Threat modelling for developers

Arne Padmos
ASKING AIRCRAFT DESIGNERS ABOUT AIRPLANE SAFETY:
Nothing is ever foolproof, but modern airliners are incredibly resilient. Flying is the safest way to travel.

ASKING BUILDING ENGINEERS ABOUT ELEVATOR SAFETY:
Elevators are protected by multiple tried-and-tested failsafe mechanisms. They're nearly incapable of falling.

ASKING SOFTWARE ENGINEERS ABOUT COMPUTERIZED VOTING:
That's terrifying.

WAIT, REALLY?
Don't trust voting software and don't listen to anyone who tells you it's safe.

WHY?
I don't quite know how to put this, but our entire field is bad at what we do, and if you rely on us, everyone will die.

THEY SAY THEY'VE FIXED IT WITH SOMETHING CALLED "BLOCKCHAIN."
AAAAA!!
Whatever they sold you, don't touch it.
Bury it in the desert.
Wear gloves.
Safety vs Security
Are we doomed?
“Building security in”

“Security by design”

“Shifting security left”
“If we ... could do only one thing to improve software security ... we would do threat modelling every day of the week.”

— Howard & Lipner
“If we ... could do only one thing to improve software security ... we would do threat modelling every day of the week.”

— Howard & Lipner
Requirements engineering & Architectural analysis
What’s your threat model?

( security assumptions )
On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.

I. INTRODUCTION

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a “passive” eavesdropper, namely, one who merely taps the communication line and tries to decipher the intercepted message. However, as pointed out in Needham and Schroeder [8], an improperly designed protocol could be vulnerable to an “active” saboteur, one who may impersonate another user and may alter or replay the message. As a protocol might be compromised in a complex way, informal arguments that assert the security for a protocol are prone to errors. It is thus desirable to have a formal model in which the security issues can be discussed precisely. The models we introduce will enable us to study the security problem for families of protocols, with very few assumptions on the behavior of the saboteur.

We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user $X$ has an encryption function $E_X$ and a decryption function $D_X$, both are mappings from $\{0,1\}^*$ (the set of all finite binary sequences) into $\{0,1\}^*$. A secure public directory contains all the $(X, E_X)$ pairs, while the decryption function $D_X$ is known only to user $X$. The main requirements on $E_X$, $D_X$ are:

1) $E_X D_X = D_X E_X = 1$, and
2) knowing $E_X(M)$ and the public directory does not reveal anything about the value $M$.

Thus everyone can send $X$ a message $E_X(M)$, $X$ will be able to decode it by forming $D_X(E_X(M)) = M$, but nobody other than $X$ will be able to find $M$ even if $E_X(M)$ is available to them.

We will be interested mainly in protocols for transmitting a secret plaintext $M$ between two users. To give an idea of the way a saboteur may break a system, we consider a few examples. A message sent between parties in the network consists of three fields: the sender’s name, the receiver’s name, and the text. The text is the encrypted part of the message. We will write a message in the format: sender’s name, text, receiver’s name.

Example 1: Consider the following protocol for sending messages:

"sender X sends message M to receiver Y"
“More precisely, we will assume the following about a saboteur:”

– obtain any message
– initiate any conversation
– be a receiver to any user
GFE = Google Front End Server

SSL Added and removed here! 😊

Traffic in clear text here.
What could possibly go wrong?
What could possibly go wrong?
& how
Types of threat modelling

– Attacker-centric
– Asset-centric
– System-centric
Initial Foothold: Compromised System
- Reconnaissance
- Weaponization
- Delivery
- Social Engineering
- Exploitation
- Persistence
- Defense Evasion
- Command & Control

Pivoting

Network Propagation: Internal Network
- Discovery
- Privilege Escalation
- Execution
- Credential Access
- Lateral Movement

Access

Action on Objectives: Critical Asset Access
- Collection
- Exfiltration
- Target Manipulation
- Objectives
This action does not apply to this asset, based on the asset's type in the Data Model tab.

(Never) The system should never let this actor take this action on this asset.

(Conditionally) The system should let this actor take this action on this asset when certain conditions (typically documented in the cell comment) are met.

(Always) The system should always let this actor take this action on this asset.

Example:

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>R</th>
<th>X</th>
<th>Author</th>
<th>Editor</th>
<th>Reader</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>D</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asset</td>
<td>Blog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Blog Post</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Eleanor Saitta et al.
Popular approaches (system-centric)

- STRIDE
- Trike
- PASTA
Relevant questions

1. What are we working on?
2. What can go wrong?
3. What are we going to do?
4. Did we do a good job?
Lightweight methodology

1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work
Lightweight methodology

1. **Draw data flows**
2. **Elicit threats**
3. **Ranking + controls**
4. **Check your work**
Lightweight methodology

1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work
Confidentiality
Integrity
Availability
Authentication
Authorisation
Accountability
Information disclosure
Tampering
Denial of service
Spoofing
Elevation of privilege
Repudiation
“STRIDE”
Customer’s premises

End-user computer

Internet

SWIFT messaging software

SWIFT connectivity software

Back-office applications and printers

SWIFT
Lightweight methodology

1. Draw data flows
2. Elicit threats
3. **Ranking + controls**
4. Check your work
Risk \approx \text{likelihood} \times \text{impact}
Lightweight methodology

1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work
“All models are wrong, some models are useful.”

— George Box
1. information disclosure
2. data interpreted as code
3. resource exhaustion/denial
4. race conditions
5. canonicalisation
6. insufficient access control
7. environment (mis)configuration
8. logic errors
9. predictability
10. poor usability
"I wish these parts could communicate more easily."

"Ooh, this new technology makes it easy to create arbitrary connections, integrating everything!"

"Ooh, this new technology makes it easy to enclose arbitrary things in secure sandboxes!"

"Uh-oh, there are so many connections it's creating bugs and security holes!"
Lightweight methodology

1. Draw data flows
2. Elicit threats
3. Ranking + controls
4. Check your work
The washing machine has sensors that can measure the size and type of the load. Using this information, it determines the most suitable washing programme.

The correct amount of laundry detergent is calculated based on the size of the load detected by the sensors.

The washing machine is connected to Wi-Fi, allowing it to be controlled remotely by a smartphone.

The smartphone and the washing machine are connected via the cloud. Both the status of the washing machine and its instructions are transmitted via the cloud, which is managed by the app provider (usually the manufacturer of the washing machine).

Access to the cloud can be regulated by the manager of the cloud service, but may also be outsourced to other parties – it may be possible to log in using a Google or Facebook account, for example.

A smartphone app enables the washing machine to be turned on and off (with an optional timer). Users can also select their own programmes. Apps are provided by the manufacturer of the washing machine, but can also be developed and supplied by other parties.
What could possibly go wrong?
Arne Padmos
hello@arnepadmos.com
github.com
/arnepadmos/resources

my “toy collection”