OWASP Testing & Code Review Guide
- OWASP Xenotix Exploit Framework

Yong-Sik Choi, Ph.D.
OWASP Korea Chapter Member
IDLE co., ltd
yongsik.choi@owasp.org

OWASP
2014.06.17

The OWASP Foundation
http://www.owasp.org

Index

- Software Security
- OWASP Code Review Guide
- OWASP Code Testing Guide
- OWASP Xenotix Exploit Framework
- Reference
- FAQ

Software Security

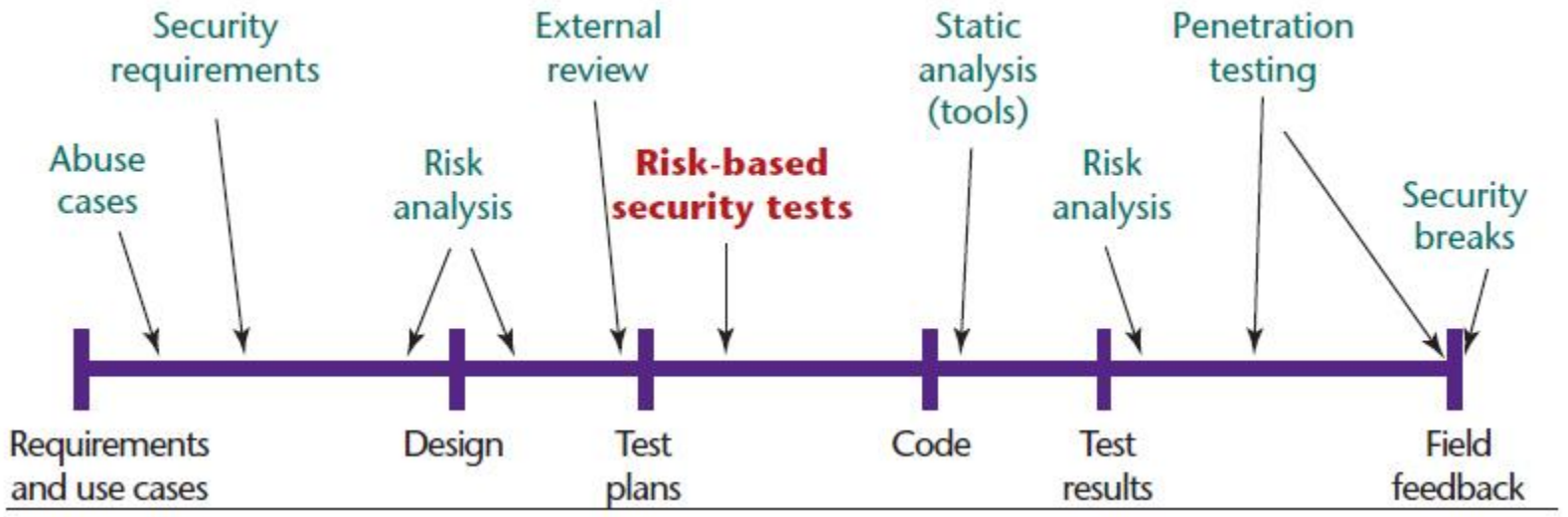
- SW 보안
- SW 개발 보안
- SW 취약점과 약점
- SW 보안 테스팅

SW 보안

- 국가 주요 시스템, 비즈니스 SW에 의해 운영
- 사회기반시설(에너지, 수송, 항공제어)
- 안전하고 신뢰적인 SW 의존
- SW는 데이터와 자원들을 보호하도록 설계되고 구현되어야 하며 SW 자체에 대한 보안 제공
- SW 보안이란 악의적인 공격 하에서도 SW가 정상적인 기능(역할)을 유지하도록 하는 것
- SW 취약점, 악성코드, 결함 등의 위험 완화

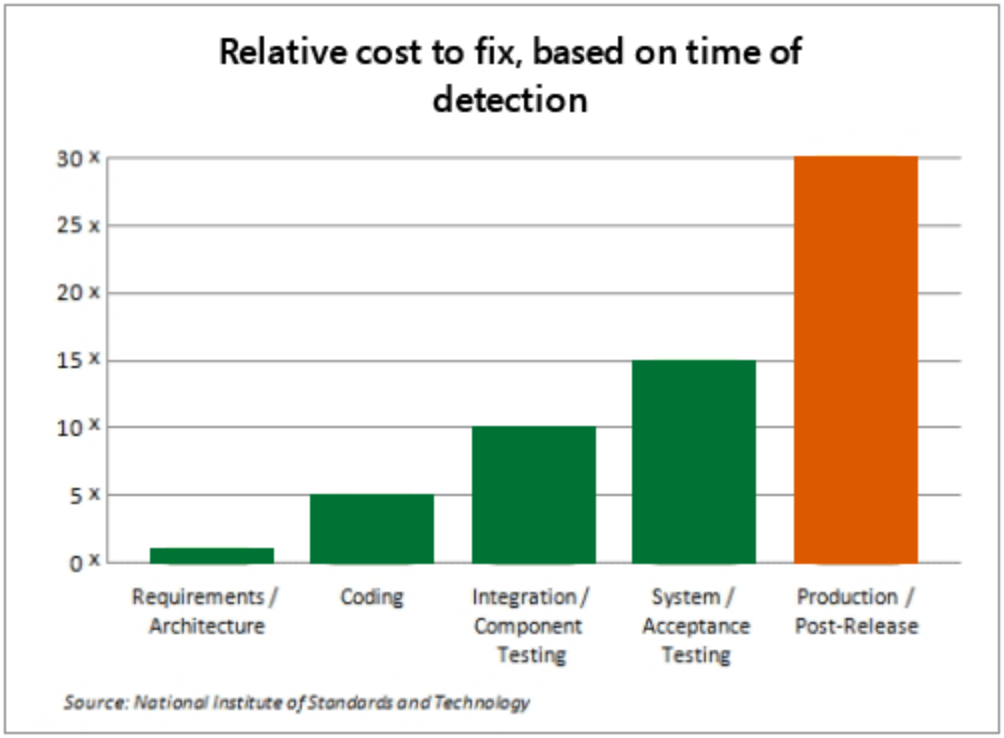
SW 개발 보안

- 안전한 SW 개발을 위해 소스코드 등에 존재하는 잠재적인 보안 취약점을 제거
- 보안을 고려하여 기능을 설계, 구현



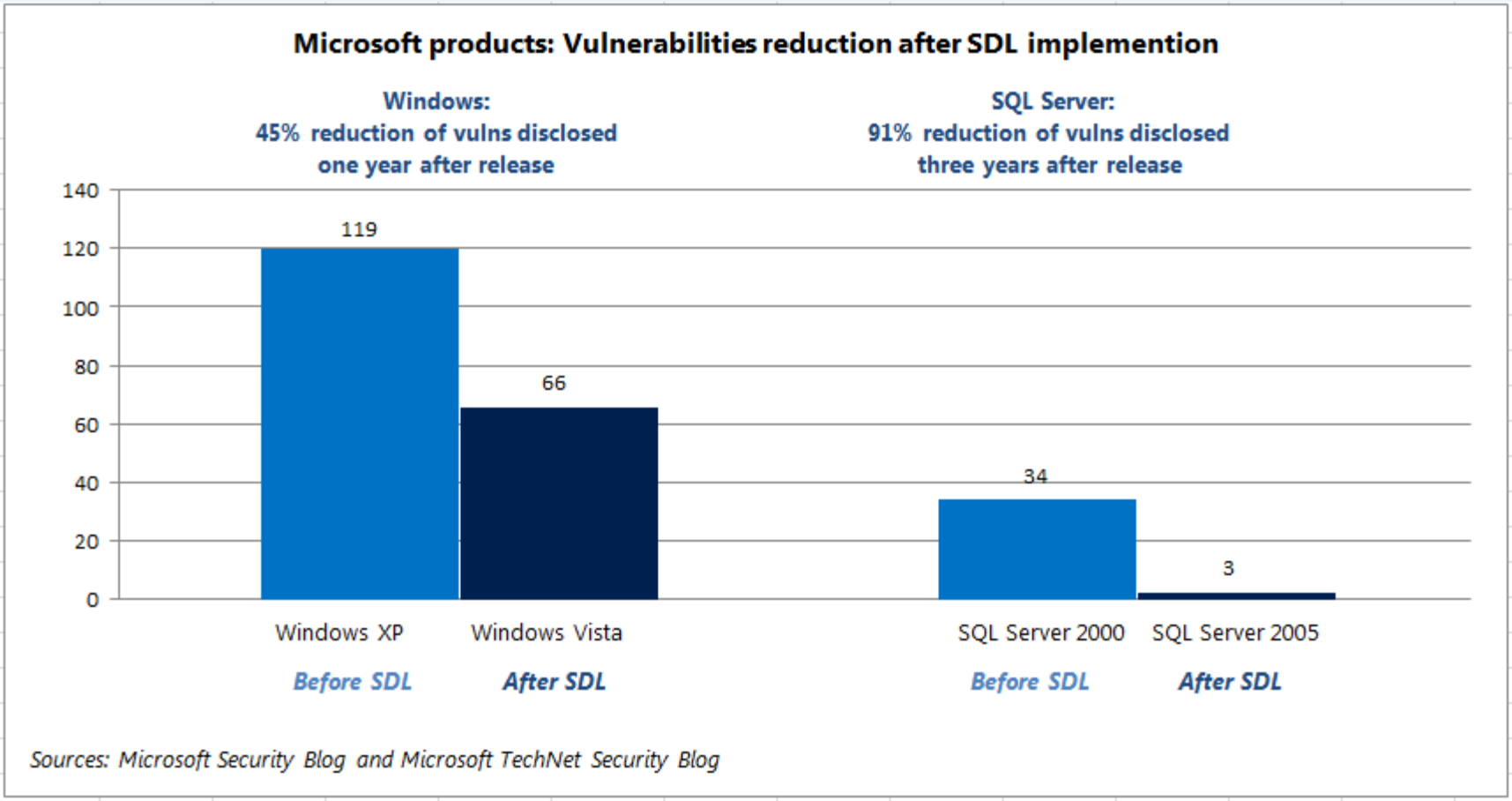
- Software development life cycle, Gray McGraw

- 엄격한 QA(Quality Assurce)를 거치더라도 천라인당 5 ~ 50개 정도의 오류 존재
- 코드 1000라인 당 약 20여개의 보안 결점(Flaw)
  - NTST 보고서
  - 상업 SW 경우 더 많은 오류 내포할 가능성 있음
  - SW 배포 후에 코드를 수정하는 비용은 설계단계에서 수정하는 비용보다 30배 소요
  - 개발 후반부로 갈수록 다 많은 비용 소요



- SW 배포 후에 코드를 수정하는 비용은 설계단계에서 수정하는 비용보다 30배 소요
- 개발 후반부로 갈수록 다 많은 비용 소요

■ MS SDL, http://www.microsoft.com/security/sdl/



- MS SDL(Secure Development Lifecycle) 적용효과

■ MS SDL, http://www.microsoft.com/security/sdl/

SW 취약점(Vulnerability)과 약점(Weakness)

- 프로그래머는 습관적으로 동일 코딩 기법을 사용하기 때문에 동일한 오류 반복
- SW 보안 문제가 지속적으로 증가(Gray McGraw)
  - 연결성(Connectivity)
    - 네트워크상에 연결되어 공격 벡터 증가
  - 확장성(Extensibility)
    - SW 유연한 확장성, 인터페이스
    - 플러그-인 구조나 동적 적재 모듈성
  - 복잡성(Complexity)
    - SW 시스템의 크기, 복잡성 증가
    - 결합 비율(Defect Rate)은 코드 크기의 제곱에 비례

- OWASP 정의
  - "설계상의 결점이나 구현상의 버그와 같은 허점(hole)/약점(weakness)"으로, 공격자에 의해 이용되어 애플리케이션 관계자에게 해를 끼칠 수 있다
- 보안공격의 원천
- SW 결함(버그 50%, 결점50%)
  - SW 설계단계의 결점(Flaw)
  - 비논리적인 접근제어
  - 일관성 결여된 에러 핸들링
  - 메서드 오버라이딩
  - 버그(부정확한 입력 검증, 자원 누수, 환경변수 등)

SW 보안 테스팅

- 사용자나 악의적 사용자에 의해 보안 결점들이 발견될 가능성을 줄임
  - 예측 가능한가?
  - 취약점, 약점이 들러나는가?
  - 오류나 예외 핸들링 루틴이 안전한 상태인가?
  - 명시된, 암묵적 비기능적 보안 요구사항 만족하는가?
  - 명시된 보안 제약 사항을 위반 하는가?
- 데이터를 보호하고 기능을 유지하고 있는지 확인하는 과정

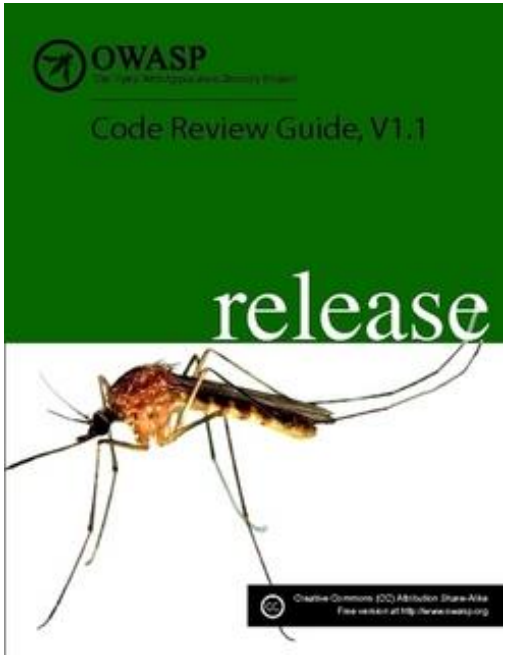
- 위험 분석
  - 보안 요구 사항 검토, 보안 위험 식별
  - 위협 모델링(Threat Modeling)
- 동적, 정적 분석
  - 코드 수준에서 취약점, 약점 탐지
- 퍼즈 테스팅(Fuzz Testing)
  - 취약점 동적 분석/테스트
  - 랜덤(또는 규칙적) 데이터 입력하여 예외, 오류 분석
- 취약점 스캐닝
- 침투 테스트

OWASP Code Review Guide

- History
- 기본점검사항, 발견:정보수집, 컨텍스트, 체크리스트
- 코드 리뷰 범위
- 애플리케이션 위협 모델링
- 코드 리뷰 Metrics
- 코드 리뷰 결과 보고서

History

- OWASP Code Review Guide V2.0, 2014 예정
- OWASP Code Review Guide V1.1 한글 2014 예정
- OWASP Code Review Guide V1.1, 2008



기본점검사항, 발견:정보수집, 컨텍스트, 체크리스트

- 기본 점검 사항
  - 어플리케이션 용도와 주요 사업 방향에 대한 이해
  - 서로 다른 주요 위협, 동기, 잠재적인 공격 식별
  - 코드: 이슈사항, 문제점, 모범 사례
  - 컨텍스트: 동작여부/환경검토, 데이터 유형 파악 및 데이터 유출될 경우 피해 파악
- 정보수집
  - 설계 문서, 요구사항, 기능명세, 테스트 결과
  - 개발자와 애플리케이션 설계 담당자와 대화
- 컨텍스트
  - 유무형의 자산, 영업비밀과 같은 정보와 자산 보호/보증
  - 비즈니상에서의 컨텍스트의 이해, 상위수준 위협 모델 준비
- 체크리스트

코드 리뷰 범위

- 공격 접점의 이해
  - “모든 입력값은 그 상응하는 동등한 출력값을 가진다”
  - 예) 브라우저 쿠키값, 쿠키, 속성 파일, 외부 프로세스, 데이터 입력, 서비스 응답, 플랫 파일, 커맨드 라인 파라미터
- 환경변수
  - 동적 및 정적 데이터 흐름 분석
  - 애플리케이션 내의 모든 트랜젝션 분석
    - 모든 신뢰되지 않은 출처로부터의 데이터/입력값 검증
    - 인증, 인가, 세션 관리, 암호, 에러 처리/정보 누수, 감사 기록
    - 보안 코드 환경
- 리뷰 대상에 대한 이해

애플리케이션 위협 모델링

- 시스템에 대한 전반적인 이해: 애플리케이션의 엔트리 포인트 위치, 각 엔트리 포인트와 관련된 위협 인지
- 위협 모델링 절차
  - STEP 1: 애플리케이션 분리
  - STEP 2: 위협 순위
  - STEP 3: 대응방안과 완화방안의 수립

- STEP 1: 애플리케이션 분리
  - 애플리케이션이 외부 엔티티와 어떻게 반응하는가와, 애플리케이션에 대한 이해
  - 잠재적인 공격자 관점에서 애플리케이션과 어떻게 상호작용하는지를 확인
    - 엔트리 포인트를 확인
    - 공격자가 관심 있게 볼만한 항목이나 영역
    - 애플리케이션이 외부 엔티티에 대해 접근을 허용할 신뢰도 수준을 확인, 사용자 케이스 생성
  - 위협 모델(Threat Model) 문서에 명세
  - 데이터 플로우 다이어그램(Data Flow Diagrams, DFDs)

- STEP 2: 위협 순위
  - STRIDE와 같은 위협 범주
  - 감사, 인증, 인가, 설정 관리, 통신 및 스토리지 내의 데이터 보호, 데이터 검증, 에외 관리와 같이 위협을 정의
  - 공격자로부터의 위협과(STRIDE) 방어하는 관점 모두에서 위협을 확인
  - ASF (Application Security Frame) 범주화
    - 위협, 보안 관제 취약성을 확인
    - 보안 장치들이 어떻게 우회
    - 보안이 적용되었을 때 결함이 어디에 존재
  - 각 위협에 대한 보안 위험도 측정

- STEP 3: 대응방안과 완화방안의 수립
  - 위협 대응 매핑 리스트, 우선순위
  - 위험 완화 전략
    - 유형 및 사례
    - 대응 방향 제시
    - 사업적 위협 평가

코드 리뷰 **Metrics**

- 개발 정책, 지침, 해석
- 검토 과정의 정확성, 성능, 효과 기록
  - 측정하는 것을 결정
    - 결함도, 코드라인, 기능라인, 위험도, 복잡도
  - 최소 기준 설정
  - 보고에 대한 요구 사항 정의
    - 검사율, 결함 결출율, 코드 커버리지, 결함 수정율, 재검토 결함율

코드 리뷰 결과 보고서

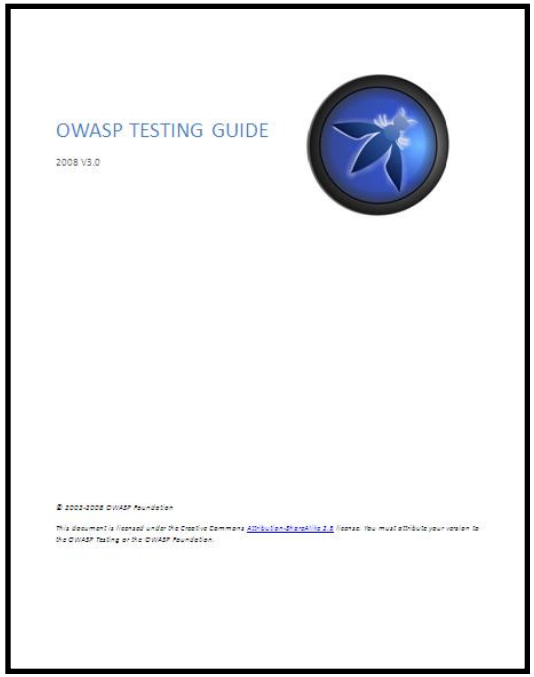
검토 /계약 참조:			
패키지/콤포넌트/클래스명/행번호			
검토결과서 제목			
중요성: 높음			
결과물 내용	위치	위험 설명	권고내용
HTTPRequest object.getID() 함수에서 입력값 검증절차 부재 입력값 검증부재로 인해 다양한 종류의 주입공격이 가능함	com.inc.dostuff.java 행번호 20, 55,106 com.inc.main.java 행번호 34, 99	취약점이 악용 되는 경우의 가능성과 사업상 영향에 대한 논의	운영환경에 투입되기 이전에 해결되어야 하는 중대 문제임

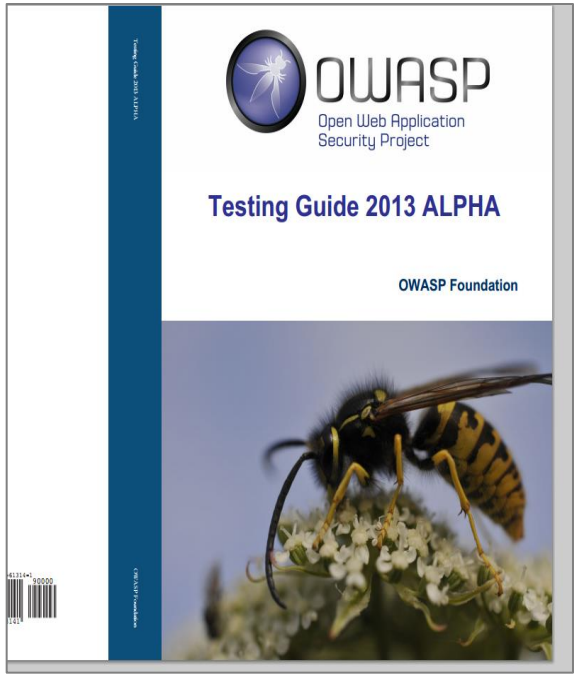
OWASP Testing Guide

- History
- 테스팅 원칙
- 테스팅
- 테스팅 결과 보고서

History

- OWASP Testing Guide V4.0, 2014 예정
- OWASP Testing Guide V3.0 한글 2014 예정
- OWASP Testing Guide V3.0, 2008





테스팅 원칙

- 묘책은 없다, 전략적으로 판단
- 일찍 테스트 그리고 빈번하게
- 보안 범위 파악
- 고정관념을 버려라, 컨텍스트 이해
- 올바른 도구의 사용, 실행
- 소스 코드 검토
- Metrics
- 테스트 결과 문서화

보안 테스팅

- **보안테스트 케이스 개발**
  - Positive functional 보안 테스트 케이스
  - Negative 테스트 케이스
  - 공통 취약점 도출
- **개발자와 테스터의 테스트 통합**
  - 정적, 동적 테스팅, 단위 테스팅
  - 통합 시스템 테스트, 운영 테스트
- **테스트 결과 분석 및 보고**
  - 결함(Defect) 관리
  - 근본 원인 식별

테스트 결과 보고서

범주	참고 번호	테스트 이름	발견한 내용	해결 방안	위험
정보 수집	OWASP-IG-001	스파이더, 로봇 및 크롤러			
	OWASP-IG-002	검색 엔진 발견/정찰			
	OWASP-IG-003	응용 프로그램 시작점 파악			
	OWASP-IG-004	웹 응용 프로그램 지문 테스팅			
	OWASP-IG-005	응용 프로그램 검색			
	OWASP-IG-006	오류 코드 분석			
구성 관리 테스팅	OWASP-CM-001	SSL/TLS 테스팅 (SSL 버전, 알고리즘, 키 길이, 디지털 인증서 유효성)			
	OWASP-CM-002	DB 리스너 테스팅			
	OWASP-CM-003	인프라 구성 관리 테스팅			
	OWASP-CM-004	응용 프로그램 구성 관리 테스팅			
	OWASP-CM-005	파일 확장자 처리 테스팅			
	OWASP-CM-006	오래된, 백업 및 비 참조 파일			
	OWASP-CM-007	인프라 및 응용 프로그램 관리자 인터페이스			
	OWASP-CM-008	HTTP 메쏘드와 XST 테스팅			

OWASP Xenotix Exploit Framework

- Cross Site Scripting (XSS)
- OWASP Xenotix Exploit Framework
- SCANNER MODULES
- INFORMATION GATHERING MODULES
- EXPLOITATION MODULES
- UTILITY MODULES
- XENOTIX SCRIPTING ENGINE

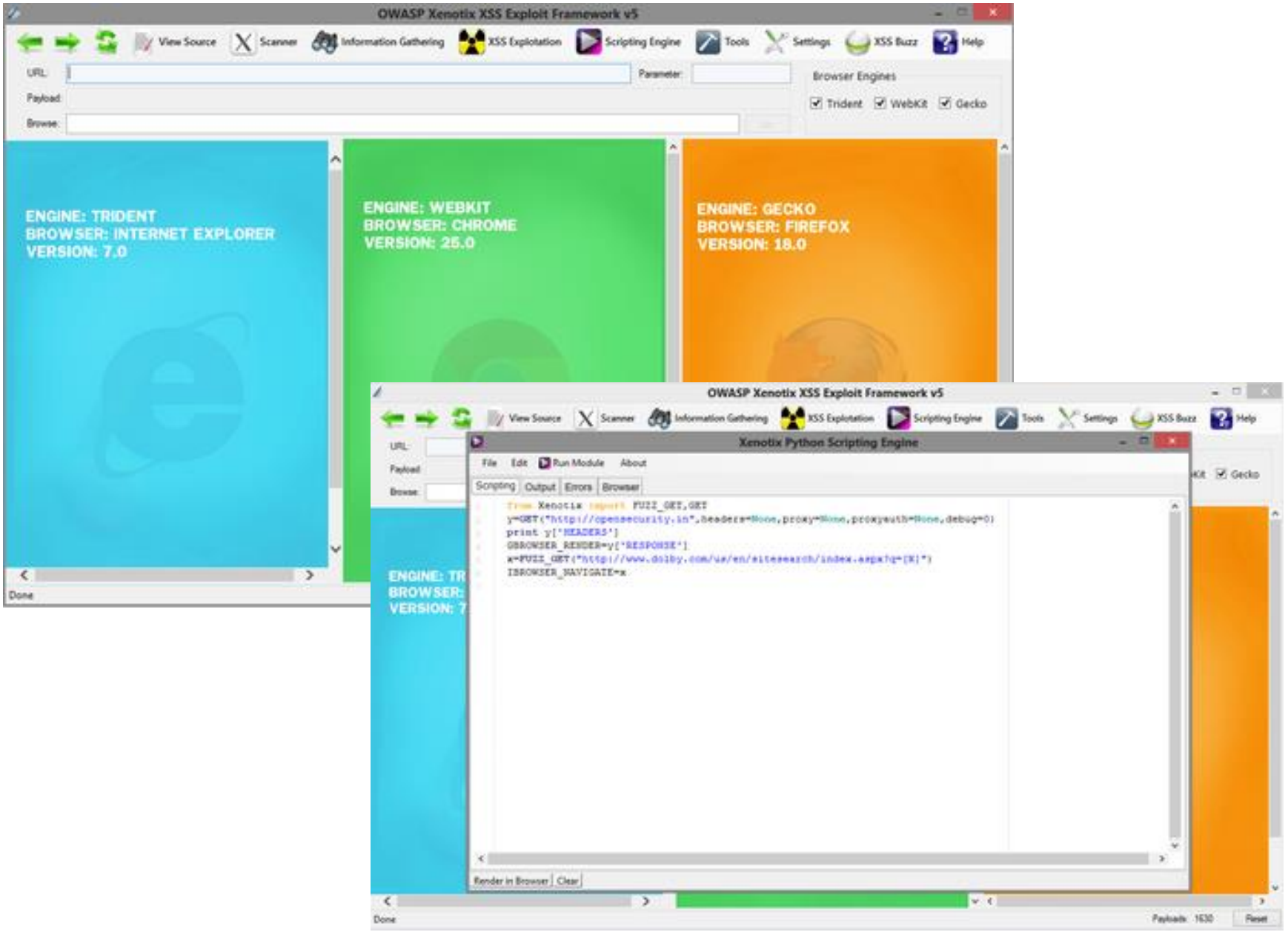
A3- 크로스 사이트 스크립팅 (XSS)

- 애플리케이션이 신뢰할 수 없는 데이터를 가져와 적절한 검증이나 제한 없이 웹 브라우저로 보낼 때 발생
- 공격자가 피해자의 브라우저에 스크립트를 실행하여 사용자 세션 탈취, 웹 사이트 변조, 악의적인 사이트로 이동

특정 애플리케이션	공격 가능성 평균	취약점 분포 매우 광범위	탐지 가능성 쉬움	영향도 중간	특정 애플리케이션 / 사업
외부 사용자, 내부 사용자 및 관리자를 포함하여 시스템에 신뢰할 수 없는 데이터를 보낼 수 있는 사람을 고려해야 한다.	공격자는 브라우저에서 인터프리터를 공격할 수 있는 텍스트 기반 공격 스크립트를 전송한다. 데이터베이스에서 오는 내부 데이터뿐만 아니라 거의 모든 형태의 데이터가 공격에 활용될 수 있다.	XSS는 가장 널리 알려진 웹 애플리케이션 보안 결함이다. XSS는 애플리케이션에서 브라우저로 전송하는 페이지에서 사용자가 입력하는 데이터를 검증하거나 이스케이핑하지 않을 때 발생한다. XSS 결함은 3가지 종류가 있는 것으로 알려져 있다. 1) 저장 XSS, 2) 반사 XSS, 3) DOM 기반 XSS. 대부분의 XSS 결함은 시험 또는 코드 분석을 통해 쉽게 검출할 수 있다.		공격자는 사용자 세션 하이재킹, 웹사이트 변조, 공격 콘텐츠 삽입, 사용자 리디렉션, 악성코드를 사용한 사용자 브라우저를 하이재킹하기 위하여 피해자 브라우저에서 스크립트를 실행할 수 있다.	영향 받는 시스템의 사업적 가치와 처리되는 모든 데이터를 고려해야 한다. 또한 취약점이 일반에게 공개될 경우에 사업적 영향을 고려해야 한다.

OWASP Xenotix Exploit Framework

- XSS 취약점 탐지 및 개발 프레임워크
  - Visual Basic.NET, C++, Java
- 내장 브라우저(Trident, WebKit, Gecko) 엔진 및 스캐너로 영 거짓 양성 검색 결과
- Xenotix 스크립팅 엔진
  - Xenotix의 API를 통해 사용자 정의 테스트 케이스 및 애드온 작성 가능
  - 공격 프레임 워크는 XSS 침투 테스트를 위한 모듈과 증거 포함


OWASP Xenotix XSS Exploit Framework v5
View Source
Scanner
Information Gathering
XSS Exploitation
Scripting Engine
Tools
Settings
XSS Buzz
Help
URL:
Payload:
Browse:
Parameter:
Browser Engines
Trident
WebKit
Gecko
ENGINE: TRIDENT
BROWSER: INTERNET EXPLORER
VERSION: 7.0
ENGINE: WEBKIT
BROWSER: CHROME
VERSION: 25.0
ENGINE: GECKO
BROWSER: FIREFOX
VERSION: 18.0
Done
OWASP Xenotix XSS Exploit Framework v5
Xenotix Python Scripting Engine
File
Edit
Run Module
About
Scripting
Output
Errors
Browser
from Xenotix import FUZZ_GET,GET
y=GET("http://opensecurity.in",headers=None,proxy=None,proxyauth=None,debug=0)
print y['HEADERS']
GBROWSER_RENDER=y['RESPONSE']
x=FUZZ_GET("http://www.dolby.com/us/en/sitesearch/index.aspx?q=[X]")
IBROWSER_NAVIGATE=x
Render in Browser
Clear
Done
Payloads: 1630
Reset

SCANNER MODULES

- GET Request Manual Mode
- GET Request Auto Mode
- DOM Scanner
- Multiple Parameter Scanner
- POST Request Scanner
- Request Repeater
- URL Fuzzer
- Hidden Parameter Detector

INFORMATION GATHERING MODULES

- WAF Fingerprinting
- Victim Fingerprinting
- Browser Fingerprinting
- Browser Features Detector
- Get Network IP
- Ping Scan
- Port Scan
- Internal Network Scan

EXPLOITATION MODULES

- Send Message
- Cookie Thief
- Phisher
- Tabnabbing
- Keylogger
- HTML5 DDoSer
- Load File
- Grab Page Screenshot
- JavaScript Shell

UTILITY MODULES

- WebKit Developer Tools
- Payload Encoder
- JavaScript Beautify
- Hash Calculator
- Hash Detector

XENOTIX SCRIPTING ENGINE

- Xenotix API
- IronPython Scripting Support
- Trident and Gecko Web Engine Support

- XSS Filter Bypass, Detection and Explanation with OWASP Xenotix v5
  - https://www.youtube.com/watch?v=loZSdedJnqc#t=1027

Reference

[1] OWASP Code Review Project Site, https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project

[2] OWASP Testing Project Site, https://www.owasp.org/index.php/Category:OWASP_Testing_Project

[4] The OWASP Source Code Flaws Top 10, https://www.owasp.org/index.php/OWASP_Source_Code_Flaws_Top_10_Project_Index

[6] OWASP Xenotix XSS Exploit Framework Site, https://www.owasp.org/index.php/OWASP_Xenotix_XSS_Exploit_Framework

[5] AJIN ABRAHAM, Detecting and Exploiting XSS With OWASP XENOTIX XSS EXPLOIT Framework, Blackhat Europe 2013

[6] Gary McGraw, "Software Security", IEEE Security & Privacy, pp. 80-83, 2004.3.

[7] Gary McGraw, "Software Security Testing", IEEE Security and Privacy archive, Page 81-85, 2004.9.

[8] F. Piessens, "A Taxonomy of Causes of Software Vulnerabilities in Internet Software", Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering, pp 47-52, 2002.11.

[9] Microsoft SDL Site, www.microsoft.com/security/sdl/

[10] 조성제, 김동진, "소프트웨어 취약점, 보증 및 보안 테스팅", 정보과학회지 제30권 제2호, 2012.2.

FAQ

- Q & A
- Suggestion (yongsik.choi@owasp.org)

- The OWASP Foundation
  - https://www.owasp.org
- OWASP Korea Chapter
  - http://www.owasp.or.kr
  - https://www.owasp.org/index.php/Korea
  - https://www.facebook.com/groups/owaspk