DefenseCode

Security Products Development

Leon Juranic
leon@defensecode.com

Security Products Development

- Q: Why I picked this boring topic at all?
- A: Avoidance of any hacking-related topics for fsec (khm.) :)

Security Products Development

- For last 4-5 years, I've been working on commercial security products
- This is my story on security products development
  - Idea
  - Motivation
  - Obstacles
  - Goals
  - Results

Security Products Development

- Why would someone start to develop commercial security tools on his own?
- Main reason - I was never thrilled to use other people's software.
  - Ego stuff? Possible.
  - Money? Definitely.

Security Products Development

- For last 13 years, I coded my own security tools:
  - port scanners
  - vulnerability scanners
  - fuzzers
  - exploits
  - shellcodes
  - network exploitation tools
  - misc. junk
  - etc.
- I have this retarded obsession that I have to know it under the hood, in details...
- Few years ago, I decided it's time to make some money on it. :)

Security Products Development

- I decided to go for Web Application security. Why?
- Simple - ROI.
- "Cool" hacking stuff takes much more resources...
  - Buffer overflows are kinda rare these days
  - High profile b0fs are hard to find (takes time)
  - Low-hanging fruits are mostly gone
  - Hard to exploit (takes time)
  - Buffer Overflows - Questionable ROI

Security Products Development

On the other side...

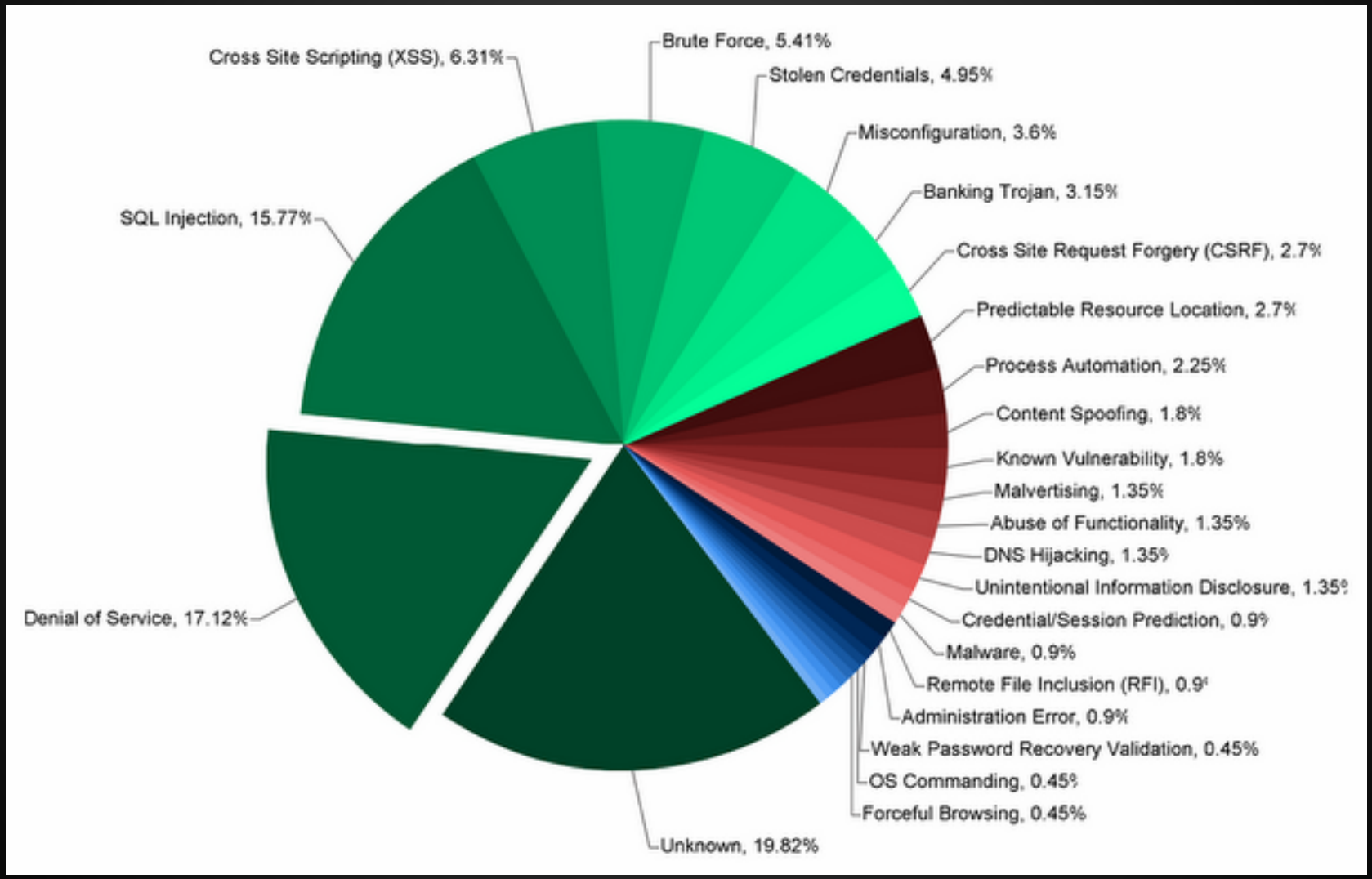
- Web Applications Security - Good ROI
- Everyone has at least one website
- Web Apps are mostly vulnerable as hell (more than 80% of web sites vulnerable)
- Nowadays, more than 60% of server-side intrusions are result of poor web apps

Security Products Development

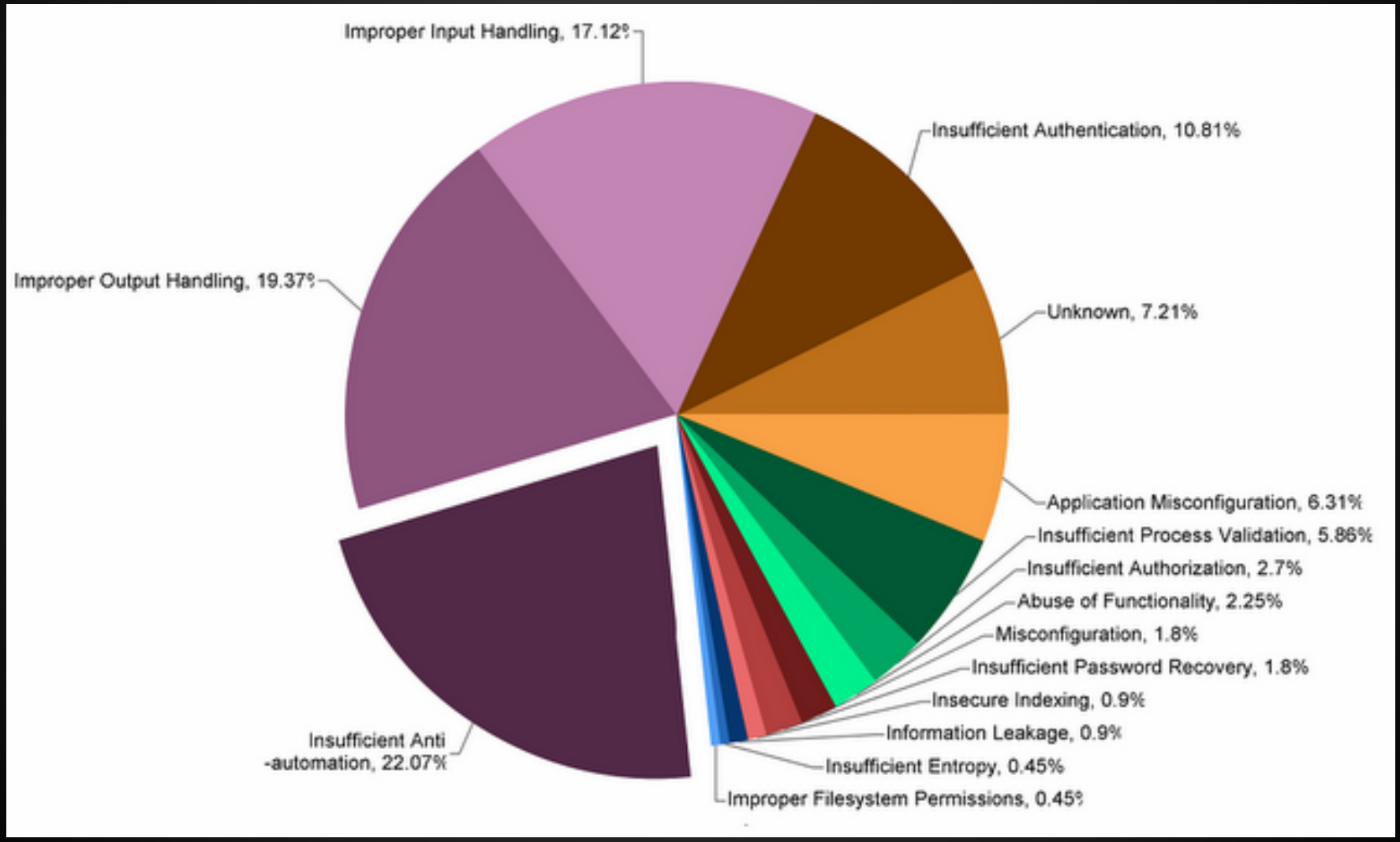
"Did they get you to trade? Your heroes for ghosts". - Pink Floyd

- Sellout?
- No...
- Simple shift of focus from "hard core" resource consuming hacking stuff, to more cost-effective security research/audit
- Attacking the weakest link (Web Apps)

WASC 2010 WEB APP VULNS



WASC 2010 WEB APP VULNS



Security Products Development

- GOAL: To create products that will be able to discover vulnerabilities in web applications like I'm doing it manually
- To summarize knowledge and experience gained over the years, and transform it to security products

Security Products Development

- In past 4-5 years, I've coded engines/tools:
  - **Web Security Scanner**
    (Black-Box)
  - **Web Application Source Code Security Analysis Scanner**
    (White-Box)

Web Security Scanner

Web Security Scanner

- I've started to work on my own Web Security Scanner
- Tool for fully automated black-box security audit of web applications
- Simple: Just set it to target URL and hit start
- Goal: To discover all security vulnerabilities present in target web application/web site

Web Security Scanner

- BENCHMARK-LIKE-GOAL:
  - To develop a tool that will discover 0days in popular web apps.
- GOAL: Comprehensive, effective and fast
- GOAL: It has to be simple to use
- GOAL: Nice, and user-friendly GUI

Web Security Scanner

- Web Security Scanner components under the hood:
  - **Crawling engine**
  - **Security testing (attack) engine**
  - **Reporting engine**
  - **Brute-force engine**
- All components equally important

Web Security Scanner Crawling

- GOAL: To crawl and record every single link and form on website

- Problems:
  - Various technologies used on modern web sites
  - HTML, JavaScript, Flash, AJAX, JSON, etc. etc.
  - - You have to cover **ALL** of them
  - - If you miss single link/file, you could miss critical security vulnerability
  - How to write something effective, but still non-intrusive

Web Security Scanner Crawling

- Problems:
  - You don't want to crash http server or cause DoS on it
  - Authentication (Basic, Digest, NTLM, Form, Cookie)
  - Security testing of links and forms (POST, GET, HEAD, PUT)
  - Testing Cookies, link rewrite...
  - Exclusions (sometimes you don't want to scan everything)
  - HTTP/1.0, HTTP/1.1
  - etc.
  -

Web Security Scanner Testing

- GOAL: Identify security vulnerabilities in links/forms/scripts previously collected in crawling engine

- Problems:
  - You have to create test cases for every possible security vulnerability class
  - SQL Injection, XSS, Command Execution, File Disclosure, etc. etc. etc.
  - Over 40 vuln. classes
  - Special modules for Blind SQL Injections and stuff

Web Security Scanner Testing

- Problems:
  - Hidden resources brute-force (files, dirs, backup)
  - Support for all known security vulnerabilities
  - Various technologies (Apache, IIS, PHP, Java, ASP. Net, ASP, CF, etc.)
  - Various OSes (Linux, Windows, *BSD, Solaris, etc.)
  - You have to create huge database of security checks
  - Most tests based on response analysis and error messages
  - JavaScript emulation engine
  - Flash engine

DEMO

Web Application Static Source Code Security Analysis Scanner

Web App Static Source Code Security Analysis

- Web Application Static Source Code Security Analysis Scanner (White-Box)
  - Tool to discover security vulnerabilities in web application source code
- GOAL: To discover all security vulnerabilities present in target web application source code
- GOAL: Low, very low false positive rate

Web App Static Source Code Security Analysis

- It has to be effective
- It has to be simple to use
- It has to be fast
- BENCHMARK-LIKE-GOAL: To develop a tool that will discover 0days in popular web apps.
- If it can't discover really complex vulnerabilities - no use of it. :)

Web App Static Source Code Security Analysis

Under the hood:

- Code analysis engine
- Data flow analysis engine
- Tainted input recognition engine
- Security scanning and validation engine

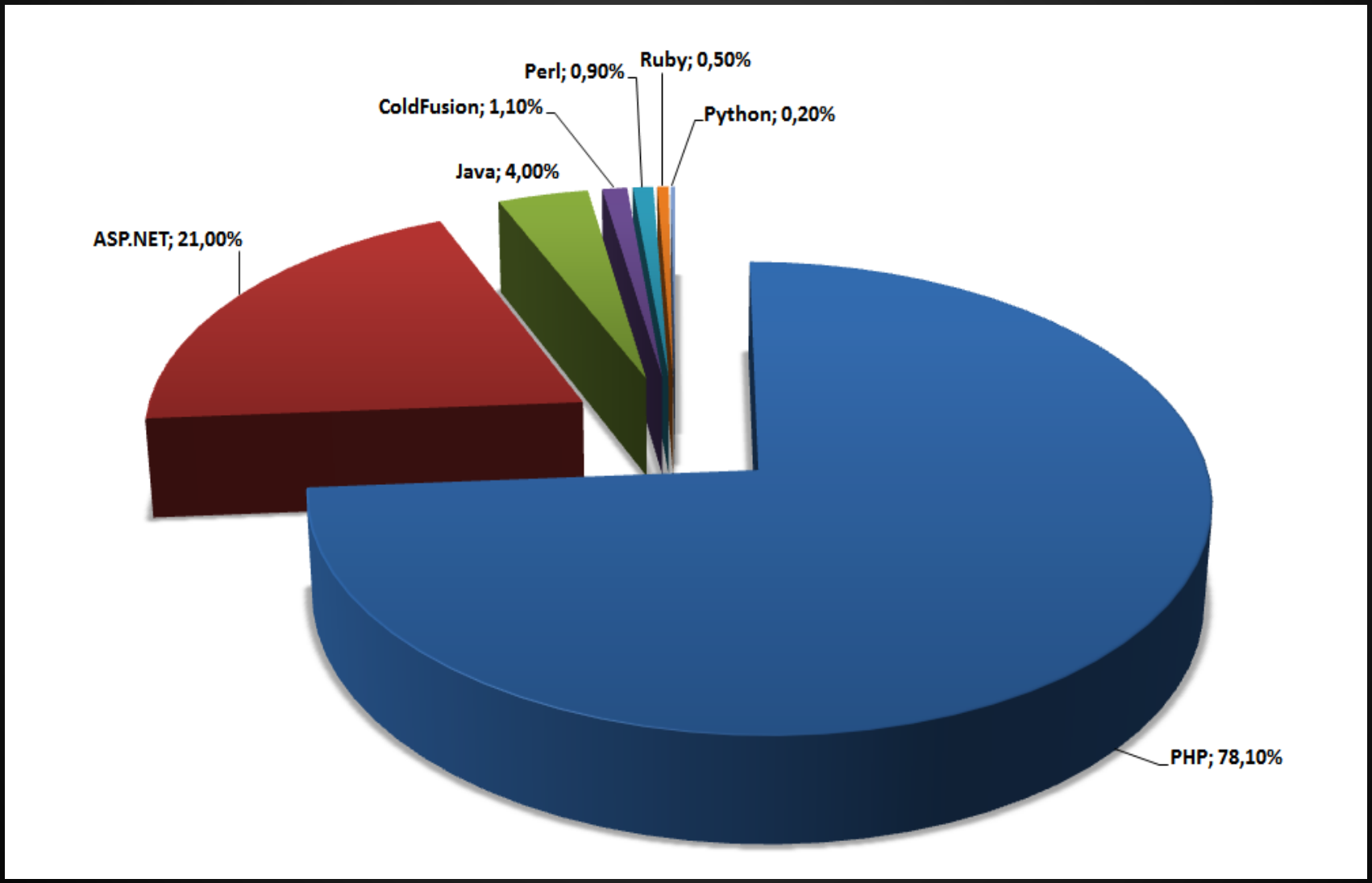
Web App Static Source Code Security Analysis

- I had two choices - lexer or parser.
- I did it with parser
- Simple - Easier to implement, and less work :)

Web App Static Source Code Security Analysis

- Problems:
  - I really can't list them all.. :)
  - Various programming languages
  - So we have to cut them down to most popular (market share)
- ASP.Net, Java, PHP, VB.Net, ASP
  - Various coding practices/styles
  - Third-party frameworks
  - **Simple pattern matching for simple vulnerabilities is just not acceptable**

Web App Static Source Code Security Analysis - **Market Share**



Web App Static Source Code Security Analysis

How it works in a few short steps...

1. Analyze code files/includes/libraries
2. Discover all custom classes/methods/ functions/entry points
3. Simulate code execution
4. Track and follow user input through code
5. Discover vulnerable functions
6. Decide is it false positive
7. If not, report that stuff. :)

DEMO

Conclusion

- Don't do it! :)
- It's never ending game...
- If you really want to do it right, you have to give up on:
  - Common sense!!!
  - Sleep!!!
  - Time...

- Results
  - Still waiting.... :)