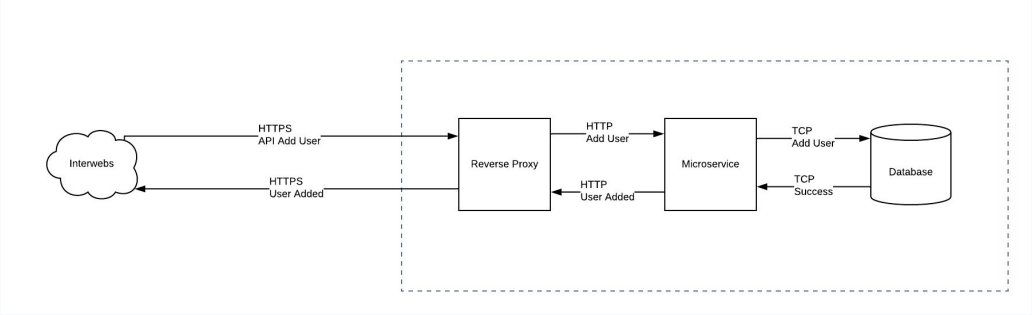
<talk about specific organizational vulnerabilities, ideally give an example that you have encountered in your career>

There are many mitigations for tampering vulnerabilities. Input validation is a great way to spend your efforts. You should understand the nature of data that is coming back. Never rely on client-side checks, they are great for the user's experience, but we need to ensure that we have server side checks to validate the data. Use least privilege whenever possible, if an adversary were to get past our checks, we want to make sure that there is a small blast radius.

Tampering — Add User API

Interfere with something in order to cause damage or make unauthorized alterations.

Where can things be tampered with? Where can we pick a lock, smash a window?



Let’s look for tampering vulnerabilities in this system.

What type of tampering questions do you have?

Repudiation

The ability of denying that an action or an event has occurred.

What is Repudiation? The ability of denying that an action or an event has occurred.

Is there a way to relate repudiation attacks back to software? Here are some examples:

- Not creating any access or application logs at all, client may state that they didn't execute an action in our system when they clearly did
- Need to ensure that users don't use shared username/passwords to access an account in our system
- An attacker deleting log messages because they lack strong integrity controls (both tampering and repudiation)

Side note:
This is a great time to tell a story about a repudiation vulnerability. Talk about a famous attack where the adversary was able to delete authentication logs after they gained access to the system and were slowly able to gain access to production. The adversary were smart because they used IP addresses from the city that the company was located in. They accessed the systems during business hours.

Repudiation at <Organization>

Security Tickets

- <Organization ticket related to repudiation>
- <Organization ticket related to repudiation>
- <Organization ticket related to repudiation>

Mitigations

- Log user actions within the application (who, what, when)
- Ensure that the logs are secure, not easy for adversary to overwrite
- Audit the logs and ensure that they are reviewed on a regular basis

Organizations are not immune to repudiation vulnerabilities.

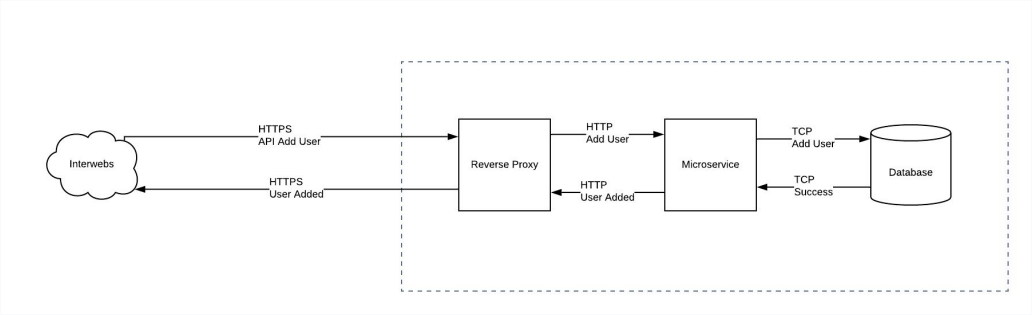
<talk about specific organizational vulnerabilities, ideally give an example that you have encountered in your career>

There are many mitigations for repudiation vulnerabilities. It is critically important that we are logging the right things in our application logs. Make sure that we know who performed the action, what the action was and when they performed it. This will greatly help understand what happened. Use the tooling built for application logging, we ensure that our logs are secure and difficult for an adversary to read or overwrite. Another control is to review your logs on a regular basis, see how users are interacting with your feature, make sure nothing surprises you.

Repudiation — Add User API

The ability of denying that an action or an event has occurred.

Where can things be repudiated? Where can we lie about not being in the system?

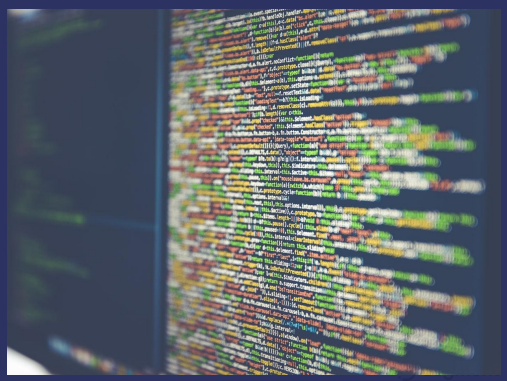


Let’s look for tampering vulnerabilities in this system.

What type of repudiation questions do you have?

Information Disclosure

Exposing information to someone not authorized to see it.



What is Information Disclosure? Exposing information to someone not authorized to see it.

Is there a way to relate information disclosure attacks back to software? Here are some examples:

- An open S3 bucket
- Making a private GitHub repo public by accident
- SQL Injection attack where you can exfiltrate data
- The HTTP protocol

Side note:
This is a great time to tell a story about an information disclosure vulnerability. Talk about the time that your favorite cloud security instructor discussed their mistake and they accidentally published a secret in their GitHub repo and how an adversary was able to take advantage of that mistake.

Information Disclosure at <Organization>

Security Tickets

- <Organization ticket related to information disclosure>
- <Organization ticket related to information disclosure>
- <Organization ticket related to information disclosure>

Mitigations

- Store as little data that you need to do your job
- Encrypt sensitive data
- Ensure that only authorized parties can see your data

Organizations are not immune to information disclosure vulnerabilities.

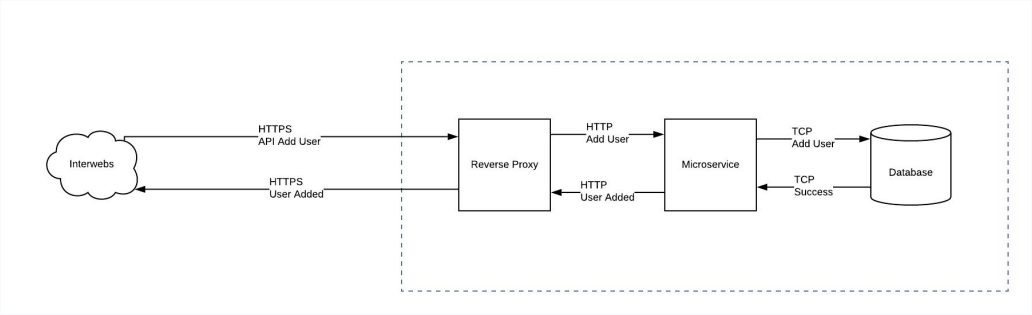
<talk about specific organizational vulnerabilities, ideally give an example that you have encountered in your career>

There are many mitigations for information disclosure vulnerabilities. I am a big fan of the Privacy team and their mantra is to store as little data that you need to do your job. After your job is done, get rid of the data (when possible). If we don't have sensitive information in our systems, then we have less to worry about. If we do have sensitive information, look to encrypt (or hash) whenever reasonable. It can be difficult, but make sure that when you build new features, understand who needs to know and see that information. Deny everyone else access to it.

Information Disclosure — Add User API

Exposing information to someone not authorized to see it

Where can information leak?



Let’s look for information disclosure vulnerabilities in this system.

What type of information disclosure questions do you have?

Denial of Service

Deny or degrade service to users.

What is Denial of Service? Deny or degrade service to users.

Is there a way to relate denial of service attacks back to software? Here are some examples:

- Filling up the file system with files until there is no more disk space
- Keep opening http connection and hold them open as long as possible, the adversary is trying to exhaust the number of threads available
- Application Denial of Service, which is hitting an endpoint that consumes a lot of CPU, the adversary is trying to “peg” the CPU and not let any other requests be processed
- Distributed Denial of Service, which is having so many requests coming in that overwhelm the server

Side note:
This is a great time to tell a story about a denial of service vulnerability. Talk about the time many companies ago the bug bounty researcher uploaded a zip bomb and it caused the hard drive to fill up quickly.

Denial of Service at <Organization>

Security Tickets

- <Organization ticket related to denial of service>
- <Organization ticket related to denial of service>
- <Organization ticket related to denial of service>

Mitigations

- Permission checks
- Rate limits
- Quotas

Organizations are not immune to DoS vulnerabilities.

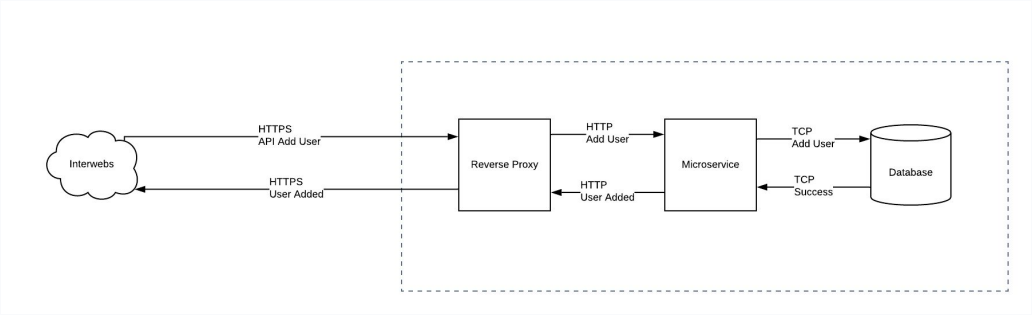
<talk about specific organizational vulnerabilities, ideally give an example that you have encountered in your career>

There are many mitigations for denial of service vulnerabilities. Some of the mitigations are not within the application itself, but the ones we can build within the app are related to rate limits, permission checks and quotas. Think about how often someone needs to perform an action, adding a user to the system. Should they be able to add 100 users/second? What types of limits should we put?

Denial of Service - Add User API

Deny or degrade service to users

Where can we deny service?



Let’s look for denial of service vulnerabilities in this system.

What type of denial of service questions do you have?

Elevation of Privilege

Elevation of Privilege refers to gaining access that one should not have.

What is Elevation of Privilege? Refers to gaining access that one should not have

Is there a way to relate elevation of privilege attacks back to software? Here are some examples:

- Lower privileged users typing in webpages in the browser (/managerusers) and being able to perform admin attacks
- Lower privileged users sending DELETE requests to REST APIs and the API doesn't validate the request
- Using server side request forgery to access a microservice that have no authentication mechanism

Side note:
This is a great time to tell a story about an elevation of privilege vulnerability. Talk about the time at a previous company where a bug bounty researcher setup a script to look for any new endpoint that we would add to the application and how they would run authorization checks on them. Most of the time there would be no authorization checks and the researcher made a lot of money.

Elevation of Privilege at <Organization>

Security Tickets

- <Organization ticket related to elevation of privilege>
- <Organization ticket related to elevation of privilege>
- <Organization ticket related to elevation of privilege>

Mitigations

- Permissions checks
- Input Validation
- Reactive measure - good application logging

Organizations are not immune to elevation of privilege vulnerabilities.

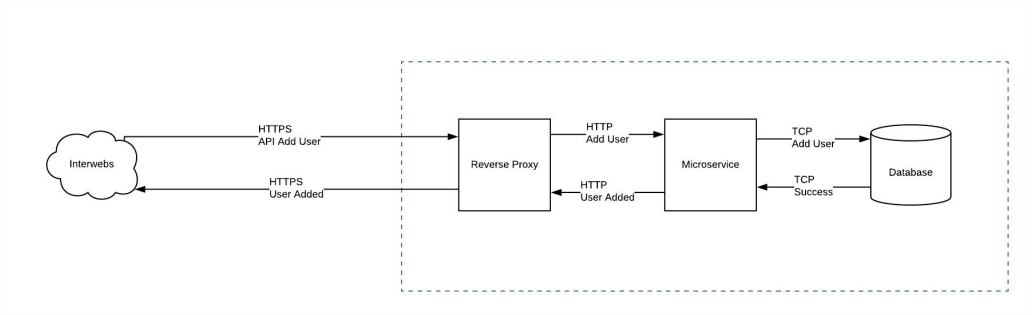
<talk about specific organizational vulnerabilities, ideally give an example that you have encountered in your career>

There are many mitigations for elevation of privilege vulnerabilities. The most basic of mitigation is to check if the current user is allowed to perform the action. A great reactive measure is to have robust application logging. If our Incident Response teams needs to dig deeper and understand what is happening in a system, having robust application logs will help.

Elevation of Privilege - Add User API

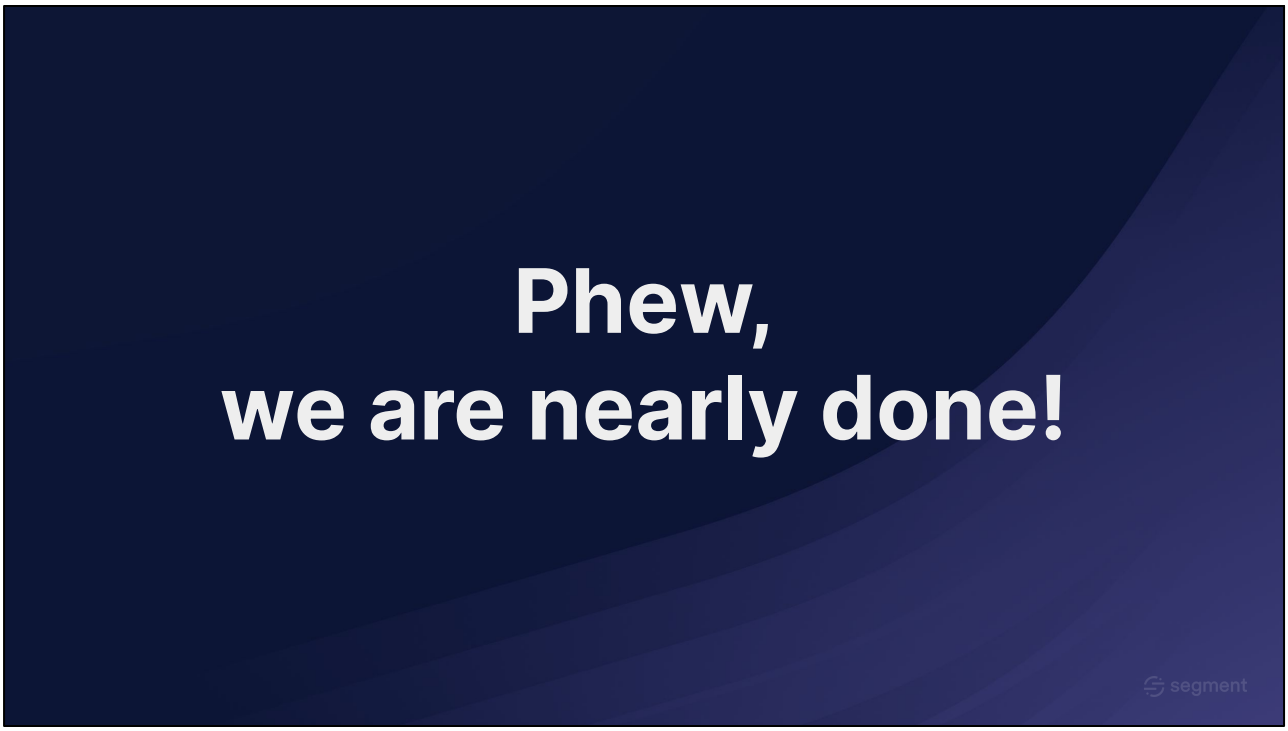
Elevation of Privilege refers to gaining access that one should not have

Where can we elevate our privileges?



Let’s look for elevation of privilege vulnerabilities in this system.

What type of elevation of privilege questions do you have?



Switching gears now, we talked about the Threat Modeling process, but how do we actually find threats?

Goals

- Get familiar with Threat Modeling
- Understand Threat Modeling workflow
- Learn about different types of vulnerabilities in systems
- ~~Become a Threat Modeling legend~~

Let's recap, are we know more familiar with Threat Modeling and the Threat Modeling workflow?

Threat Modeling workflow

- Breaking down the feature
- Finding threats
- Prioritizing threats
- Mitigating threats

Do you understand how Security Engineering discovers vulnerabilities during a Threat Modeling session? Do you remember what STRIDE means?

segment

Part 1 of 3 Complete!

GitHub Repository for Trainings

Training Part 1

Training Part 2

Training Part 3

Side note:
Good place to drop the Trustworthy Computing memo link:
https://www.wired.com/2002/01/bill-gates-trustworthy-computing/