Beyond Automated Tools:

Finding Zero-Day / Custom Vulnerabilities in Open Source Web Applications

(using OpenDocMan as a case study)

by
Kenneth F. Belva, CISSP
January 24, 2013

Why Research Open Source Software?

- First, it's open source = less potential legal issues
  - Not only in review and testing but also presenting to peers – I can actually show code samples
- Second, it's open source = more of a challenge
  - Theory: More eyeballs have seen the code so there should be less errors
  - Reality: Sometimes it's not spaghetti but it can be close! Certainly inconsistent code in the case of OpenDocMan.
- Third, code exploration lead to *finds and dead ends* so there is some interesting things to talk about

Deep Dive: Open Doc Man

Why this software:

First, I found vulnerabilities

- XSS
- SQL Injection
- File Upload XSS
- Admin password reset without authentication

Second, lots of lessons learned here

- Not all issues found are exploitable
- *Poorly written code gets protected inadvertently*!
- Different Vulnerability Types in One App

Third, “Well, I needed to pick on someone.”

Where to start?

We are looking for “interesting blocks of code”

We want functionality that does stuff and is processed

- Specifically, parameters that lead to SQL statements or to DB entries (XSS, CSRF, SQL injection)
- Any type of character filtering in the application
- File uploads (upload our own pages)
- Parameters that lead to “decisions” - that is the code acts differently when it views them (logical flaws)
- Of course, session tokens and anything related to state
- Any authentication component (bypass, spoofing, etc)

We want to find the code gaps

- We are looking for logical flaws
- Unfiltered or partially/incomplete filters

In short: we need to understand how the program works!

Errata: Additional Places to Research

(basically a few items that I couldn't fit on previous slide!)

Standard software looks for variations on single and double quotes

If you can find something like 0xbf that happens at a lower level – the db level – many filtering routines at the higher level will be vulnerable

In business environments: code changes especially where there was a rush to complete the job or time pressure to make the change

Code changes to internal apps by third parties: they sometimes do not understand the standards

While beyond the scope of this presentation, look in areas where there is mixed technology: i.e., java applets embedded

A Few Quick Words on Filtering & Fuzzing...

A Tale of Two Layers*
The App and the Interpreter

First set of controls are the web app filter

Single and double quotes, left and right brackets, etc.

Second set of “controls” is the underlying app

db query interpreter (MySQL, MSSQL, Oracle)

- Properly formatted queries
- 0xbf2 interpretation is at db level to do conversion

Web server (Apache / IIS)

- Null Byte upload issue on name of file

*** This dual layer issue comes into play later in the analysis of OpenDocMan.**

And Now A Quick Word on Fuzzing

Why fuzz?

- It allows testing of a lot of different parameters and a large surface area of an application in an effective and efficient manner

Fuzzing allows for automation

Fuzzing allows for a lot of possibilities – different combinations of parameters to be submitted – that an individual tester 1) may accidentally skip over while testing or 2) much higher volume than could do manually

In short, I'm a fan!

“Beyond Fuzzing”

If the purpose of fuzzing is to determine program errors through tons of parameter submissions.....

Going “beyond fuzzing” is *a direct or indirect examining the internal logic of the application*

It consists of reading /examining specific functionality of the code to determine flaws that might exist but cannot be found through just submitting parameters

We see this often in the case of reverse engineering and debugging.... it's beyond just fuzzing and often used in combination with it

Some Web App Techniques

Find filtering routine and run independent test to determine exactly what comes out the other side

I use fuzzing here. What mistakes will the code make?

Search strings throughout the application for specific information and routines

*** How is the db query executed? Once we find this we look for instances when it's not done the “correct” way! (centralized vs decentralized)

Functionality where we send data and files to app

General Non-Automated Source Code Analysis Techniques (and not just for web apps)

Look for specific, weak functions and determine where it's used throughout application

- I coded small grep utility that will help me trace function through different files (include and exclude) : function tracing, not just parameter tracing!

Rip out and test web app filters directly!

- One benefit of Open Source Software is you can take same code and "sandbox" it in your own environment and test it thoroughly
- SQL Injection: Usually will tell you if it's just UNION Select or other injection
- XSS: Will tell you what type of encoding will bypass filters...

Look for where files are uploaded to: under path of applications? What are names of files?

Examine Obscure functionality closely!

- If it's not heavily used code then chances are it's not been heavily debugged for issues

Read actual code that builds and submits query to DB

OpenDocMan v1.2.6.2: Issues/Code to Explore

Note: For this presentation we will focus on server side issues not client side ones

Four Zero-days to Explore

From OpenDocMan we will examine the following:

- First, a quick review of the UI so we can understand the app
- Second, a demonstration of ripping and testing filters
- Third, upload XSS Issue – uploading files and internal storage (OSVDB: 88443)
- Fourth, admin password reset issue (OSVDB: 88441)
- Fifth, a “partial” SQL injection: works but we can't get data out of app (OSVDB: 88442)
- Finally, File Access Control Issue – It's all about scope (OSVDB: 88444)

Ken's Bio

Kenneth F. Belva is the Publisher and Editor-in-Chief of bloginfosec.com. He previously managed an Information Technology Risk Management Program for a bank whose assets are Billions of dollars. He reported directly to the Senior Vice President and Deputy General Manager (CFO).

ITsecurity.com recognized him as one of the top information security influencers in 2007.

In 2009, he was published in the Information Security Management Handbook, Sixth Edition, edited by Hal Tipton and Micki Krause. He also co-authored one of the central chapters in Enterprise Information Security and Privacy, edited by Warren Axelrod, Jennifer L. Bayuk and Daniel Schutzer.

In addition to his daily corporate responsibilities, he is currently on the board of the New York Metro Chapter of the Information Systems Security Association (ISSA).

He recently co-authored a paper entitled “Creating Business Through Virtual Trust: How to Gain and Sustain a Competitive Advantage Using Information Security” with Sam Dekay of The Bank of New York. of security breaches on stock prices.

Mr. Belva frequently presents at information security conferences around the US as well as globally. He writes on day-to-day information security experiences in a non-essay format at SecurityMaverick.com when time permits and can be followed on twitter @infosecmaverick

Ken's Contact Info

Website: http://silverbackventuresllc.com

Email: ken@silverbackventuresllc.com

Research: http://securitymaverick.com

Essays: http://www.bloginfosec.com

Twitter: http://twitter.com/infosecmaverick

Stop by and say, 'Hi'!