Designing Secure Software

Nick Sumner
wsumner@sfu.ca
Secure software can be challenging to design.
Secure software can be challenging to design

- `send_message(from,to,text)` charges a fee to sender
- `send_credit(from,to,value)` credits recipient & charges sender
- `summarize_account(target_path)`
Secure software can be challenging to design

Client

Server

Client

send_message(from,to,text)
charges a fee to sender

send_credit(from,to,value)
credits recipient & charges sender

What could possibly go wrong?
Secure software can be challenging to design

Client

send_message(from,to,text)
charges a fee to sender

send_credit(from,to,value)
credits recipient & charges sender

send_credit(from,to,-100)

Server

Client

summarize_account(target_path)

Alice

Bob
Secure software can be challenging to design

How should you prevent it?
Secure software can be challenging to design

send_message(from,to,text)
charges a fee to sender

send_credit(from,to,value)
credits recipient & charges sender

summarize_account(target_path)

send_credit(from,to,-100)

How should you prevent it?
Secure software can be challenging to design

send_message(from,to,text) charges a fee to sender
send_credit(from,to,value) credits recipient & charges sender
summarize_account(target_path)

How should you prevent it?
Secure software can be challenging to design

send_message(from,to,text) charges a fee to sender

send_credit(from,to,value) credits recipient & charges sender

summarize_account(target_path)

summarize_account(“path/to/bill”)
Secure software can be challenging to design

How should you prevent it?
Secure software can be challenging to design

send_message(from,to,text) charges a fee to sender

send_credit(from,to,value) credits recipient & charges sender

summarize_account(target_path)

summarize_account("path/to/bill")

How should you prevent it?
Secure software can be challenging to design

How should you prevent it?
Secure software can be challenging to design

**Client**

- Alice

**Server**

- send_message(from,to,text)
  - charges a fee to sender

- send_credit(from,to,value)
  - credits recipient & charges sender

- summarize_account(target_path)

**Client**

- Bob

- send_credit(from,to,-100)

- summarize_account(“path/to/bill”)

- send_message(Alice,Bob, ”xxxx send_credit(Bob,Alice,200)” )
Secure software can be challenging to design

send_message(from,to,text) charges a fee to sender
send_credit(from,to,value) credits recipient & charges sender
summarize_account(target_path)

send_credit(from,to,-100)
summarize_account("path/to/bill")

How should you prevent it?
Secure software can be challenging to design

Client

send_message(from,to,text) charges a fee to sender

send_credit(from,to,value) credits recipient & charges sender

Client

send_credit(from,to,-100)

summarize_account(target_path)

summarize_account("path/to/bill")

send_message(Alice,Bob, "xxxx send_credit(Bob,Alice,200)" )

How should you prevent it?
Secure software can be challenging to design

send_message(from,to,text) charges a fee to sender

send_credit(from,to,value) credits recipient & charges sender

summarize_account(target_path)

send_credit(from,to,-100)

summarize_account("path/to/bill")

send_message(Alice,Bob, "xxxx send_credit(Bob,Alice,200)"

How should you prevent it?
Secure software can be challenging to design

send_message(from,to,text) charges a fee to sender

send_credit(from,to,value) credits recipient & charges sender

How should you prevent it?
Secure software can be challenging to design

send_message(from,to,text) charges a fee to sender

send_credit(from,to,value) credits recipient & charges sender

summarize_account(target_path)

send_credit(from,to,-100)

send_credit(Alice,Bob,100)

send_credit(Alice,Bob,100)

summarize_account("path/to/bill")

send_message(Alice,Bob, "xxxx send_credit(Bob,Alice,200)"

How should you prevent it?
Secure software can be challenging to design

- `send_message(from,to,text)` charges a fee to sender
- `send_credit(from,to,value)` credits recipient & charges sender
- `summarize_account(target_path)`
Secure software can be challenging to design

send_message(from,to,text)

charges a fee to sender

summarize_account(target_path)

send_credit(from,to,value)

credits recipient & charges sender

How should you prevent it?
Defining the threats

- *Security* is about maintaining desired properties in the face of an adversary
Defining the threats

- **Security** is about maintaining desired properties in the face of an adversary
- Before addressing issues, you must define your adversary
  - *threat modeling* / risk analysis
Defining the threats

- **Security** is about maintaining desired properties in the face of an adversary
- Before addressing issues, you must define your adversary
  - threat modeling / risk analysis
- Threat models define the abilities of an attacker
  - Too weak – you won’t defend what you need to
  - Too strong – you will expend resources unnecessarily defending
Defining the threats

- **Security** is about maintaining desired properties in the face of an adversary
- Before addressing issues, you must define your adversary
  - *threat modeling* / risk analysis
- Threat models define the abilities of an attacker
  - Too weak – you won’t defend what you need to
  - Too strong – you will expend resources unnecessarily defending
  - Your threat model should define realistic abilities
Defining the threats

- **Security** is about maintaining desired properties in the face of an adversary
- Before addressing issues, you must define your adversary
  - threat modeling / risk analysis
- Threat models define the abilities of an attacker
  - Too weak – you won’t defend what you need to
  - Too strong – you will expend resources unnecessarily defending
  - Your threat model should define realistic abilities
- An explicit threat model focuses attention
Defining the threats

• Several methodologies that can be used
  – STRIDE, ASSET, OCTAVE
  – Basing a model on existing models & knowledge mitigates internal uncertainty
Defining the threats

- Several methodologies that can be used
  - STRIDE, ASSET, OCTAVE
  - Basing a model on existing models & knowledge mitigates internal uncertainty
  - Knowing realistic attack patterns is necessary
Defining the threats

- Several methodologies that can be used
  - STRIDE, ASSET, OCTAVE
  - Basing a model on existing models & knowledge mitigates internal uncertainty
  - Knowing realistic attack patterns is necessary
    - Invalid or malformed inputs
Defining the threats

- Several methodologies that can be used
  - STRIDE, ASSET, OCTAVE
  - Basing a model on existing models & knowledge mitigates internal uncertainty
  - Knowing realistic attack patterns is necessary
    - Invalid or malformed inputs
    - Duplicating, dropping, or changing network traffic
Defining the threats

- Several methodologies that can be used
  - STRIDE, ASSET, OCTAVE
  - Basing a model on existing models & knowledge mitigates internal uncertainty
  - Knowing realistic attack patterns is necessary
    - Invalid or malformed inputs
    - Duplicating, dropping, or changing network traffic
    - Observing traffic to learn patterns
Defining the threats

- Several methodologies that can be used
  - STRIDE, ASSET, OCTAVE
  - Basing a model on existing models & knowledge mitigates internal uncertainty
  - Knowing realistic attack patterns is necessary
    - Invalid or malformed inputs
    - Duplicating, dropping, or changing network traffic
    - Observing traffic to learn patterns
    - Direct access to database? Meaningful for colocated attackers!
Defining the threats

- Several methodologies that can be used
  - STRIDE, ASSET, OCTAVE
  - Basing a model on existing models & knowledge mitigates internal uncertainty
  - Knowing realistic attack patterns is necessary
    - Invalid or malformed inputs
    - Duplicating, dropping, or changing network traffic
    - Observing traffic to learn patterns
    - Direct access to database? Meaningful for colocated attackers!
  - e.g. STRIDE: Spoofing, Tampering, Repudiation, Information Disclosure, DOS, Elevate privilege
What properties do we care about?

- CIA Model – classic security properties
What properties do we care about?

- CIA Model – classic security properties
  - **Confidentiality**
    - Information is only disclosed to those authorized to know it
What properties do we care about?

- CIA Model – classic security properties
  - Confidentiality
  - Integrity
    - Only modify information in **allowed ways by authorized parties**
What properties do we care about?

- CIA Model – classic security properties
  - Confidentiality
  - Integrity
    - Only modify information in allowed ways by authorized parties
    - Do what is expected
What properties do we care about?

- CIA Model – classic security properties
  - Confidentiality
  - Integrity
  - Availability
    - Those authorized for access are not prevented from it
Most security problems are design problems

- 50% of security issues are actually design issues [McGraw 2006]
Most security problems are design problems

- 50% of security issues are actually design issues [McGraw 2006]
- The responses that you employ will be driven by risk assessment
  - Risk = Probability * Impact
  - E[cost of breach] vs. E[cost of mitigation]
Most security problems are design problems

- 50% of security issues are actually design issues [McGraw 2006]
- The responses that you employ will be driven by risk assessment
  - Risk = Probability * Impact
  - E[cost of breach] vs. E[cost of mitigation]
- Can we characterize the core problems in our initial design?
Most security problems are design problems

- 50% of security issues are actually design issues [McGraw 2006]
- The responses that you employ will be driven by risk assessment
  - Risk = Probability * Impact
  - E[cost of breach] vs. E[cost of mitigation]
- Can we characterize the core problems in our initial design?

```
send_credit(from,to,-100)
```

```
summarize_account("path/to/bill")
```

```
send_message(Alice,Bob,
"xxxx send_credit(Bob,Alice,200)"
)
```
Managing threats

- Threat management strategies focus on 3 approaches:
Managing threats

- Threat management strategies focus on 3 approaches:
  - Prevention
  - Mitigation
  - Detection & Response
Threat management strategies focus on 3 approaches:
- Prevention
- Mitigation
- Detection & Response

If we have the first, why do we need others?
Managing threats

- Threat management strategies focus on 3 approaches:
  - Prevention
  - Mitigation
  - Detection & Response

- **Key principles** [Salzer, 1975]
Managing threats

- Threat management strategies focus on 3 approaches:
  - Prevention
  - Mitigation
  - Detection & Response

- Key principles [Salzer, 1975]
  - Keep the design as simple and small as possible
Managing threats

- Threat management strategies focus on 3 approaches:
  - Prevention
  - Mitigation
  - Detection & Response

- **Key principles** [Salzer, 1975]
  - Keep the design as simple and small as possible
  - Complete mediation
Managing threats

• Threat management strategies focus on 3 approaches:
  – Prevention
  – Mitigation
  – Detection & Response

• Key principles [Salzer, 1975]
  – Keep the design as simple and small as possible
  – Complete mediation
  – Separation of privilege
Managing threats

- Threat management strategies focus on 3 approaches:
  - Prevention
  - Mitigation
  - Detection & Response

- **Key principles** [Salzer, 1975]
  - Keep the design as simple and small as possible
  - Complete mediation
  - Separation of privilege
  - Least privilege
  - ...
Applying least privilege

- What can our server & clients do?
Applying least privilege

- What can our server & clients do?
  - Send network traffic
  - Parse strings
  - Transfer funds
  - Create files
Applying least privilege

- What can our server & clients do?
  - Send network traffic
  - Parse strings
  - Transfer funds
  - Create files

- What can it do when compromised?
Applying least privilege

- What can our server & clients do?
  - Send network traffic
  - Parse strings
  - Transfer funds
  - Create files

- What can it do when compromised?
  - The same
Compartmentalization and least privilege

- Instead, we can separate these responsibilities into separate components with limited rights
Compartmentalization and least privilege

- Instead, we can separate these responsibilities into *separate components* with limited rights.

Diagram:

- Client
- Action Selection
- Message Parsing
- Display
- Networking
Instead, we can separate these responsibilities into separate components with limited rights.
Compartmentalization and least privilege

- Instead, we can separate these responsibilities into *separate components* with limited rights.

```python
send_message(Alice, Bob, "xxxx send_credit(Bob, Alice, 200)"
```
Compartmentalization and least privilege

Instead, we can separate these responsibilities into separate components with limited rights

Client
- Action Selection
- Display
- Networking
- Message Parsing

Server
- Billing Database
- File Access
- Message Dispatch
- Networking

What is the impact?

send_message(Alice,Bob, “xxxx send_credit(Bob,Alice,200)” )
Instead, we can separate these responsibilities into separate components with \textit{limited rights}. If the message parser has the \textit{right} to \textit{communicate to the network}, we achieved nothing!
Compartmentalization and least privilege

- Instead, we can separate these responsibilities into separate components with *limited rights*

If the message parser has the *right* to communicate to the network, we achieved nothing!
Compartmentalization and least privilege

- Instead, we can separate these responsibilities into separate components with *limited rights*

![Diagram showing client and server components with actions and rights]

If the message parser has the *right* to *communicate to the network*, we achieved nothing!

Use capabilities, pledges, ... to sandbox & limit rights
Compartmentalization and least privilege

- Instead, we can separate these responsibilities into separate components with *limited rights*

```
Client
  - Action Selection
  - Message Parsing
  - Display
  - Networking

Server
  - Billing Database
  - File Access
  - Message Dispatch
  - Networking
```

How does this relate to what we have discussed in class?
Compartmentalization and least privilege

- Instead, we can separate these responsibilities into separate components with *limited rights*

![Diagram showing client and server components]

- This approach to sandboxing responsibilities is also employed:
  - Browsers
  - Android
  - ...

Stepping back
Stepping back
The opportunity for secure software can only exist when the software is designed cleanly.