Section 5: Secure Systems Design and Development

195

Having learned basics of computer security and data security, in this section, you will learn how to develop secure systems.

In particular, we will learn threat modeling process during secure system design.

Objectives

- To introduce application types
- Understand secure software development process
- Learn threat modeling
- Case Study – Example threat modeling

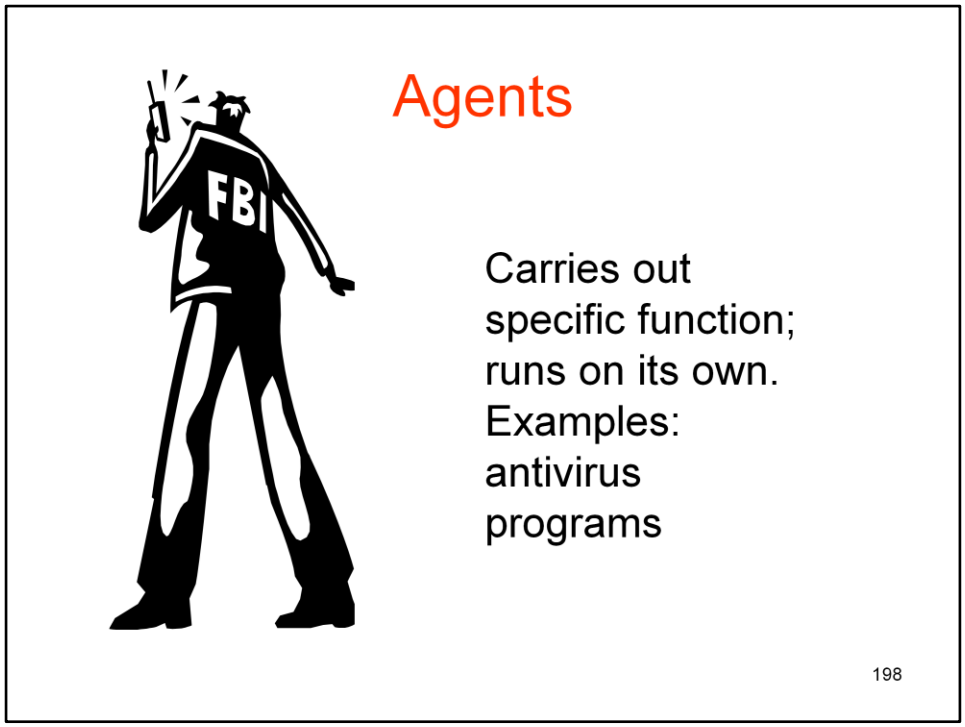
196

Some Application Types

- Agents
- Applet
- Client Server Applications
- Distributed Applications
- Web Applications
- Others?

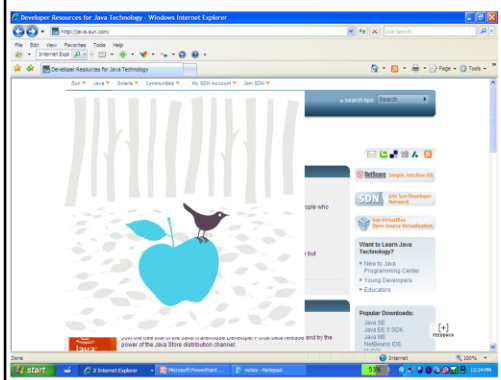
197

Software applications can be categorized into the above types. Application type generally influences the type of architecture and threat models to be used.



Agents run unassisted and carry out a specific function. An agent can communicate with other agents and users. A software agent can perceive the changes in the environment and react to them. Agents are useful in distributed applications and there is need for communication, especially over the Internet. Agents have ability to reason and can work based on messages received and changes to environment.

Applets

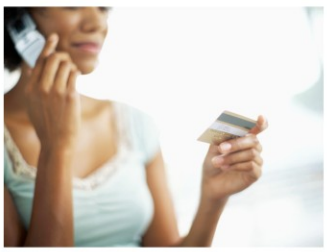


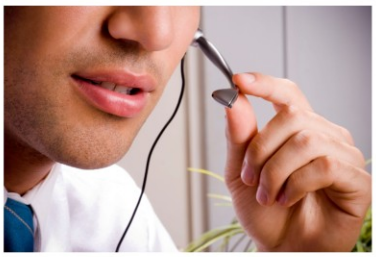
Applets run within another program. Example: java applets within web browser.

199

Applets may be put inside web page to make it dynamic. In that case, applets are downloaded from the web server just like any other resource (e.g., audio or image). Applets are usually written in JAVA. Applet has a very specific and narrow function and does not run independently. Applets are usually run in a “sandbox”, thus putting security restrictions on their access to resources such as hard drive.

Client – Server Application





200

Client-Server computing is distributed computing in which tasks are divided into server side tasks and client side tasks. Clients and Server often run on different hardware and interact over the network. Server usually listens to connection and client initiates the connection.

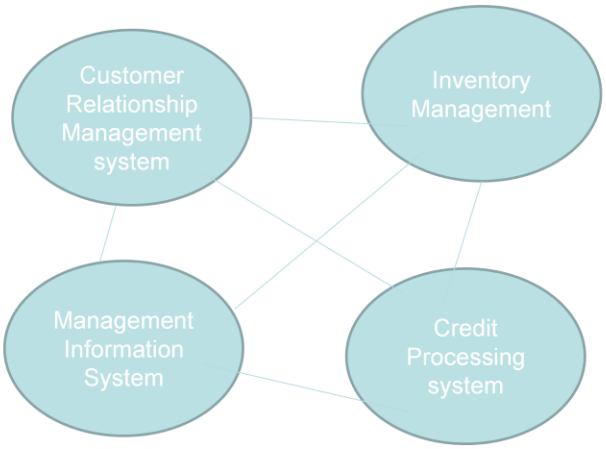
Client Application

- Contains User Interface Logic
- Accepts inputs from users
- Runs on PCs, Work Stations etc.

Server Applications

- Server accepts client requests and processes them
- Connects to database and other components

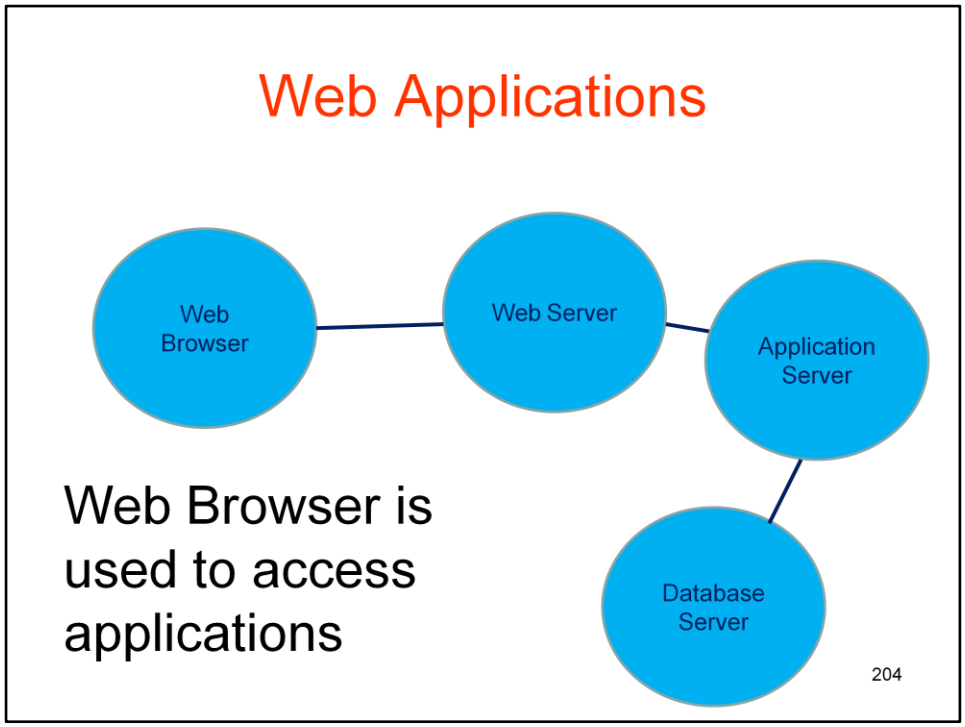
Distributed Applications



Software components running on different systems

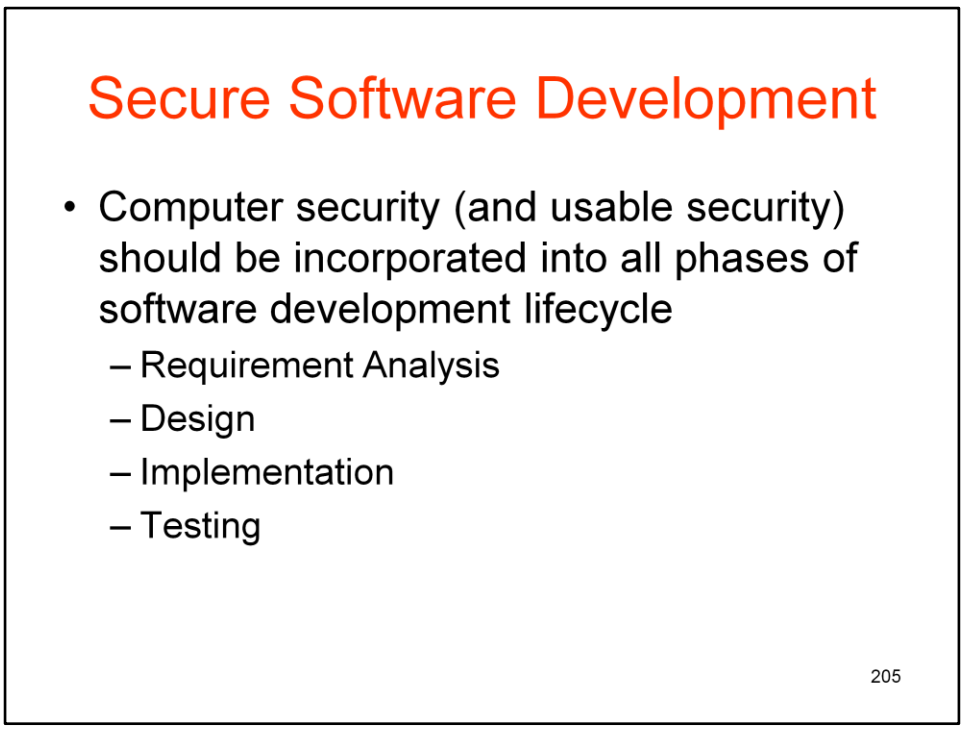
203

General distributed applications. Programs running on, often, different hardware interact with each other by sending and receiving messages.



Most web applications are client-server applications. Web client (such as browser) initiates a connection to the web server and requests a resource. The resource may be a static HTML file, which may be found on a local file system at the server. If the requested page has dynamic information, such as customer information, it needs to contact an application server, which in turn contacts a database server. Usually database server stores and protects information.

In this section, we focus on web applications due to their relevancy and interest from (usable) security view point.



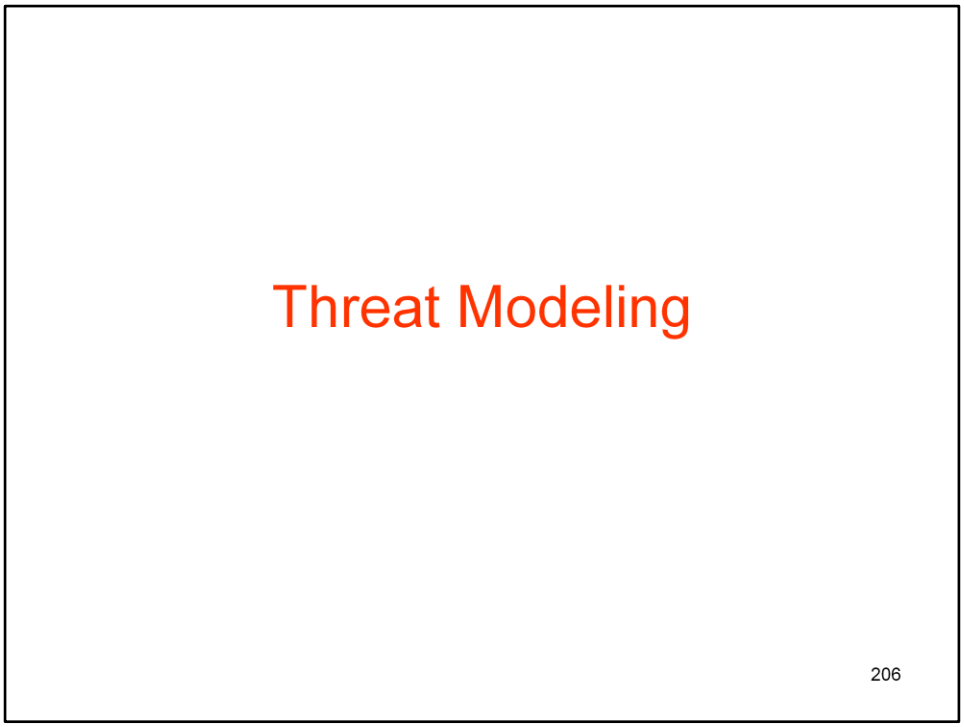
Security must be incorporated to all phases of software development lifecycle.

During the requirement phase, the project team can explore how security could be integrated into the development process, identify important security objectives, and list all security feature requirements. Security requirements may be requested by the client or may be required by regulation.

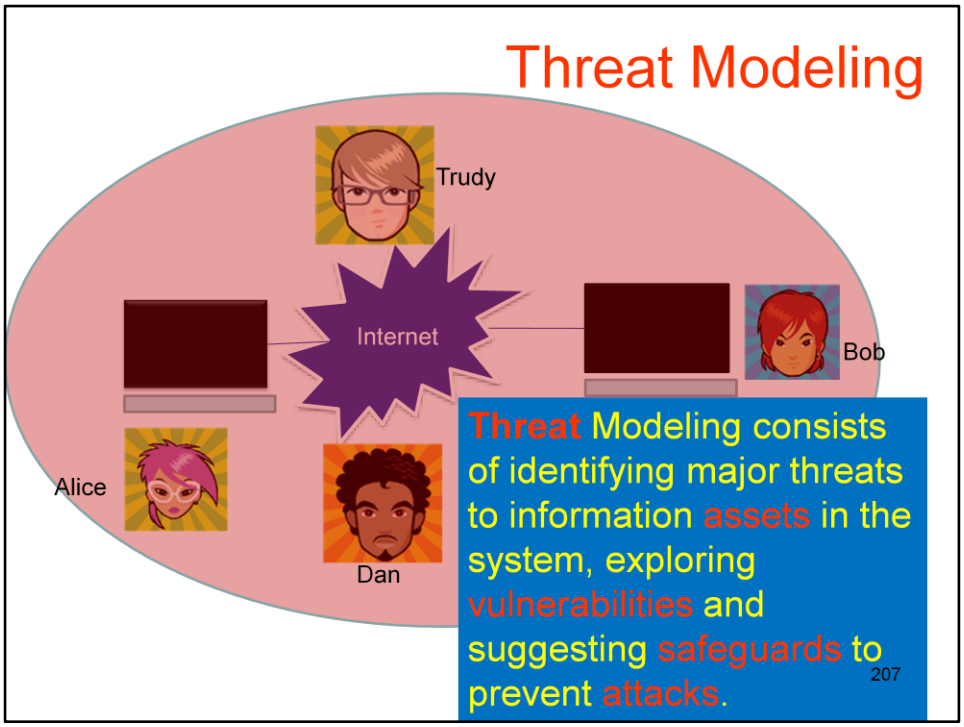
During the design phase security is achieved by threat modeling as explained later.

During the implementation phase secure coding guidelines must be followed.

During testing, testing tools such as fuzzer may be used to test the software. A fuzzer supplies invalid input to the program to determine vulnerabilities. Static code analysis tools may be used to detect programming flaws. Finally code reviews, where experts go through the code and identify potential problems.



Threat modeling is a systematic process to ensure application security.



Reference:

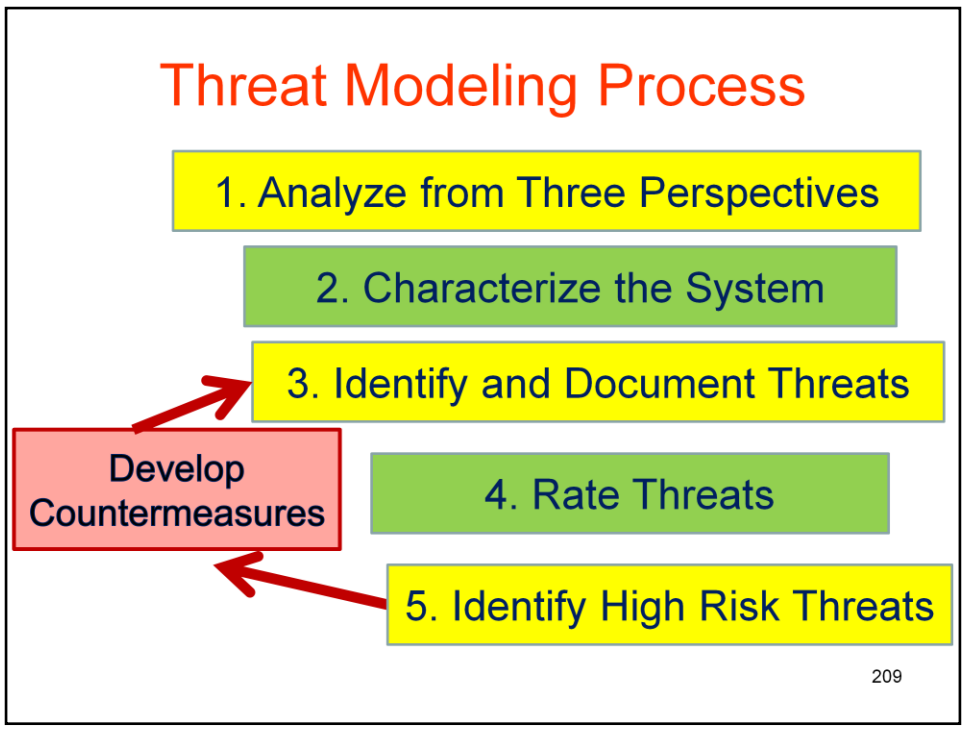
1. http://www.sans.org/reading_room/whitepapers/securecode/threat_modeling_a_process_to_ensure_application_security_1646
2. Threat Modeling (Microsoft Professional Books Series) *by* Frank Swiderski, Window Snyder, Window Snyder, *Microsoft Press, June 2004*

Data Flow Approach

- Follow the flow of data through the system
- Identify the key processes
- Identify threats to those processes considering both technical and human factors

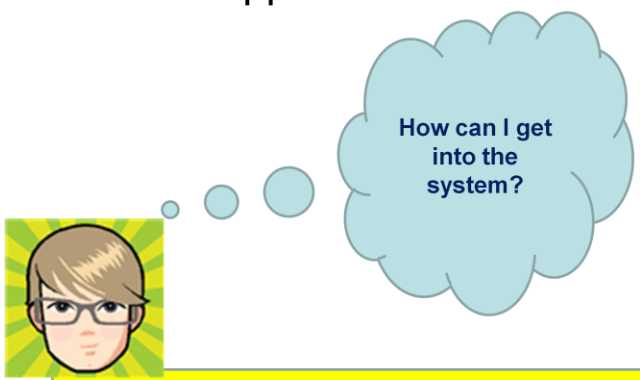
208

Data flow approach is one of the approaches for visualizing system design. However, other methods may be used as well.


Threat Modeling Process
1. Analyze from Three Perspectives
2. Characterize the System
3. Identify and Document Threats
Develop Countermeasures
4. Rate Threats
5. Identify High Risk Threats
209

Identify Entry & Exit Points

- Identify the points where data enters and exits the application



1. Analyze from Attackers' Perspective

Identify Assets

- Identify all information assets and list them
- Example: customer credit card information database

1. Analyze from Administrator's Perspective

Identify the trust levels

- Define privileges of an External Entity to access system through entry/exit points

1. Analyze from Administrator's Perspective

Identify User's Security Needs

- Find out and list user's security needs.
- What security functions are to be performed by the user?

1. Analyze from User's Perspective [213]

Use Scenarios

- How the system should be used or should not be used

User Model

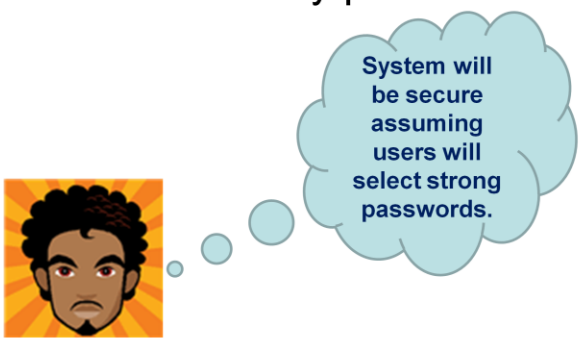
- Identify Usage Scenarios
- For each usage scenario:
  - Identify security changes intended by user in each usage scenario.
  - Identify the necessary feedback mechanism to inform users about the security state of the system.
- List users expectations about security and explain how those expectations are met

User Interface Design

- Design multiple design of each security related user Interface
- Perform usable security analysis and improve the interface as necessary

External Dependencies

- Define systems dependence on outside resources and security policies



External Security Notes

- Provides security and integration information
- For example, a note may require end users to change their password once in three months. A note may require the programmer to pre-filter the input for SQL injection attacks before submitting to the system.

Internal Security Notes

- Any notes that would make the threat model easier to understand
- Notes specify what is included and any concessions made in the design

Implementation Assumptions

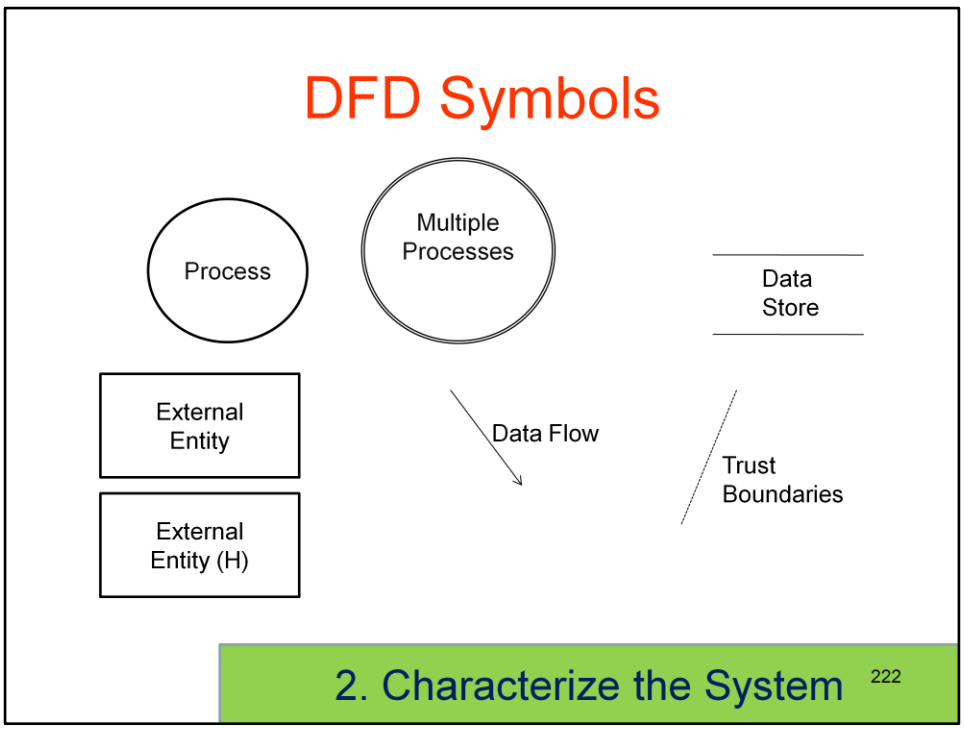
- Details of features that would be developed later

Who cares about the notes?

System Modeling

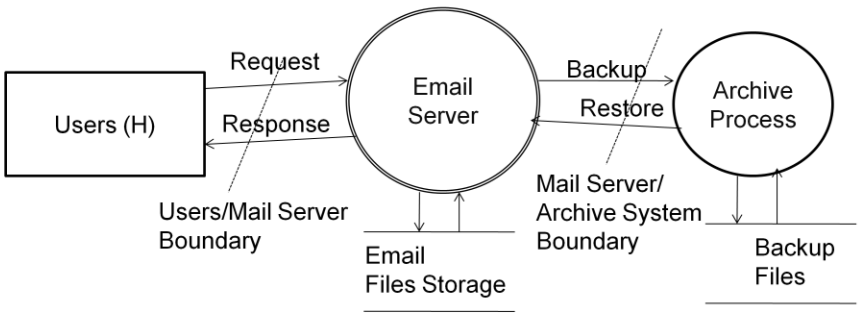
- Visual representation of how subsystems work together within the system.
- Data Flow Diagrams (DFD) may be used for visualizing the system.

We use DFD for simplicity. UML can be good choice as well.



For performing STRIDE+H threat analysis, in DFDs, the external entities may be identified with (H) to clearly distinguish them from other (non-human) external systems or entities. This will help us focus on identifying human factor threats.

DFD Email Example



Threat Profile

- Identify the goals of adversary
- Identify the vulnerabilities in the system because of these goals

3. Identify and Document Threats 224

Identify the threats

- Analyze each entry/exit point and determine how these may be attacked

3. Identify and Document Threats 225

Questions to be asked

- How can the adversary
  - Modify or control the system?
  - Retrieve information within the system?
  - Manipulate information within the system?
  - Cause the system to fail or become unusable?
  - Gain additional rights? [SANS05]

  (Exploiting both human and technical factors)

3. Identify and Document Threats 228

Questions to be asked

- Can the adversary access the asset
  - Without being audited?
  - And skip any access control checks?
  - And appear to be another user?

[SANS05]

STRIDE+H model

- Use security model (e.g. STRIDE+H) to classify and document threats
  - Spoofing (gaining access with false identity)
  - Tampering (integrity violation)
  - Repudiation (users may deny their actions)
  - Information Disclosure (confidentiality violation)
  - Denial of Service
  - Elevation of Privilege
  - Human Factors

3. Identify and Document Threats 228

Rate Threats w/ DREAD model

- DREAD model - threats may be rated on a 1-10 scale in terms of :
  - Damage Potential
  - Reproducibility
  - Exploitability
  - Affected Users
  - Discoverability

Identify High Risk Threats

- Rate all threats using DREAD model
- Pick the most important threats to be addressed

5. Identify High Risk Threats

Counter Measures

- Counter Measures could be a mechanism or policy to prevent or mitigate the threats
- Go back to Step 3 (Identify and Document Threats), if needed.

Develop Countermeasures

Example: Add two numbers

Requirement Analysis

Original Requirement Statement by the client:

"Add two numbers"

It would be tempting to go ahead and start implementing. Often, client gives you sketchy requirements, at best. So it is important that we find out more.

Analysis

- What are the other requirements?
  - Are these real numbers or integers?
  - What kind of interface must be provided?
- What are the security requirements?
  - Are these numbers confidential? (like salary)
  - What are the usable security requirements? (can users make typos?)
  - What are the integrity requirements (like overflow errors?)

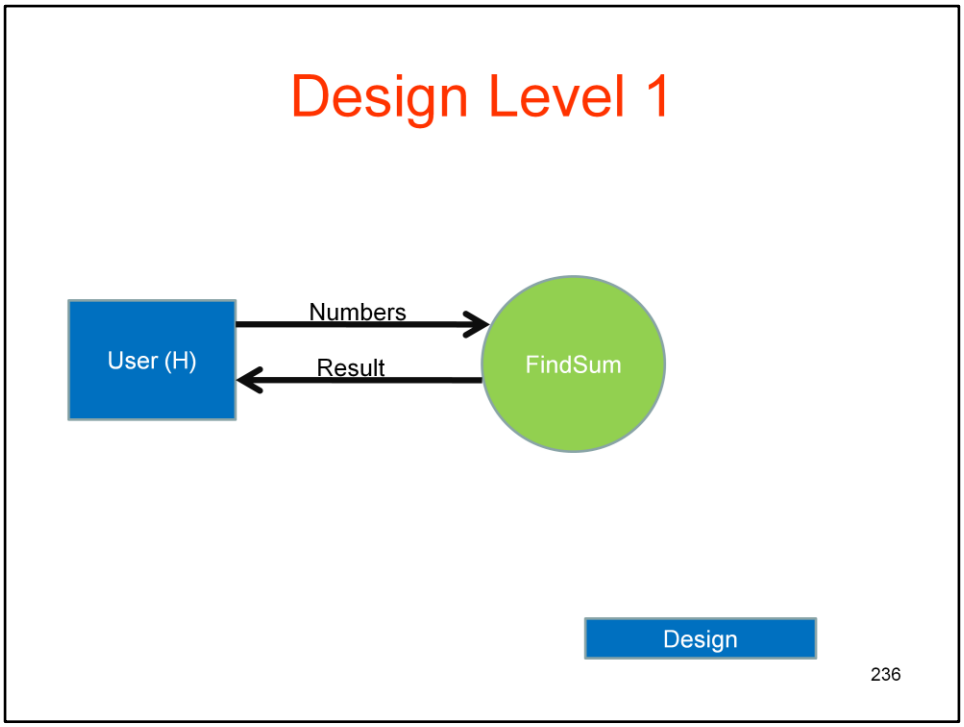
Requirement Analysis

234

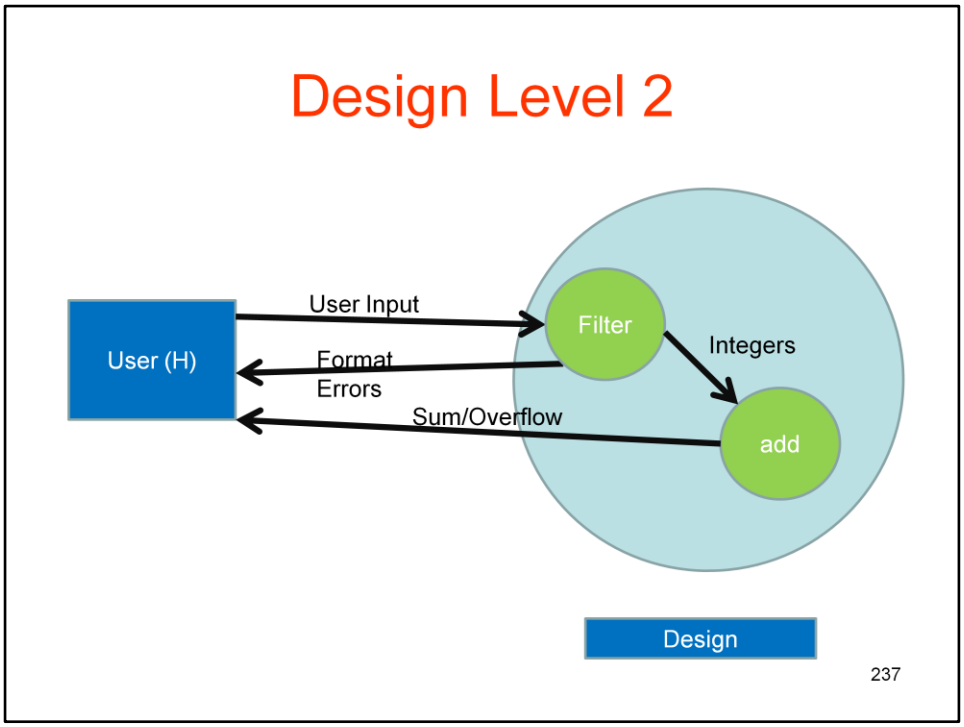
Ask additional questions to find out more information. The more information you can find, the better it is.

Revised Requirement Statement

- Users may enter two integers using a web form and the program should display the sum. Program should display errors when the input numbers are not in the right format or there are overflow errors.



Draw DFDs to visualize the system (you may also use UML). Entities are represented by rectangles. The processes are represented by circles.



Level 1 DFD is expanded. The filter process obtain two integers from the user. These are passed to add. Add adds these integers and displays the result for the user.

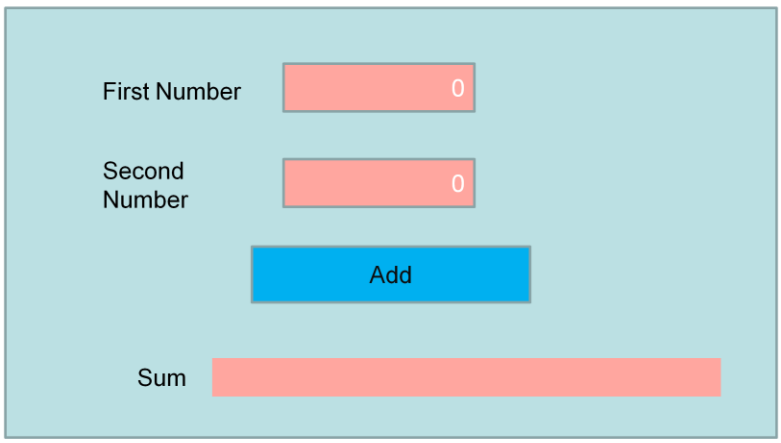
Modules

- Filter: prompt user for numbers, read numbers, make sure that the numbers are in the right format. Calls add method with two integers.
- Add: adds two integers and displays the result. If there is an overflow displays error.

Design

238

User Interface Design



239

Filter Method

```
int a, b;
boolean done=false;
while(!done)
{
Read two numbers
If (Numbers in the correct format)
   {
    add(number1, number2);
    done = true;
   }
}
```

Design

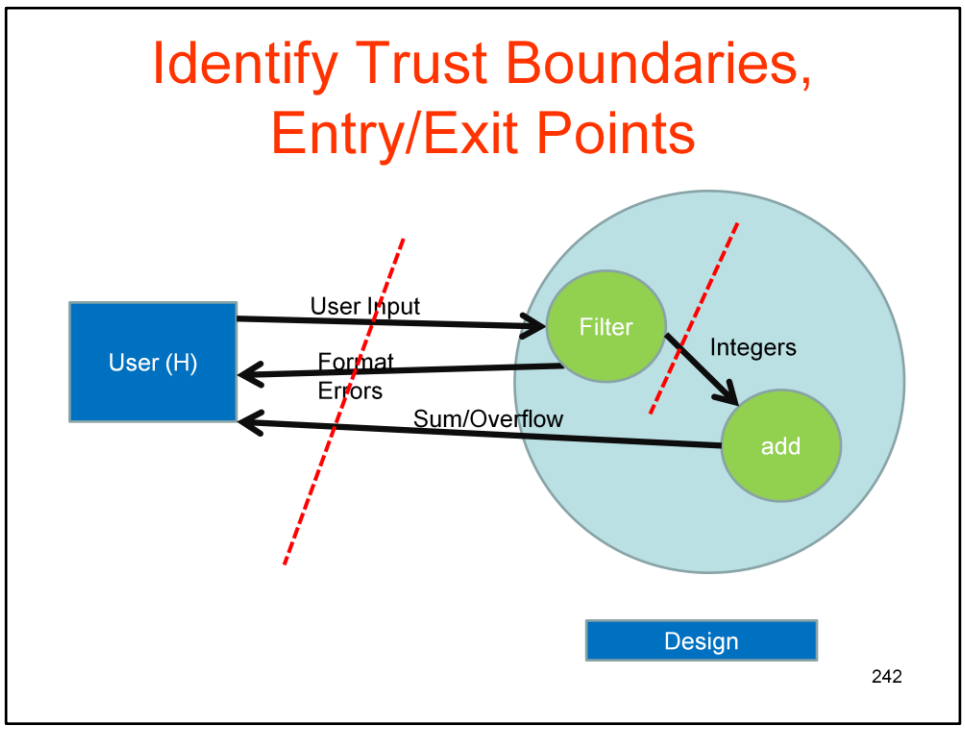
240

Write psuedo code for the method

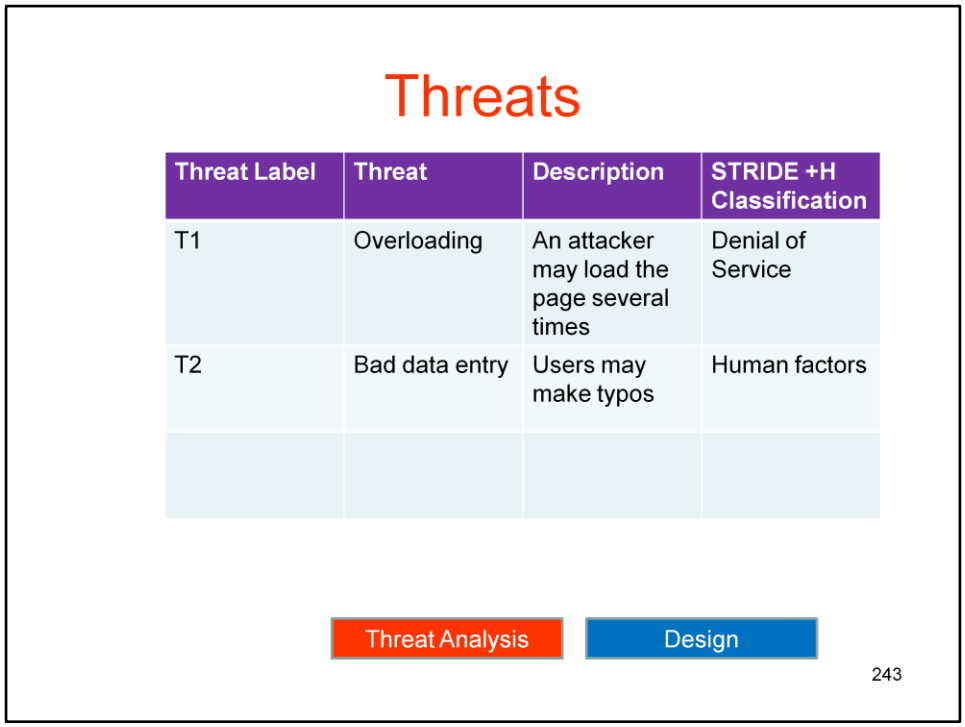
Add Method

```
Add(int a, int b)
{
int sum = a + b;
If (sum < a || sum < b)
    Out("Overflow error");
else
    Out(sum);
}
```

Design

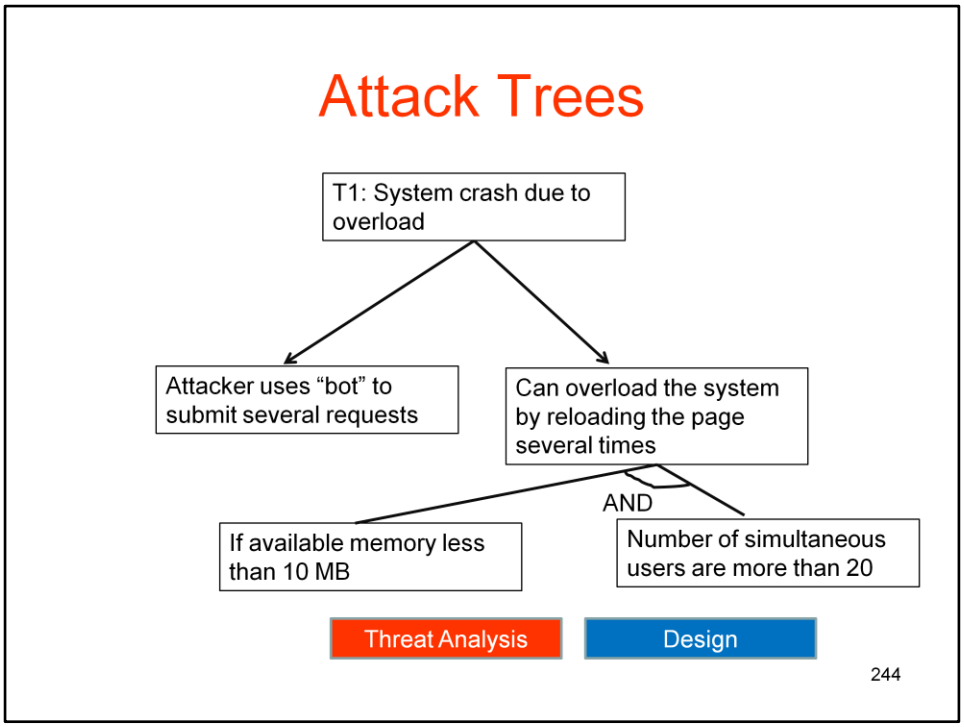


Trust boundaries are identified as above. Entry/Exit point to the system would be the web input form. Anyone can read the web page and submit information. Users are not allowed to modify web pages. The second trust boundary is between Filter and add. User's don't have direct access to add method. Add assumes that the inputs are integers – no error checking is done inside. Only applications can cross through this boundary.



Analyze system from three perspectives: Use, Defense and Offense to list potential threats.

Threats may be rated using a model (such as DREAD) to spot the important threats.



Draw attack tree to understand more about the threat.

The system may crash due to over load if:

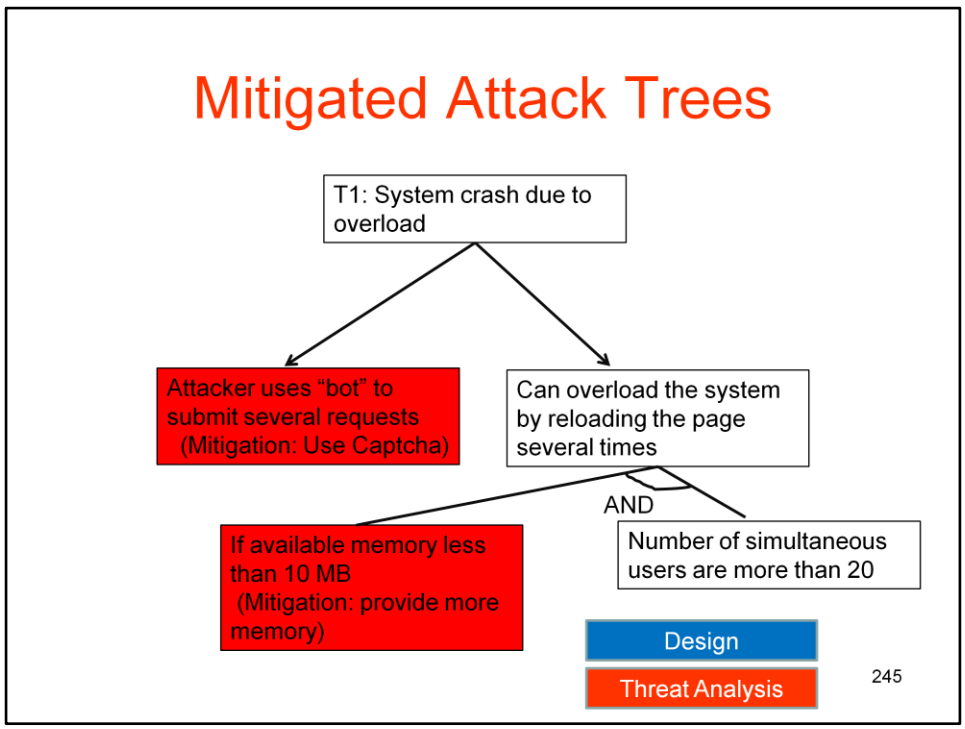
An attacker user uses bot to submit several automated requests

OR

If an attacker overload the system by reloading several pages (that is possible if the available memory is

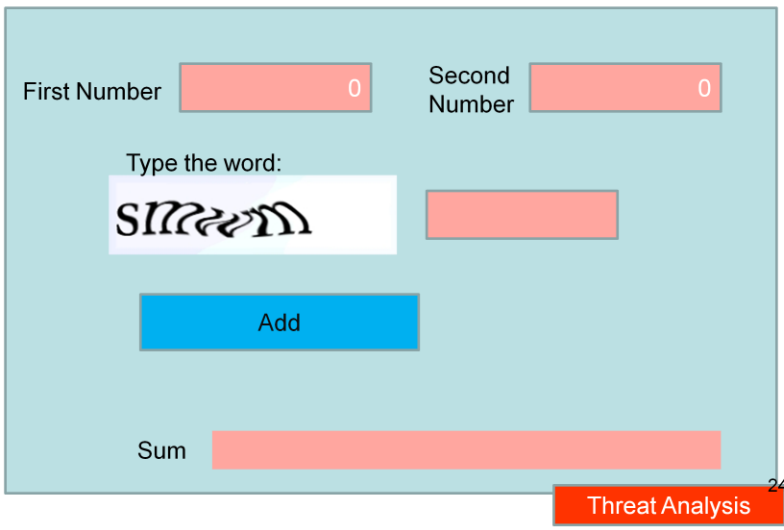
below 10 MB and the number of users are more than 20)

Attack Trees, B. Schneier http://www.schneier.com/attacktrees.pdf



The threat T1 is mitigated as follows: Provide Captcha, provide more memory.

User Interface Redesigned



246

Filter Method Redesigned

```
int a, b;
boolean done=false;
while(!done)
{
   if (captcha is correctly verified){
      Read two numbers
      If (Numbers in the correct format)
      {
         add(number1, number 2);
         done = true;
      }
   }
}
```

Threat Analysis | Design

247

Make necessary changes to mitigate the threat.

Add Method Redesigned

```
private Add(int a, int b)
{
   int sum = a + b;
   If (sum < a || sum < b)
      Out("Overflow error");
   else
      Out(sum);
}
```

Design

248

No direct access is allowed to this method. (this is just an example of using a safeguard – this is definitely implementation dependent)

Test Plan

- Develop a test plan for each module.

Example, for filter module: if the following inputs are specified, the expected output would be:

First Number	Second Number	Expected Output/Action
1	2	Call add
23.3	14.6	Round numbers and call add
xslkl	2	Display error
		Display Error
3O	2I	Display Error

Design

249

For each module, specify how you are going to test it.

Implementation Strategy

- Select appropriate implementation platform (e.g., Java, Netbeans, etc.)
- Specify network security mechanisms (such as SSL)
- Specify Operating System Security Mechanisms (e.g., program is made read/execute – no write permission for user)
- Specify application Security (e.g., how to handle overflows)

Design

250

Perform additional threat modeling based on selected implementation platform, operating system, etc.

Implementation and Testing

- Follow secure coding practices (see the next slide)
- Test each module; Integrate and test
- Use automatic web input generators for testing

Software Security Principles [viega]

- Secure the weakest link
- Practice defense in depth
- Fail security – don't fall apart
- Assign the least possible privilege to do the task
- Compartmentalize systems and isolate security problems
- Make it as simple as possible
- Promote privacy
- Can't hide security problems or vulnerabilities
- Trust as little as possible
- Use community resources for security

252

For developing secure software, a number of design principles should be followed. First, secure the weakest link. For example, the weakest link may be an end-user thus GUI design must follow the usable security requirements. Write simple code or may be misunderstood or may be difficult to debug and can result in programmer (human) errors. Practice defense in depth – i.e., add multiple layers of security. Fail Security means – the program should not fall apart because of an error – it should be handled. Assign least possible privilege to do the task. Compartmentalizing system would help prevent problem in one compartment from affecting other areas of the system. System design should be simple and easy to follow. Assume that information by de-facto is private unless stated otherwise. Don't assume that security problems or vulnerabilities can be hidden from a possible attacker. Do not trust. Use community resources for security.

Secure Coding Guidelines for Java are available at:

http://www.oracle.com/technetwork/java/seccodeguide-139067.html

Case Study: E-Tailing Application

253

E-Tailing application (Developing an online store such as amazon.com)

Case Study: Application Name and Description

- Web E-Tail application using a database server as backend and a web browser as the front end.

Case Study: Security Objectives

- Protect the confidentiality of customer data
- Ensure availability of application
- Ensure the integrity of the data
- Ensure non-repudiation of customer /employee actions
- Ensure usability of security functions

Case Study: Application Overview

- Uses web-browser as the front end; customers can browse product catalog
- Uses database server as the backend
- Business Logic resides on the web server

256

Identify Entry & Exit Points

ID	Name	Description	Trust Levels
E1	HTTP Port	Non-secure web pages accessed through this port	Anonymous user Authenticated Users Non-Authenticated Users Administrators
E2	Login Page	Users enter their credential to enter	Authenticated Users Non-Authenticated Users Administrators
E3	Customer Profile Page	Display customer contact information	Authenticated Users Administrators

1. Analyze from Attackers' Viewpoint

257

Identify Assets

ID	Name	Description	Trust Levels
A1	User Login Data	Login password, Last login, etc.	Authenticated Users Administrators
A2	Administrator Data Information	Login password, Last login, etc.	Administrators
A3	Customer Profile Page	Customer address, phone number, email	Authenticated Users Administrators

(Assets an attacker would be interested in)

1. Analyze from Attackers' Viewpoint

Identify the trust levels

ID	Trust Levels	Description
T1	Anonymous User	Any user who accesses the website, but not logged in
T2	Authenticated Users	Internet Users submitted valid credentials
T3	Administrator Read Only	Administrator Access; but read only. Cannot update or change information.

1. Analyze from Attackers' Viewpoint

Use Case Scenarios

ID	Use Case Scenario
U1	An anonymous user accesses the system for browsing products
U2	An anonymous user adds product into the shopping cart
U3	User logs into save the shopping cart
U4	User tags the computer so he/she does not have to enter password to login next time

User Model

- U4: User tags the computer so that he/she does not have to enter password to login next time
  - The user identifying information should be stored into the Session – Automatic login should be enabled
  - Users should be advised against using this feature in public computers; should be told whether the feature is currently enabled and should be informed how this feature may be disabled, if users wish so.
- User expects that his/her order history would be private. This expectation is met by requiring user authentication before access to history.

Design User Interface and Conduct Usability Studies

- Example: Login Page

262

Conduct Usability Studies to improve the interface

Inquiry: survey

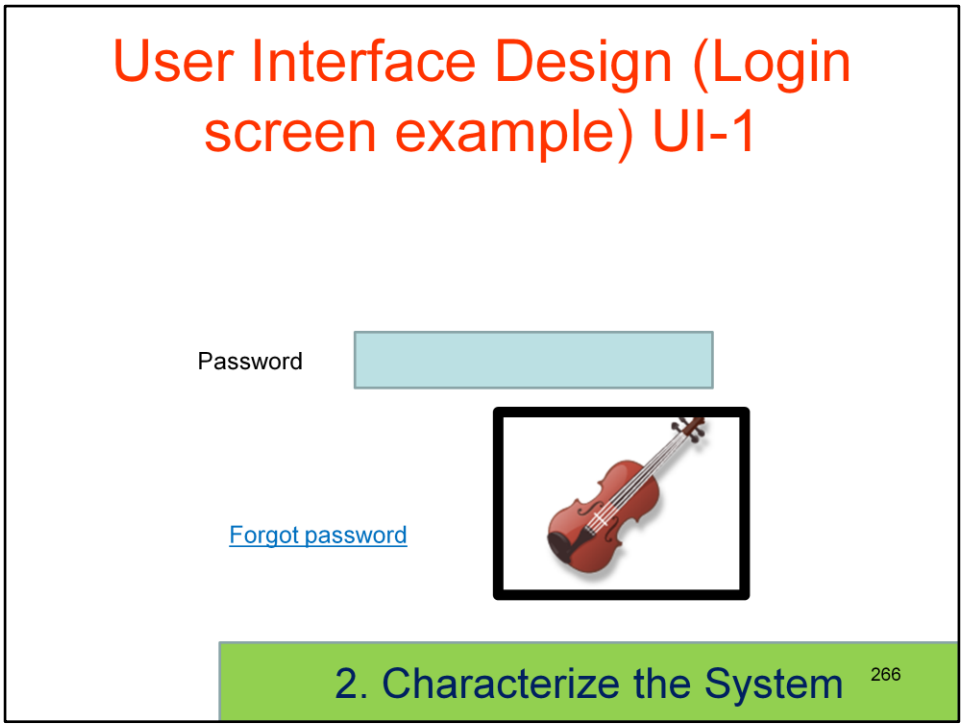
- What features would you like on the login page?

Inspection: Cognitive Walkthrough

- Talk about different functionalities of login page
- What must be included – how would the users use them?

Testing

- Low-fidelity prototyping (see next slide) and test using “Think Aloud Protocol”



A personal picture selected during the signup process can act as the login button. This is to reassure users that they are at the right site. The username may be either entered in the previous screen or obtained from session. In this step, security related interfaces should be designed and usability analysis should be performed. (See Usable Security Section)

External Dependencies

ID	External Dependencies
ED1	E-Tailing Application would execute on Linux System
ED2	Apache web server will be used
ED3	MySQL will be the database

External Security Notes

ID	External Security Notes
ES1	Website administrator should upgrade apache and enable SSL
ES2	Database administrator should disable windows login

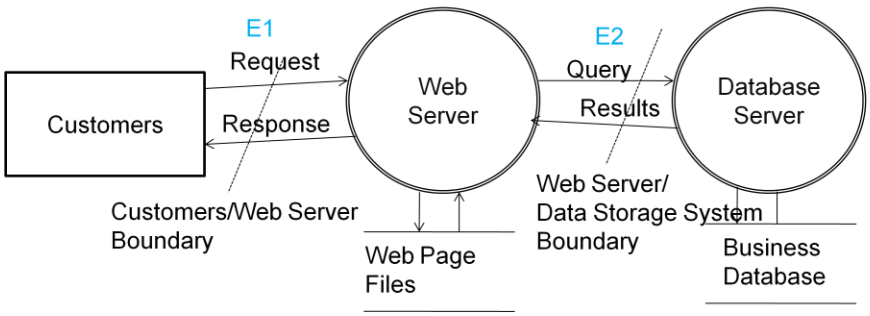
Internal Security Notes

ID	Internal Security Notes
IS1	Database Server trusts the connections from the server side application.
IS2	Web Application authenticates Internet Users

Implementation Assumptions

ID	Implementation Assumptions
IA1	Current Implementation does not support paypal payment system
IA2	No webservices are provided at this point

Dataflow Diagram



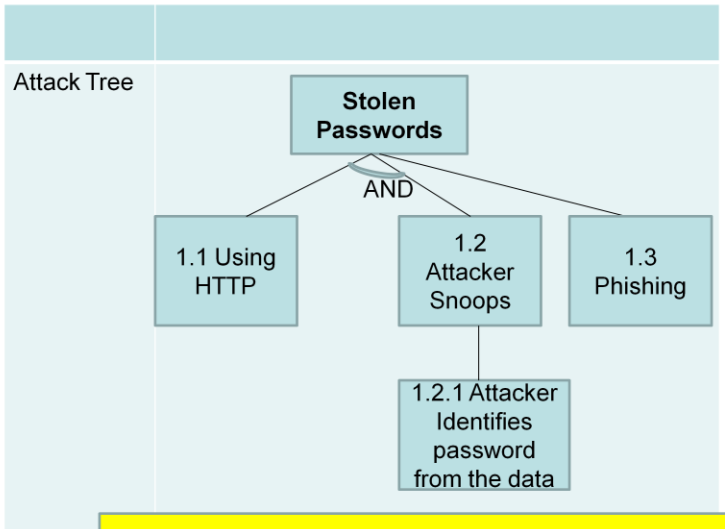
2. Characterize the System 271

Threat T1

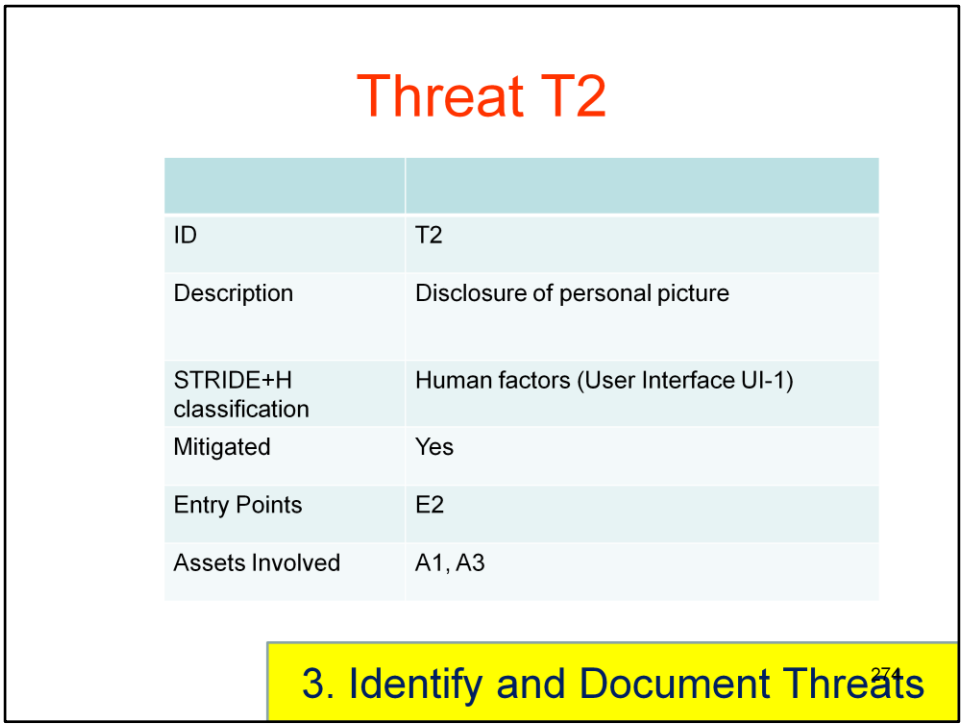
ID	T1
Description	Disclosure of private information; if an attacker uses stolen user credentials, will be able to access the users account
STRIDE+H classification	Spoofing; Information Disclosure
Mitigated	Yes
Entry Points	E1, E2
Assets Involved	A1, A3

3. Identify and Document Threats 272

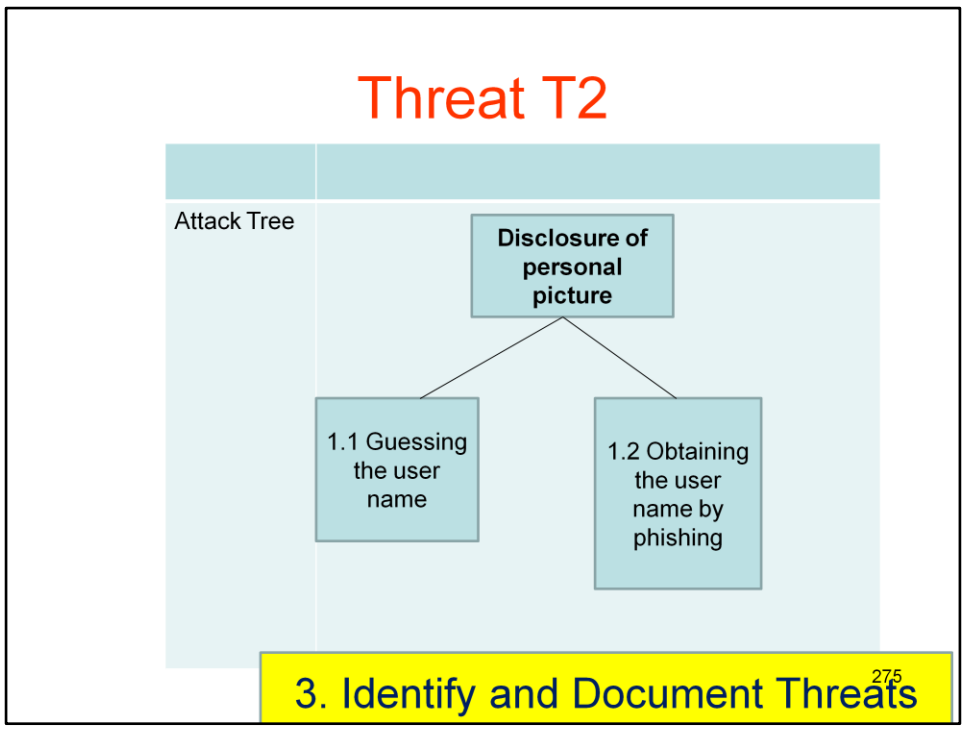
Threat T1



3. Identify and Document Threats



Disclosure of personal picture may make it easy for someone to obtain the password by conducting a phishing attack.



Guessing the username by knowing the first name and last name of the user may be mitigated by encouraging users to select unpredictable user names. To reduce a site obtaining the user name by phishing and then performing a man-in-the-middle attack to display the personal picture on phisher's website, users may be asked randomly selected additional security questions if the request is coming from an untagged computer.

DREAD Rating

- Rate each threat using DREAD:
  - Damage Potential
  - Reproducibility
  - Exploitability
  - Affected Users
  - Discoverability

4. Rate Threats 276

DREAD Ratings for Threat T1

Category	Rating (Out of 10)
Damage Potential	8
Reproducibility	9
Exploitability	2
Affected Users	2
Discoverability	9
Average	**6**

4. Rate Threats 277

Active Learning Task

- Find and document other threats (draw attack trees) and calculate DREAD ratings
- Identify the most important threats

5. Identify High Risk Threats

Review: Identify Vulnerabilities

- Identify vulnerabilities for the most important threats
- Find solutions; i.e., by designing safeguards

Develop Countermeasures

279

Vulnerability Analysis

ID	V1
Description	Web Server accepts HTTP connections
STRIDE+H classification	Spoofing; Information Disclosure
Threat ID	T1
Mitigated?	Yes
Mitigation	Place Login forms in the HTTPS area; Disable Administrator login from Web.
Entry Points	E1, E2
Assets Involved	A1, A2, A3

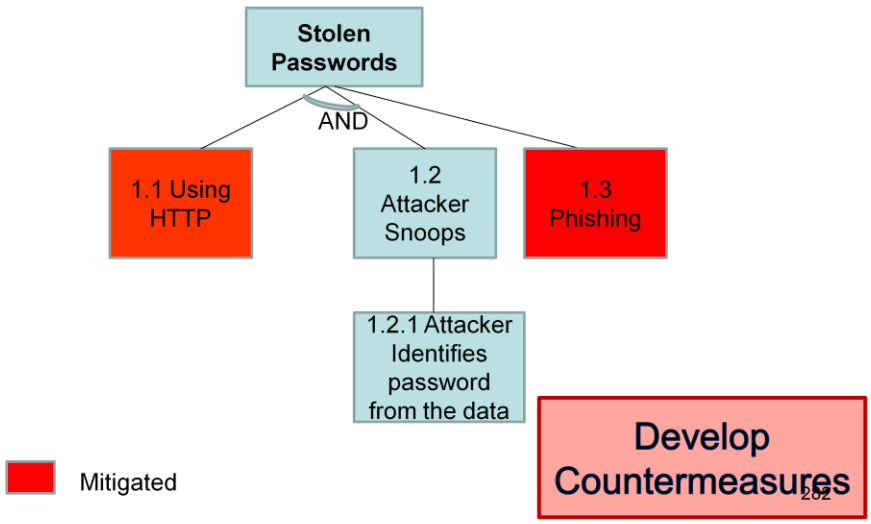
Develop Countermeasures

Vulnerability Analysis

ID	V2
Description	Users may be subject to phishing attacks
STRIDE+H classification	Spoofing; Information Disclosure
Threat ID	T1
Mitigated?	Yes
Mitigation	Employ Phishing filters; Educate users about the phishing attacks
Entry Points	E1, E2
Assets Involved	A1, A3

Develop Countermeasures

Mitigated Attack Tree



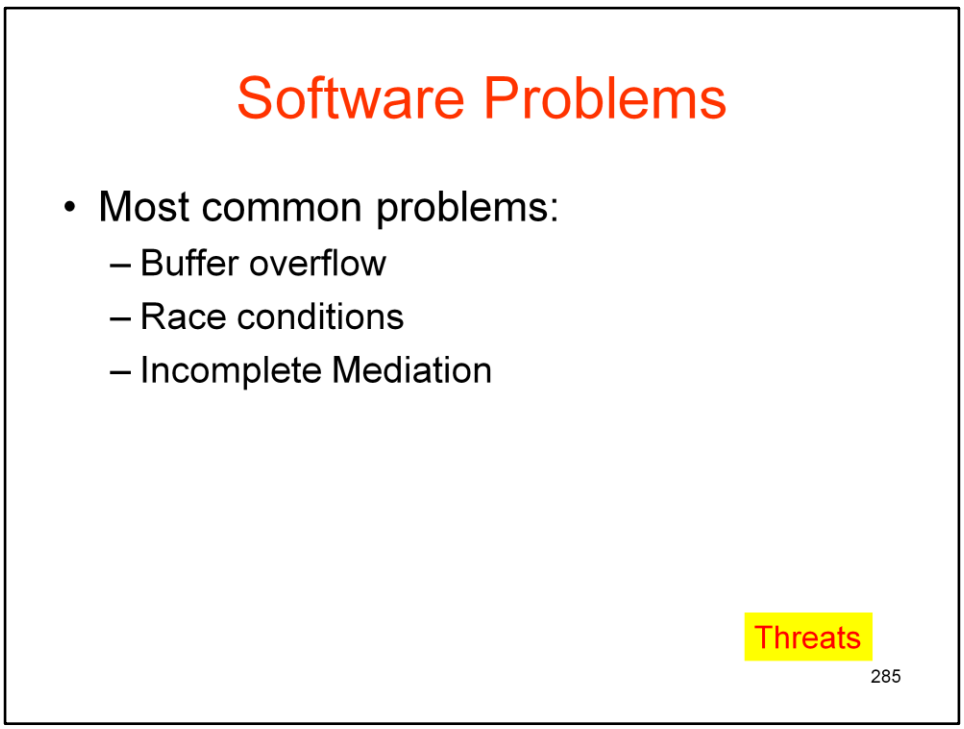
Active Learning Task

- Complete the threat analysis, identify potential vulnerabilities, suggest safeguards, and draw the mitigated attack trees.
- Reminder: Build Defense in depth!!!

283

Assignment

- Perform threat analysis on an application you like (such as facebook, webCT, or Zimbra). Identify potential vulnerabilities from both technical and human perspectives and suggest safeguards; draw attack trees and mitigated attack trees.



In this section we take a closer look at Software Security. Major problems with software security are, buffer overflow, race conditions and incomplete mediation.

Buffer Overflow

- char data[30];
  int loggedIn = 0;
  data[30] = 1; //buffer overflow
  if (!loggedIn)
      askForPassword();
- A buffer overflow can cause the program to malfunction or can be carefully manipulated to alter the control flow

Threats

286

Buffer overflow is shown in the slide, where there is a space of 30 bytes are reserved for data. Exceeding the allocated space by storing 1 in the location data[30], creates a buffer overflow causing the program to circumvent and important security check. (Programmer needs some usable security as well – just remember that C starts counting from 0)

A great link for OS related vulnerabilities:

http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/15_Security.html

Incomplete Mediation

- If data exchanged between components of the system (such as client and server) is compromised or altered system may fail.
- Example: Form data

http://www.wiu.edu/users/mfbg/submit?user='binto'&operation='viewaccount'

submitted from a web browser may be altered before reaching the server

Threats

287

The data is exchanged between a web client and server as part of URL (in GET mode). This URL may be intercepted by an adversary and be changed, thus server receiving unexpected input.

http://www.wiu.edu/users/mfbg/submit?user='binto'&operation='viewaccount'

An adversary can intercept this and change the operation into, say, "changepassword", or much worse, something invalid. This confuses the server and the server may shutdown.

Race Conditions

- Security processes may have multiple steps, but they are supposed to be atomic, i.e., execute all or nothing.
- An attacker may be able to make some changes between steps thereby compromising security

Threats

288

For example an attacker can try to stop a program after the verification phase and before the execution phase. For example, a system program may need to create a temporary file. After the program creates a temporary file, an attacker can replace the temporary file with a link to the system password file. Once this is done, when the former program writes to the temporary file, the linked password file would be overwritten.

Malware

- Virus – relies on an external entity to propagate from one system to another
- Worm – worm propagates itself without help on an external entity
- Trojan horse – appear to be a useful program doing something bad behind
- Trapdoor – with a trapdoor added to a system someone can access it
- Rabbit – A program designed to exhaust all system resources.

Threats 289

A virus may be a file virus, macro virus, boot sector virus or partition virus. An example for trapdoor would be a bank consultant leaving a secret entry point into the system, which may be used later to access customer information. An easy to use firewall software and antivirus scanners can detect and remove most malware or protect systems from attacks.

Human Factor Threats

- Can the page may be spoofed?
- Does it have a recognizable URL? (or could be easily confused with a fake URL)
- Are the passwords guessable?
- Can users forget their password?
- Will users set one of the most common passwords?
- Will users set the same password on multiple websites?

290

More Human Factor Threats

- Is the secondary authentication is as strong as primary authentication?
- Can the password lock out mechanism be used for Denial of Service?
- Could the password be captured during login?

Denial of Service (DoS) Attacks

Threats

DoS Attacks

- DoS attacks don't often include intrusion to the system
- Prevents legitimate users from accessing the system
- Works by overloading the system by exceeding the limits of the system such as memory, disk space or network bandwidth

Threats

293

DoS attacks work by making the system to do more than it can handle thus slowing it down or crashing it.

DoS Attack Examples

- TCP SYN Flood Attack
- Smurf IP Attack
- UDP Flood Attack
- ICMP Flood Attack
- Ping of Death
- Teardrop Attack
- Land Attack
- Echo/Chargen Attack

Threats

294

TCP SYN Flood Attack

- Memory space is allocated on initiation of every TCP/IP connection
- Requesting several connections and leaving them half open server can run out of buffer space
- Legitimate connections are delayed or timed out because of this

Threats

295

Prevention techniques

- SYN Cookies: instead of allocating memory, it sends a SYNACK with an indentifying "cookie" which the client needs to send back to initiate the connection
- RST Cookies: server sends wrong SYNACK; in normal case client should then generate RST to indicate that something is wrong

Threats

296

Prevention Techniques

- Stack Tweaking: reduce the timeout in the TCP stack thus removing the half open connection from the queue faster

Bottom line: don't allocate resources until verifying that the connection is legitimate.

Threats

297

Land Attack

- Attacker sends a forged packet with same source address and destination IP address (of the victim)
- Host sends messages back and forth to itself
- This causes the system to shutdown

Threats

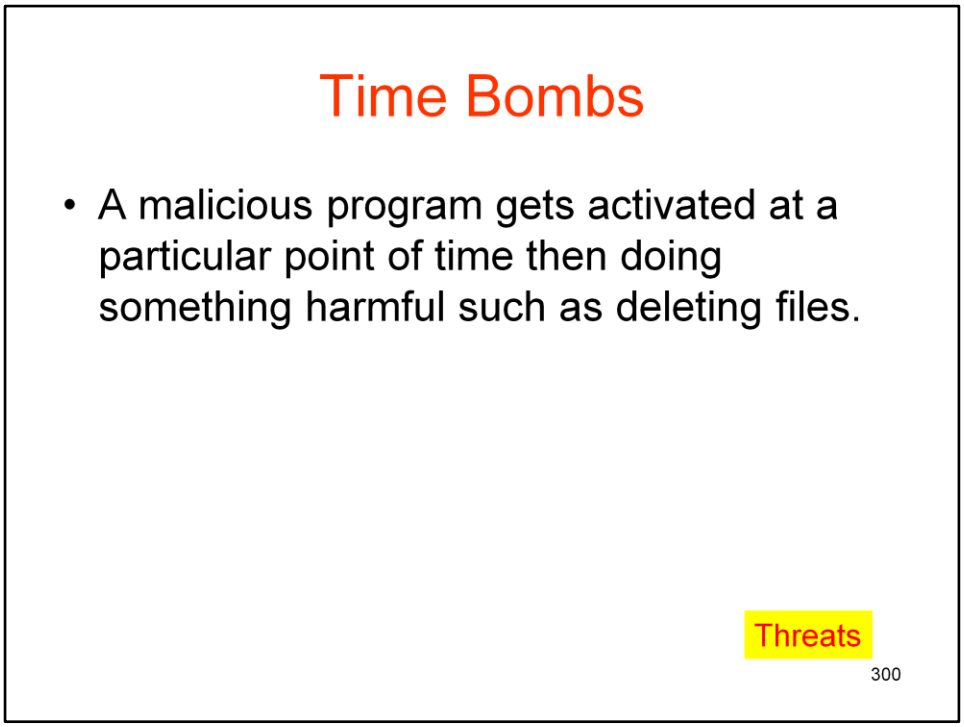
298

Echo/Chargen Attack

- Character Generator (chargen) service generates a random stream of characters
- Echo service just bounces the packet back to the source
- An attacker can simply connect a chargen service to echo service thus generating large amount of data in the network

Threats

299



An employee leaving an organization could plant a “time bomb”, which may be activated at a pre-programmed time. A terrorist may setup time bombs on many system, triggering at particular time.

Software Reverse Engineering

- Someone can analyze a software and find out how it works and reverse engineer it
- This can be accomplished by using a debugger and setting break points

Safeguards

301

Setting break points will let an attacker to stop at different points, examine memory locations, and processor registers to find out what a program does.

Anti-disassembly

- An executable file can be encrypted
- But the code must be decrypted before it can be executed
- Some invalid instructions may be added to confuse someone trying to disassemble

Safeguards

302

Although code may be encrypted for protection, before executing it, it has to be decrypted thereby subjecting the code to attack. To confuse an attacker, some invalid instructions may be added to the code, but this doesn't completely prevent attacks although this can considerably increase the time for an attacker to understand the code.

Anti-debugging techniques

- A program can monitor certain flags and see if the program is being debugged, if so the program can be terminated
- An instruction can dynamically correct the about to be executed instruction by overwriting it, and the program works fine if instructions are pre-fetched
- In debug mode instructions are not pre-fetched thus the program crashes.

Safeguards

303

The technique involves, deliberately keeping some instructions incorrect. These incorrect instructions can be dynamically corrected at the time of execution. Modern processors pre-fetch instructions for efficiency. Thus, many instructions are brought into the processer instruction cache. In debug mode, however, pre-fetch is disabled. If there is no pre-fetch the instruction never gets corrected, thus executing incorrect code.

Software Tamper resistance

- Guards – using a hash check parts of the program could be verified
- Obfuscation – makes the code difficult to understand
- Metamorphism – change copies of software to defeat the “break one, break all” paradigm

Safeguards

304

Metamorphism prevents an attacker from designing code which will affect all copies of an executable program. By making the executable codes different, it would be hard for an attacker to design code that would affect all computers the same.

Digital Rights Management

- Attempt to retain control over the digital contents even after the distribution to enforce conditions such as:
  - No copying
  - Read just once
  - Do not open until a particular time
  - No forwarding or making additional copies

Safeguards

305

Digital rights management is an attempt to manage control over digital products. For example, a music file may be sold for playing 10 times, but not more.

DRM Example

- Adobe PDF can have passwords to open or other constraints enforced by acrobat reader.

Safeguards

306

Active Learning Task

- Research on DRM has shown that removal of the restrictive DRM policies can in fact reduce piracy! Discuss this counter-intuitive result.

Reference

- **Improving Web Application Security: Threats and Countermeasures** J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla and Anandha Murukan
  Microsoft Corporation
- Threat Modeling: A Process To Ensure Application Security, SANS Institute