Developing defensible web applications on IBM i

Peter Helgren
Value Added Software, Inc.
San Antonio, TX
GIAC Secure Software Programmer-Java

pete@valadd.com

www.petesworkshop.com

Agenda

- Internet security – why IBM i users should care.
- Cover the top 10 web exploits
- Best practices when developing web applications

www.valadd.com www.petesworkshop,com (c) copyright 2014

Security begins with you

- You need to use best practices on your Windows systems/servers
- Top 4
  - Patching third party applications
  - Patching operating systems
  - Minimizing administrative privileges
  - Application whitelisting

Why should IBM i users and developers care?

- Web applications are increasing common on IBM i.
- RPG programmers may be unfamiliar with web programming and exploits.
- Fewer tools/libraries available for mitigating web exploits in RPG (alas! No ESAPI for RPG)
- More open source use means possible “import” of security vulnerabilities. Can you say “Heartbleed”?

ESAPI

Enterprise Security API's

ESAPI (The OWASP Enterprise Security API) is a free, open source, web application security control library that makes it easier for programmers to write lower-risk applications. The ESAPI libraries are designed to make it easier for programmers to retrofit security into existing applications.

Java and PHP on IBM I can take advantage of these API's.

OWASP Top 10
(Open Web Application Security Project)

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross-Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross-Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	A8 – Failure to Restrict URL Access
A9 – Insecure Communications	A9 – Insufficient Transport Layer Protection
<not in T10 2007>	A10 – Unvalidated Redirects and Forwards (NEW)
A3 – Malicious File Execution	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>

2013 OWASP Top 10

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

www.valadd.com www.petesworkshop,com (c) copyright 2014

SANS Top 10
(ALL software)

- [1] Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
- [2] Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
- [3] Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
- [4] Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

www.valadd.com www.petesworkshop,com (c) copyright 2014

SANS Top 10
(ALL software) cont.

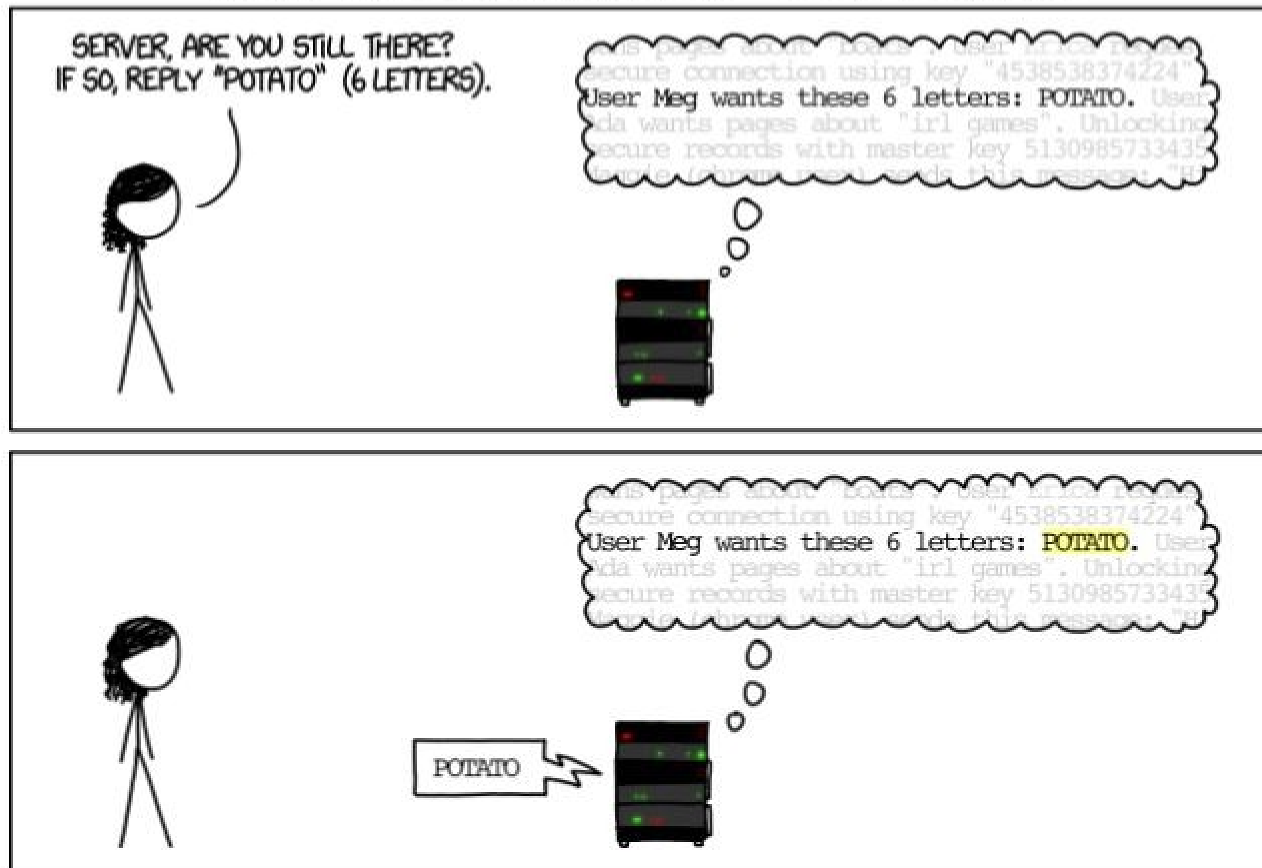
- [5] Missing Authentication for Critical Function
- [6] Missing Authorization
- [7] Use of Hard-coded Credentials
- [8] Missing Encryption of Sensitive Data
- [9] Unrestricted Upload of File with Dangerous Type
- [10] Reliance on Untrusted Inputs in a Security Decision
- [12] Cross-Site Request Forgery (CSRF)

for Business

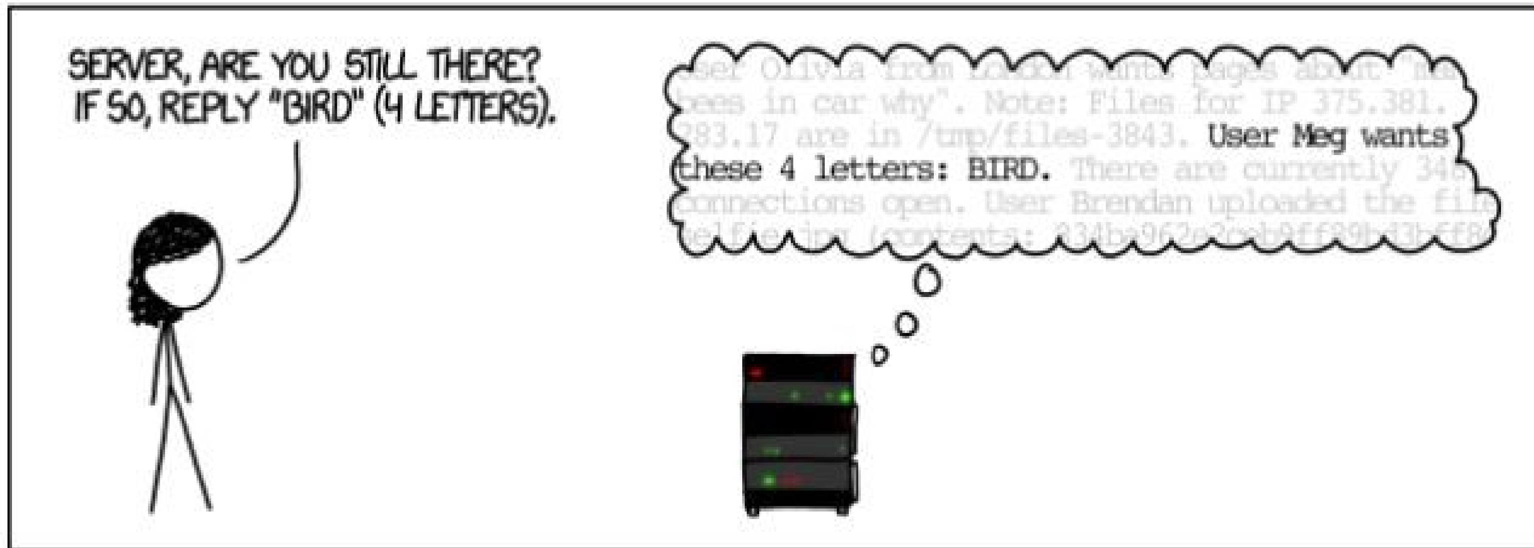
Heartbleed

- Classic buffer overflow (buffer abuse) #3 on the SANS list

HOW THE HEARTBLEED BUG WORKS:

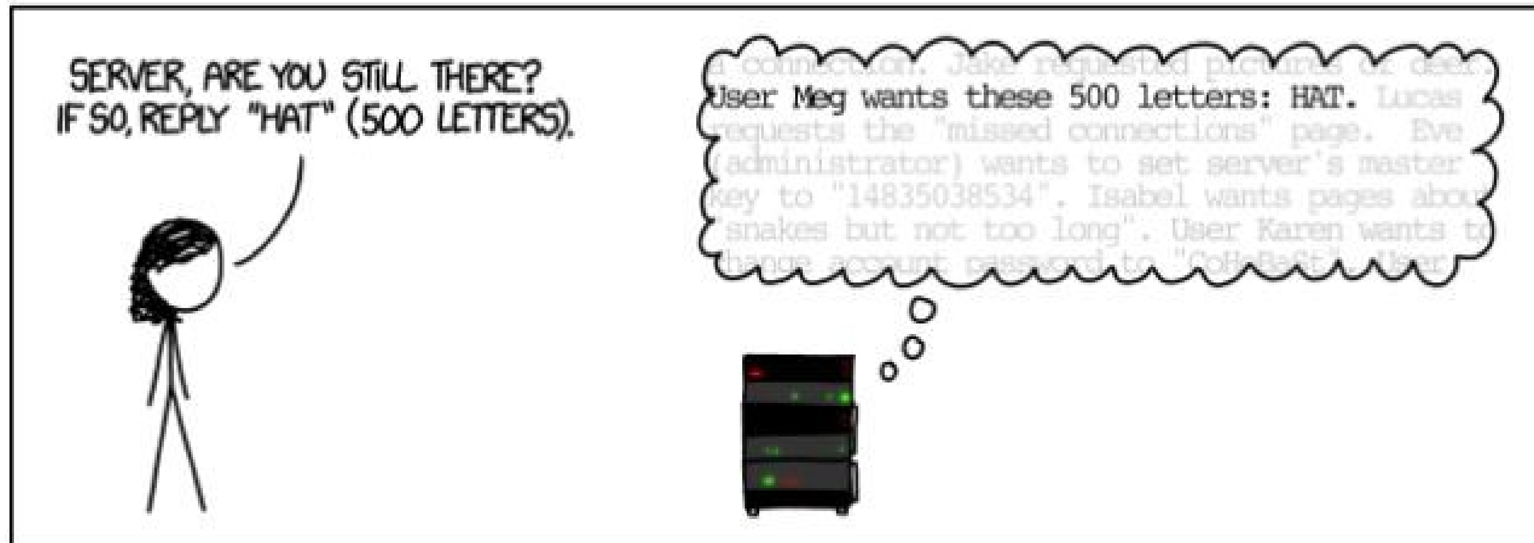


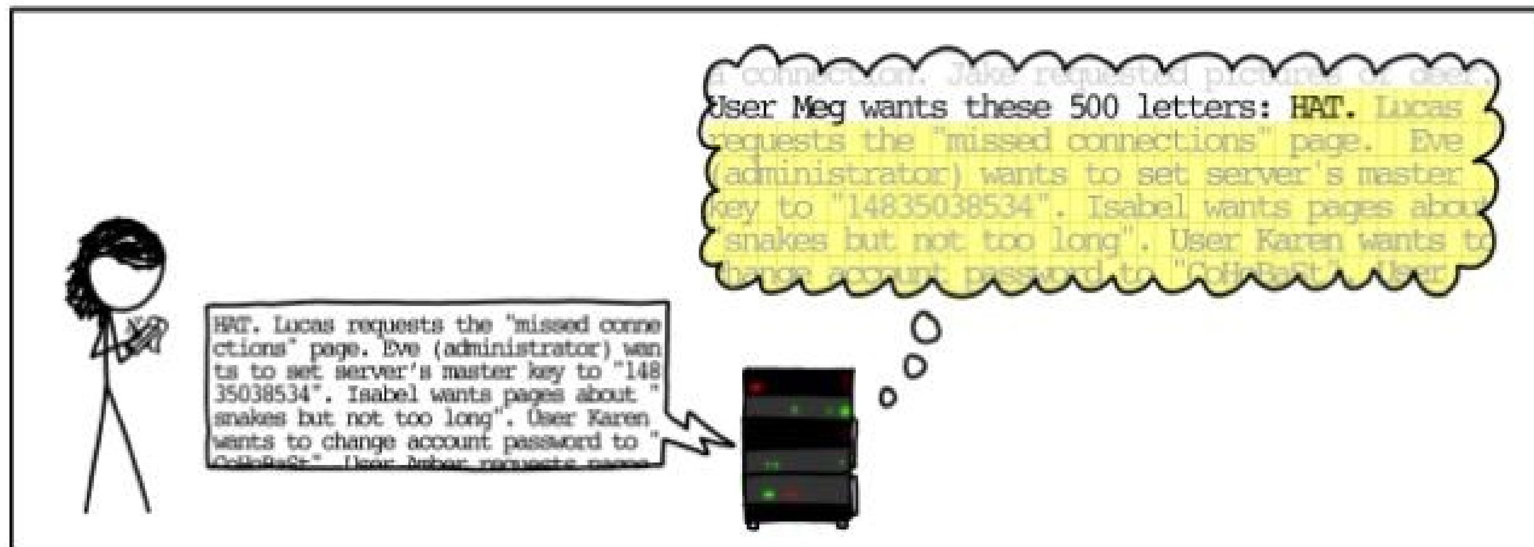
Heartbleed



www.valadd.com www.petesworkshop,com (c) copyright 2014

Heartbleed





SQL Injection

- Relies on unsanitized input to a dynamic SQL statement.
- Classic login SQL:

  “Select * from users where user ='” + user + “' and pswd =' ” + pass +”'”;
- If a record is returned then the program logic returns that the login was successful and the user is authenticated.

SQL Injection

- Submitted login form is parsed by the program and parameters passed to the SQL statement.
- What if :
  - USER = pete' OR '1'='1' --
  - PASS = whocares
- The resulting SQL:

  “Select * from users where user ='pete' OR '1'='1' -- ' and pswd ='whocares'”;
- The “- -” is the comment character for many SQL implementations so remaining characters are ignored

SQL Injection

- The hacker gains access and, because the first users in a table tend to be administrative users, the hacker could gain access to admin rights as well.
- Plenty of other ways to exploit dynamic SQL
- A nice, real world understandable step by step can be found here:
- http://www.unixwiz.net/techtips/sql-injection.html

Injection example

select firstname,lastname from user where userid = ' +userid + 'and password = ' + password + "'";

userid = pete password = mypassword
select * from user where userid = 'pete' and password = 'mypassword'

user = pete' OR '1'='1 -- password = dontcare

select * from user where userid = 'pete' OR '1'='1' -- and password = dontcare

SQL Injection - fixes

Process SQL queries using prepared statements, parameterized queries, or stored procedures. These features should accept parameters or variables and support strong typing. Do not dynamically construct and execute query strings within these features using "exec" or similar functionality, since you may re-introduce the possibility of SQL injection.

Cross site scripting - XSS

- Two types:
  - Stored (persistent)
  - Reflected
- Cause:
  - Unsanitized content is stored by the application which can be a malicious link, or worse, javascript.

Cross Site Scripting - XSS

- Example:
  - <script>window.location=" http://www.petesworkshop.com" </script>
  - <script> alert('Gotcha!")</script>
- A recent attack described here (YouTube?)

http://www.scmagazine.com/xss-vulnerability-in-popular-video-site-enables-unique-ddos-attack/article/341453/

Cross Site Scripting (XSS)

- Stored
  - The unsanitized data is entered in a web form and stored. Blogs and “bulletin boards” can suffer from this issue. User “comment” areas on a web site are notorious for this
- Reflected
  - Unsanitized data isn't stored but is immediately executed in the browser. Many phishing schemes can use this approach

Cross Site Scripting (XSS)

- Rule: Don't trust data going in or going out!
- Fixes:
  - If using Java you can use the ESAPI libraries.
  - Whitelist characters
  - Blacklist characters e.g.<,>, [,], ?, *
  - Escape characters
    - & --> &
    - < --> <
    - > --> >
    - " --> "
    - ' --> '   ' is not recommended
    - / --> /   forward slash is included as it helps end an HTML entity

www.valadd.com www.petesworkshop,com (c) copyright 2014

Authentication Issues

- Missing Authentication for Critical Function
  - Failure to authenticate a user before allowing a function.
  - Note: Because a user's authorization is checked before using commands on IBM i we often forget to add that feature in our applications
- Missing authorization check
  - Example: Pass access level in URL after authenticated (could be changed/elevated by malicious user).

Authentication Issues

- Weak passwords.
- No account lockout mechanism.
- Weak challenge questions on password reset.
- Examples:
- Hard-coded credentials (can you say Stuxnet?)
- Rugged.com flaw ….

Rugged.com (backdoor) example

- The login credentials for the backdoor include a static username, “factory,” that was assigned by the vendor and can’t be changed by customers, and a dynamically generated password that is based on the individual MAC address, or media access control address, for any specific device.
- Hmmm...I wonder how I can find out the MAC address.....

Session Management

- URL rewriting with session ID e.g.
  - http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii
  -
- Failure to invalidate a session (above session could be used indefinitely without a timeout)
- No log out with session.invalidate

Insecure Direct Object References

(Similar to missing authorization checks)

- Allows malicious users to change parameter values without validation.
- e.g. http://example.com/app/accountInfo?acct=notmyacct
- Fixes
  - Use indirect object references e.g. use values in the browser that are random and generated by the server
  - Check authorization – always.

Cross-Site Request Forgery(CSRF)

- Just about all sites have some form of this vulnerability.
- Particularly dangerous with XSS
- Generally requires two browser sessions. One open to a secure site (to be exploited) and a malicious site.
- Phishing is a blatant form of CSRF.

Missing Encryption of Sensitive Data

- Classic example is credit card information
- Other things that should be encrypted:
  - Social Security Numbers
  - Any personally identifiable information
    - Birth dates
    - Place of birth
  - Passwords (uh, duh!)
- Remediation: Strong encryption of the data
- Secure Transport (TLS)

Unrestricted Upload of File with Dangerous Type

- Payloads cannot be judged by their extensions!
- Executable files come in many forms
- Remediation: Generate your own filenames
  - Store files outside the web root (to prevent execution)
  -

Execution with Unnecessary Privileges

- Can you say “QSECOFR”?
- Tightly control file permissions on files that execute. *public *all or *public *RWX for IFS files isn't always necessary.

Unvalidated Redirects and Forwards

- Pretty rare to have code that intentionally forwards users to another Page using “redirect” or “forward”. This is kind of a JSP thing.

- Remediation: Don't forward or redirect (let the server do it)

www.valadd.com www.petesworkshop,com (c) copyright 2014

Thanks! Questions?

- Resources
  - SANS.org
  - OWASP.org
  -