

# SW 보안약점 소개

SIGPL Workshop, KCC2013

2013. 6. 28

한국항공대학교 안준선

## 목차



- 시큐어 코딩, 보안약점, 보안 취약점
- SW 보안약점의 유형
  - 입력 데이터 검증 및 표현 (Input Validation and Representation)
  - API 오용(API Abuse)
  - 보안 기능 (Security Features)
  - 시간 및 상태 (Time and State)
  - 에러 처리(Error Handling)
  - 코드 오류(Indicator of Poor Code Quality)
  - 캡슐화(insufficient Encapsulation)
- 관련 연구

# 시큐어 코딩 (Secure Coding)

- 보안취약점 없는 강건한 프로그램을 작성하는 것

... One of the key things that developers can do to help secure their systems is to write code that can **withstand attack and use security features properly** ...

[<http://msdn.microsoft.com/en-us/security/aa570401.aspx>]

...Easily avoided **software defects** are a primary cause of commonly exploited software vulnerabilities. ...

[<http://www.cert.org/secure-coding/>]

3

# 2<sup>nd</sup> Software Crisis : Trustworthiness

The collage consists of several overlapping elements:


- ITBUSINESSEDGE** logo at the top left.
- VeriSign** logo at the top left.
- nakedsecurity** logo at the top center.
- PCMag.com** logo on the left side.
- LAPTOPS | DESKTOPS** text on the left side.
- Stuxnet malware is ... Iran's Bushehr n** article by Mark Clayton, Staff writer | September 21, 2010. It includes a photo of the Bushehr nuclear power plant.
- Kindergarten-Level Computer Security** article by Kenrick Vezina on June 23, 2011. Sub-headline: "How Citigroup let itself fall prey to an easy hack." It includes a photo of a Citibank building.
- FA FOR W** logo on the right side.

4

## 침해 양상의 변화

- 침해의 난이도가 쉬워지고 있음
- 침해의 변화 경향
  - 특정 시스템 목표
  - Social Engineering
  - 목적 : 단순 파괴 → 이윤추구/정치적 목적

Recent Market Price	
Credit Card Number	\$0.50~\$20
Full Identity	\$1~\$15
Bank Account	\$10~\$1000



You've visited this page 3 times. Last visit: 6/25/13

5

## 시큐어 코딩의 중요성

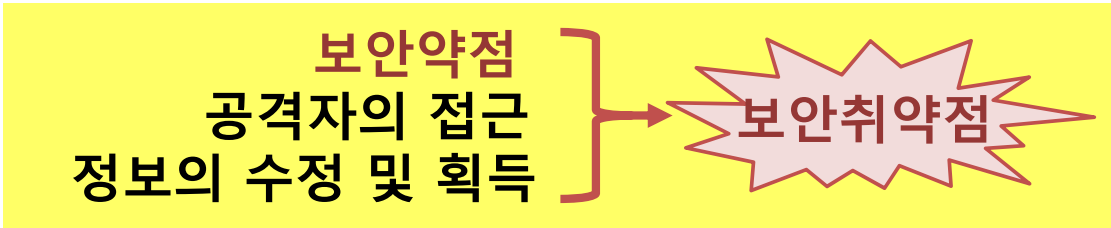
- 보안 침해의 75% 이상이 응용 소프트웨어와 관련 (Gartner, 05)
- 보안 침해의 소프트웨어 보안 결함으로 연간 1800억달러 손실 (Geeknomics, Davis Rice)
- 운영단계의 취약점 제거 비용은 개발단계에서보다 많은 비용 소요 (IBM 60~80배, MS 100배)
- MS에서는 SDL (Secure Development Lifecycle)을 SQL Server 2005에 적용한 결과 SQL Server 2000 대비 3년간 발견된 취약성이 91% 감소

6

# I 보안약점, 보안취약점



	보안약점 (Weakness)	보안취약점 (Vulnerability)
정의	보안취약점이 될 수 있는 소프트웨어의 결함, 실수, 버그 등	공격자가 이용하였을 경우에 시스템의 보안정책을 침해하게 되는 시스템의 허점
특성	일반적 형태 / 근원적	개별적 / 상황 의존적



→ 보안약점을 제거함으로써, 보안취약점에 대한 근원적인 대처가 가능 😊

# SW 보안약점 유형



- 주요 유형
  - 입력 데이터 검증 및 표현 (Input Validation and Representation)
  - API 오용(API Abuse)
  - 보안 기능 (Security Features)
  - 시간 및 상태 (Time and State)
  - 에러 처리(Error Handling)
  - 코드 오류(Indicator of Poor Code Quality)
  - 캡슐화(insufficient Encapsulation)

[Seven Pernicious Kingdoms : A Taxonomy of Software Security Errors, IEEE Security & Privacy, 3(6):81-84, 2005]

# SW 보안약점 유형 1 : 입력 데이터 검증 및 표현

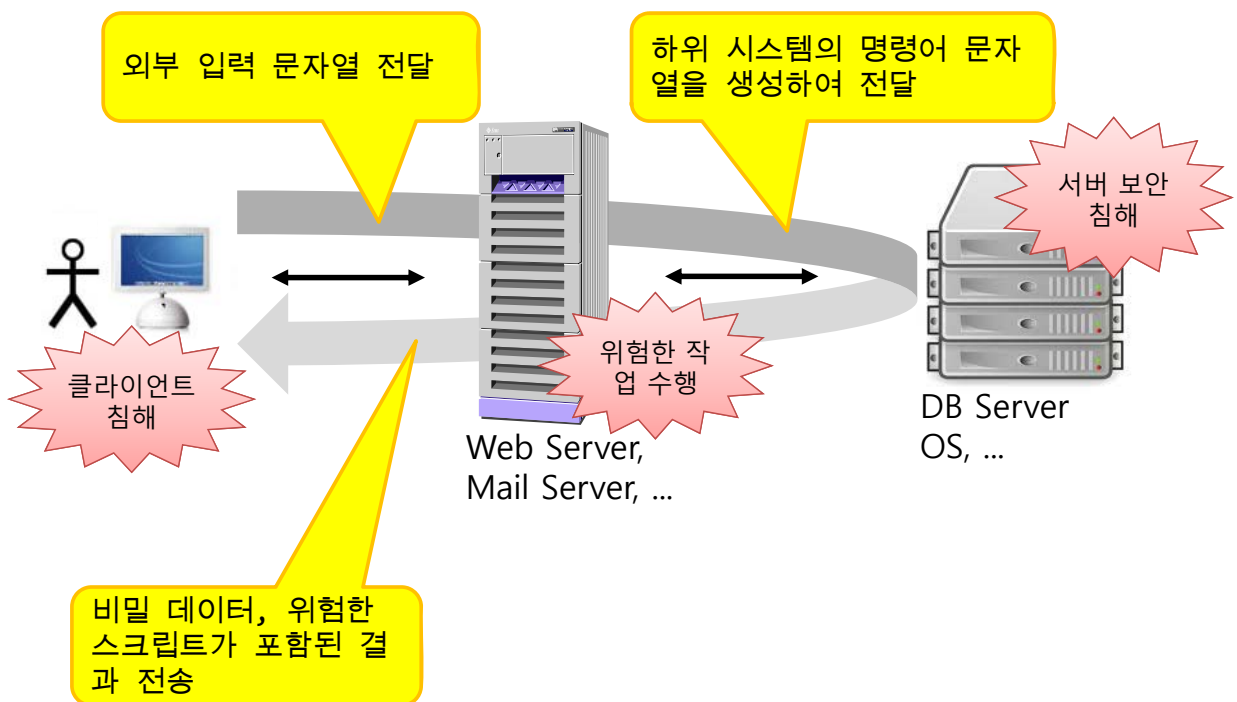
- 적절한 검증 없이 사용된 외부 입력으로 인한 SW 취약점

## • 관련 약점

- 문자열 삽입
  - SQL 삽입
  - 크로스 사이트 스크립트
  - OS 명령어 삽입
- 버퍼 넘침
- 자원 삽입
- 숫자값 오류



# 문자열 삽입 약점



## 문자열 삽입 약점



보안약점	생성 문자열
XQuery 삽입	XQuery
SQL 삽입	SQL
XSS (Cross Site Scripting)	HTML 파일
XPath 삽입	XPath
OS 명령어 삽입	Shell
HTTP 응답 분할	HTML 헤더
LDAP 삽입	LDAP
디렉토리 경로 조작	파일 경로
로그 삽입	로그
XML 삽입	XML

11

## SQL 삽입



Identification

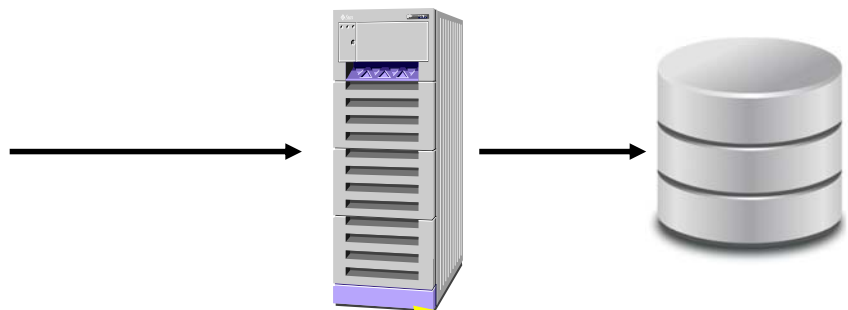
**Log On.** Enter your username and password below.

User name

Password

Remember me?

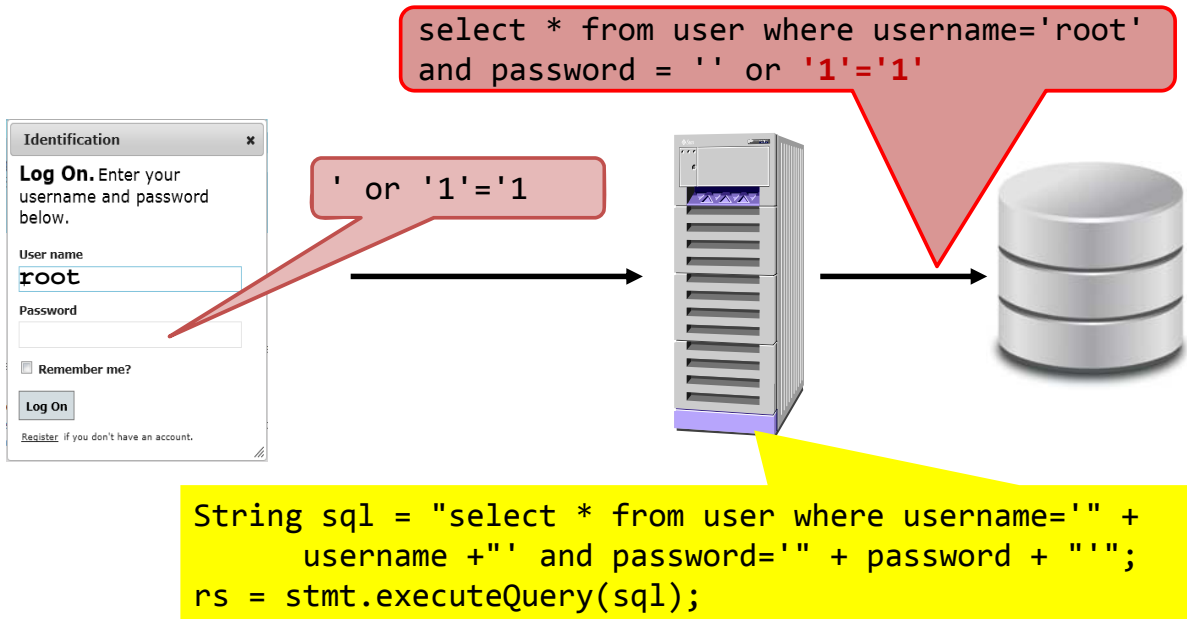
Register if you don't have an account.



```
String sql = "select * from user where username='" +  
username + "' and password='" + password + "'";  
rs = stmt.executeQuery(sql);
```

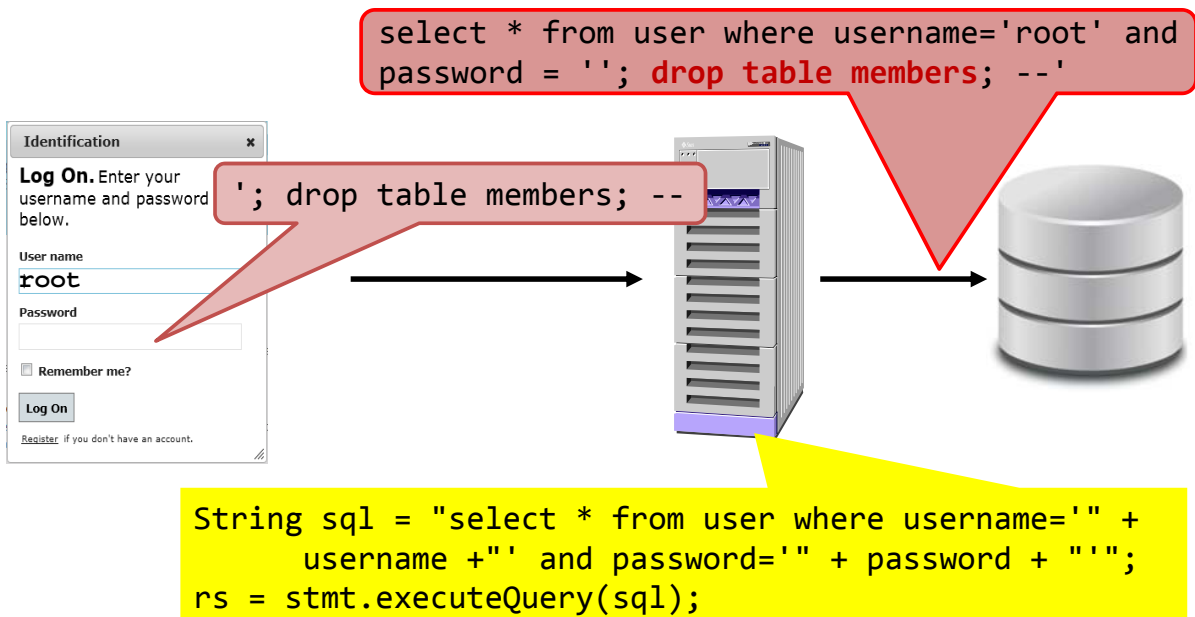
12

# SQL 삽입



13

# SQL 삽입



14

## SQL 삽입 😊



15

## SQL 삽입 약점의 제거

- Prepared statements

```
PreparedStatement pstmt = con.prepareStatement(
    "select * from user where username=? and password = ?");
pstmt.setString(1, username);
pstmt.setString(2, password);
rs = pstmt.executeQuery();
```

- Stored statements

```
CallableStatement cs = connection.prepareCall(
    "{call sp_getUserRecord(?)}");
pstmt.setString(1, username);
pstmt.setString(2, password);
rs = cs.executeQuery();
```

16



## SQL 삽입 공격의 방어

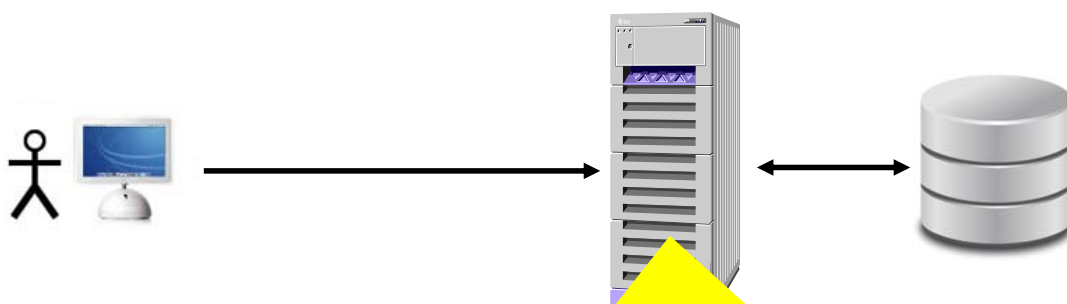
- 탈출문자열 (Escaping sequence)
  - DB에서 일반 문자로 인식되도록 기호를 대치
  - 문제
    - DB에 따라 탈출 문자열이 다름

```
' → '\'' /* MySQL */  
" /* Oracle */
```

- ESAPI (Enterprise Security API)

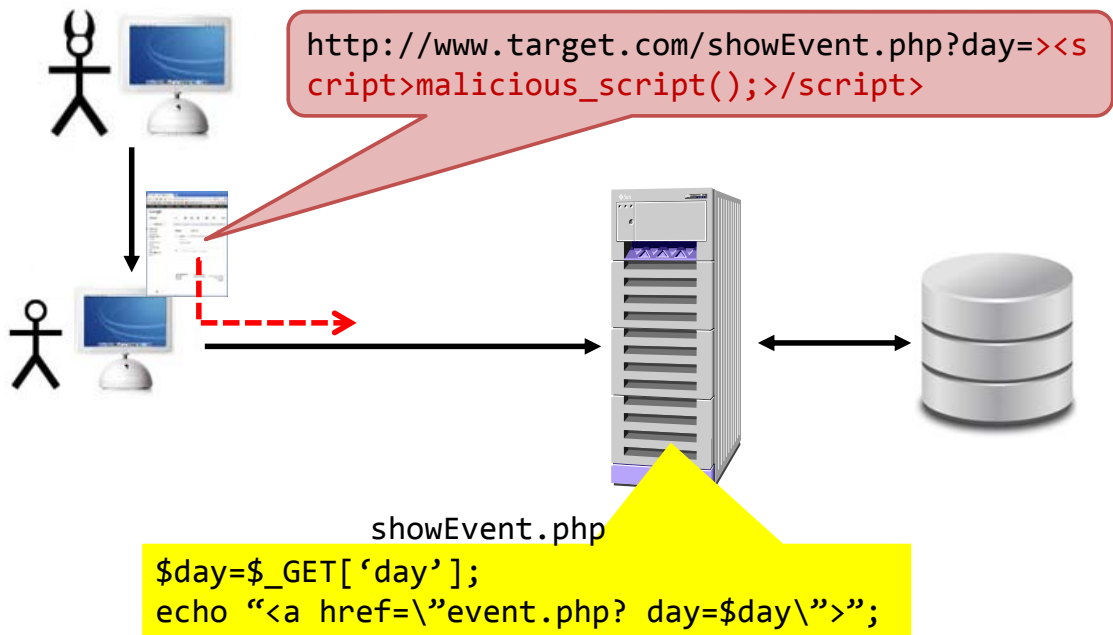
17

## 크로스 사이트 스크립트(XSS)

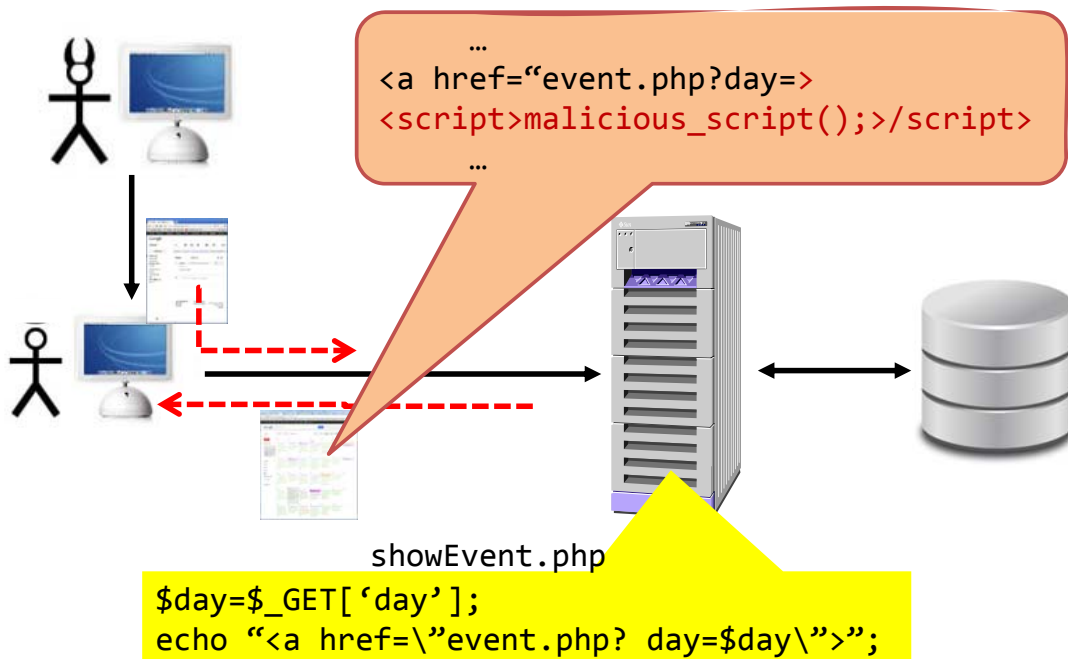


18

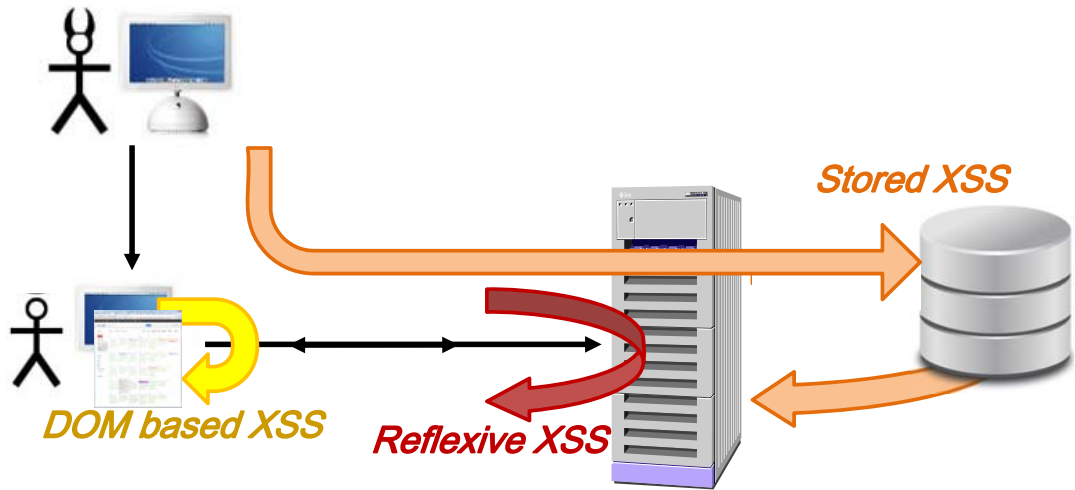
# 크로스 사이트 스크립트(XSS)



# 크로스 사이트 스크립트(XSS)

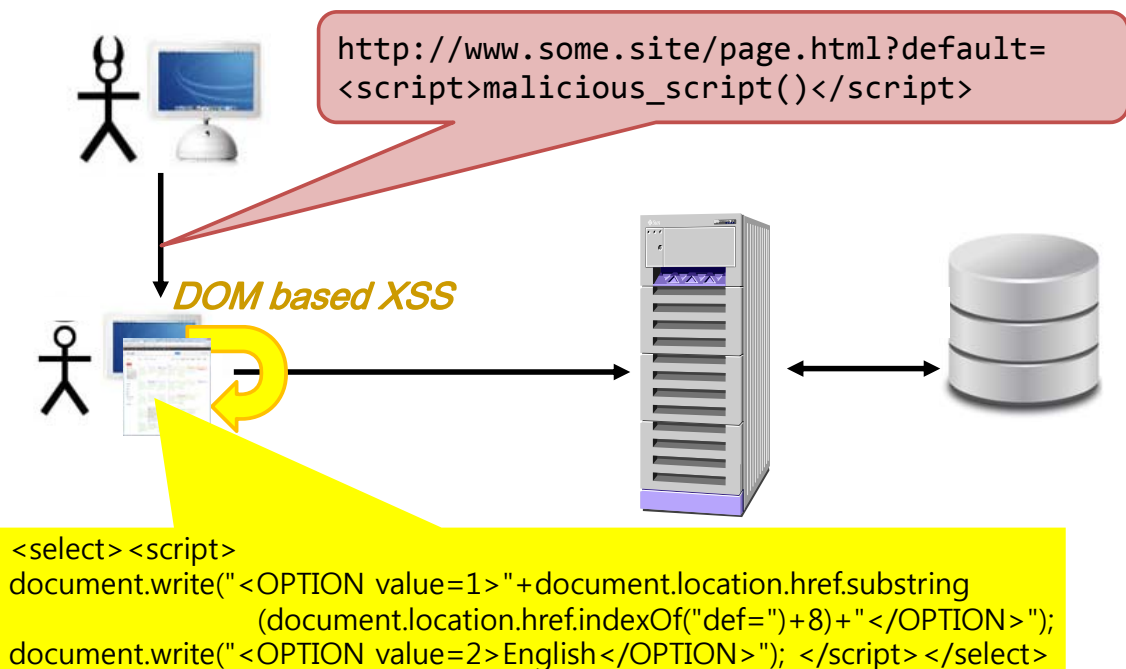


# 크로스 사이트 스크립트(XSS)



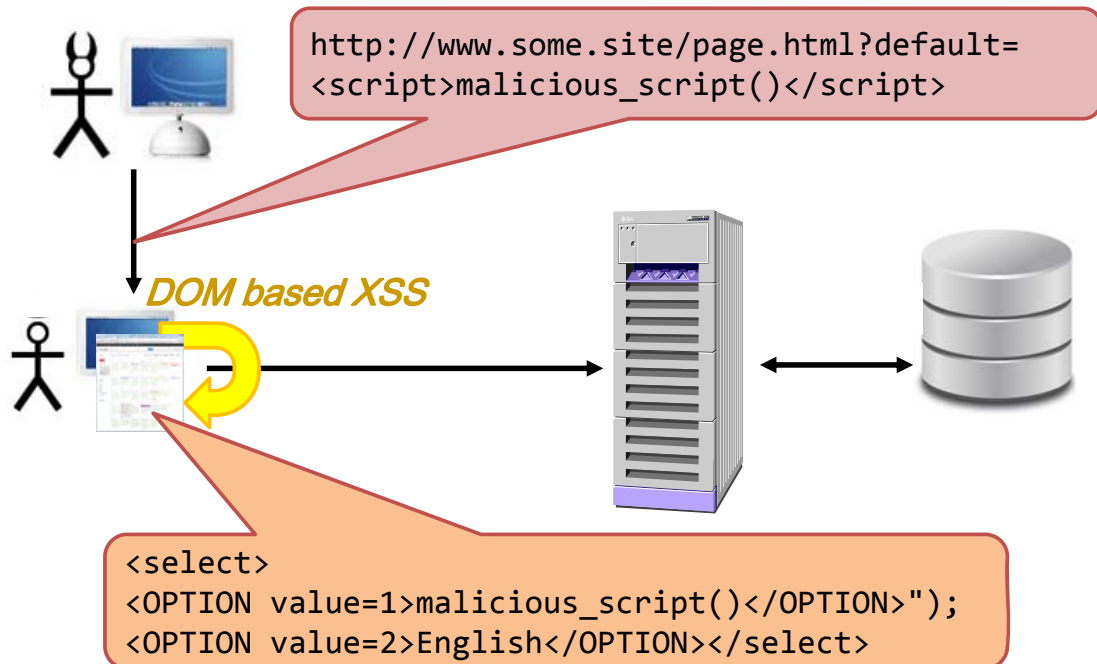
21

# 크로스 사이트 스크립트(XSS)



22

## 크로스 사이트 스크립트(XSS)



23

## XSS 약점의 제거

- 신뢰할 수 없는 데이터에 대한 검증
  - Whitelist 검증 사용
- 문맥을 고려한 인코딩
  - OWASP ESAPI
  - Microsoft Anti-Cross Site Scripting Library

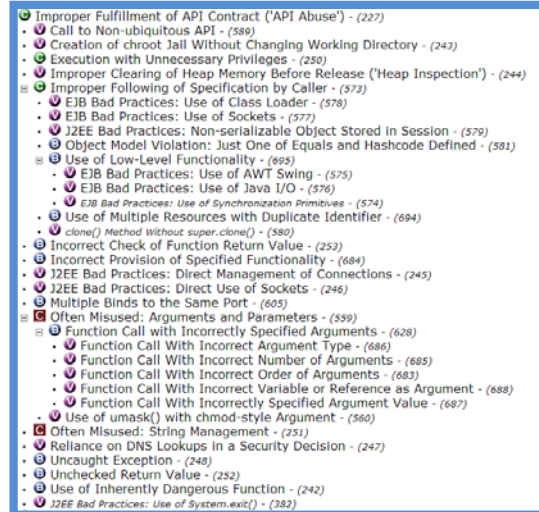
24

## SW 보안약점 유형 2 : API 오용

- 적합하지 않은 인자를 사용한 API의 사용이나 API 결과에 대한 적절하지 않은 가정에 의한 약점

- 관련 약점들

- DNS lookup에 의존한 보안결정
- 결과값 검사 누락
- 안전하지 않은 함수의 사용
- 덜 지워진 않은 힙 메모리 데이터



25

## DNS lookup에 의존한 보안 결정

- 도메인명에 의존하여 보안 결정을 할 경우 DNS 서버 캐시 오염을 통한 공격이 가능

```
InetAddress addr = InetAddress.getByName(ip);
if addr.getCanonicalHostName().endsWith("trustme.com"){
    trusted = true;
}
```

```
if (Ip.equals(trustedAddr)) {
    trusted = true;
}
```

26

## SW 보안약점 유형 3 : 보안 기능

- 인증, 접근제어, 암호화, 권한 관리 등과 관련된 소프트웨어 약점

- 관련 약점들

- 취약한 암호화 알고리즘 사용
- 적절한 인증 없는 중요기능 허용
- 중요 자원에 대한 잘못된 권한 설정
- 사용자 중요 정보 평문 저장
- 솔트 없이 일방향 해쉬 함수 사용
- 무결성 검사 없는 코드 다운로드

...



27

## 취약한 암호화 알고리즘 사용

- 도메인명에 의존하여 보안 결정을 할 경우 DNS 서버 캐시 오염을 통한 공격이 가능

```
Cipher c = Cipher.getInstance("DES");
```

```
Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
```

- 취약한 알고리즘 : MD4, MD5, RC2, RC4, RC5, DES, 2DES
- 안전한 알고리즘 : SHA-256, AES, SEED, ARIA 등

28

## SW 보안약점 유형 4 : 시간 및 상태

- 복수의 스레드나 프로세스들의 병행 수행과 관련하여 시점과 상태를 적절히 처리하지 못하여 발생하는 약점
- 관련 약점들
  - 경쟁조건 : 검사시점과 가용시점
  - 제대로 제어되지 않은 재귀

Time and State - (361)
Concurrency Issues - (557)
Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') - (362)
Context Switching Race Condition - (368)
Race Condition within a Thread - (366)
Signal Handler Race Condition - (364)
Time-of-check Time-of-use (TOCTOU) Race Condition - (367)
Race Condition During Access to Alternate Channel - (421)
Covert Timing Channel - (385)
Double-Checked Locking - (609)
Exposure of Resource to Wrong Sphere - (668)
External Influence of Sphere Definition - (673)
Improper Control of a Resource Through its Lifetime - (664)
Improper Synchronization - (662)
Improper Locking - (667)
Incorrect Synchronization - (821)
Missing Synchronization - (820)
Incorrect Resource Transfer Between Spheres - (669)
Insufficient Control Flow Management - (691)
Insufficient Session Expiration - (613)
Operation on a Resource after Expiration or Release - (672)
Redirect Without Exit - (698)
Session Fixation - (384)
Signal Errors - (387)
State Issues - (371)
Symbolic Name not Mapping to Correct Object - (386)
Technology-Specific Time and State Issues - (380)
Temporary File Issues - (376)
Uncontrolled Recursion - (674)
Unrestricted Externally Accessible Lock - (412)
Use of a Non-reentrant Function in a Concurrent Context - (663)

29

## 경쟁조건 : 검사시점과 가용시점

- 자원에 대한 권한 검사 시점과 자원의 사용시점간의 자원의 상태 변화에 따른 허점

```

if(!access(file,W_OK)) {
    f = fopen(file,"w+");
    operate(f);
    ...
}
else {
    fprintf(stderr,"Unable to open file %s.\n",file);
}
    
```

### Notes

**Warning:** Using `access()` to check if a user is authorized to, for example, open a file before actually doing so using `open(2)` creates a security hole, because the user might exploit the short time interval between checking and opening the file to manipulate it. **For this reason, the use of this system call should be avoided.** (In the example just described, a safer alternative would be to temporarily switch the process's effective user ID to the real ID and then call `open(2)`.)

`access()` always dereferences symbolic links. If you need to check the permissions on a symbolic link, use `faccessat(2)` with the flag `AT_SYMLINK_NOFOLLOW`

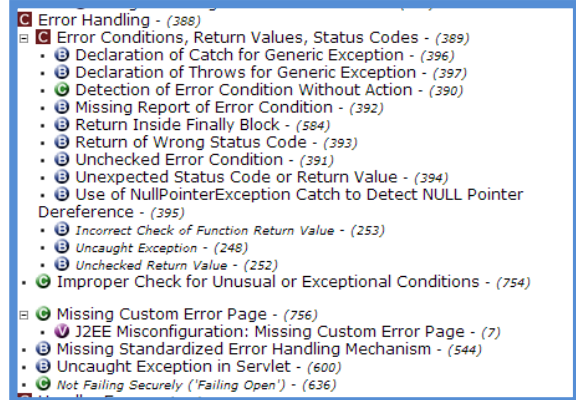
30

## SW 보안약점 유형 5: 에러 처리

- 에러를 처리하지 않거나, 불충분하게 처리하여 발생하는 시스템의 불안정이나 정보 누출과 관련한 약점

- 관련 약점들

- 오류메시지 통한 정보 노출
- 오류 상황 대응 부재
- 적절하지 않은 예외 처리



31

## 오류메시지 통한 정보 노출

- 시스템에 대한 민감한 정보를 포함하는 오류 메시지를 생성하여 외부에 제공하여 공격을 도와주는 경우



32

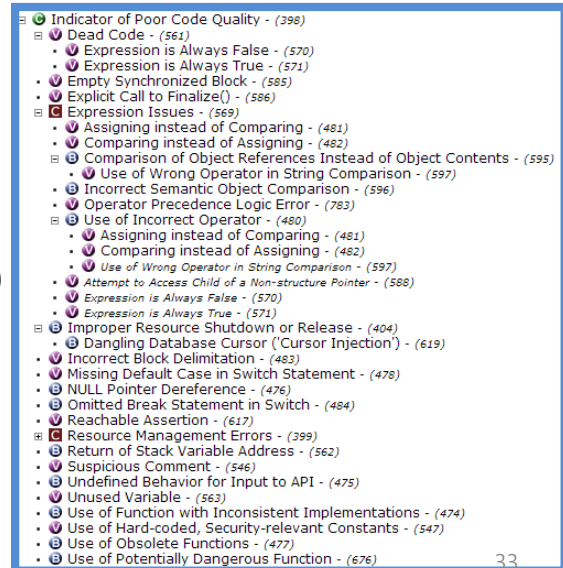


## SW 보안약점 유형 6: 코드 오류

- 소프트웨어의 질관리가 부실함을 보여주는 약점들로 일반적으로 취약점으로 직접 연관되지는 않는다.

- 관련 약점들

- 부적절한 자원 해제
- 스택 변수 주소의 반환
- 널 포인터 역참조
- 사용되지 않는 코드 (Dead Code)
- 사용되지 않는 변수



## 부적절한 자원 해제

- 파일, 힙메모리, 소켓 등의 자원 사용 후 적절한 반환이 이루어지지 않는 경우

```
Try {  
    Class.forName("com.mysql.jdbc.Driver");  
    conn = DriverManager.getConnection(url);  
    ...  
    conn.close();  
}  
Catch (Exception e) { }
```

# SW 보안약점 유형 7: 캡슐화

- 중요한 데이터나 함수에 대한 불완전한 캡슐화

- 관련 약점들

- 잘못된 세션에 의한 데이터 누출
- 제거되지 않고 남은 디버거 코드
- 공용 메서드로부터 반환된 private 배열-유형필드
- private 배열-유형 필드에 공용 데이터 할당



# 잘못된 세션에 의한 데이터 노출

- 다중 쓰레드 환경에서 싱글톤 객체 필드에 대한 경쟁 조건 발생으로 인한 데이터 노출 가능

```
Public class MyServlet extends HttpServlet {
    private String name;
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        name = request.getParameter("name");
        ...
        out.println(name+", thanks for visiting");
        ...
    }
    ...
}
```

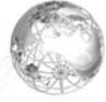
# 관련 연구 : 보안약점 / 보안취약점 목록



# 관련 연구 : 주요 보안약점 목록




# 관련 연구 : Secure Coding Guides



**CERT** | **Software Engineering Institute**  
Carnegie Mellon

HOME | Software Assurance | Secure Systems | Organizational Security |

**Secure Coding**  
Easily avoided software defects are a primary cause of commonly exploited software vulnerabilities. CERT



**OWASP**  
The Open Web Application Security Project

Page Discussion

Navigation  
Home

**OWASP Secure Coding Practices - Quick Reference**

**msdn** United States (English) | Search MSDN with Bing

Home Library Learn Samples Downloads Support Community Forums

MSDN Library  
Development Tools and Languages  
Visual Studio 2008  
Visual Studio

**Secure Coding Guidelines**  
.NET Framework 3.5 | Other Versions  
4 out of 7 rated this helpful - Rate this topic