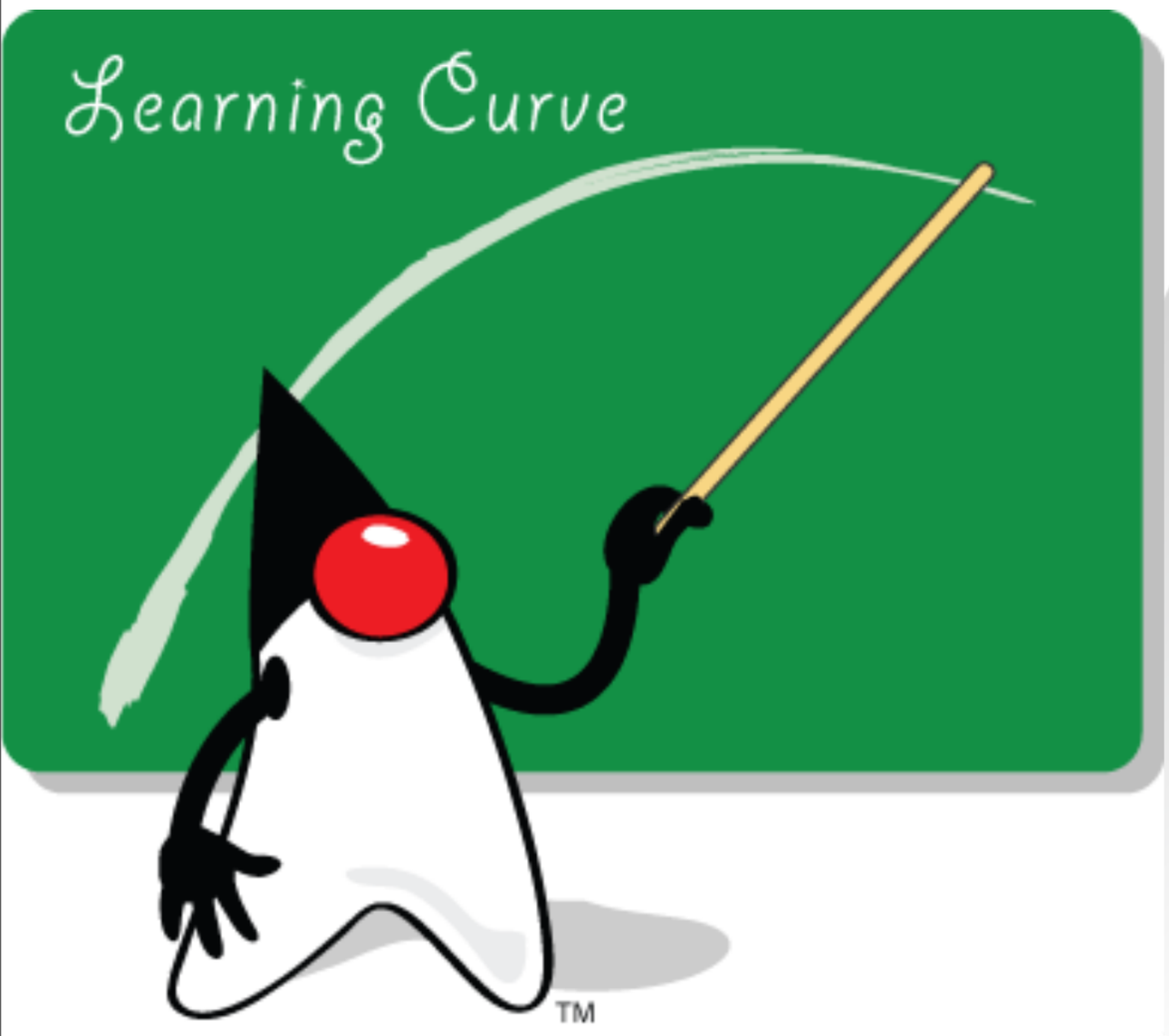
Secure Coding for Java

An Introduction

Sebastien Gioria
OWASP France Leader

OWASP Montreal
Feb 26th 2013

Agenda



- Introduction
- Using OWASP materials to secure code
- Secure Coding principles

http://www.google.fr/#q=sebastien gioria

➡Application Security Consultant

➡OWASP France Leader & Founder - Evangéliste

➡ OWASP Global Education Comittee Member
(sebastien.gioria@owasp.org)

➡Leader and technical advisor on the Web App security group at CLUSIF

Twitter :@SPoint

CISA && ISO 27005 Risk Manager

★More than 15 years of manager and technical leads in differents firms ; bank, insurance, telecom, startups, ...

★Technical Expertise

- ★Securing SDLC
- ★Pentesting
- ★CodeReview
- ★Risk management, audits
- ★Security and Network training

ForeWords

- **This is a presentation made from my own experience with a big number of company using OWASP materials.**

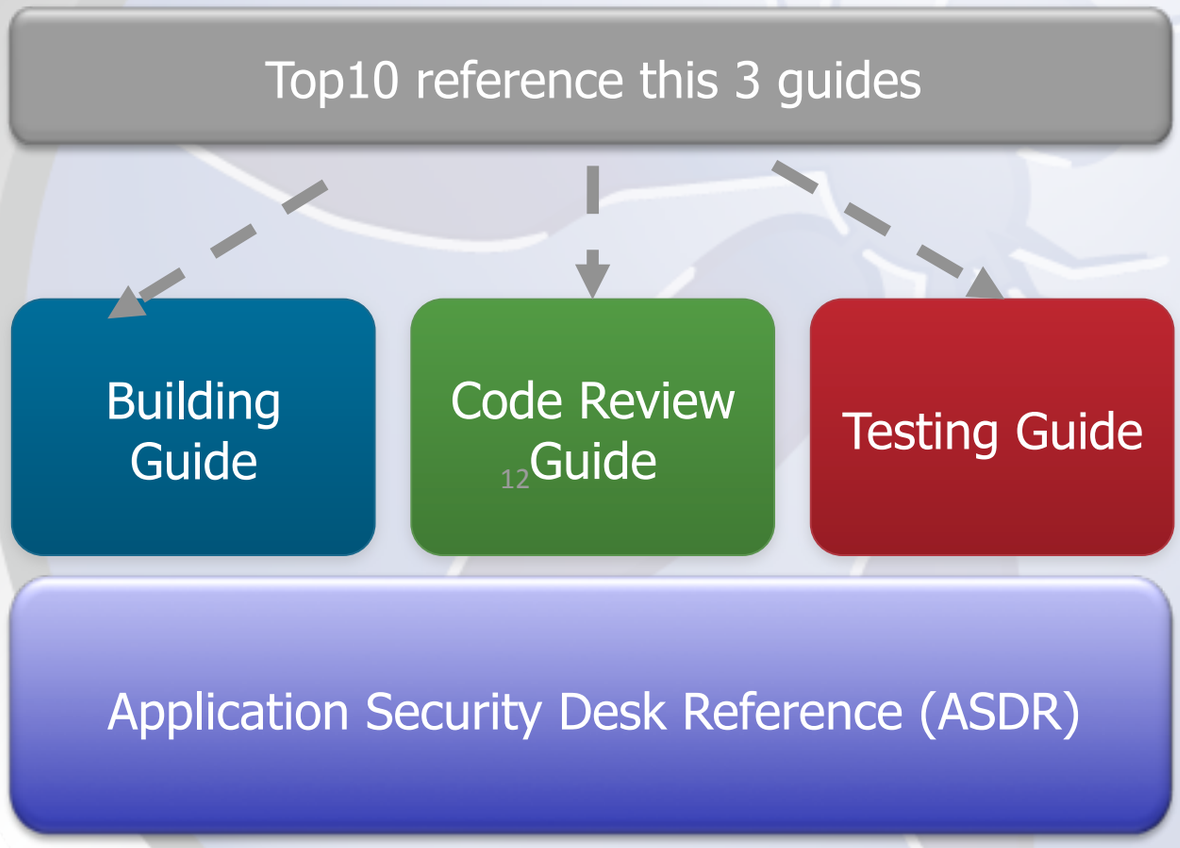
- Only the documents from OWASP wiki are OWASP officials (see https://www.owasp.org)

- Some extracts come from document I wrote as OWASP leader, this is why you could find it elsewhere.

Majors OWASP publications we can use

- All are on the wiki https://www.owasp.org
- All are under GPL or friendly licenses
- Majors publications you can use to secure your projects/SDLC

- OWASP Top10
- Auditor/Testing Guide
- Code Review Guide
- Building Guide
- Application Security Verification Standard (ASVS)
- Secure Coding Practices



Learning

Learning


OWASP
OWASP Top 10 - 2010
release


OWASP
OWASP Secure Software
Contract Annex v1.0
release

Learning

Contract

Learning

Contract

Learning

Contract

Testing

Learning

Contract

Testing

Learning

Contract

Build

Testing

Learning

Contract

Build

Testing

Learning

Contract

Build

Check

Testing

Learning

Contract

Build

Check

Testing

Learning

Contract

Build

Check

Testing

Progress

Introduction

Consequences of bad or no security

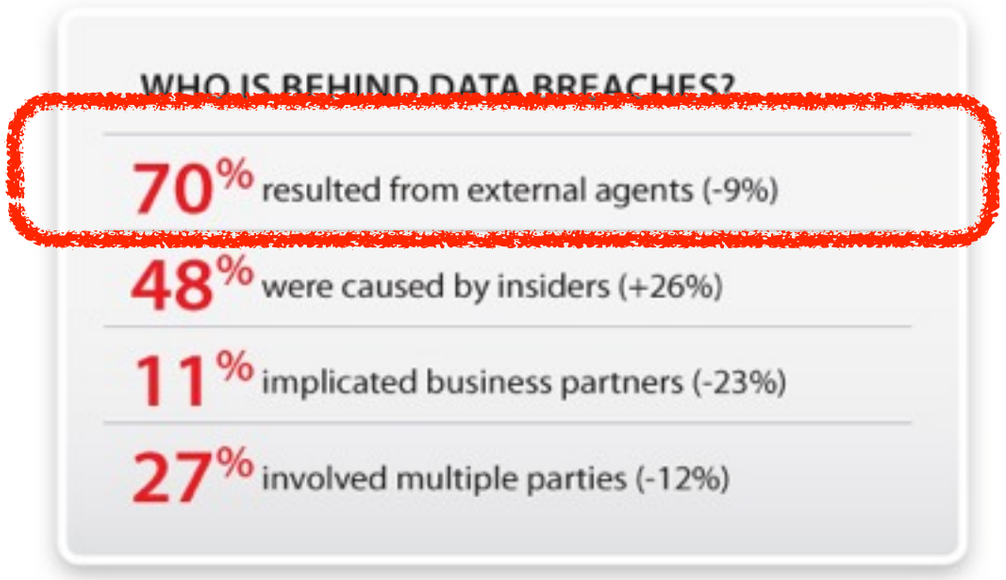
- Identity theft
- Hardware theft
- Bad Media coverage
- Customers loss
- Legals/business penalty
- Financials loss
- IT downtime

What Verizon (PCI-DSS company) said ?

WHO IS BEHIND DATA BREACHES?

- **70%** resulted from external agents (-9%)
- **48%** were caused by insiders (+26%)
- **11%** implicated business partners (-23%)
- **27%** involved multiple parties (-12%)

What Verizon (PCI-DSS company) said ?



What Verizon (PCI-DSS company) said ?

WHO IS BEHIND DATA BREACHES?

- **70%** resulted from external agents (-9%)
- **48%** were caused by insiders (+26%)
- **11%** implicated business partners (-23%)
- **27%** involved multiple parties (-12%)

HOW DO BREACHES OCCUR?

- **48%** involved privilege misuse (+26%)
- **40%** resulted from hacking (-24%)
- **38%** utilized malware (<>)
- **28%** employed social tactics (+16%)
- **15%** comprised physical attacks (+6%)

What Verizon (PCI-DSS company) said ?

WHO IS BEHIND DATA BREACHES?

- **70%** resulted from external agents (-9%)
- **48%** were caused by insiders (+26%)
- **11%** implicated business partners (-23%)
- **27%** involved multiple parties (-12%)

HOW DO BREACHES OCCUR?

- **48%** involved privilege misuse (+26%)
- **40%** resulted from hacking (-24%)
- **38%** utilized malware (<>)
- **28%** employed social tactics (+16%)
- **15%** comprised physical attacks (+6%)

What Verizon (PCI-DSS company) said ?

WHO IS BEHIND DATA BREACHES?

- **70%** resulted from external agents (-9%)
- **48%** were caused by insiders (+26%)
- **11%** implicated business partners (-23%)
- **27%** involved multiple parties (-12%)

HOW DO BREACHES OCCUR?

- **48%** involved privilege misuse (+26%)
- **40%** resulted from hacking (-24%)
- **38%** utilized malware (<>)
- **28%** employed social tactics (+16%)
- **15%** comprised physical attacks (+6%)

WHAT COMMONALITIES EXIST?

- **98%** of all data breached came from servers (-1%)
- **85%** of attacks were not considered highly difficult (+2%)
- **61%** were discovered by a third party (-8%)
- **86%** of victims had evidence of the breach in their log files
- **96%** of breaches were avoidable through simple or intermediate controls (+9%)
- **79%** of victims subject to PCI DSS had not achieved compliance

What Verizon (PCI-DSS company) said ?

WHO IS BEHIND DATA BREACHES?

70% resulted from external agents (-9%)

48% were caused by insiders (+26%)

11% implicated business partners (-23%)

27% involved multiple parties (-12%)

HOW DO BREACHES OCCUR?

48% involved privilege misuse (+26%)

40% resulted from hacking (-24%)

38% utilized malware (<>)

28% employed social tactics (+16%)

15% comprised physical attacks (+6%)

WHAT COMMONALITIES EXIST?

98% of all data breached came from servers (-1%)

85% of attacks were not considered highly difficult (+2%)

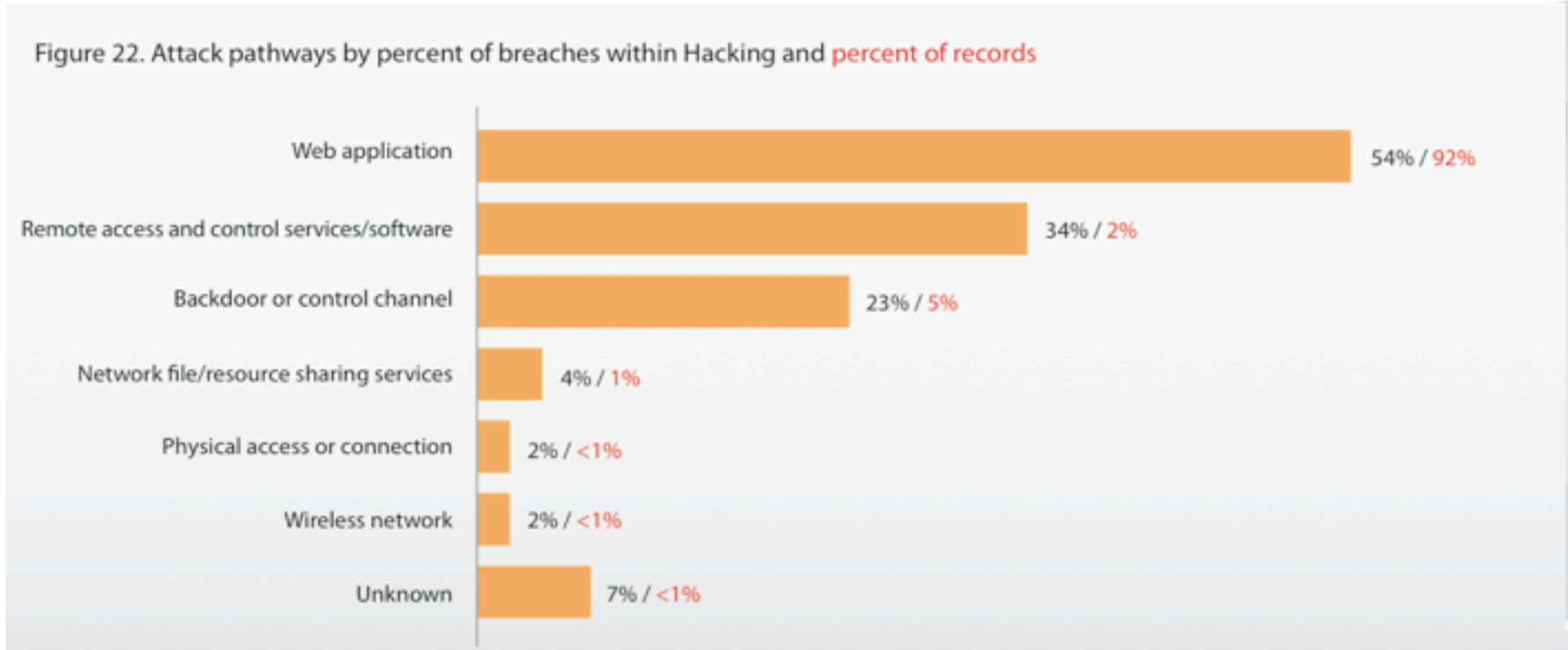
61% were discovered by a third party (-8%)

86% of victims had evidence of the breach in their log files

96% of breaches were avoidable through simple or intermediate controls (+9%)

79% of victims subject to PCI DSS had not achieved compliance

Verizon Study



Verizon Study

Figure 22. Attack pathways by percent of breaches within Hacking and percent of records

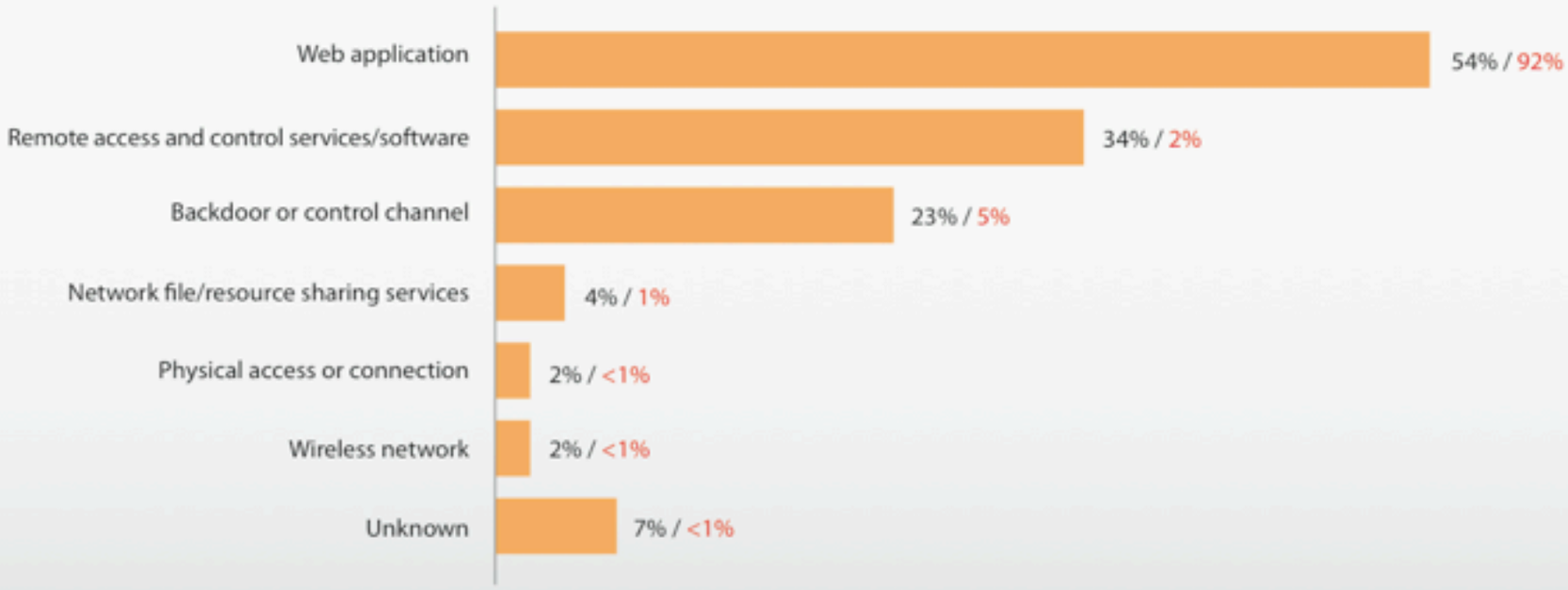
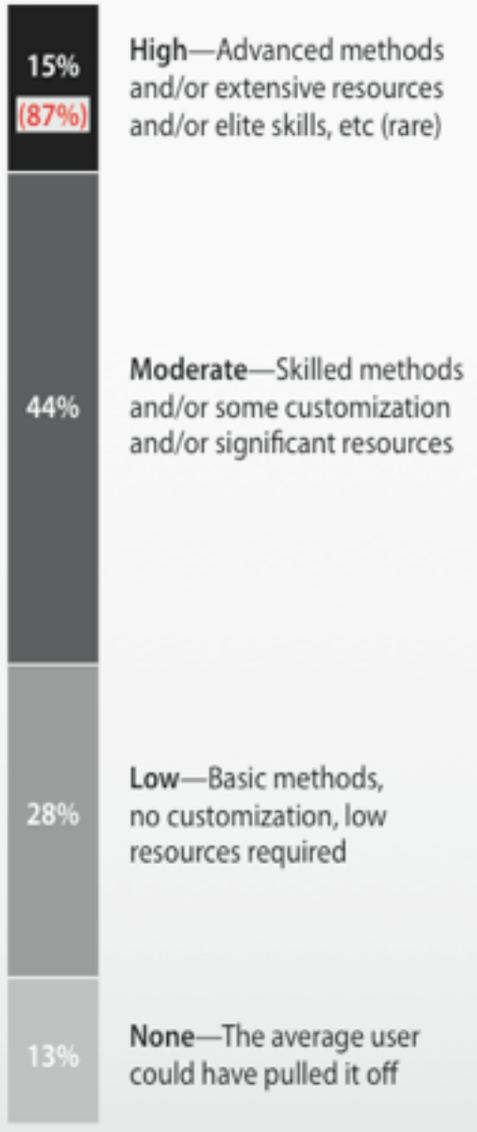
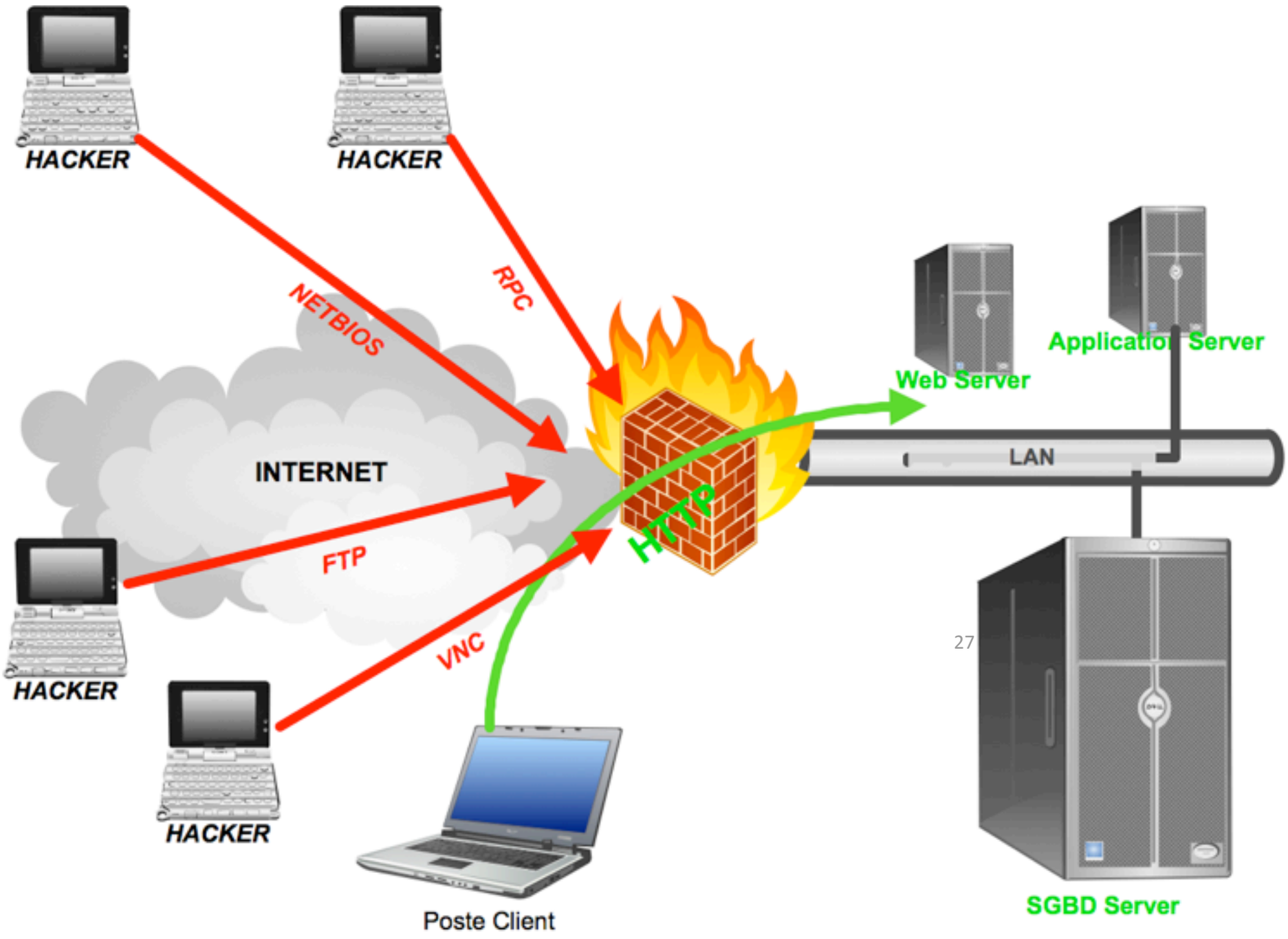


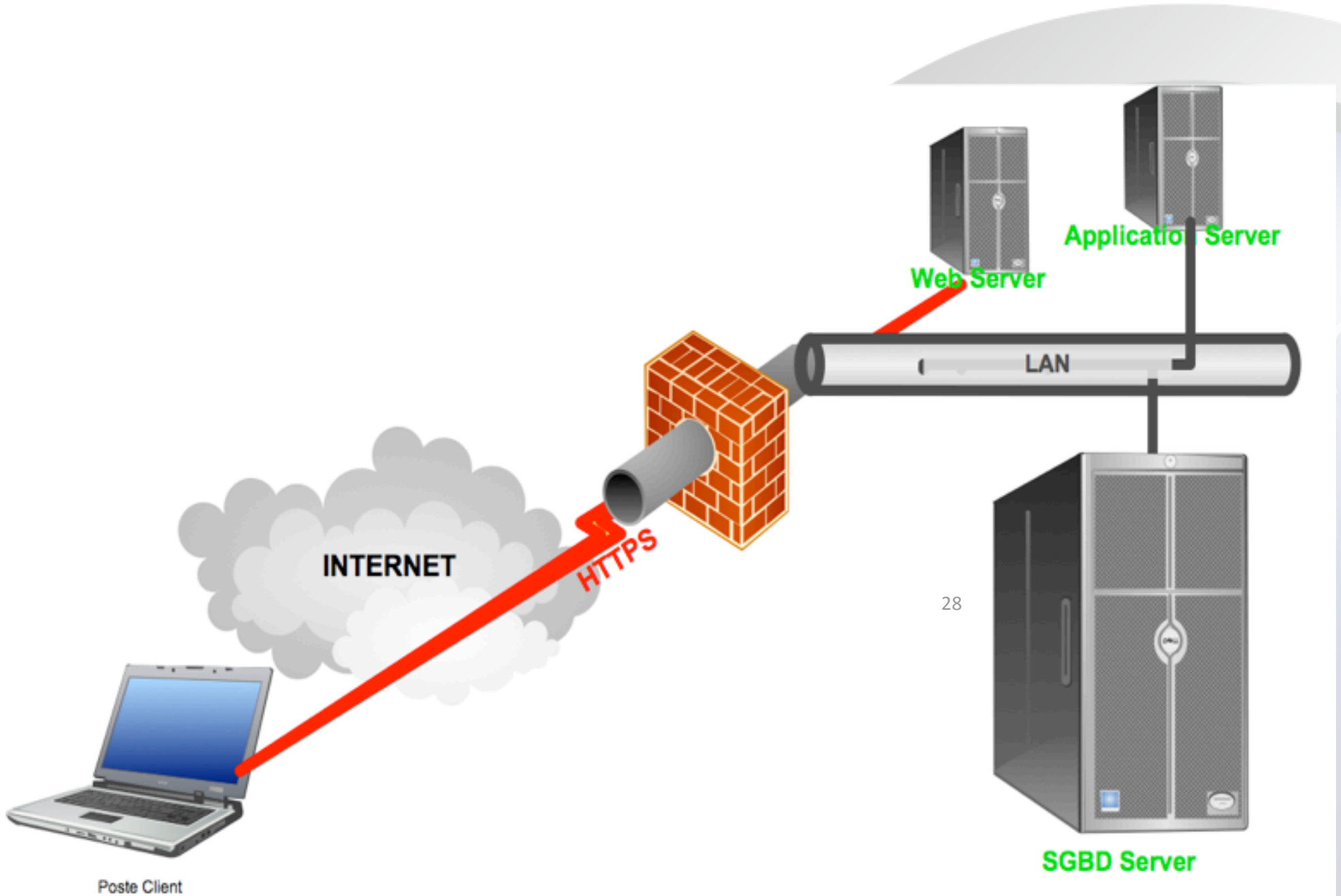
Figure 32. Attack difficulty by percent of breaches and records*



What you CIO Said : I got a Firewall !



What your business user said : I have SSL based Web Site



What your business user said : only the hacker can attack my website

- Tools are more and more simples.
- Try a simple request on google website on SQL Injection and look at it.
- An attack on a Web Server cost 100$/200$ per day on the underground market.

What your user said : a vulnerability on internal WebApp is not critical.

- No, The web is anywhere, and CSRF, HTML5 CORS and more can make this completly destructive

- Be aware and share this :
  - **AJAX doing a lot of things without you**

- Be aware and share this :
  - HTML5 will come with “nice” user functionnality , but with big impact on security (WebSocket, CORS, ...)

OWASP Application Security Verification Standard

What is ASVS ?

- A standard that provides a basis for the verification of web applications application-independent.
- A standard life-cycle model independent.
- A standard that define requirements that can be applied across applications without special interpretation.

What are ASVS responses ?

- How much trust can be placed in a web application?
- What features should be built into security controls?
- How do I acquire a web application that is verified to have a certain range in coverage and level of rigor?

ASVS secure controls requirements

Security Area	Level 1A	Level 1B	Level 2A	Level 2B	Level 3	Level 4
V1 – Security Architecture Verification Requirements	1	1	2	2	4	5
V2 – Authentication Verification Requirements	3	2	9	13	13	14
V3 – Session Management Verification Requirements	4	1	6	7	8	9
V4 – Access Control Verification Requirements	5	1	12	13	14	15
V5 – Input Validation Verification Requirements	3	1	5	7	8	9
V6 – Output Encoding/Escaping Verification Requirements	0	1	2	8	9	10
V7 – Cryptography Verification Requirements	0	0	2	8	9	10
V8 – Error Handling and Logging Verification Requirements	1	1	2	8	8	9
V9 – Data Protection Verification Requirements	1	1	2	3	4	4
V10 – Communication Security Verification Requirements	1	0	3	6	8	8
V11 – HTTP Security Verification Requirements	3	3	6	6	7	7
V12 – Security Configuration Verification Requirements	0	0	0	2	3	4
V13 – Malicious Code Search Verification Requirements	0	0	0	0	0	5
V14 – Internal Security Verification Requirements	0	0	0	0	1	3
Totals	22	12	51	83	96	112

But ASVS stand for Verification ?

- ASVS just said functionals needs for controls.
- We could use it as a Secure Coding Policy.

★**Don’t be medium(ASVS Level1/2), just target excellence (ASVS Level 4)**

Using ASVS as a secure coding policy

ASVS : Verify that all password fields do not echo the user's password when it is entered.

- **All Password fields must be define as HTML passwd fields and must not echo user passwd.**
- **All login forms must include autocomplete=off tag**

ASVS : Verify that all input validation is performed on the server side.

- **Performs all input validation on the server. Nothing in the browser**

Positive attitude

- Negative
  - The tester shall search for XSS holes
- Positive
  - Verify that the application performs input validation and output encoding on all user input

See: http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

OWASP Secure Coding Practices

OWASP Secure Coding Practices

- Small document (only 9 pages)
- Could be use as an simple checklist for your policy.
- Could be use together with ASVS or alone.
- More technical and deeper approach than ASVS .
- Wrote and use by Boeing :)

Secure Coding Practices Contents

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handling and Logging
- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices

Now the torture room

Let talk Secure Coding now

(extracts from OWASP Secure Coding Practices/
OWASP CheatSheets OWASP ASVS, ...)

Some secures principles to follow

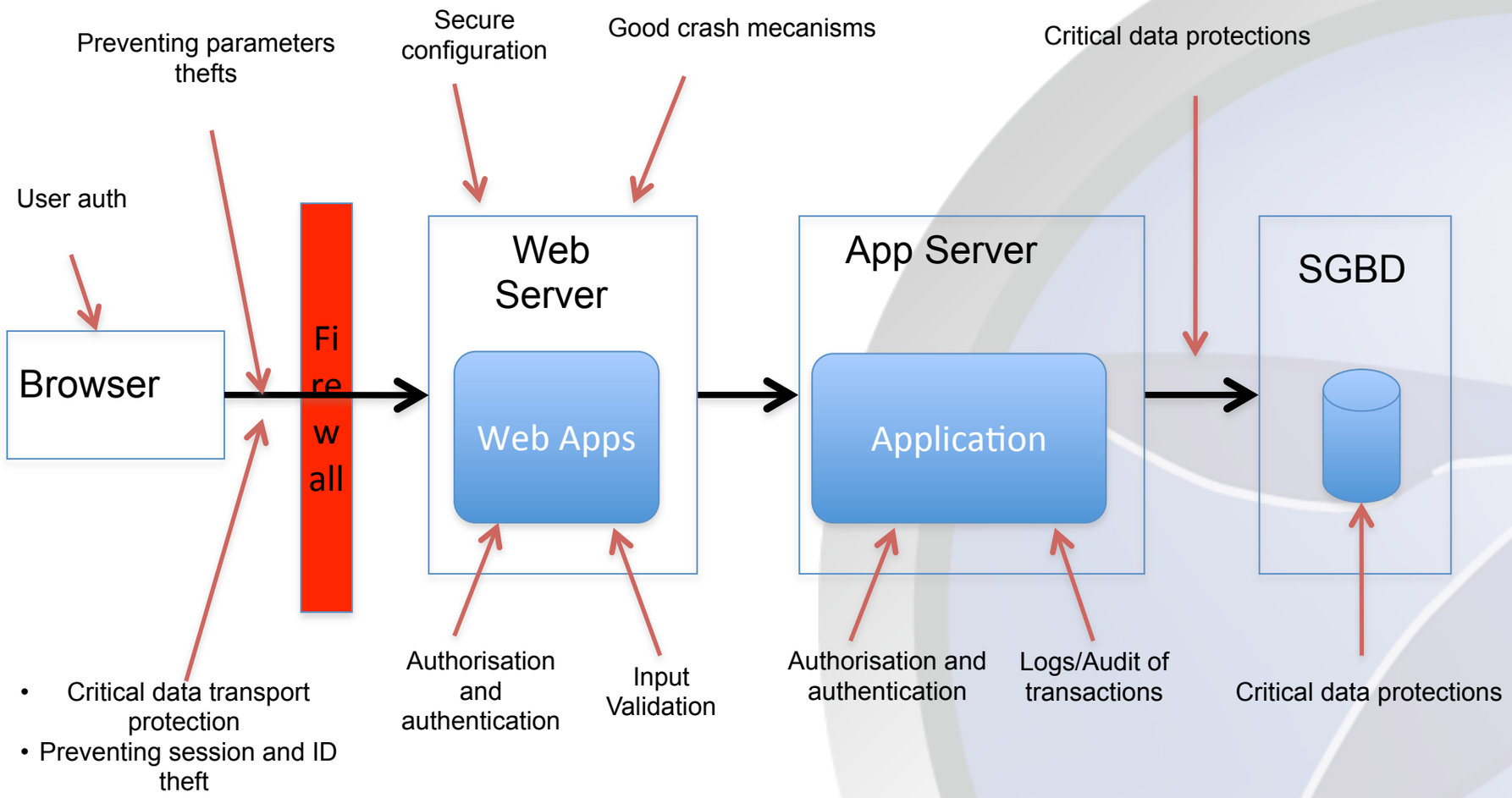
- **Deep defense of application is mandatory**

- Following less privileges is the best solution

- Segregate duty more that user think

➡ **Remember that application need to answer user needs and not security pleasure.**

Deep defense of an Application (example)



Fail securely

Don’t give user technical details of the crash.

Example :

- 404

Vous tentez d'accéder à une ressource qui n'existe pas sur ce site internet.
Vous pouvez, si vous le souhaitez prendre contact avec notre service assistance
afin d'avoir de plus amples informations.

- 500

Une erreur vient de se produire pendant votre session de travail n° B70C25E4C8C0098BBEE9FECA8FD3D725
Vous pouvez contacter notre **Service Assistance.**

Fail Securely

```
public class MyAccessController {

    public boolean testAccess(Object securedResource) {
        boolean accessGranted = false;

        try {
            if ( ESAPI.AccessController().isAuthorizedForData(securedResource) ) {
                accessGranted = true;
            }
        }
        catch (Exception ex) {
            // Log result
            accessGranted = false;
            logger.audit("Access control failure", ...);
        }

        return accessGranted;
    }
}
```

Don’t try to make obscure things

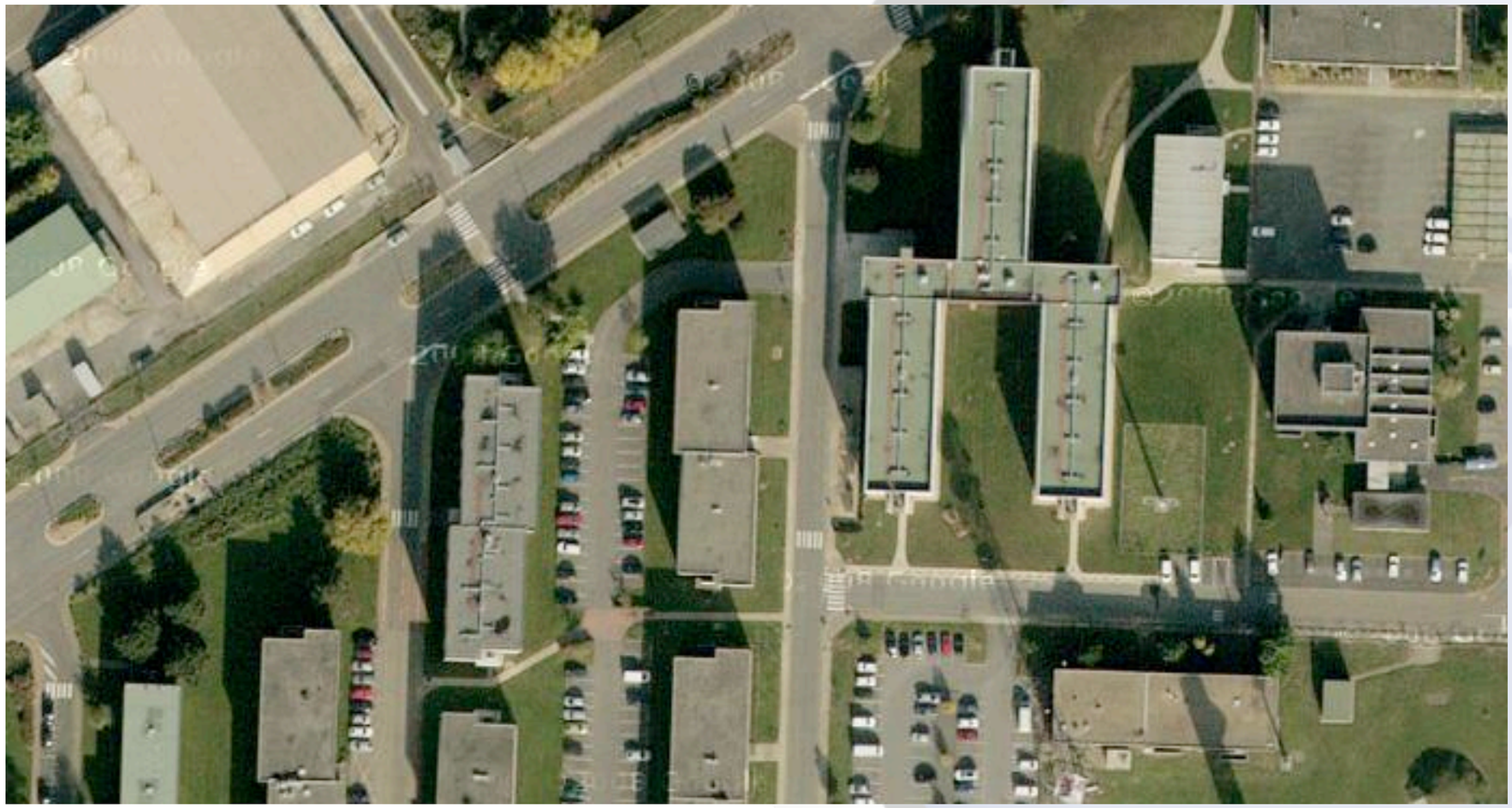
Don’t try to make obscure things

GEOPORTAIL

Don’t try to make obscure things

Don’t try to make obscure things

GOOGLE MAPS



Controls

- Controls need :
  - to be simple
  - to be used correctly
  - functional
  - present in every part of the application

Bad understanding of a control result of unused it by developers and application will be vulnerable.

Minimals controls to have

You must have at least this components in your application :

- Authentication
- Authorization
- Logging and audit
- Secure Storage
- Secure transport
- Secure input and output manipulation of data

Authentication

Implement good passwd strategy

Password length

- Categorize applications :
  - Important : at least 6 characters
  - Critical : at least 8 characters and perhaps multi-factors authentication
  - High Critical : at least 14 characters and multi-factors authentication

Password strength

- Implement passwd complexity with previous categories
  - at least : 1 upper, 1 lower, 1 digit, 1 special
  - don’t allow dictionnary passwd
  - don’t allow continuous characters

Implement good passwd strategy

- Let the user choose it
- Force the user to change it regulary, and add no reuse capability.
- Don’t allow too much “I forgot my passwd”
- Don’t allow change of passwd without user approval; require actual passwd from the user and more for high critical.
- **Add sleep strategy !**
- **Add detection of misuse strategy !**
- **Don’t store passwd in clear !!!!! use hash !**

Multi-Factor authentication

- Passwds are bad
- Passwds are guessable
- Multi-factor combine:
  - something you have (token, mobile, ...)
  - something you know (details about you, passwd, ...)
  - sometime, something you are (biometrics)
  - Use it for high critical applications.

Implement good global strategy

- Ask second authentication for critical transactions (with multi-factor auth...)
- Force authentication to be in TLS/SSL
- Regenerate Session ID after authentication
- Force Session ID to be “secure”
- Limiting forgotten passwd,change of login/passwd

How to do ?

- Authenticate all pages but not public pages (login, logout, help, ...)
- Don't allow more than one authentication mecanism
- **Authenticate on the SERVER**
- Simply send back "user or passwd mismatch" and nothing else after a failed authentication.
- Logged all failed and all correct authentication
- After each authentication give the user the last status of his authentication.

Session Management

Session

- Use Default Java Framework Generator
- Use other name than the default name of the Framework (rename JSESSIONID...)
- Force transport of ID authentication on SSL/TLS.
- Don’t allow Session ID in URL !
- If using cookie :
  - Secure Cookie
  - HTTPOnly Cookie
  - Limiting path + domain
  - Max Age and expiration

Session tricky

Automatic expiration

- categorize applications :
  - default : 1 hour
  - critical (some transaction) : 20mns
  - high critical (financials or account impact) : 5mns

Renew Session ID after any privilege change

Don’t allow simultaneous logon

Add Session Attack Detection

- add in-session tips : ip of session, other random number, ...

Browser defenses

Bind JavaScript events to close session

- on window.close()
- on window.stop()
- on window.blur()
- on window.home()

Use Javascripts timer to automatic close session in high critical applications

Disable WebBrowser Cross-tab Session if possible...(bad user experiences....)

- **If you use cookie, this is not possible !!!!**

```
Cookie ck = new Cookie ("nom", valeur);

ck.setPath("/mypath");
ck.setDomain("server1.example.com");
ck.setMaxAge(-1); // Cookie de session, non stocké
ck.setSecure();

//Ajout du HTTPOnly
String sessionid = request.getSession().getId();
response.setHeader("SET-COOKIE", "JSESSIONID=" + sessionid + "; HttpOnly");
```

Using Servlet 3.0 ?

```
<session-config>
  <cookie-config>
    <http-only>true</http-only>
    <secure>true</secure>
  </cookie-config>
</session-config>
```

Access Controls

Remember

Remember

(1)Without access control, you can't control the user in your application

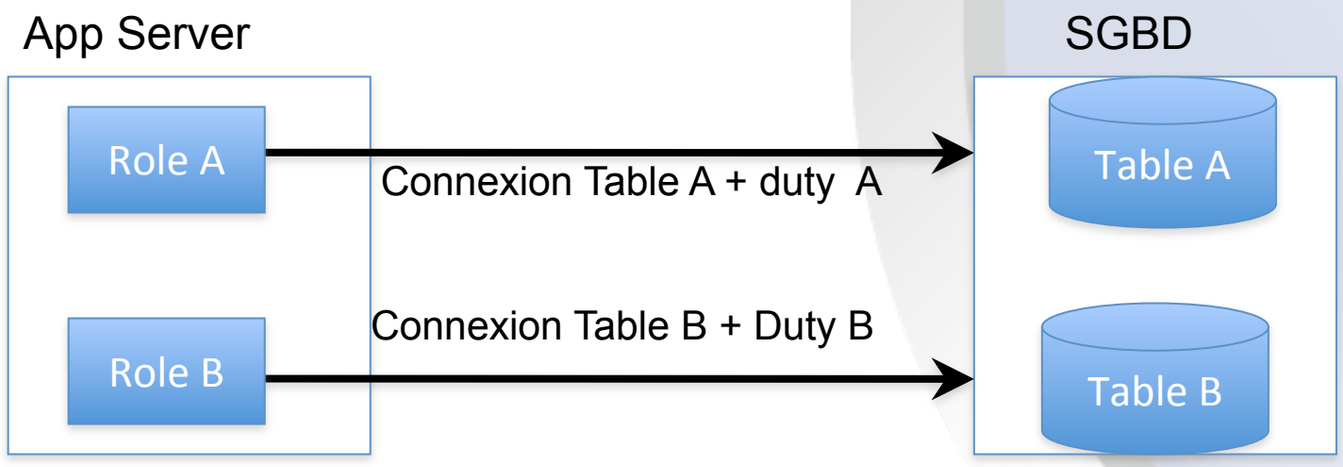
Remember

(1)Without access control, you can't control the user in your application

(2)Client inputs are EVIL

Authentication && Authorization

- Two Levels of authentication and authorization are needed
  - –In the Application
  - –In infrastructure



Authorization

Have in mind the rule :

- Nothing by default

Centralize all authorization code **on the SERVER**

If client state are mandatory, use encryption and integrity checking on the server side to catch state tampering.

Limit number of transaction per user at a interval time.

Authorization

Enforce :

- protection of URL to authorized account only
- protection of function to authorized account only
- protection of file access to authorized account only

Application need to terminate session when authorization failed.

Split administrative and user authorization

Enforce dormant account :

- loss privileges.
- “disable account”
- alerts

Input Validation

Ensure all data validation are done on **THE SERVER.**

- If you do something on client side we can said you do “painting”

Classify your data :

- Trusted Data
- Untrusted Data

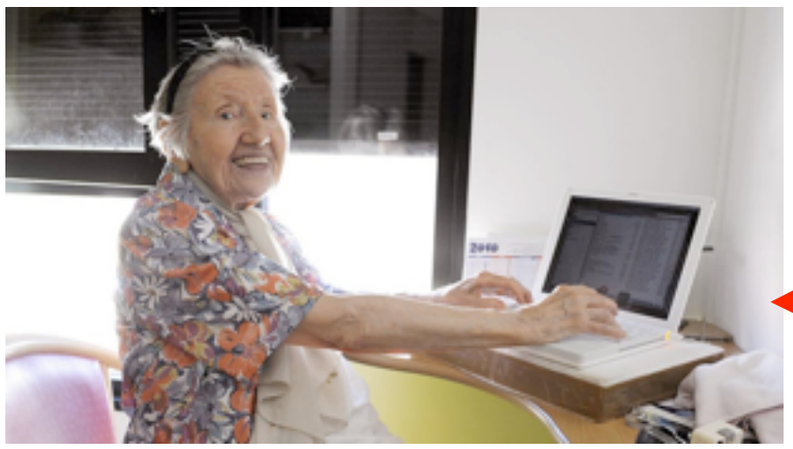
Conduct trusted path.

Centralize your data validation

Use parametrize query when exists (SQL)

Border validation

Consider validating data along all the entry points of your Application border



Input Validation

Use proper characters set for all input

Encode all data to the same character set before doing anything <=>Canonicalize

Reject all not validated datas

Validate data :

- expected type (convert as soon as possible to Java Types)
- expected range
- expected length
- expected values
- expected “white list” if possible

Input Validation

Be careful of using “hazardous” characters (ex: <>’,”!(+)&\ %.)

Add specific validation :

- check for null bytes (%00)
- check for new lines (%0D, %0A, \n, \r, ...)
- check for dot-dot-slashes (../)

Be careful of encoding for specific validation...

<script>alert(XSS);</script>

URL

%3c%73%63%72%69%70%74%3e%61%6c
%65%72%74%28%58%53%53%29%3b%3c%2f%73%63%72%69%70%74%3e
%0a

HTML

<script>ale&#x7
2;t(XSS);</sc&#x
72;ipt>

UTF-8

%u003c%uff53%uff43%uff52%uff49%uff50%uff54%u003e%uff41%uff4c
%uff45%uff52%uff54%uff08%uff38%uff33%uff33%uff09%u003c
%u2215%uff53%uff43%uff52%uff49%uff50%uff54%u003

One space ?

< s c r i p t > a l e r t (X S S) ; < / s c r i p t >

Validating Datas

```
public Boolean validateSSN(String ssnVal) {
    if (ssnVal.matches("^[1|2] [0-9]{2} \\d{2} \\d{2} \\d{6} \\d{2}$")) {
        return true;
    } else {
        return false;
    }
}
```

```
public boolean validateAmount (String amount) {
    if (validate(amount, "^\\d+$")){
        if (Long.parseLong(amount) < 10000L){
            return true;
        }
    }
    return false;
}
```

SQL => bad

```
class Login {
    public Connection getConnection() throws SQLException {
        DriverManager.registerDriver(new com.microsoft.sqlserver.jdbc.SQLServerDriver());
        String dbConnection = PropertyManager.getProperty("db.connection");
        // can hold some value like
        // "jdbc:microsoft:sqlserver://<HOST>:1433,<UID>,<PWD>"
        return DriverManager.getConnection(dbConnection);
    }
    String hashPassword(char[] password) {
        // create hash of password
        return ("Hash");
    }

    public void doPrivilegedAction(String username, char[] password) throws SQLException {
        Connection connection = getConnection();
        if (connection == null) {
            // handle error
        }
        try {
            String pwd = hashPassword(password);
            String sqlString = "SELECT * FROM db_user WHERE username = '" + username +"' AND password = '" + pwd + "'";
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(sqlString);
            if (!rs.next()) {
                throw new SecurityException("User name or password incorrect" );
            }
            // Authenticated; proceed
            } finally {
                try {
                    connection.close();
                    } catch (SQLException x) {
                        // forward to handler
                    }
                }
        }
    }
```

SQL => bad

```
class Login {
    public Connection getConnection() throws SQLException {
        DriverManager.registerDriver(new com.microsoft.sqlserver.jdbc.SQLServerDriver());
        String dbConnection = PropertyManager.getProperty("db.connection");
        // can hold some value like
        // "jdbc:microsoft:sqlserver://<HOST>:1433,<UID>,<PWD>"
        return DriverManager.getConnection(dbConnection);
    }
    String hashPassword(char[] password) {
        // create hash of password
        return ("Hash");
    }

    public void doPrivilegedAction(String username, char[] password) throws SQLException {
        Connection connection = getConnection();
        if (connection == null) {
            // handle error
        }
        tr{
            String pwd = hashPassword(password);
            String sqlString = "SELECT * FROM db_user WHERE username = '" + username +"' AND password = '" + pwd + "'";
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(sqlString);
            if (!rs.next()) {
                throw new SecurityException("User name or password incorrect" );
            }
            // Authenticated; proceed
            } finally {
                try {
                    connection.close();
                    } catch (SQLException x) {
                        // forward to handler
                    }
                }
        }
    }
```

SQL => bad

```
class Login {
    public Connection getConnection() throws SQLException {
        DriverManager.registerDriver(new com.microsoft.sqlserver.jdbc.SQLServerDriver());
        String dbConnection = PropertyManager.getProperty("db.connection");
        // can hold some value like
        // "jdbc:microsoft:sqlserver://<HOST>:1433,<UID>,<PWD>"
        return DriverManager.getConnection(dbConnection);
    }
    String hashPassword(char[] password) {
        // create hash of password
        return ("Hash");
    }

    public void doPriv      dAction(String username, cha      assword) throws SQLException {
        Connection c          = getConnection();
        if (connec
            // ha
        }
        try {
            String pwd
            String sqlStri                                    ERE username = '" + username +"' AND password = '" + pwd + "'";
            Statement stmt =                                  nt();
            ResultSet rs =
            if (!rs.nex
                throw                                         word incorrect" );
            }
            // Au
            } fina
                try
                    c       on.close();
                    } catch (SQLException x) {
                        // forward to handler
                    }
                }
    }
}
```

SQL => a little bit better

```
Connection connection = getConnection();
if (connection == null) {
    // Handle error
}
try {
    String pwd = hashPassword(password);
    // Ensure that the length of user name is legitimate if ((username.length() > 8) {
    // Handle error }
    String sqlString = "select * from db_user where username=? and password=?";
    PreparedStatement stmt = connection.prepareStatement(sqlString);
    stmt.setString(1, username);
    stmt.setString(2, pwd);
    ResultSet rs = stmt.executeQuery();
    if (!rs.next()) {
        throw new SecurityException("User name or password incorrect");
    }
    // Authenticated, proceed
} finally {
try {
    connection.close();
} catch (SQLException x) {
    // forward to handler
}
```

XML => bad

```
public class bad {
    private void createXMLStream(BufferedOutputStream outStream, String quantity) throws IOException {
        String xmlString;
        xmlString = "<item>\n<description>Widget</description>\n" +
        "<price>500.0</price>\n" +
        "<quantity>" + quantity + "</quantity></item>"; outStream.write(xmlString.getBytes());
          outStream.flush();
        }
}
```

XML => bad

```
public class bad {
    private void create                    fferedOutput                    tream, String quantity) throws IOException {
        String xmlStr
        xmlString = "<i                                                    ption>\n" +
        "<price>500.0</price
        "<quantity>" + quanti                                              m>"; outStream.write(xmlString.getBytes());
          outStream.flush
        }
}
```

XML => Validating

```
public class good {
    private void createXMLStream(BufferedOutputStream outStream, String quantity) throws IOException {
        // Write XML string if quantity contains numbers only.
        // Blacklisting of invalid characters can be performed
        // in conjunction.
        if (!Pattern.matches("[0-9]+", quantity)) {
            // Format violation
        }
        String xmlString = "<item>\n<description>Widget</description>\n" + "<price>500</price>\n" +
        "<quantity>" + quantity + "</quantity></item>"; outStream.write(xmlString.getBytes());
          outStream.flush();
        }
}
```

Better, a XML schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="item">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="description" type="xs:string"/>
                        <xs:element name="price" type="xs:decimal"/>
                        <xs:element name="quantity" type="xs:integer"/>
                </xs:sequence>
        </xs:complexType>
 </xs:element>
</xs:schema>
```

XML => XML Parser

```
private void createXMLStream(BufferedOutputStream outStream, String quantity)
        throws IOException {
    String xmlString;
    xmlString = "<item>\n<description>Widget</description>\n" + "<price>500.0</price>\n" +"<quantity>" + quantity + "</quantity></item>";
    InputSource xmlStream = new InputSource(new StringReader(xmlString) );

    // Build a validating SAX parser using our schema
    SchemaFactory sf = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
    DefaultHandler defHandler = new DefaultHandler() {
        public void warning(SAXParseException s) throws SAXParseException {throw s;}
        public void error(SAXParseException s) throws SAXParseException {throw s;}
        public void fatalError(SAXParseException s) throws SAXParseException {throw s;}
    };

    StreamSource ss = new StreamSource(new File("schema.xsd"));
    try {
        Schema schema = sf.newSchema(ss);
        SAXParserFactory spf = SAXParserFactory.newInstance();
        spf.setSchema(schema);
        SAXParser saxParser = spf.newSAXParser();
        // To set the custom entity resolver,
        // an XML reader needs to be created
        XMLReader reader = saxParser.getXMLReader();
        reader.setEntityResolver(new CustomResolver());
        saxParser.parse(xmlStream, defHandler);
    } catch (ParserConfigurationException x) {
            throw new IOException("Unable to validate XML", x);
    } catch (SAXException x) {
        throw new IOException("Invalid quantity", x);
    }
    // Our XML is valid, proceed outStream.write(xmlString.getBytes()); outStream.flush();
}
```

LDAP => bad

```
DirContext ctx = new InitialDirContext(env);

String managerName = request.getParameter("managerName");

//retrieve all of the employees who report to a manager

String filter = "(manager=" + managerName + ")";

NamingEnumeration employees = ctx.search("ou=People,dc=example,dc=com",
                                         filter);
```

LDAP => bad

```
DirContext ctx = new InitialDirContext(env);

String managerN        uest.ge        r("managerName");

//retrieve all of t                  report to a manager

String filter = "(                    ame + ")";

NamingEnumeratio       yees = ctx.    ("ou=People,dc=example,dc=com",
                                      ilter);
```

LDAP => better

```
public NamingEnumeration extractEmployees {
    DirContext ctx = new InitialDirContext(env);
    String managerName = request.getParameter("managerName");
    //retrieve all of the employees who report to a manager
    if (validate (managerName, "^\\w$")){
        String filter = "(manager=" + managerName + ")";
        NamingEnumeration employees = ctx.search("ou=People,dc=example,dc=com", filter);
    }
}
public boolean validate (String s, String pat) {
    return (Pattern.matches(pat, s));
}
}
```

Using OWASP ESAPI

```
public class MyInputValidation {

    public String realFileNameFromInput(String inputFileName) {
        ArrayList list = new ArrayList();
        list.add( new WindowsCodec() );
        Encoder encoder = new DefaultEncoder( list );

        // Only allow singly-encoded strings
        String clean = encoder.canonicalize( request.getParameter( "input" ), false);

        return clean;
    }
}
```

Output Encoding

Output encoding

It's a Defense in depth mechanism

Encode **ON THE SERVER**

Centralize the encoder functions

Sanitize all data send to the client

- HTMLEncode is a minimum but did not work on all cases

Essai 1 => bad

```
public static String sanitize(String string) {
      return string
         .replaceAll("(?i)<script.*?>.*?</script.*?>", "")   // case 1
         .replaceAll("(?i)<.*?javascript:.*?>.*?</.*?>", "") // case 2
         .replaceAll("(?i)<.*?\\s+on.*?>.*?</.*?>", "");     // case 3
   }
```

Essai 1 => bad

```
public static Str     nitize(String     {
        return st
            .replace                ript.*?>", "")    // case 1
            .replaceAll(            .*?>.*?</.*?>", "") // case 2
            .replaceAll("           >.*?</.*?>", "");     // case 3
    }
```

Essai 2 => it's bad, but better than nothing

```
class HtmlSanitizer
{
    /// A regex that matches things that look like a HTML tag after HtmlEncoding to &#DECIMAL; notation. Microsoft AntiXSS 3.0 can be used to preform this. Splits th
    /// chunks that start with &#60; and ends with either end of line or &#62;
    private static readonly Regex _tags = new Regex(@"&\#60;(?!&\#62;).+?(&\#62;|$)", RegexOptions.Singleline | RegexOptions.ExplicitCapture | RegexOptions.Compiled)
    /// A regex that will match tags on the whitelist, so we can run them through
    /// HttpUtility.HtmlDecode
    /// FIXME - Could be improved, since this might decode &#60; etc in the middle of
    /// an a/link tag (i.e. in the text in between the opening and closing tag)
    private static readonly Regex _whitelist = new Regex(@"^&\#60;(&\#47;)? (a|b(lockquote)?|code|em|h(1|2|3)|i|li|ol|p(re)?|s(ub|up|trong|trike)?|ul)&\#62;$|^&\#60;
      RegexOptions.Singleline | RegexOptions.IgnorePatternWhitespace |RegexOptions.ExplicitCapture | RegexOptions.Compiled);
    /// HtmlDecode any potentially safe HTML tags from the provided HtmlEncoded HTML input using
    /// a whitelist based approach, leaving the dangerous tags Encoded HTML tags
    public static string Sanitize(string html)
    {
        Match tag;
        MatchCollection tags = _tags.Matches(html);

        // iterate through all HTML tags in the input
        for (int i = tags.Count - 1; i > -1; i--)
        {
            tag = tags[i];
            string tagname = tag.Value.ToLowerInvariant();

            if (_whitelist.IsMatch(tagname))
            {
                // If we find a tag on the whitelist, run it through
                // HtmlDecode, and re-insert it into the text
                string safeHtml = HttpUtility.HtmlDecode(tag.Value);
                html = html.Remove(tag.Index, tag.Length);
                html = html.Insert(tag.Index, safeHtml);
            }
        }
        return html;
    }
}
```

Essai 2 => it’s bad, but better than nothing

```
class HtmlSanitizer
{
    /// A regex that matches t      that look like a HTML tag aft      mlEncoding to &#DECIMAL; notation. Microsoft AntiXSS 3.0 can be used to preform this. Splits th
    /// chunks that start wi      d ends with either end      &#62;
    private static readonl      ew Regex(@"&\#6      (&\#62;|$)", RegexOptions.Singleline | RegexOptions.ExplicitCapture | RegexOptions.Compiled)
    /// A regex that will      itelist      hrough
    /// HttpUtility.HtmlDe
    /// FIXME - Could be impr      the middle of
    /// an a/link tag (i.e. in th      closing tag)
    private static readonly Regex _wh      60;(&\#47;)? (a|b(lockquote)?|code|em|h(1|2|3)|i|li|ol|p(re)?|s(ub|up|trong|trike)?|ul)&\#62;$|^&\#60;
      RegexOptions.Singleline | Reg      ace |RegexOptions.ExplicitCapture | RegexOptions.Compiled);
    /// HtmlDecode any potenti      HtmlEncoded HTML input using
    /// a whitelist based ap      TML tags
    public static string
    {
        Match tag;
        MatchCollection tags      atches(html);

        // iterate through all HTML tags in the input
        for (int i = tags.Count - 1; i > -1; i--)
        {
            tag = tags[i];
            string tagname = tag.Value.ToLowerInvariant();

            if (_whitelist.IsMatch(tagname))
            {
                // If we find a tag on the whitelist, run it through
                // HtmlDecode, and re-insert it into the text
                string safeHtml = HttpUtility.HtmlDecode(tag.Value);
                html = html.Remove(tag.Index, tag.Length);
                html = html.Insert(tag.Index, safeHtml);
            }
        }
        return html;
    }
}
```

A good solution with a robust Sanitizer :)

```
// AntiSammy
//http://code.google.com/p/owaspantisamy/downloads/list
Policy policy = Policy.getInstance(POLICY_FILE_LOCATION);
AntiSamy as = new AntiSamy();
CleanResults cr = as.scan(dirtyInput, policy);
MyUserDAO.storeUserProfile(cr.getCleanHTML()); // some custom function
```

```
// OWASP Java Sanitizer
// http://owasp-java-html-sanitizer.googlecode.com/svn/trunk/distrib/javadoc/org/owasp/html/Sanitizers.html
 PolicyFactory sanitizer = Sanitizers.FORMATTING.and(Sanitizers.BLOCKS);
  String cleanResults = sanitizer.sanitize("<p>Hello, <b>World!</b>");
```

Error Logging

Error Handling

Your Application will crash !

Catch all exceptions without exception (remember the null pointer exception !)

- Clean all exception code of sensitive datas
- Don’t give user any details about crash, just said “It’s a crash, try again later”

Logs are sensitive, you **MUST PROTECT THEM**

Log :

- input validation failures
- authentication request; especially failures
- access control failures
- systems exceptions
- administrative functionality
- crypto failures
- invalid/expired session token access

Logging/Errors

Split your logs with categories, examples :

- Access
- Error
- Debug
- Audit

Use log4j for standard logging

Log4J Example

```
import com.sec.dev;

 // Import log4j classes.
 import org.apache.log4j.Logger;
 import org.apache.log4j.BasicConfigurator;

 public class SecLogger {

   // Define a static logger variable so that it references the
   // Logger instance named "MyApp".
   static Logger logger = Logger.getLogger(MyApp.class);

   public static void main(String[] args) {

     // Set up a simple configuration that logs on the console.
     BasicConfigurator.configure();

     logger.setLevel(Level.DEBUG); // optional if log4j.properties file not used
     // Possible levels: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL

     logger.info("Entering application.");
     Bar bar = new Bar();
     bar.doIt();
     logger.info("Exiting application.");
   }
 }
```

Bad handling of Exception

```
class ExceptionExample {
    public static void main(String[] args) throws FileNotFoundException {
        // Linux stores a user's home directory path in
        // the environment variable $HOME, Windows in %APPDATA%
        FileInputStream fis = new FileInputStream(System.getenv("APPDATA") + args[0]); }
}
```

```
try {
FileInputStream fis = new FileInputStream(System.getenv("APPDATA") + args[0]);
} catch (FileNotFoundException e) {
    // Log the exception
    throw new IOException("Unable to retrieve file", e);
}
```

Bad handling of Exception

```
class ExceptionExample {
    public static void main(String[] args) throws FileNotFoundException {
        // Linux stores a user's home directory path in
        // the envir  ent variable $HOME, Windows in  PPDATA%
        FileInputS  = new FileInputStrea  tenv("APPDATA") + args[0]); }
}
```

```
try {
FileInputStrea  putStream(  DATA") + args[0]);
} catch (FileNot  on e) {
    // Log the exce  n
    throw new IOException("Unable to retrieve file", e);
}
```

Good handling of exception

```
class ExceptionExample {
    public static void main(String[] args) {
        FileInputStream fis = null; try {
            switch(Integer.valueOf(args[0])) {
                case 1:
                    fis = new FileInputStream("c:\\homepath\\file1");
                    break;
                case 2:
                    fis = new FileInputStream("c:\\homepath\\file2");
                    break; //...
                default:
                    System.out.println("Invalid option");
                    break;
            }
        } catch (Throwable t) {
            MyExceptionReporter.report(t); // Sanitize }
        }
}
```

```
<error-page>
  <exception-type>java.lang.Throwable</
exception-type>
  <location>/error.jsp</location>
 </error-page>
```

Data Protection

Data protection

Protect sensitive datas, don’t store them in clear.

Store sensitive datas in trusted systems

Don’t use GET request for sensitive data.

Disable client site caching

Disable Client Side caching

```
import javax.servlet.*;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Date;

public class CacheControlFilter implements Filter {

    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain chain) throws IOException, ServletException {

        HttpServletResponse resp = (HttpServletResponse) response;
        resp.setHeader("Expires", "Tue, 03 Jul 2001 06:00:00 GMT");
        resp.setHeader("Last-Modified", new Date().toString());
        resp.setHeader("Cache-Control", "no-store, no-cache, must-revalidate, max-age=0, post-check=0, pre-check=0");
        resp.setHeader("Pragma", "no-cache");

        chain.doFilter(request, response);
    }

}
```

web.xml

```
<filter>
    <filter-name>SetCacheControl</filter-name>
    <filter-class>com.sec.dev.cacheControlFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>SetCacheControl</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Acces to FileSystem

Absolute Path is bad

```
public static void badSecureOpenFile(String[] args) {
    File f = new File(System.getProperty("user.home") + System.getProperty("file.separator") + args[0]);

    String absPath = f.getAbsolutePath();
    if (!isInSecureDir(Paths.get(absPath))) {
        throw new IllegalArgumentException();
    }
    if (!validate(absPath)) {
        // Validation
        throw new IllegalArgumentException();
    }
}
```

Absolute Path is bad

```
public static void badSecureOpenFile(String[] args) {
    File f = new File(System.getProperty("user.home") + System.getProperty("file.separator") + args[0]);

    String absPath = f.getAbsolutePath();
    if (!isInSecureDir(Paths.get(absPath))) {
        throw new IllegalArgumentException();
    }
    if (!validate(absPath)) {
        // Validation
        throw new IllegalArgumentException();
    }
}
```

Absolute Path is bad

```
public static vo    dSecureOpenFile(St    ] args) {
    File f = n      tem.getPro          home") + System.getProperty("file.separator") + args[0]);

    String absPa
    if (!isInSecureDi                  ))) {
        throw new I                   ();
    }
    if (!vali                         ) {
        // Vali
        throw new IllegalArgumentException();
    }
}
```

Canonicalisation is good

```
public static void goodSecureOpenFile (String[] args) throws IOException {
    File f = new File(System.getProperty("user.home") + System.getProperty("file.separator")+ args[0]);

    String canonicalPath = f.getCanonicalPath();
    if (!isInSecureDir(Paths.get(canonicalPath))) {
        throw new IllegalArgumentException();
    }
    if (!validate(canonicalPath)) {
        // Validation
        throw new IllegalArgumentException();
    }
}
```

Secure Communications

Secure Communications

Use TLS/SSL :

- at least SSL v3.0/TLS 1.0
- minimum of 128bits encryption
- use secure crypto : AES is good

Don’t expose critical data in the URL

Failed SSL/TLS communications should not fall back to insecure

Validate certificate when used

Protect all page, not just logon page !

Force TLS/SSL Response

Use HTTP Strict Transport Security (HSTS).

- Available on some browsers (not IE)
- draft IETF : http://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-04

```
HttpServletResponse ...;
response.setHeader("Strict-Transport-Security", "max-age=7776000;
includeSubdomains");
```

Configuration

Review all properties, configuration files

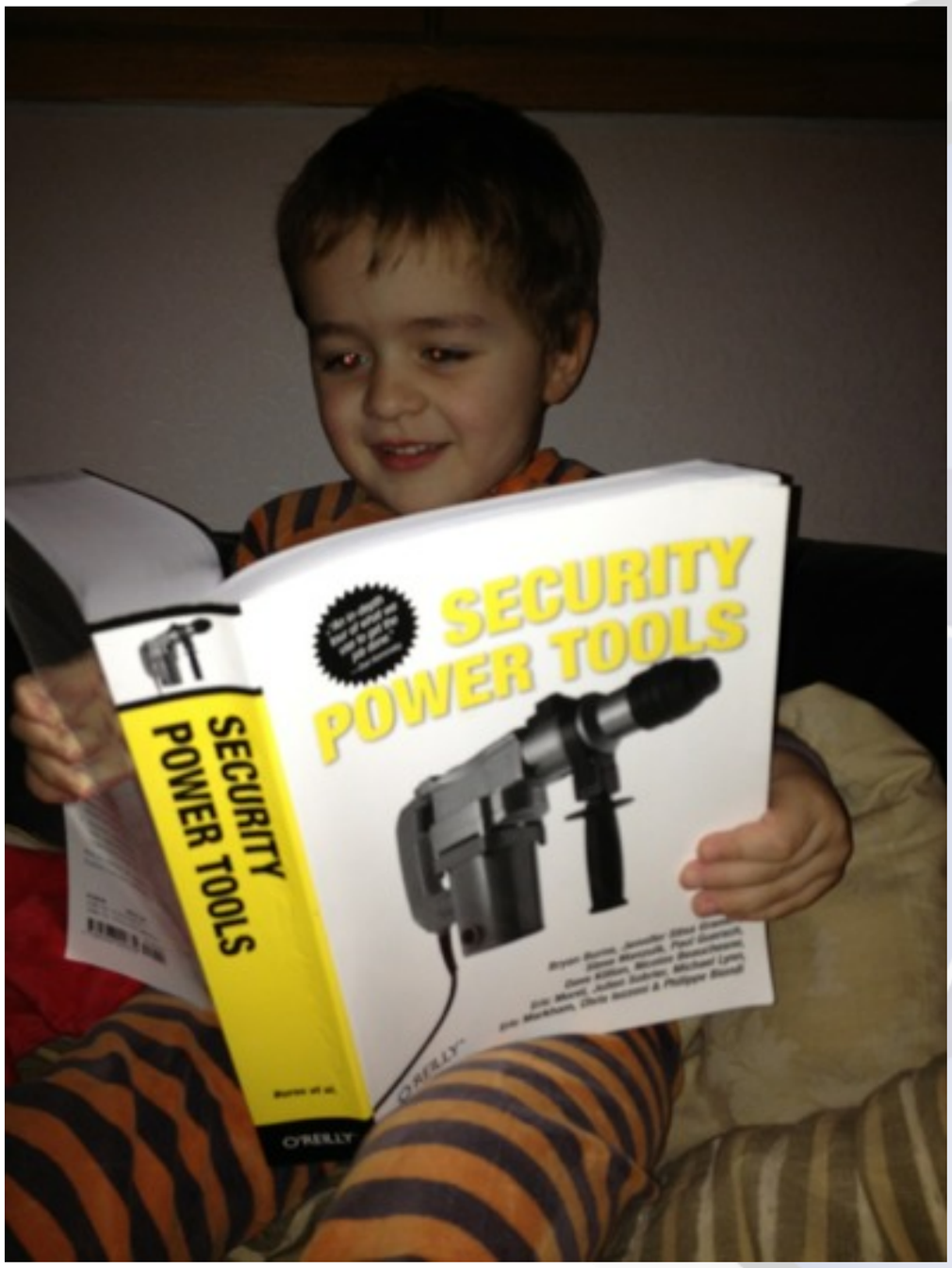
Be careful of default passwds...

Remove, and not just desactivate, unused functions/modules

Use sandbox system when available :

* **Be careful of Java Signed code who execute with more privileges !**

Now you can protect against him



License


Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions 3.0 France