

TRAIL
OF **BITS**

COMP 427, Rice University
April 11, 2023
Kelly Kaoudis

Systems security in practice

Intro to threat modeling

nice to meet you

Currently: security engineer at Trail of Bits

- Research: vulnerabilities in parsers
- Assurance: threat modeling

How I got here...

- Previously: tech lead for application security at Twitter (left mid 2022)
- Before that: backend software engineering
- Before *that*: started a PhD in computer science, didn't finish

agenda

- Basics (in no particular order)
 - What, when, who, why
 - Threats, vulnerabilities, and attack vectors
 - Trust boundaries and scope
- How it all fits together
- What do people do with the results?

main idea threat modeling informs and enables making good system-level security decisions!

threat modeling

1. Agree on scope
2. Determine what can go wrong
3. Analyse how to prevent the scenario(s)
4. Try it
5. Did it work?
6. goto 1

why: systems thinking

- Everything is interconnected (dependencies)
- Emergent properties
- Protections and countermeasures will layer
- Boundaries: input, output, exchange
- ***Design-level*** weaknesses and vulns



rice comp 427: intro to threat modeling

img src: https://en.wikipedia.org/wiki/Sinking_of_the_Titanic, Willy Stöwer, 1912

<https://mechse.illinois.edu/news/blogs/titanic-material-failure>

<https://www.britannica.com/summary/Titanic>

Margaret Brown

文A 41 langu

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

From Wikipedia, the free encyclopedia

For other people with similar names, see [Margaret Brown \(disambiguation\)](#) and [Molly Brown \(disambiguation\)](#).

Margaret Brown (née **Tobin**; July 18, 1867 – October 26, 1932), posthumously known as "**The Unsinkable Molly Brown**", was an [American socialite](#) and [philanthropist](#). She was a passenger on the [RMS *Titanic*](#) which [sank in 1912](#) and she unsuccessfully urged the crew in [Lifeboat No. 6](#) to return to the debris field to look for survivors.^[1]

During her lifetime, her friends called her "Maggie", but by her death, obituaries referred to her as the "Unsinkable Mrs. Brown".^[2] The "Molly" nickname was coined by [1960 Broadway musical](#) based on her life and its [1964 film](#) adaptation which were both entitled *The Unsinkable Molly Brown*.

Margaret Brown



assumptions

- Enough lifeboats
- Lifeboat process is well-defined and practiced by crew (and passengers?)
- Sufficient distress signal monitoring
- Rivets and welds are safe
- Hull material is safe for expected water temp range
- Bulkheads are tall enough
- “unsinkable”

when to threat model

- Worst case: before the boat goes in the water ;)
- Software development life cycle (SDLC):

Requirements ->

Design ->

Implementation ->

Testing ->

Deployment ->

Maintenance ->

...

when to threat model

- Software development life cycle (SDLC):

Requirements ->

Design ->

Implementation ->

Testing ->

Deployment ->

Maintenance ->

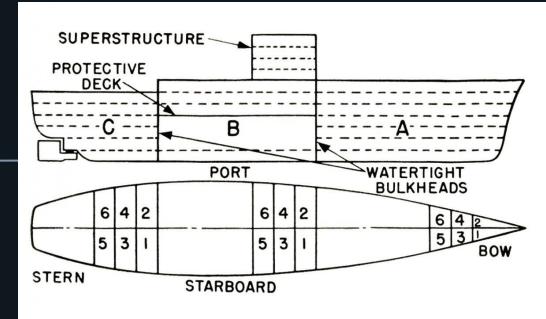
...

vulnerability

NIST: “a vulnerability is any **trust** assumption involving people, processes, or technology that can be violated in order to exploit a system”

ship vulns!

- Hull material or rivets are weak
- Bulkheads not tall enough
- No process for evacuation and insufficient lifeboats
- Also possible:
 - Engine overheats at top speed
 - Propeller gets stuck in shallow water
 - Crew mutiny
 - Crew member can be paid off to let in pirates
 - Hatch seal leaks in stormy conditions



expertise: who's in the room?

- **Development:** how individual pieces work, expected failures
- **Architecture:** how components fit together, systemic failure modes
- **Ops:** why is it like this actually? How did it previously fail?
- **Security:** how would an attacker _? What else can go wrong?

who's there?

- Users, admins, attackers...
- What do they want?
 - Sensitive data
 - Privileged access
 - Persistence
- What *should* they be able to do? What *can* they do?



examples: actors

- A user on the public network (internet) from which at least one application instance is accessible
- An administrator who can deploy the application, and responds to production alerts which impact the application
- A local user who controls a process or user account on the same host as the application developer

scope

- Simple: what could be threatened (data, components, people ...)
- More complicated: second degree involvement (dependencies)
- Balance cost (\$, time) with completeness
- Psychological acceptability

plausible problems

Practically speaking...

- threat \neq risk \neq vulnerability
- More complex system \Rightarrow more things can go wrong

Exploit...

- how system components will fail (also need to know how they *should* work)
- assumptions about what failure looks like
- unknown unknowns

plausible problems

Practically speaking...

- threat \neq risk \neq vulnerability
- More complex system \Rightarrow more things can go wrong

Exploit...

- how system components will fail (also need to know how they *should* work)
- assumptions about what failure looks like
- unknown unknowns



plausible problems

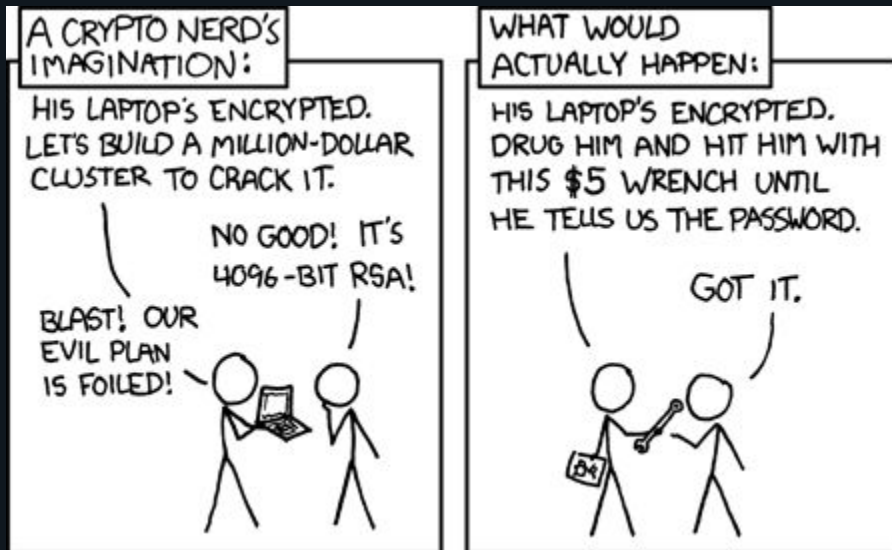


Reports that say that something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns—the ones we don't know we don't know. And if one looks throughout the history of our country and other free countries, it is the latter category that tends to be the difficult ones.^[1]


- assumptions about what failure looks like
- unknown unknowns

threat (scenarios)

NIST: “adversely impact[ing] organizational operations and assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, or modification of information, and/or denial of service”



threat: motive and method

-  : error / fault that makes software output / behavior not match expectations
- **Vulnerability**: *incorrect assumption* of security where there is actually weakness (subclass of all bugs)
- **Threat** = motive (attacker's desire) + method (exploiting the vuln)

threats: STRIDE

- **Spoofing**
- **Tampering**
- **Repudiation**
- **Information disclosure**
- **Denial of service**
- **Expansion of authority (elevation of privilege)**

STRIDE

- **Spoofing** violates authenticity
- **Tampering** violates integrity
- **Repudiation** violates non-repudiation
- **Information disclosure** violates confidentiality
- **Denial of service** violates availability
- **Expansion of authority** violates authorization

STRIDE: things that break user trust

- **Spoofing** violates authenticity
- **Tampering** violates integrity
- **Repudiation** violates non-repudiation
- **Information disclosure** violates confidentiality
- **Denial of service** violates availability
- **Expansion of authority** violates authorization

303 2.1.3 Attack Vector

304 An *attack vector* is a segment of the entire pathway that an attack uses to access a vulnerability. Each
305 attack vector can be thought of as comprising a *source* of malicious content, a potentially vulnerable
306 *processor* of that malicious content, and the nature of the malicious *content* itself. Examples of attack
307 vectors are:

- 308 • Malicious web page content (content) downloaded from a web site (source) by a vulnerable web
309 browser (processor);
- 310 • A malicious email attachment (content) in an email client (source) rendered by a vulnerable
311 helper application (processor);
- 312 • A malicious email attachment (content) downloaded from an email server (source) to a vulnerable
313 email client (processor);
- 314 • A network service with inherent vulnerabilities (processor) used maliciously (content) by an
315 external endpoint (source);
- 316 • Social engineering-based conversation (content) performed by phone from a human attacker
317 (source) to get a username and password from a vulnerable user (processor);
- 318 • Stolen user credentials (content) typed in by an attacker (source) to a web interface for an
319 enterprise authentication system (processor); and
- 320 • Personal information about a user harvested from social media (content) entered into a password
321 reset website by an attacker (source) to reset a password by taking advantage of weak password
322 reset processes (processor).

PASTA (process for attack simulation and threat analysis)

- Objectives
- Scope
- Decompose the application
- Analyse
 - Application / system weaknesses
 - Potential attack paths
 - Threats
- Determine impact (and likelihood)

PASTA (process for attack simulation and threat analysis)

- Objectives
- Scope
- Decompose the application
- Analyse
 - Application / system weaknesses
 - Potential attack paths
 - Threats
- Determine impact (and likelihood)



Determine attackers/actors

PASTA (process for attack simulation and threat analysis)

- Objectives
- Scope
- Decompose the application
- Analyse
 - Application / system weaknesses
 - Potential attack paths
 - Threats
- Determine impact (and likelihood)



Determine attackers/actors



Draw trust boundaries

trust boundary

- **OWASP**: “A trust boundary is a location in data flow where level of trust changes”

trust boundary

- **Adam Shostack**: “An attack surface is a **trust boundary** and a **direction** from which an attacker could launch an attack... a trust boundary is where entities with different **privileges** interact”
- **OWASP**: “A trust boundary is a location in data flow where level of trust changes”

trust boundary

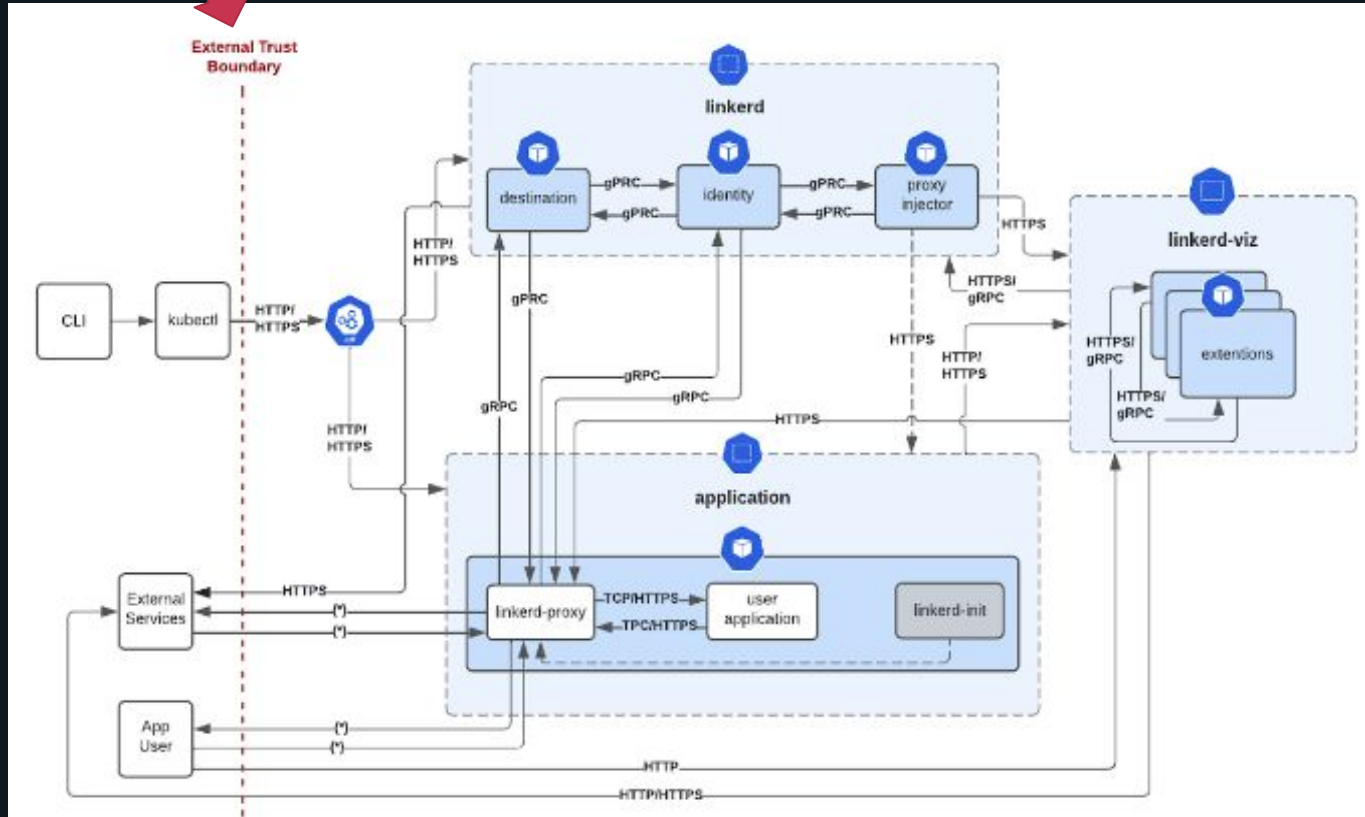
vector!

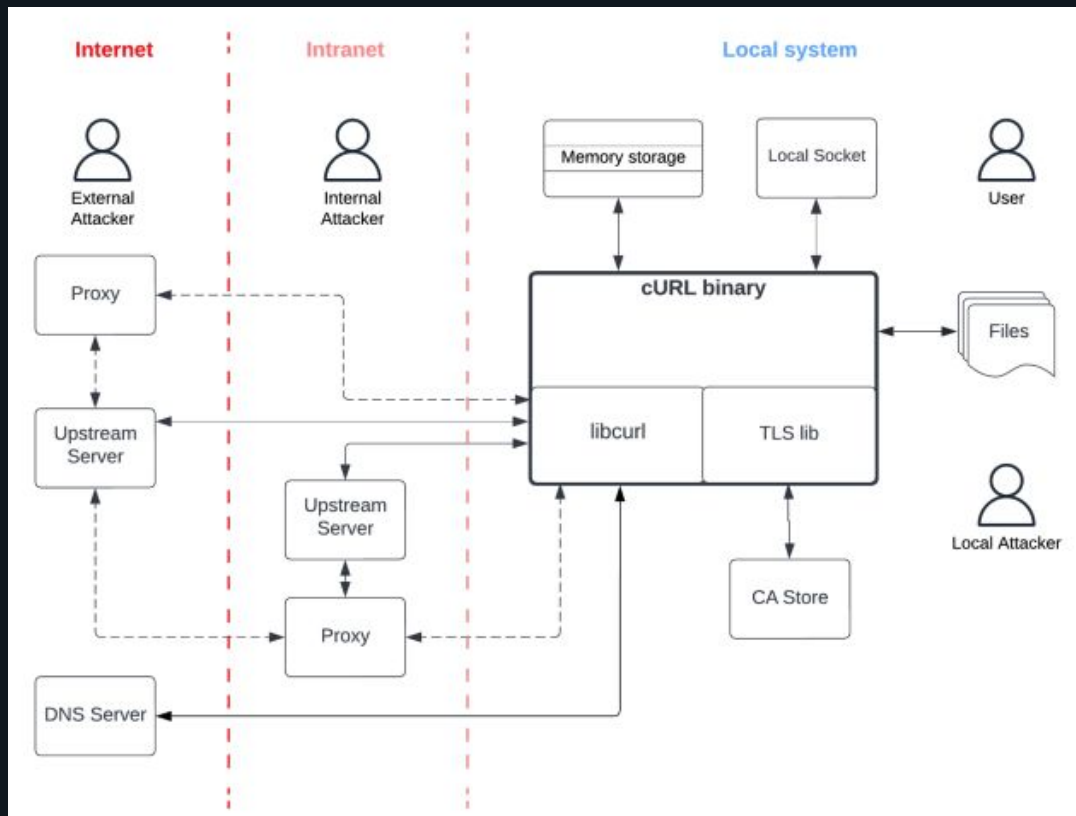


- **Adam Shostack:** “An attack surface is a **trust boundary** and a direction from which an attacker could launch an attack... a trust boundary is where entities with different **privileges** interact”
- **OWASP:** “A trust boundary is a location in data flow where level of trust changes”

trust boundary

- A location where system component(s) check privileges in order to allow or deny data flow
- “Between” components in data flow terms





cURL



PASTA (process for attack simulation and threat analysis)

- Objectives
- Scope
- Decompose the application
- Analyse
 - Application / system weaknesses
 - Potential attack paths
 - Threats
- Determine impact (and likelihood)



Determine attackers/actors



Draw trust boundaries

Mozilla Rapid Risk Assessment

- What does the system or application **do**?
- What **data** can it process or store?
- Confidentiality: What happens if all the data is disclosed to the world?
- Integrity: What if data is incorrect, misleading, website defaced, etc.?
- Availability: What if data or service is missing, deleted, unreachable?
- **Impact** (reputation, finances, productivity, system usability...)

NIST's Data-Centric Threat Modeling

- Step 1: identify and characterise the **system** and **data** of interest
- Step 2: identify and select the **attack vectors** to be included in the model
- Step 3: characterise the security **controls** for mitigating the attack vectors
- Step 4: analyse the resulting **threat model**

in practice: Trail of Bits

- System expertise + security expertise
- Make lists!
 - *Assumptions*
 - Actors
 - Data
 - Components
 - Client concerns
 - Operational controls

in practice: Trail of Bits

- Combine expertise areas
- Make lists!
 - *Assumptions*
 - Actors
 - Data
 - Components
 - Client concerns
 - Operational controls
- Draw a diagram
- A data flow through the system is...
- Could an attacker...
- Could a user accidentally...
- Could a developer...
- Could an admin...

in practice **tl;dr communication**

- Combine expertise areas
- Make lists!
 - *Assumptions*
 - Actors
 - Data
 - Components
 - Client concerns
 - Operational controls
- Draw a diagram
- A data flow through the system is...
- Could an attacker...
- Could a user accidentally...
- Could a developer...
- Could an admin...

***a finding* = plausible threat + lacking mitigation**



research

Try* to answer before discussion w/ client:

- What version of this dependency is in use? What weakness(es) could it add?
- If a user sends a message through component A, does it automatically trigger a message sent from component A to component B?
- What kinds of error messages get shown when the component fails?
- What classes can access this piece of data?

*resources like documentation, source code access, etc are not always provided, though...

discussion with the client

- What do you think is the most common use case?
- What is supposed to happen when the component fails?
- What was the design decision that led to this outcome?
- If a user sends a message through component A, does it automatically trigger a message sent from component A to component B?
- What *operational control* will apply when this component fails?



operational controls

- Process for assessment and recovery when \$BAD_THING happens
 - Diagnosis mechanisms? Who gets the alerts?
 - Incident (production, security) response
 - Who is responsible for what actions
 - Communication and managing expectations (leadership, user, internal...)
 - Service Level Objectives (and Service Level Agreements)
 - Timeframes for diagnosis, recovery
 - *Documentation*

***a finding* = plausible threat + lack of mitigation**

Voatz (2020)

Internal Processes	33
TM5. Incident Response is not automated and is under-documented	33
TM6. Risk Management is lacking	35
Voting Processes	38
TM7. Voter identity verification is manual with minimal training support	38
TM8. Internal team has full access to voter PII	39
TM9. Voters or admins could be blacklisted in denial-of-service attacks	40
TM10. Post-election handling processes increase risk of mishandling	41
External or Third-Party Storage	42
TM11. Post-election information shared via File Hosting Provider folders	42
TM12. Manual process to purge post-election data from shared File Hosting Provider folders	43
TM13. Cloud Storage Service storage is used for multiple elections and jurisdictions	44

Voatz

Internal Processes

- TM5. Incident Response is not automated and
- TM6. Risk Management is lacking

Voting Processes

- TM7. Voter identity verification is manual with
- TM8. Internal team has full access to voter PII
- TM9. Voters or admins could be blacklisted in c
- TM10. Post-election handling processes increa

External or Third-Party Storage

- TM11. Post-election information shared via File
- TM12. Manual process to purge post-election c
- folders
- TM13. Cloud Storage Service storage is used fo

TM5. Incident Response is not automated and is under-documented

Severity: High
Type: IR,RA
Component(s): All

Difficulty: High
Finding ID: TOB-VOATZ-TM05

Description

The implementation team noted that Incident Response and Threat Hunting processes were neither automated nor directly documented. Most IR or Hunt activities involved systems administrators sifting through logs manually via tools such as grep. Manual tooling increases the chances that an incident will be missed, both in terms of how long an incident occurs and what is the actual impact of the incident.

Justification

The severity is High for the following reasons:

- Missing or incomplete documentation does not in and of itself impact the normal operation of the system.
- However, missing documentation may hinder the correct implementation, remediation, or related activities such as incident response by the implementation team.
- Additionally, not alerting on incidents in an automated fashion increases the chance that incidents may be missed, or that more serious incidents will be missed by disaggregate data.

The difficulty is High for the following reasons:

- The implementation team must perform an inventory of all assets and data throughout the system.
- The team must also have previously completed [TOB-VOAT-TM02: Missing and Incomplete Data Classification](#).

TM21. Administrator commands are not logged

Severity: High
Type: AU, CM
Component(s): All

Difficulty: Medium
Finding ID: TOB-VOATZ-TM21

Description

The implementation team noted that system administrator's commands are not currently logged or monitored related to the Voatz application infrastructure. This may allow for malicious behavior that would not be detected or alerted. Furthermore, due to the manual nature of the configuration management within the Voatz system, an attack by a Malicious Internal Attacker could go unnoticed for some time.

Justification

The severity is High for the following reasons:

- The configuration management is manual, meaning that regular administrator access is not anomalous.
- Administrators do not need further authorization other than VPC access and credentials (username and key material) to access production instances.
- A Malicious Internal Attacker could alter configurations or install malicious software on the servers and there would be no detective controls like audit logging to help alert or determine what happened.

The difficulty is Medium for the following reasons:

- The implementation team must add command introspection to all host-login actions.
- Additional storage and processing would be needed to handle the additional log volume that would result from implementing these controls.

TM9. Voters or admins could be blacklisted in denial-of-service attacks

Severity: High

Type: DS, SC-5

Component(s): Admin Portal, Apache WS, ALB, WAF

Difficulty: Medium

Finding ID: TOB-VOATZ-TM09

Description

The controls put in place to help defend against brute forcing voter accounts, could be turned against the system to blacklist voters and admins in a denial of service attack. Credential stuffing and brute force attacks would generate the traffic that would result in account locks for voters and IP Blacklisting for admins. This finding assumes that there is a Malicious Internal User or Internal Attacker with sufficient position to affect this attack.

Justification

The severity is High for the following reasons:

- If a denial of service attack against the voters attempting to utilize the Voatz application was successful it would be immensely damaging to the trust in Voatz
- Denying election admins access to their management portal could cast doubt on the integrity of the election results.

The difficulty is Medium for the following reasons:

- To be effective, it would need to be a targeted attack with knowledge of eligible voters with email address and/or mobile phone number
- Implementing authentication controls with the proper balance between defense against brute force without sizably increasing the risk of a denial of service attack can be difficult

TM29. Infrastructure hosted outside the US

Severity: Medium
Type: SC-1, AC-20
Component(s): All

Difficulty: Low
Finding ID: TOB-VOATZ-TM29

Description

There are no procedural protections to prevent Voatz infrastructure from being hosted outside of the United States. Currently, Voatz has three servers physically located in Canada provided by [OVH](#), a French hosting company. Voatz indicated that these servers were test infrastructure. It is unclear whether any of these servers have been used in prior or ongoing elections.

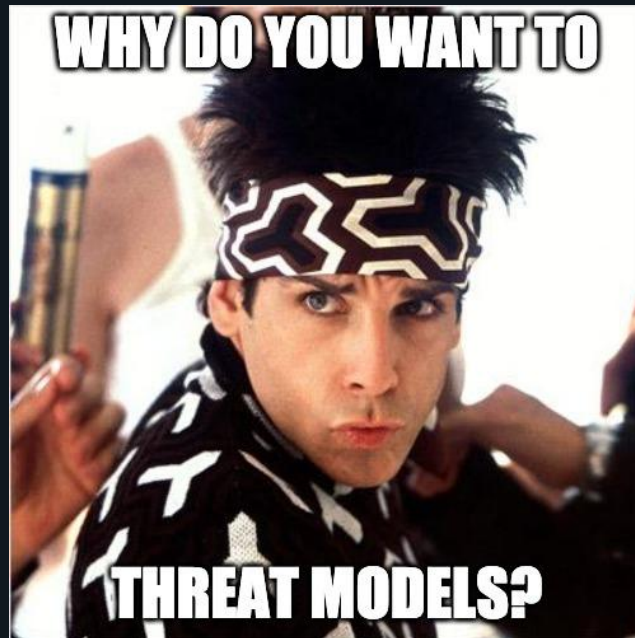
Justification

The severity is Medium for the following reasons:

- Voatz has claimed that Canada is still considered an acceptable jurisdiction in which to host servers.
- There is no evidence that Voatz has provisioned infrastructure in countries other than the US and Canada.
- Many hosting providers run datacenters in less friendly countries. A simple provisioning error changing "CA" to "CN" could result in infrastructure being provisioned in China.

The difficulty is Low for the following reasons:

- Election servers hosted in an adversarial country with unilateral control over its Internet infrastructure could trivially and selectively deny service to Voatz.
- Voatz infrastructure hosted in the jurisdiction of an adversarial country could be subpoenaed or confiscated.



threat model uses and outcomes


- Threat != risk
- Sufficient risk: report a finding (vulnerability, plausibility)
- PASTA, Mozilla RRA: what's the worst that could happen?
- What people actually do with a threat model:
 - How should an uncovered threat scenario be mitigated?
Who does the work to fix it? Who checks the fix works?
 - When should a threat scenario *not* be mitigated? What then?



security of systems: so what?

- Aiming for *acceptably secure*, not perfect
- Users' desire-paths (cat wants to be in box!)
- Maintainability, runnability
- User-centric design
 - Remove the footguns
 - Foster and maintain **trust**

security of systems: so what?

- Aiming for *acceptably secure*, not perfect
 - Users' desire-paths
 - Maintainability, runnability, compliance
 - User-centric design
 - Remove the footguns
 - Foster and maintain **trust**
- primary reason
- 

thank you!



Kelly Kaoudis

Security Engineer

kelly.kaoudis@trailofbits.com

trailofbits.com