David Rook

The Principles of Secure Development

OWASP Ireland Conference, Dublin

if (slide == introduction) System.out.println("I’m David Rook");

SECURITY

- Security Analyst, Realex Payments, Ireland

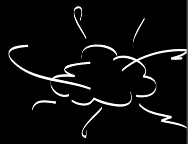
  CISSP, CISA, GCIH and many other acronyms

- Security Ninja (www.securityninja.co.uk)

- Speaker at security events (national and international)

- IIA Web Development Working Group

- Facebook hacker and published security author (insecure magazine, bloginfosec etc)



realex

The real time payment exchange

Agenda

- It is broken so lets fix it
- The current approach
- The Principles of Secure Development
- An example of a real world implementation

It is broken so lets fix it

- Cross Site Scripting, 10 years old?
- SQL Injection, 11 years old?

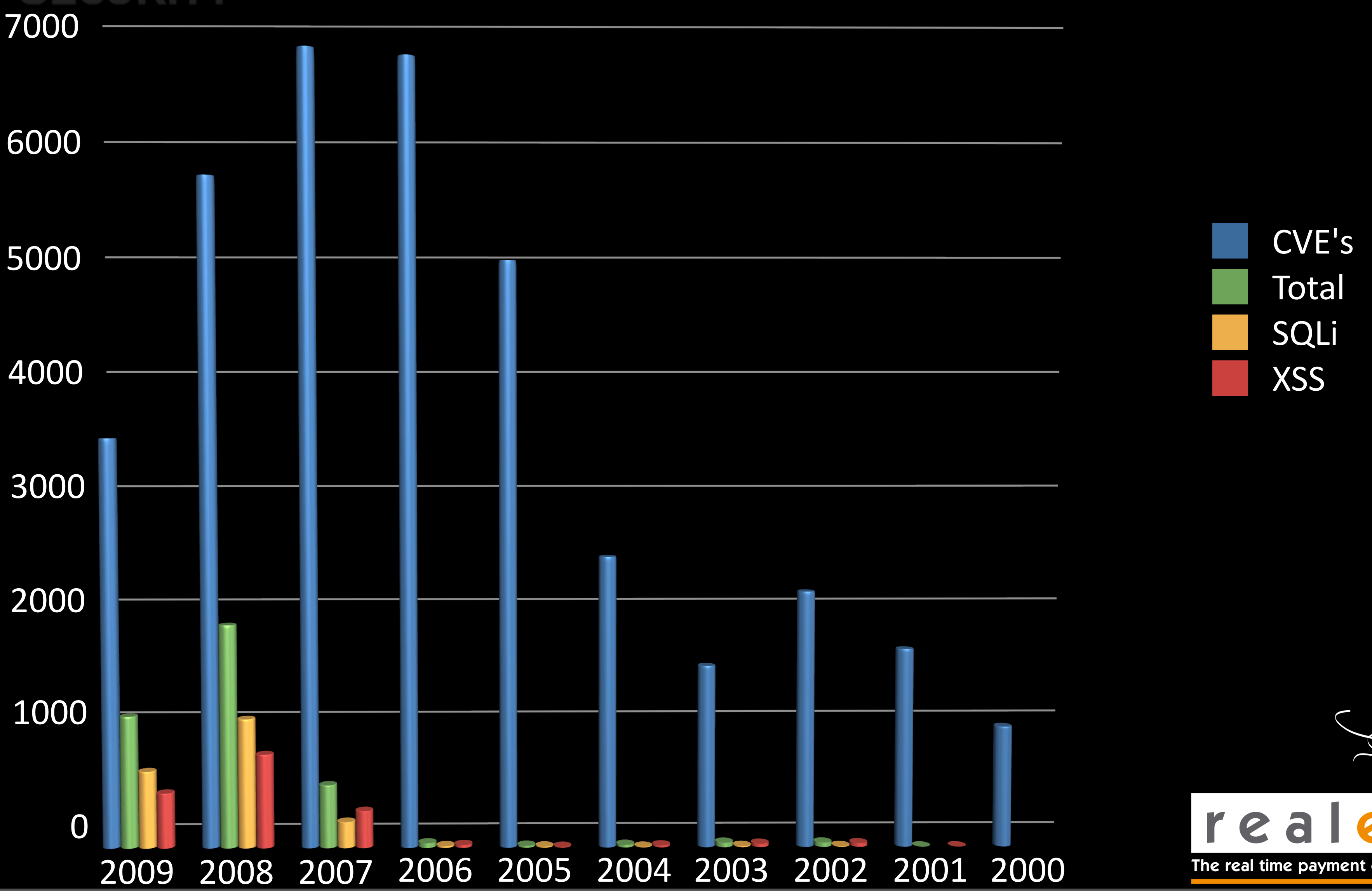
33% of all vulnerabilities in 2008 and 2009 (so far) are XSS or SQL Injection (Based on CVE numbers)

CVE statistics: http://web.nvd.nist.gov/view/vuln/statistics

It is broken so lets fix it

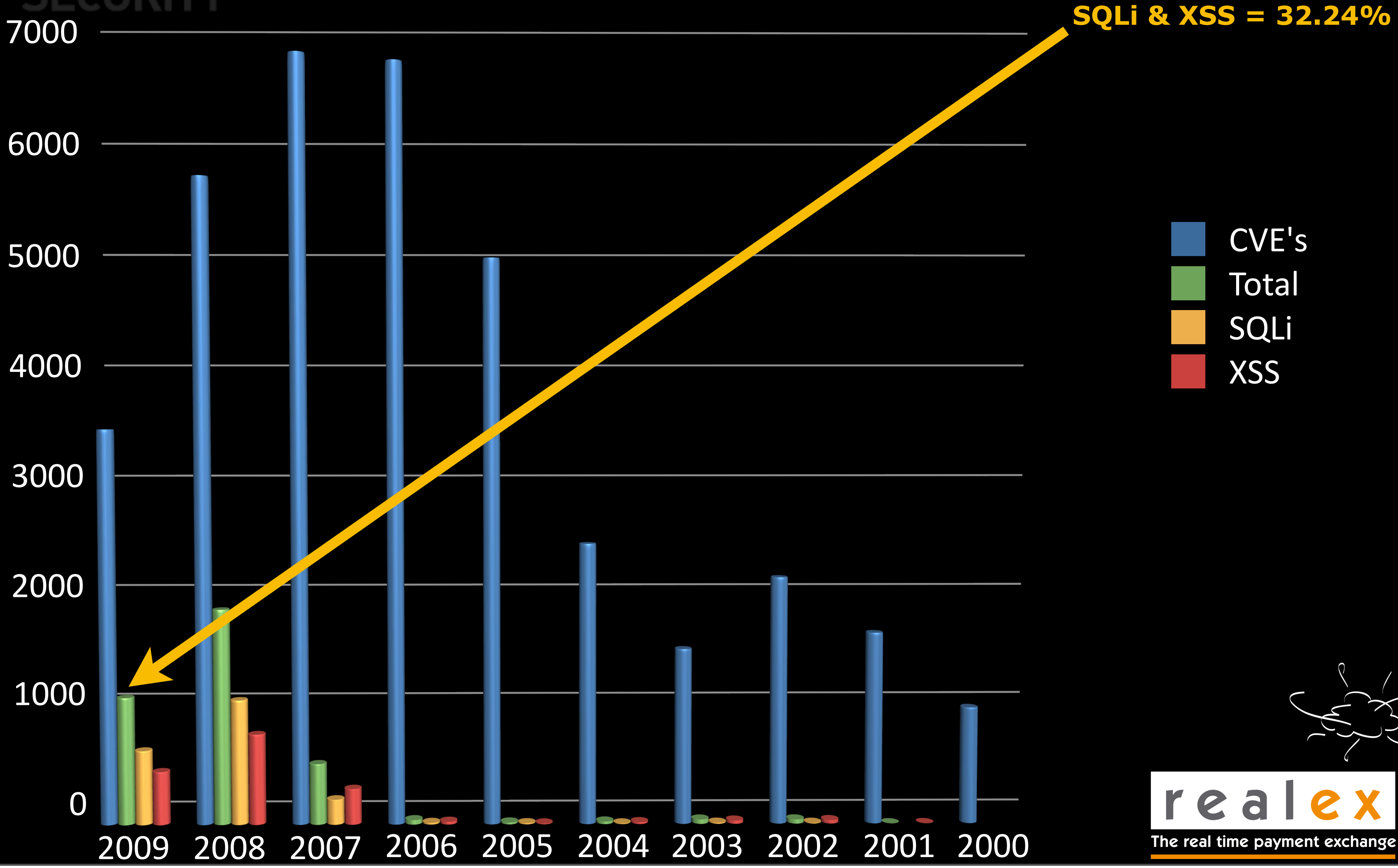
- 🟦 CVE's
- 🟩 Total
- 🟧 SQLi
- 🟥 XSS

It is broken so lets fix it



realex
The real time payment exchange

It is broken so lets fix it



realex

The real time payment exchange

Friday, 18 September 2009

Philosophical Application Security

Give a man a fish and you feed him for a day, teach him to fish and you feed him for a lifetime.

Philosophical Application Security

Give a man a fish and you feed him for a day, teach him to fish and you feed him for a lifetime.

I want to apply this to secure application development:

Teach a developer about a vulnerability and he will prevent it, teach him how to develop securely and he will prevent many vulnerabilities.

The current approach

(And why I think it fails to deliver secure applications)

- The cart before the horse
  - Security guys tell developers about specific vulnerabilities
  - We hope they figure out how to prevent them
  - Inevitably security flaws end up in live code
  - Security guys complain when data gets stolen

The current approach

(And why I think it fails to deliver secure applications)

- What if we taught drivers in the same way?
  - Instructor tells driver about the different ways to crash
  - We hope the driver figures out how not to crash
  - Inevitably the driver will crash
  - People complain when they get crashed into

The current approach

(And why I think it fails to deliver secure applications)

- Many lists of vulnerabilities
  - OWASP Top 10
  - White Hat Sec Top 10
  - SANS Top 25
  - Others??
- != Secure development guidance

SECURITY

The current approach

(And why I think it fails to deliver secure applications)

- Many lists of vulnerabilities
  - OWASP Top 10
  - White Hat Sec Top 10
  - SANS Top 25
  - Others??
- != Secure development guidance
- 45 vulnerabilities, 42 unique names
- 8 secure coding principles to prevent them

realex
The real time payment exchange

What we need to do

- Put the application security horse before the cart
  - Security guys tell developers how to write secure code
  - Developer doesn’t need to guess anymore
  - Common vulnerabilities prevented in applications
  - Realistic or just a caffeine fueled dream?

Lets make secure development easier

- Keep It Short and Simple (KISS)
  - The principles must be clearly defined
  - Language/Platform/Framework independent
  - Should cover more than just the common vulnerabilities
  - More secure software and greater ROI on security training?

The Principles of Secure Development

Input Validation

Output Validation

Error Handling

Authentication and Authorisation

Session Management

Secure Communications

Secure Storage

Secure Resource Access

The Principles of Secure Development

- Input Validation
  - Identify and define the data your application must accept

realex
The real time payment exchange

The Principles of Secure Development

- Input Validation
  - Identify and define the data your application must accept
  - Create regex's to validate each data type (content and size)
  - For example, a credit card number data type: \d{12,16}$

The Principles of Secure Development

SECURITY

- Input Validation
  - Identify and define the data your application must accept
  - Create regex’s to validate each data type (content and size)
  - For example, a credit card number data type: \d{12,16}$
  - Use whitelisting for validation where possible

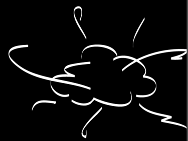
realex

The real time payment exchange

The Principles of Secure Development

- Input Validation
  - Identify and define the data your application must accept
  - Create regex’s to validate each data type (content and size)
  - For example, a credit card number data type: \d{12,16}$
  - Use whitelisting for validation where possible
  - Blacklisting approach harder and potentially less secure
  - Blacklist example, replacing single quotes:

    s.replaceAll(Pattern.quote(" ' "),
    Matcher.quoteReplacement(" " "))



The Principles of Secure Development

- Output Validation

  - Identify and define the data your application must output

The Principles of Secure Development

- Output Validation
  - Identify and define the data your application must output
  - Understand where (i.e. in a URL) your data should end up
  - Choose the correct output encoding for the data's destination

The real time payment exchange

The Principles of Secure Development

- Output Validation
  - Identify and define the data your application must output
  - Understand where (i.e. in a URL) your data should end up
  - Choose the correct output encoding for the data's destination
  - Proper encoding means this attack:

www.examplesite.com/home.html?day=<script>alert(document.cookie)</script>

Becomes:

day=%3Cscript%3Ealert%28document.cookie%29%3C/script%3E

The Principles of Secure Development

- Error Handling
  - Even the best apps will crash at some point, be prepared!

realex
The real time payment exchange

The Principles of Secure Development

- Error Handling
  - Even the best apps will crash at some point, be prepared!
  - Crashes/errors can help an attacker if you don’t handle them

The Principles of Secure Development

SECURITY

- Error Handling
  - Even the best apps will crash at some point, be prepared!
  - Crashes/errors can help an attacker if you don’t handle them
  - Handle error conditions securely, sanitise the message sent

realex
The real time payment exchange

SECURITY

The Principles of Secure Development

- Error Handling
  - Even the best apps will crash at some point, be prepared!
  - Crashes/errors can help an attacker if you don’t handle them
  - Handle error conditions securely, sanitise the message sent
  - No error handling = information leakage

```
Microsoft OLE DB Provider for ODBC
Drivers(0x80040E14)
 [Microsoft][ODBC SQL Server Driver]
 [SQL Server]Invalid column name

/examplesite/login.asp, line 10
```

realex
The real time payment exchange

The Principles of Secure Development

- Authentication and Authorisation
  - Even simple apps often have a need to authenticate users

The Principles of Secure Development

- Authentication and Authorisation
  - Even simple apps often have a need to authenticate users
  - Often at least two levels of authorisation

The Principles of Secure Development

- Authentication and Authorisation
  - Even simple apps often have a need to authenticate users
  - Often at least two levels of authorisation
  - Need to prevent horizontal and vertical privilege escalation

The Principles of Secure Development

- Authentication and Authorisation
  - Even simple apps often have a need to authenticate users
  - Often at least two levels of authorisation
  - Need to prevent horizontal and vertical privilege escalation
  - Implement strong passwords and management systems

The real time payment exchange

The Principles of Secure Development

- Authentication and Authorisation
  - Even simple apps often have a need to authenticate users
  - Often at least two levels of authorisation
  - Need to prevent horizontal and vertical privilege escalation
  - Implement strong passwords and management systems
  - Ensure A+A is secure, not a false sense of security (CAPTCHA?)

realex

The real time payment exchange

The Principles of Secure Development

- Authentication and Authorisation
    - Even simple apps often have a need to authenticate users
    - Often at least two levels of authorisation
    - Need to prevent horizontal and vertical privilege escalation
    - Implement strong passwords and management systems
    - Ensure A+A is secure, not a false sense of security (CAPTCHA?)
    - Don’t rely on fields that are easily spoofed (referrer field)

The Principles of Secure Development

- Session Management
  - Used to manage authenticated users, no need to re-auth

The Principles of Secure Development

- Session Management
  - Used to manage authenticated users, no need to re-auth
  - You need to ensure that your sessionID’s have sufficient entropy

The Principles of Secure Development

- Session Management
  - Used to manage authenticated users, no need to re-auth
  - You need to ensure that your sessionID’s have sufficient entropy
  - SessionID’s must not be predictable or reusable

realex
The real time payment exchange

The Principles of Secure Development

- Session Management
  - Used to manage authenticated users, no need to re-auth
  - You need to ensure that your sessionID’s have sufficient entropy
  - SessionID’s must not be predictable or reusable
  - Never build your own session management, it will fail

The Principles of Secure Development

- Session Management
  - Used to manage authenticated users, no need to re-auth
  - You need to ensure that your sessionID's have sufficient entropy
  - SessionID's must not be predictable or reusable
  - Never build your own session management, it will fail
  - Protect sessionID's when in transit (i.e. SSL!)

The Principles of Secure Development

- Session Management
  - Used to manage authenticated users, no need to re-auth
  - You need to ensure that your sessionID’s have sufficient entropy
  - SessionID’s must not be predictable or reusable
  - Never build your own session management, it will fail
  - Protect sessionID’s when in transit (i.e. SSL!)
  - Issue a new value for sensitive actions (i.e. funds transfer)

The Principles of Secure Development

- Secure Communications
    - Protect data (i.e. CC no, passwords, sessionID’s) in transit

realex
The real time payment exchange

The Principles of Secure Development

- Secure Communications
  - Protect data (i.e. CC no, passwords, sessionID’s) in transit
  - As with all crypto, don’t create your own

realex

The real time payment exchange

The Principles of Secure Development

- Secure Communications
  - Protect data (i.e. CC no, passwords, sessionID's) in transit
  - As with all crypto, don't create your own
  - Don't use broken protection mechanisms (i.e. SSLv2)

realex

The real time payment exchange

The Principles of Secure Development

- Secure Communications
  - Protect data (i.e. CC no, passwords, sessionID’s) in transit
  - As with all crypto, don’t create your own
  - Don’t use broken protection mechanisms (i.e. SSLv2)
  - Don’t just use SSL/TLS for logon pages, protect the session!

The Principles of Secure Development

- Secure Communications
  - Protect data (i.e. CC no, passwords, sessionID’s) in transit
  - As with all crypto, don’t create your own
  - Don’t use broken protection mechanisms (i.e. SSLv2)
  - Don’t just use SSL/TLS for logon pages, protect the session!
  - Try to avoid mixing secure and insecure traffic on a page

The Principles of Secure Development

- Secure Storage
  - Protect data (i.e. CC no, passwords, sessionID’s) when stored

realex
The real time payment exchange

The Principles of Secure Development

- Secure Storage
  - Protect data (i.e. CC no, passwords, sessionID’s) when stored
  - As with all crypto, don’t create your own

The Principles of Secure Development

SECURITY

- Secure Storage
  - Protect data (i.e. CC no, passwords, sessionID’s) when stored
  - As with all crypto, don’t create your own
  - Don’t use broken protection mechanisms (i.e. DES)

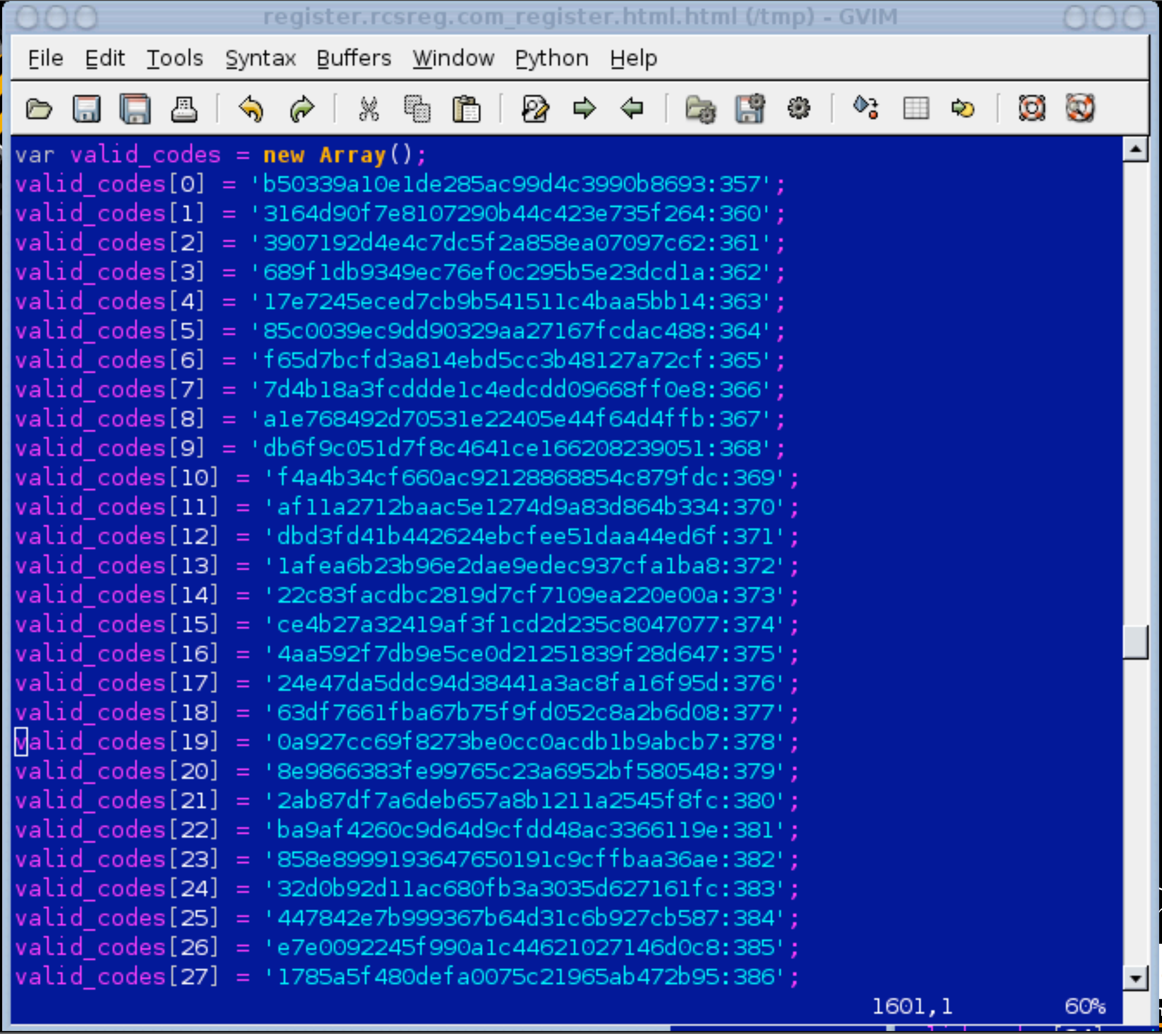
realex
The real time payment exchange

The Principles of Secure Development

- Secure Storage
  - Protect data (i.e. CC no, passwords, sessionID’s) when stored
  - As with all crypto, don’t create your own
  - Don’t use broken protection mechanisms (i.e. DES)
  - Don’t store data in places where you can’t confidently secure it

The Principles of Secure Development

- Secure Storage
  - Protect data (i.e. CC no, passwords, sessionID’s) when stored
  - As with all crypto, don’t create your own
  - Don’t use broken protection mechanisms (i.e. DES)
  - Don’t store data in places where you can’t confidently secure it
  - Strong protection mechanisms, how strong should it be?


register.rcsreg.com_register.html.html (/tmp) - GVIM
File Edit Tools Syntax Buffers Window Python Help

```
var valid_codes = new Array();
valid_codes[0] = 'b50339a10e1de285ac99d4c3990b8693:357';
valid_codes[1] = '3164d90f7e8107290b44c423e735f264:360';
valid_codes[2] = '3907192d4e4c7dc5f2a858ea07097c62:361';
valid_codes[3] = '689f1db9349ec76ef0c295b5e23dcd1a:362';
valid_codes[4] = '17e7245eced7cb9b541511c4baa5bb14:363';
valid_codes[5] = '85c0039ec9dd90329aa27167fcdac488:364';
valid_codes[6] = 'f65d7bcfd3a814ebd5cc3b48127a72cf:365';
valid_codes[7] = '7d4b18a3fcddde1c4edcdd09668ff0e8:366';
valid_codes[8] = 'a1e768492d70531e22405e44f64d4ffb:367';
valid_codes[9] = 'db6f9c051d7f8c4641ce166208239051:368';
valid_codes[10] = 'f4a4b34cf660ac92128868854c879fdc:369';
valid_codes[11] = 'af11a2712baac5e1274d9a83d864b334:370';
valid_codes[12] = 'dbd3fd41b442624ebcfee51daa44ed6f:371';
valid_codes[13] = '1afea6b23b96e2dae9edec937cfa1ba8:372';
valid_codes[14] = '22c83facdbc2819d7cf7109ea220e00a:373';
valid_codes[15] = 'ce4b27a32419af3f1cd2d235c8047077:374';
valid_codes[16] = '4aa592f7db9e5ce0d21251839f28d647:375';
valid_codes[17] = '24e47da5ddc94d38441a3ac8fa16f95d:376';
valid_codes[18] = '63df7661fba67b75f9fd052c8a2b6d08:377';
valid_codes[19] = '0a927cc69f8273be0cc0acdb1b9abcb7:378';
valid_codes[20] = '8e9866383fe99765c23a6952bf580548:379';
valid_codes[21] = '2ab87df7a6deb657a8b1211a2545f8fc:380';
valid_codes[22] = 'ba9af4260c9d64d9cfdd48ac3366119e:381';
valid_codes[23] = '858e8999193647650191c9cffbaa36ae:382';
valid_codes[24] = '32d0b92d11ac680fb3a3035d627161fc:383';
valid_codes[25] = '447842e7b999367b64d31c6b927cb587:384';
valid_codes[26] = 'e7e0092245f990a1c44621027146d0c8:385';
valid_codes[27] = '1785a5f480defa0075c21965ab472b95:386';
                                                        1601,1        60%
```

The Principles of Secure Development

- Secure Resource Access
  - Obscurity != security, don’t try to hide sensitive resources

The real time payment exchange

SECURITY

The Principles of Secure Development

- Secure Resource Access
  - Obscurity != security, don’t try to hide sensitive resources
  - Understand the users flow through an app, cover weak spots

realex

The real time payment exchange

The Principles of Secure Development

- Secure Resource Access
  - Obscurity != security, don’t try to hide sensitive resources
  - Understand the users flow through an app, cover weak spots
  - T-Mobile didn’t do the above, Paris Hiltons account hacked

The real time payment exchange

Lets redefine what secure development means

- Follow a small, repeatable set of principles
- Try not to focus on specific vulnerabilities
- Develop securely, not to prevent "hot vuln of the day"
- Build security into the code, don’t try to bolt it on at the end

Evolution, not revolution

- Don’t make things more difficult than they need to be
  - This isn’t a new wheel, its just a smoother, easier to use wheel
  - Don’t treat security as something separate, integrate it
  - By integrating security fully a security bug is just another bug
  - Secure development doesn’t have to be hard, KISS it!

SECURITY

The new approach is working

- Private banking development company, Switzerland
  - Application Security lead saw the secure development principles

realex

The real time payment exchange

SECURITY

The new approach is working

- Private banking development company, Switzerland
  - Application Security lead saw the secure development principles
  - Re-designed secure development training for his company

realex

The real time payment exchange

SECURITY

The new approach is working

- Private banking development company, Switzerland
  - Application Security lead saw the secure development principles
  - Re-designed secure development training for his company
  - Security training costs down, quicker "spin up" of developers

realex

The real time payment exchange

The new approach is working

- Private banking development company, Switzerland
  - Application Security lead saw the secure development principles
  - Re-designed secure development training for his company
  - Security training costs down, quicker "spin up" of developers
  - Security within their SDLC now based on the principles

realex
The real time payment exchange

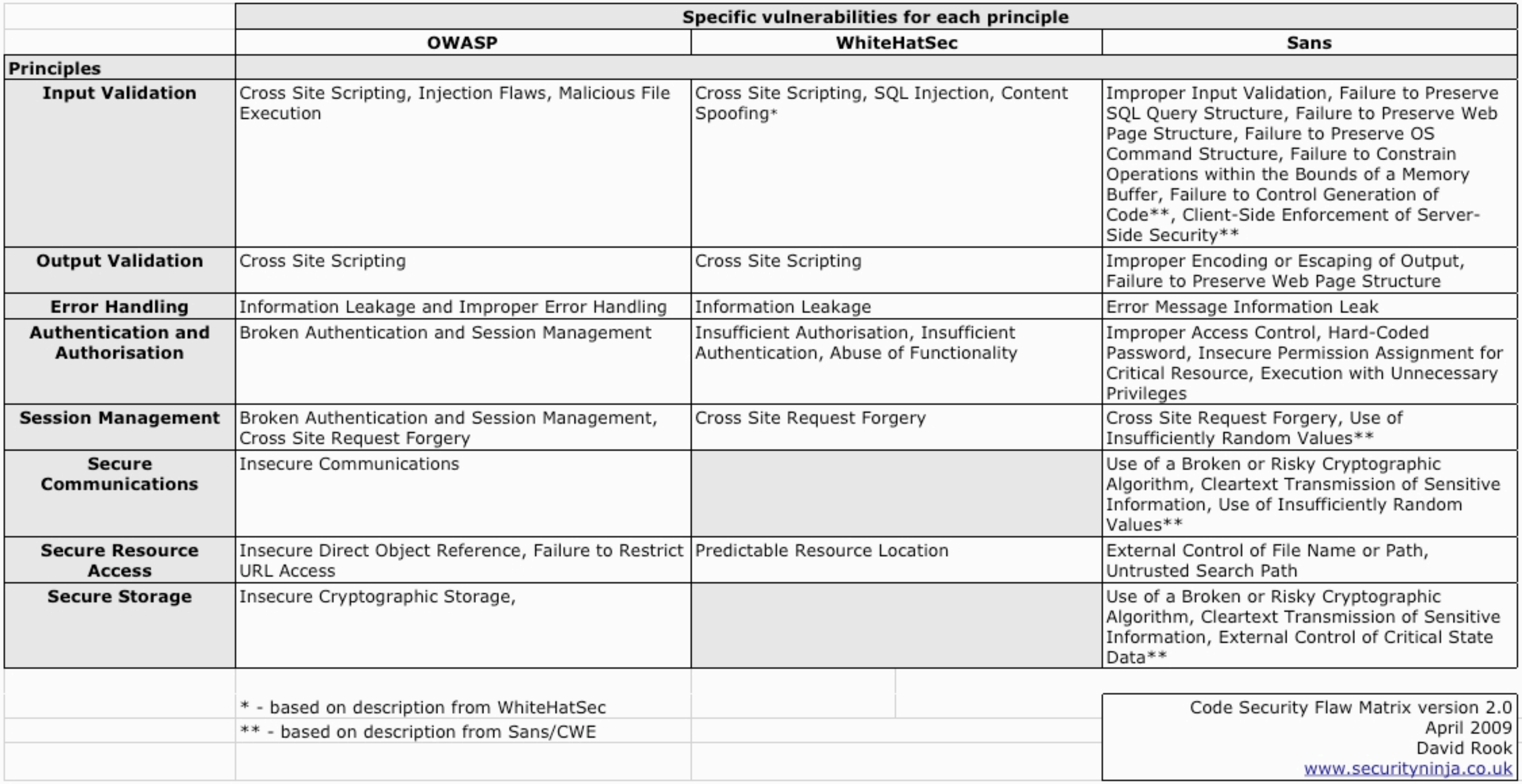
The new approach is working

- Private banking development company, Switzerland
  - Application Security lead saw the secure development principles
  - Re-designed secure development training for his company
  - Security training costs down, quicker "spin up" of developers
  - Security within their SDLC now based on the principles
  - In his own words:

You released the "secure development principles" at a time I had issues with my dev teams in how to teach them secure development. **Your approach convinced me to look in another direction, not trying to teach every vulnerability but finding the basic principles that help prevent their existence.** At that time, this was genius for me: most of my training since has been inspired by your secure development principles.

The new approach is working

They modified the principles matrix to match their own terminology



	Specific vulnerabilities for each principle		
	OWASP	WhiteHatSec	Sans
Principles			
Input Validation	Cross Site Scripting, Injection Flaws, Malicious File Execution	Cross Site Scripting, SQL Injection, Content Spoofing*	Improper Input Validation, Failure to Preserve SQL Query Structure, Failure to Preserve Web Page Structure, Failure to Preserve OS Command Structure, Failure to Constrain Operations within the Bounds of a Memory Buffer, Failure to Control Generation of Code**, Client-Side Enforcement of Server-Side Security**
Output Validation	Cross Site Scripting	Cross Site Scripting	Improper Encoding or Escaping of Output, Failure to Preserve Web Page Structure
Error Handling	Information Leakage and Improper Error Handling	Information Leakage	Error Message Information Leak
Authentication and Authorisation	Broken Authentication and Session Management	Insufficient Authorisation, Insufficient Authentication, Abuse of Functionality	Improper Access Control, Hard-Coded Password, Insecure Permission Assignment for Critical Resource, Execution with Unnecessary Privileges
Session Management	Broken Authentication and Session Management, Cross Site Request Forgery	Cross Site Request Forgery	Cross Site Request Forgery, Use of Insufficiently Random Values**
Secure Communications	Insecure Communications		Use of a Broken or Risky Cryptographic Algorithm, Cleartext Transmission of Sensitive Information, Use of Insufficiently Random Values**
Secure Resource Access	Insecure Direct Object Reference, Failure to Restrict URL Access	Predictable Resource Location	External Control of File Name or Path, Untrusted Search Path
Secure Storage	Insecure Cryptographic Storage,		Use of a Broken or Risky Cryptographic Algorithm, Cleartext Transmission of Sensitive Information, External Control of Critical State Data**

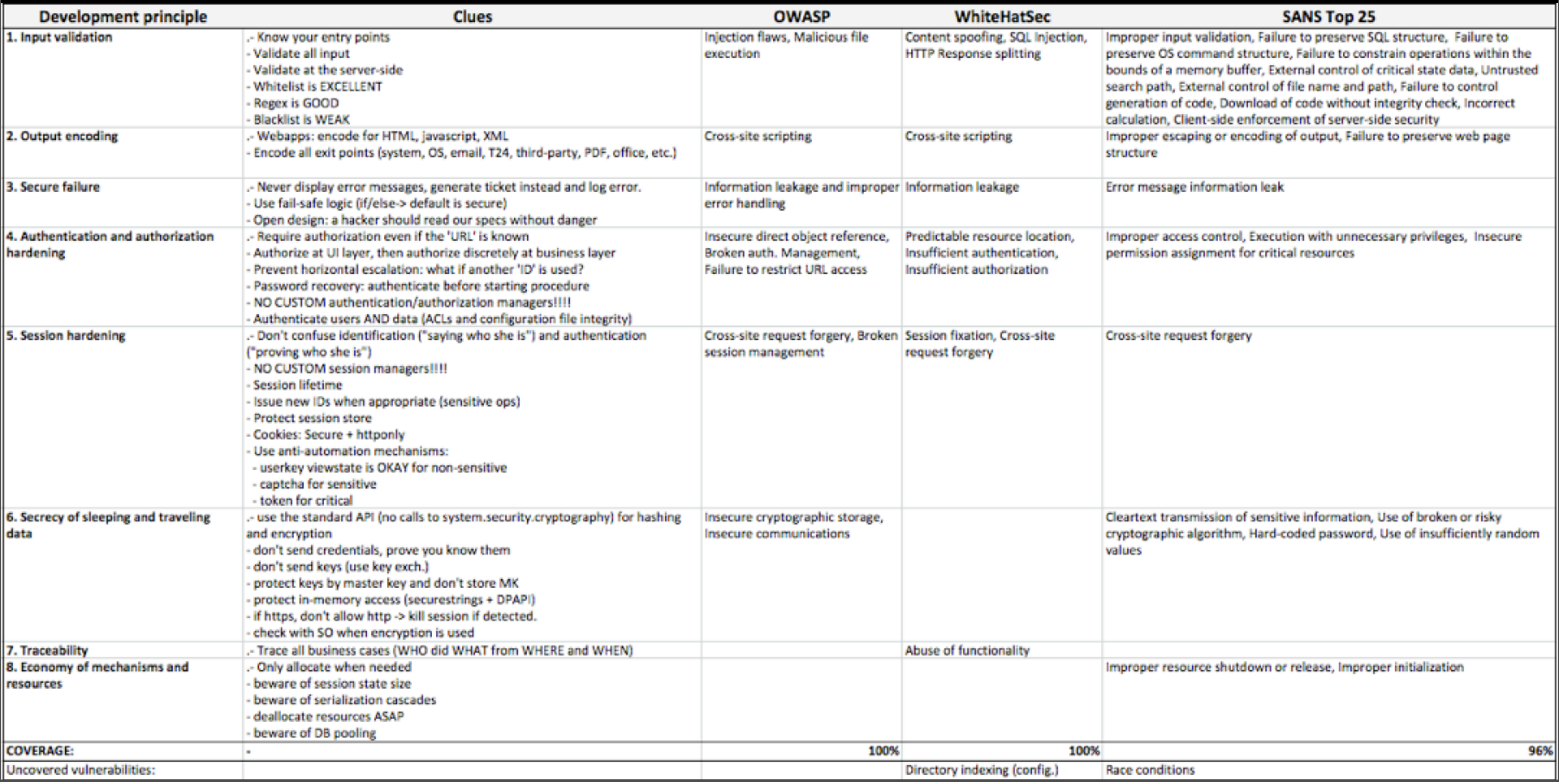
* - based on description from WhiteHatSec

** - based on description from Sans/CWE

Code Security Flaw Matrix version 2.0
April 2009
David Rook
www.securityninja.co.uk

The new approach is working

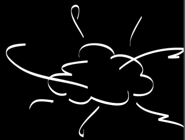
They modified the principles matrix to match their own terminology



Development principle	Clues	OWASP	WhiteHatSec	SANS Top 25
1. Input validation	.- Know your entry points - Validate all input - Validate at the server-side - Whitelist is EXCELLENT - Regex is GOOD - Blacklist is WEAK	Injection flaws, Malicious file execution	Content spoofing, SQL Injection, HTTP Response splitting	Improper input validation, Failure to preserve SQL structure, Failure to preserve OS command structure, Failure to constrain operations within the bounds of a memory buffer, External control of critical state data, Untrusted search path, External control of file name and path, Failure to control generation of code, Download of code without integrity check, Incorrect calculation, Client-side enforcement of server-side security
2. Output encoding	.- Webapps: encode for HTML, javascript, XML - Encode all exit points (system, OS, email, T24, third-party, PDF, office, etc.)	Cross-site scripting	Cross-site scripting	Improper escaping or encoding of output, Failure to preserve web page structure
3. Secure failure	.- Never display error messages, generate ticket instead and log error. - Use fail-safe logic (if/else-> default is secure) - Open design: a hacker should read our specs without danger	Information leakage and improper error handling	Information leakage	Error message information leak
4. Authentication and authorization hardening	.- Require authorization even if the 'URL' is known - Authorize at UI layer, then authorize discretely at business layer - Prevent horizontal escalation: what if another 'ID' is used? - Password recovery: authenticate before starting procedure - NO CUSTOM authentication/authorization managers!!!! - Authenticate users AND data (ACLs and configuration file integrity)	Insecure direct object reference, Broken auth. Management, Failure to restrict URL access	Predictable resource location, Insufficient authentication, Insufficient authorization	Improper access control, Execution with unnecessary privileges, Insecure permission assignment for critical resources
5. Session hardening	.- Don't confuse identification ("saying who she is") and authentication ("proving who she is") - NO CUSTOM session managers!!!! - Session lifetime - Issue new IDs when appropriate (sensitive ops) - Protect session store - Cookies: Secure + httponly - Use anti-automation mechanisms: - userkey viewstate is OKAY for non-sensitive - captcha for sensitive - token for critical	Cross-site request forgery, Broken session management	Session fixation, Cross-site request forgery	Cross-site request forgery
6. Secrecy of sleeping and traveling data	.- use the standard API (no calls to system.security.cryptography) for hashing and encryption - don't send credentials, prove you know them - don't send keys (use key exch.) - protect keys by master key and don't store MK - protect in-memory access (securestrings + DPAPI) - if https, don't allow http -> kill session if detected. - check with SO when encryption is used	Insecure cryptographic storage, Insecure communications		Cleartext transmission of sensitive information, Use of broken or risky cryptographic algorithm, Hard-coded password, Use of insufficiently random values
7. Traceability	.- Trace all business cases (WHO did WHAT from WHERE and WHEN)		Abuse of functionality	
8. Economy of mechanisms and resources	.- Only allocate when needed - beware of session state size - beware of serialization cascades - deallocate resources ASAP - beware of DB pooling			Improper resource shutdown or release, Improper initialization
COVERAGE:	-	100%	100%	96%
Uncovered vulnerabilities:			Directory indexing (config.)	Race conditions

Security Ninja new site launch!

- Security Ninja, brought you by Realex Payments
  - Free application security and compliance resource site



Security Ninja new site launch!

- Security Ninja, brought you by Realex Payments
  - Free application security and compliance resource site
  - Blog and site managed and updated by myself

The real time payment exchange

Security Ninja new site launch!

- Security Ninja, brought you by Realex Payments
  - Free application security and compliance resource site
  - Blog and site managed and updated by myself
  - Security presentations, whitepapers, videos and audio online

Security Ninja new site launch!

- Security Ninja, brought you by Realex Payments
  - Free application security and compliance resource site
  - Blog and site managed and updated by myself
  - Security presentations, whitepapers, videos and audio online
  - Secure Development Principles whitepaper available here today

www.securityninja.co.uk

Twitter: @securityninja

QUESTIONS?

www.securityninja.co.uk

Twitter: @securityninja