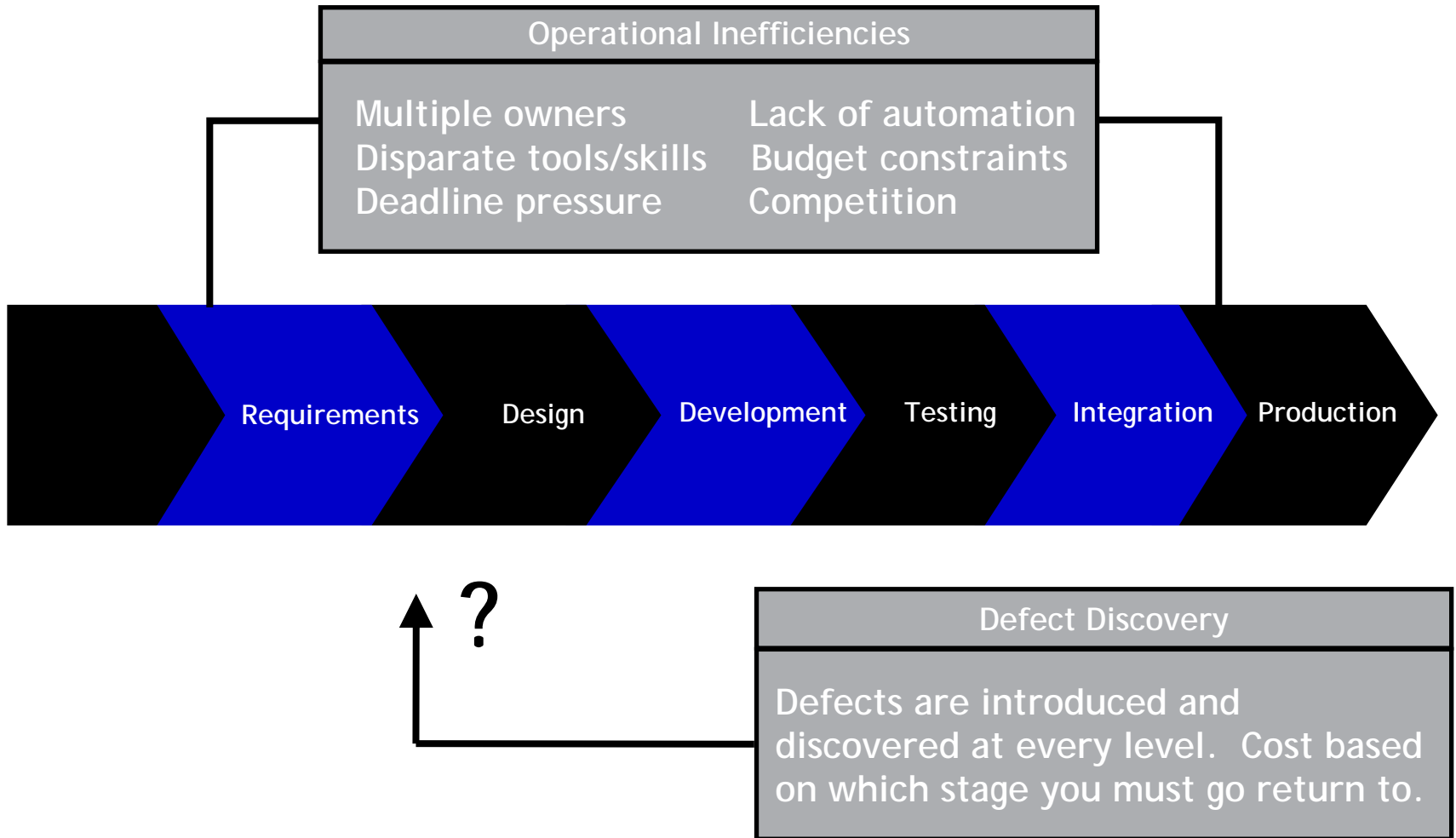


Watchfire

Web Application Security

connet

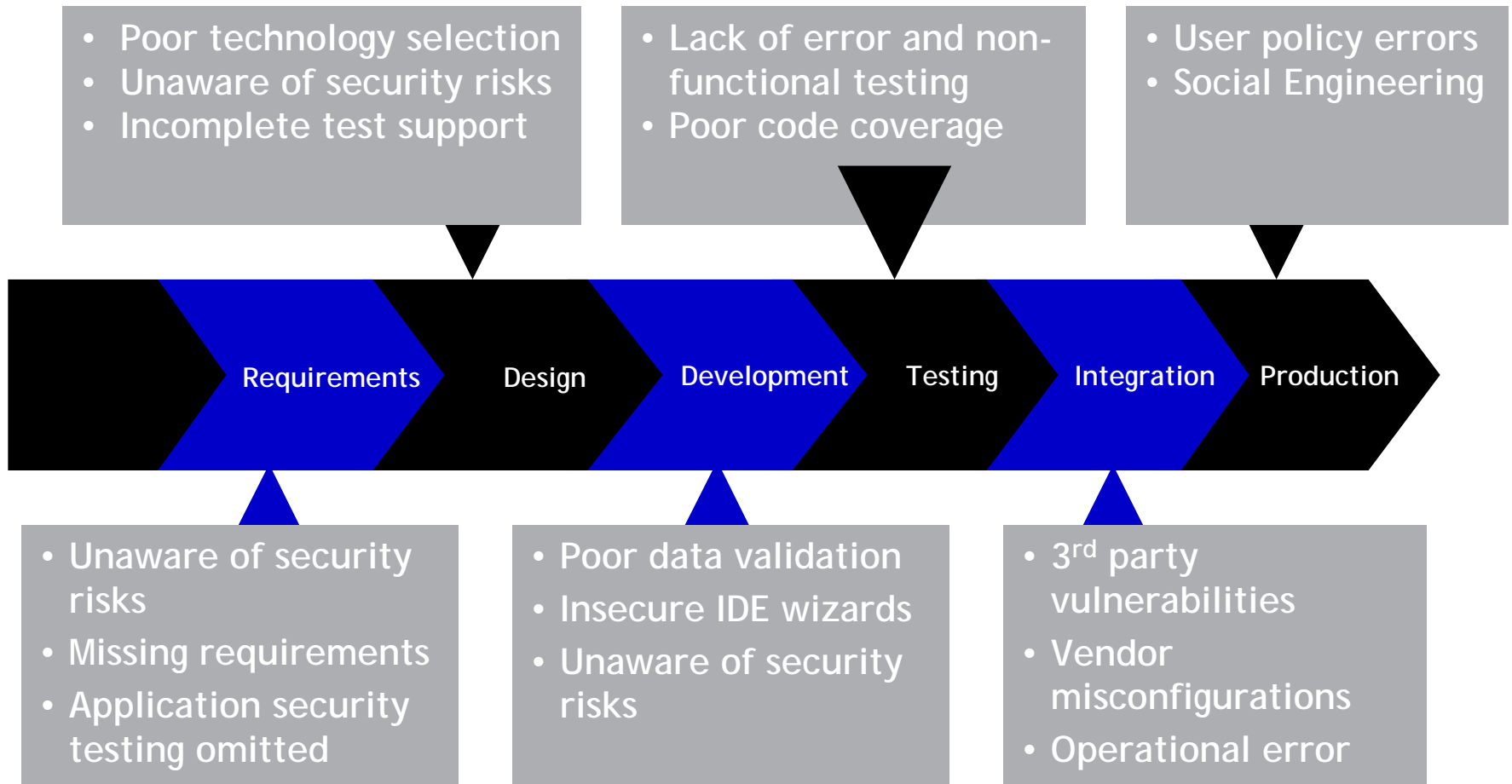
Current processes lead to security cost challenges



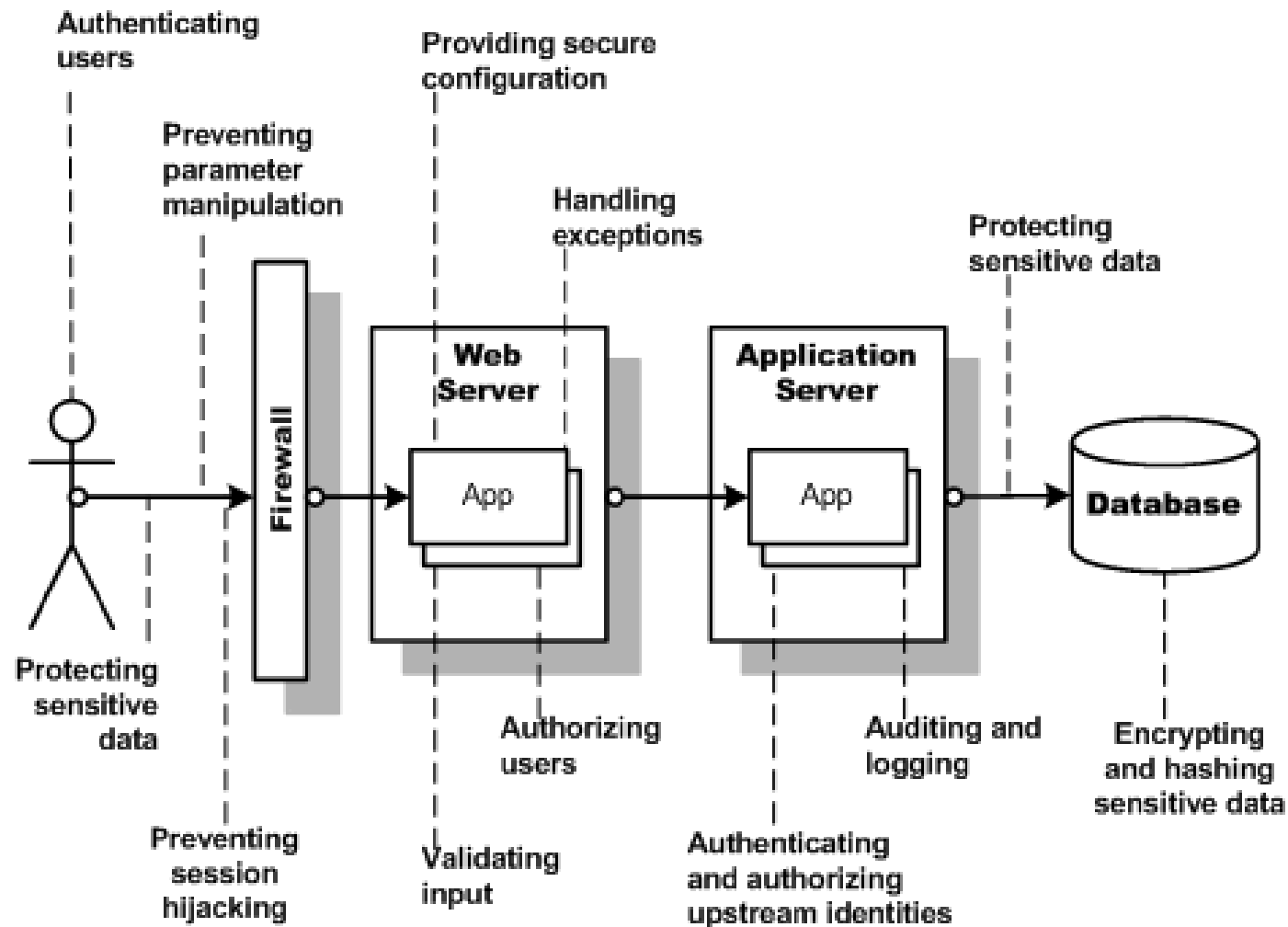
How much do security defects cost?

	Found in Design	Found in Coding	Found in Integration	Found in Beta	Found in GA
Design Errors	1x	5x	10x	15x	30x
Coding Errors		1x	10x	20x	30x
Integration Errors			1x	10x	20x

What are the common causes of defects?



What are the typical threats applications face



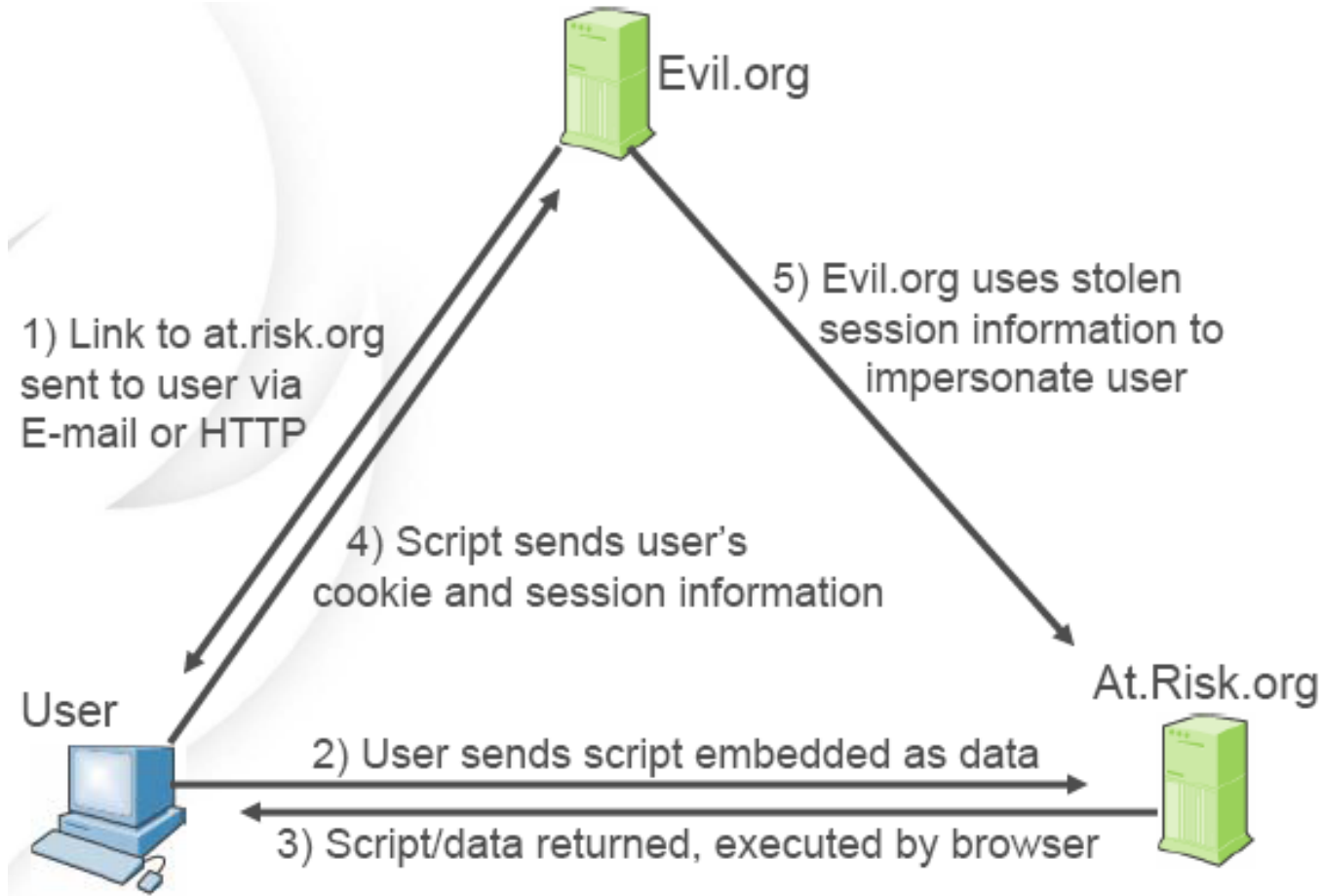
What is your Application Security Maturity?

Optimizing	<ul style="list-style-type: none">• Striving for continuous process improvement
Quantitatively Managed	<ul style="list-style-type: none">• Security reviews completed in all process stages• Published organizational standards for secure development
Defined	<ul style="list-style-type: none">• Early stage security reviews• Security awareness is provided for all process stages• Predetermined policy is enforced for the entire organization
Managed	<ul style="list-style-type: none">• Audits• Use of automated scanning tools prior to deployment• Organizational policy statement (exceptions permitted)
Performed	<ul style="list-style-type: none">• Infrastructure penetration tests• Application policy statement (limited enforcement)

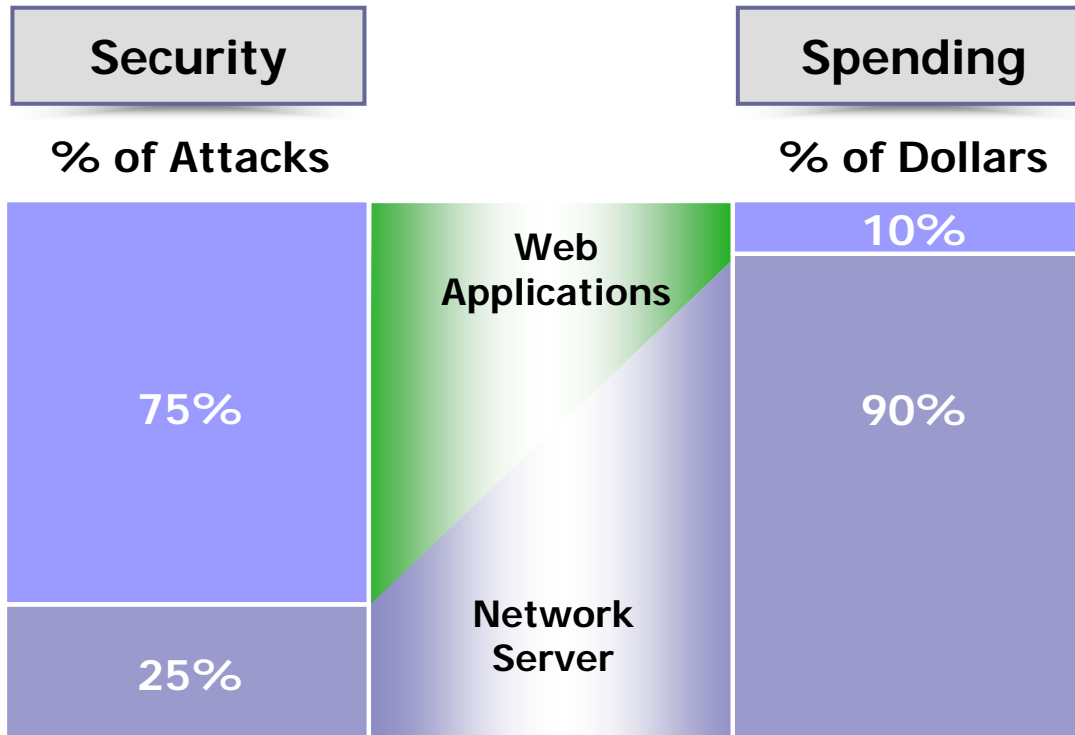
Security Defects: Those I manage vs. Those I own

	Common Web Vulnerabilities (CWVs)	Application Specific Vulnerabilities (ASVs)
Cause of Defect	Insecure application development by 3rd party SW	Insecure application development In-house
Location within Application	3rd party technical building blocks or infrastructure (web servers or databases)	Business logic -dynamic data consumed by an application
Type(s) of Exploits	Known vulnerabilities (patches issued), misconfiguration	SQL injection, path tampering, Cross site scripting, Suspect content & cookie poisoning
Detection	Match signatures & check for known misconfigurations.	Requires application specific knowledge
Business Risk	Patch latency primary issue	Requires automatic application lifecycle security
Cost Control	As secure as 3rd party software	Early detection saves \$\$\$

Cross Site Scripting – The Process



Security and Spending Are Unbalanced



- Buffer Overflow
- Cookie Poisoning
- Hidden Fields
- Cross Site Scripting
- Stealth Commanding
- Parameter Tampering
- Forceful Browsing
- SQL Injection
- Etc...

75% of All Attacks on Information Security
Are Directed to the Web Application Layer

2/3 of All Web Applications Are Vulnerable

Gartner

Web Application Hacks are a Business Issue

Application Threat	Negative Impact	Potential Business Impact
Buffer overflow	Denial of Service (DoS)	Site Unavailable; Customers Gone
Cookie poisoning	Session Hijacking	Larceny, theft
Hidden fields	Site Alteration	Illegal transactions
Debug options	Admin Access	Unauthorized access, privacy liability, site compromised
Cross Site scripting	Identity Theft	Larceny, theft, customer mistrust
Stealth Commanding	Access O/S and Application	Access to non-public personal information, fraud, etc.
Parameter Tampering	Fraud, Data Theft	Alter distributions and transfer accounts
Forceful Browsing/ SQL Injection	Unauthorized Site/Data Access	Read/write access to customer databases

Reporting System

Create Report [X]

Report Type | Layout

Select Report Type:

- Security Report
- Industry Standard
- Regulatory Compliance

Template: Developer

Executive Summary

Issues

- Include User Comments
- Include Requests
- Include Advisory and Fix Recommendations
 - General
 - .Net
 - J2EE

Minimum Severity Filter: Informational

Remediation

Application Data

Visited URLs

Help | Preview | Save Report... | Cancel

Detailed Findings

Vulnerable URL: <http://fake/fake.aspx>

Total of 2 findings in this URL

[1 of 2] Cross site scripting

Severity: **High**

Advisory & Fix Recommendation: [See Appendix 1](#)

Vulnerable URL: <http://fake/fake.aspx> (parameter = fake)

Remediation:

Sanitize user input

Variant 1 of 4 [ID=2416]

This test variant was constructed from the original request by applying the following change(s):

- Set parameter 'uid's value to '>><script>alert('Appscan%20-%20CSS%20attack%20may%20be%20used')</script>'
- Set parameter 'uid's value to '>><script>alert('Appscan%20-%20CSS%20attack%20may%20be%20used')</script>'

Request:

```
GET /bank/login.aspx?uid=>><script>alert('Appscan%20-%20CSS%20attack%20may%20be%20used')</script>&passw=Demol234&x=&y= HTTP/1.0
Cookie: ASP.NET_SessionId=3bg3jsupvfrjf0i3bphl0rql
Host: bern
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Referer: http://bern/bank/login.aspx
```

Variant 2 of 4 [ID=2418]

This test variant was constructed from the original request by applying the following change(s):

- Set parameter 'uid's value to '>><script>alert('Appscan%20-%20CSS%20attack%20may%20be%20used')</script>'
- Set parameter 'uid's value to '>><script>alert('Appscan%20-%20CSS%20attack%20may%20be%20used')</script>'

Request:

```
GET /bank/login.aspx?uid=>><script>alert('Appscan%20-%20CSS%20attack%20may%20be%20used')</script>&passw=Demol234&x=&y= HTTP/1.0
Cookie: ASP.NET_SessionId=3bg3jsupvfrjf0i3bphl0rql
Host: bern
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Referer: http://bern/bank/login.aspx
```