Protecto

Agile Collaborative Threat Modeling

Threat modeling is one of the main requirements for any Agile team. Traditional approaches provide a false sense of security, leading to products and services that attacker personas can easily exploit. Protecto fixes the four main flaws we have in our current process, which results in a big step towards creating more secure services.

Getting Started - Steps

0. **Gather your team together** to take part in the session. It's a team exercise! Pick a facilitator.

1. **Create a Low-fi Diagram:** Sketch a low-fidelity diagram of the system components, data flows, and trust boundaries. Review with your teams and add more things if needed.

2. **Brainstorm Assets:** Take a few minutes and have everyone add yellow stickies for the assets we need to protect in the system (later move them to the inventory) E.g., Access logs, credentials.

3. **Brainstorm threats:** Take some time and think individually about the Threats that can be used to get access to the asset of interest or executed against the system as a whole. Use the ideas in the **threats catalogue** to fuel discussion, but remember they are not exhaustive.

4. **Review attackers:** Threats often have a type of attacker or multiple attackers that would be the most related to the threat. You may want to duplicate threats but put a different attacker for help with the next step.

5. **Brainstorm impact/probability matrix:** Copy/Paste the threat-asset-attacker combos and bring them together to the risk assessment matrix. Place them where you think they might land. Spend a few minutes dong that and then discuss with the team and shift if needed. It doesn't need to be perfect, but this method will help you to prioritize which areas should be addressed first in the next step (top right combos)

6. **Plan mitigation:** Copy-paste the risk combo from the previous section, brainstorm possible mitigations, create a plan, add acceptance criteria, and any additional information about the implementation cost. When ready, add it to your Jira or to another system to start building your security backlog.

7. **Discuss Triggers:** Since this must be a repetitive exercise, discuss what will trigger your team to get back together and enhance your threat model. (e.g., New functionality added, new competitor, new CWE discovered)

ATTACKER PERSONAS/AGENTS

Examples are provided, but you should add specific attackers if they might change impact or probability of risk.

AT1	A Competitor
AT2	A Bot
AT3	Internet user
AT4	Internal Admin

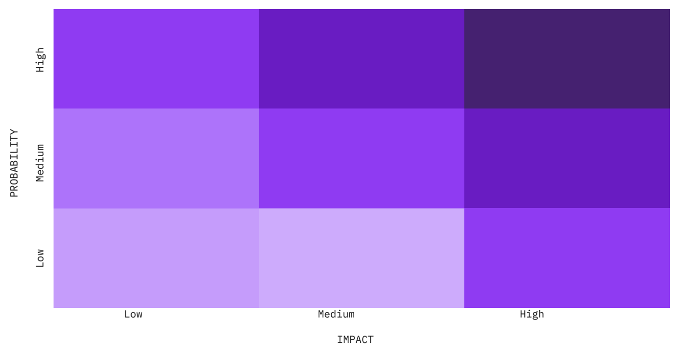
INVENTORY OF ASSETS

Examples are provided, but teams almost always add specific assets unique to your software design. Software that manages many valuable assets or privileges will need more thorough models.

A1	Credentials
A2	PII Data
A3	Application Code

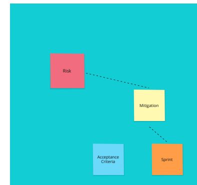
RISK MATRIX

Copy threats here from the diagram to help understand the worst risks for discussion. Typically, different combinations of assets and attackers will impact probability or impact.



RISK MITIGATION DASHBOARD

Copy risks / threats here and put together mitigation options and validation requirements to ensure the mitigation was successful. If possible, discuss cost and plan timing for mitigations.



LOW-FI DIAGRAM

Create a "whiteboard" low fidelity diagram of the system in scope (and important systems at the edge of the scope). It's common to revise the diagram throughout discussion of the model and regularly over time as needed. Use the helpers legend in the bottom right if it helps.

Low-Fi Diagram Helpers

Component

User

Data Flow

Authorization/Trust Boundary

Assets 🟨 Threats 🟥

Discuss events or situations that will trigger an update or re-assessment of the threat model. For example, perhaps a monthly review is needed, or only upon changes to specific subcomponents, etc.

TRIGGERS

If something is stolen from bar like bears

Increase in demand may need more staff or infrastructural upgrades

Customers shifting to a new bar due to more varieties.

New employees are hired or old employees have moved to a different bar

Changing demand of beer

new competitor as trigger

A new door is opened on the other end of the street

The pressure system is changed

new machine arrived

New Bartender

THREATS CATALOGUE

These are example threat types from well-known security categories -- these are not exhaustive, but they are useful for brainstorming and reminders. You should make your own threats more specific for best results.

Authenticity

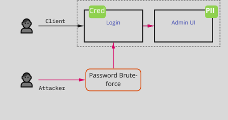
Spoofed Identity

How hard is it for an attacker to pretend to be someone with authority to use the system?

Can someone spoof an identity and then abuse its authority?
Spoofing identity allows attackers to do things they are not supposed to do.

An example of **identity spoofing** is an attacker breaking a poor user password and gaining access to the system.

Spoofed identity in the **offline** world could happen if you have a note with your pin stuck on the back of your Credit card.



S1	An attacker could squat on the random port or socket that the server normally uses
S2	An attacker could try one credential after another and there's nothing to slow them down (online or offline).
S3	An attacker can anonymously connect because we expect authentication to be done at a higher level
S4	An attacker can confuse a client because there are too many ways to identify a server
S5	An attacker can spoof a server because identifiers aren't stored on the client and checked for consistency on re-connection (that is, there's no key persistence)
S6	An attacker can connect to a server or peer over a link that isn't authenticated (and encrypted)
S7	An attacker could steal credentials stored on the server and reuse them (for example, a key is stored in a world readable file)
S8	An attacker who gets a password can reuse it (Use stronger authenticators)
S9	An attacker can choose to use weaker or no authentication
S10	An attacker could steal credentials stored on the client and reuse them
S11	An attacker could go after the way credentials are updated or recovered (account recovery doesn't require disclosing the old password)
S12	Your system ships with a default admin password, and doesn't force a change
SX	You've invented a new Spoofing attack
SP	We cannot tell which of our admins edited personal data,as admin accounts are shared.

Integrity

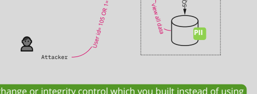
Tampering with Input

How hard is it for an attacker to modify the data they submit to your system?

Can they break a trust boundary and modify the code which runs as part of your system?

An example of **tampering with input** is when an attacker submits a SQL injection attack via a web application and uses that action to view everything in the database.

Imagine someone saying your name in the post office, and they give them all of your precious packages.



T1	An attacker can take advantage of your custom key exchange or integrity control which you built instead of using standard crypto
T2	Your code makes access control decisions all over the place, rather than with a security kernel
T3	An attacker can replay data without detection because your code doesn't provide timestamps or sequence numbers
T4	An attacker can change parameters over a trust boundary and after validation (for example, important parameters in a hidden field in HTML, or passing a pointer to critical memory)
T5	An attacker can write to a data store your code relies on
T6	An attacker can bypass permissions because you don't make names canonical before checking access permissions
T7	An attacker can manipulate data because there's no integrity protection for data on the network
T8	An attacker can provide or control state information
T9	An attacker can alter information in a data store because it has weak ACLs or includes a group which is equivalent to everyone ("all LIve ID holders")
T10	An attacker can write to some resource because permissions are granted to the world or there are no ACLs
T11	An attacker can load code inside your process via an extension point
TX	You've invented a new Tampering attack
TP1	Data in the database can be "fixed" by the admins, and nobody will ever know

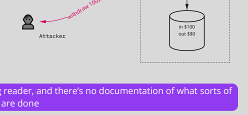
Non-repudiability

Repudiation Attack

How hard is it for users to deny performing an action?

What evidence does the system collect to help you to prove otherwise?
Non-repudiation refers to the ability of a system to ensure people are accountable for their actions.

An example of **repudiation of action** is where a user has deleted some sensitive information and the system lacks the ability to trace the malicious operations.



R1	An attacker can pass data through the log to attack a log reader, and there's no documentation of what sorts of validation are done
R2	A low privilege attacker can read interesting security information in the logs
R3	An attacker can alter digital signatures because the digital signature system you're implementing is weak, or uses MACs where it should use a signature
R4	An attacker can alter log messages on a network because they lack strong integrity controls
R5	An attacker can create a log entry without a time- stamp (or no log entry is timestamped)
R6	An attacker can make the logs wrap around and lose data
R7	An attacker can make a log lose or confuse security information
R8	An attacker can use a shared key to authenticate as different principals, confusing the information in the logs.
R9	An attacker can get arbitrary data into logs from unauthenticated (or weakly authenticated) outsiders without validation.
R10	An attacker can edit logs, and there's no way to tell (perhaps because there's no heartbeat option for the logging system)
R11	An attacker can say, "I didn't do that," and you would have no way to prove them wrong.
R12	The system has no logs.
RX	You've invented a new Repudiation attack
RP1	We log changes and deletion sof personal data, but viewing is not logged
RP2	We log personal data access,but there is no ongoing monitoring or alerting.

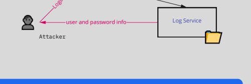
Confidentiality

Information Disclosure

Can someone view information they are not supposed to have access to?

Information disclosure threats involve the exposure or interception of information to unauthorized individuals.

An example of **information disclosure** is when a user can read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers



I1	An attacker can brute-force file encryption because no defense is in place
I2	An attacker can see error messages with security-sensitive content.
I3	An attacker can read content because messages (for example, an e-mail or HTTP cookie) aren't encrypted even if the channel is encrypted.
I4	An attacker might be able to read a document or data because it's encrypted with a nonstandard algorithm.
I5	An attacker can read data because it's hidden or occluded (for undo or change tracking) and the user might forget that it's there.
I6	An attacker can act as a "man in the middle" because you don't authenticate endpoints of a network connection.
I7	An attacker can access information through a search indexer, logger, or other such mechanism.
I8	An attacker can read sensitive information in a file with bad ACLs.
I9	An attacker can read information in files with no ACLs.
I10	An attacker can discover the fixed key being used to encrypt.
I11	An attacker can read the entire channel because the channel (for example, HTTP or SMTP) isn't encrypted.
I12	An attacker can read network information because there's no cryptography used.
IX	You've invented a new Information Disclosure attack
IP1	Personal data is being sent over a plaintext connection or email.
IP2	Personal data is being saved on unencrypted media

Availability

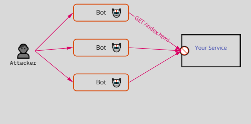
Denial of Service

Can someone break a system so valid users are unable to use it?

Denial of service attacks work by flooding, wiping or otherwise breaking a particular service or system.

An example of **denial of service** is where a Web server has been made temporarily unavailable or unusable with a flood of traffic generated by a botnet.

A DDoS attack is like an unexpected traffic jam clogging up the highway, preventing regular traffic from arriving at its destination in the **offline world.**



D1	An attacker can make your authentication system unusable or unavailable.
D2	An attacker can make a client unavailable or unusable but the problem goes away when the attacker stops.
D3	An attacker can make a server unavailable or unusable but the problem goes away when the attacker stops.
D4	An attacker can make a client unavailable or unusable without ever authenticating, but the problem goes away when the attacker stops.
D5	An attacker can make a server unavailable or unusable without ever authenticating, but the problem goes away when the attacker stops.
D6	An attacker can make a client unavailable or unusable and the problem persists after the attacker goes away.
D7	An attacker can make a server unavailable or unusable and the problem persists after the attacker goes away.
D8	An attacker can make a client unavailable or unusable without ever authenticating, and the problem persists after the attacker goes away.
D9	An attacker can make a server unavailable or unusable without ever authenticating, and the problem persists after the attacker goes away.
D10	An attacker can cause the logging subsystem to stop working. An attacker who can cause your logging to stop can execute attacks that are then harder to understand and possibly harder to remediate.
D11	An attacker can amplify a denial-of-service attack through this component with amplification on the order of 10:1
D12	An attacker can amplify a denial-of-service attack through this component with amplification on the order of 100:1.
DX	You've invented a new DoS attack
DP	Availability of certain personal data is a life-or-death matter,and our system is not as reliable as it should.

Authorization

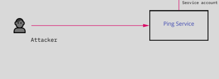
Elevation of Privilege

Can an unprivileged user gain more access to the system than they should have?

Elevation of privilege attacks are possible because authorization boundaries are missing or inadequate.

An example of **elevation of privilege** is where the attacker could use unprotected service using a service account to access the protected data.

In the **offline** world, this could be a thief breaking through the window to enter your house, otherwise protected with a very secure door.



E1	An attacker can force data through different validation paths which give different results.
E2	An attacker could take advantage of .NET permissions you ask for but don't use.
E3	An attacker can provide a pointer across a trust boundary, rather than data that can be validated.
E4	An attacker can enter data that is checked while still under the attacker's control and used later on the other side of a trust boundary.
E5	There's no reasonable way for callers to figure out what validation of tainted data you perform before passing it to them.
E6	There's no reasonable way for a caller to figure out what security assumptions you make.
E7	An attacker can reflect input back to a user, such as cross-site scripting.
E8	You include user-generated content within your page, possibly including the content of random URLs.
E9	An attacker can inject a command that the system will run at a higher privilege level.
EX	You've invented a new Spoofing attack

Ethical Design

Technology that respects **human rights** is decentralised, peer-to-peer, zero-knowledge, end-to-end encrypted, free and open source, interoperable, accessible, and sustainable. It respects and protects your civil liberties, reduces inequality, and benefits democracy.

Technology that respects **human effort** is functional, convenient, and reliable. It is thoughtful and accommodating; not arrogant or demanding. It understands that you might be distracted or differently-abled. It respects the limited time you have on this planet.

Technology that respects human experience is beautiful, magical, and delightful.It just works. It's intuitive. It's invisible. It recedes into the background of your life. It gives you joy. It empowers you with superpowers. It puts a smile on your face and makes your life better.

ED1	There is no review process for introducing new trackers or adve-tising providers on the web pages; whatever our designers like, or marketing sells, will be used
ED2	Our telemetry is tied to the users, even though our analytics couldn't care less who the user actually is
ED3	A neural network makes customer-related decisions,but nobody can really explain to the customers what the model is based on.
ED4	Your service uses one of the black hat design patterns.
ED5	Sed quis custodiet ipsos custodes?
ED6	We don't prioritize Accessibility.
EDX	The team came up with a new ethical design principle.

LEGAL NOTES