SANS AppSec 2012

AppSec – what can you learn from small companies?
What Works and What Doesn’t
About Me

- 25 years experience in software development and Ops
- Mostly in small companies designing and building software for big companies
- Last 15 years, managed technology for stock exchanges
- Consulted for IBM’s international financial services practice
- 5 years ago, co-founded BIDS Trading, where I built and manage the Technology group
- BIDS Trading is an ATS for fair and efficient institutional stock trading in the US – performance, reliability, security
- I’m a software guy, not a security guy – but security is an important part of my job
## Timeline

<table>
<thead>
<tr>
<th>Year</th>
<th>Stage</th>
<th>AppSec Work</th>
</tr>
</thead>
</table>
| 2006   | Startup                                    | Built application framework  
                  Internal AppSec review  
                  Static analysis in CI  
                  Expert secure design review  
                  Application pen test  
                  Expert secure code review  
                  Ops hardening |
| 2007   | Phased Launch                              | Change Control, Tripwire, ...  
                  Incident Response + RCA  
                  Secure coding training  
                  Code reviews with checklists  
                  Secure SDLC roadmap |
| 2008-now | Moved to Agile short release cycle (Rapid AND Safe) | Tried Web Scanning + Fuzzing  
                  Developer testing training  
                  Secure Ops training  
                  Regular pen tests  
                  Regular vulnerability scans |

Copyright © 2012 BIDS Holdings, L.P. BIDS Trading, L.P. - Member FINRA, NYSE and SIPC. All Rights Reserved.
What worked, what didn’t (YMMV)

<table>
<thead>
<tr>
<th>Practice / Tools</th>
<th>What we Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Analysis / Web Scanners?</td>
<td>Hard for black box testers to use/understand Tools need to be trained and retrained... Still noisy, crash/hang, Ajax problems Good for finding naïve mistakes in web apps Not good if you have many different interfaces</td>
</tr>
<tr>
<td>Static Analysis / Source code Scanners</td>
<td>Fast feedback/quick ROI – easy to integrate Makes code reviews easier, catches stupidity Different engines find different problems</td>
</tr>
<tr>
<td>Attack Surface Analysis and Threat Modeling</td>
<td>New or different interfaces? Changed security features/plumbing? Threat modeling (informal) for high risk areas Doing is more important than documenting</td>
</tr>
<tr>
<td>Vulnerability Scanning</td>
<td>Basic part of operational hardening (O/S, stack) Easy to bring in-house Upgrade, patch, re-configure, then do it again...</td>
</tr>
</tbody>
</table>
# What worked, what didn’t (YMMV)

<table>
<thead>
<tr>
<th>Practice / Tools</th>
<th>What we Learned</th>
</tr>
</thead>
</table>
| Exploratory and Adversarial Testing | Unit testing and functional testing not enough  
Fat tests that cover a lot, try to break things  
Getting dev and test used to this takes time  
Improves security AND reliability  
BSIMM top practice first step to security testing |
| Fuzzing?                         | Smart fuzzing needs technical smarts  
But developers don’t like doing it  
Dumb fuzzing only good if you have lots of files |
| Bug Tracking                     | Track everything in the same issue database  
Security vulnerabilities are bugs  
Full traceability to deployment |
| Code Reviews                     | Correctness and defensive coding first  
Later maintainability  
Security-specific reviews on “pointy hat stuff”  
Keep checklists short |
## What worked, what didn’t (YMMV)

<table>
<thead>
<tr>
<th><strong>Practice / Tools</strong></th>
<th><strong>What we Learned</strong></th>
</tr>
</thead>
</table>
| **Pen Testing**               | Can find good testers for web apps  
Hard to find good testers for other apps/APIs  
Be transparent – tell the pen tester everything  
Learn everything you can, mostly from first test  
Treat serious findings seriously: escalation/RCA |
| **Expert Design Review**      | Found real problems early – “pointy hat” stuff  
Reinforces what you are doing right  
Lesson: Sometimes simple design is too simple |
| **Expert Code Review?**       | Expensive, time-consuming  
Have to hold reviewer’s hand to be worth it  
Forces you to learn so that you can assess risks  
Finds “pointy hat” problems and quality issues |
| **Training**                  | Train everyone in basics and defensive coding  
Managers and security lead advanced training  
Train everyone in testing too |
Lessons Learned

- It’s hard to build secure software on your own...
  - Everybody needs training (developers, testers, managers, ops)
  - Get help from consultants on the “pointy hat” stuff (crypto, sessions, ...)
  - Work with your customers: some of them know more than you

- Spend time on upfront design - a good framework will pay off big!
  - Application framework for database access, logging, error handling, ...
  - Security framework/libraries for crypto, authentication, sessions, ...

- Tie software security into code quality (some security comes free)
  - Start with defensive coding
  - Static analysis and code reviews
  - System testing (not just functional and unit testing)

- Security has to be part of development and ops, part of everyone’s job
  - Make AppSec a “black belt” problem: extra training, extra responsibility
  - If your best people take it seriously, the rest of the team will too
BIDS Trading is a joint venture of: Bank of America/Merrill Lynch (NYSE: BAC), Citi (NYSE: C), Credit Suisse Group (NYSE: CS), Deutsche Bank (NYSE: DB), The Goldman Sachs Group, Inc. (NYSE: GS), JPMorgan Chase & Co. (NYSE: JPM), Knight Capital Group, Inc. (NYSE: KCG), Lehman Brothers (NYSE: LEH), Morgan Stanley (NYSE: MS), NYSE Euronext (NYSE: NYX), and UBS (NYSE: UBS). The BIDS Trading ATS will be open to all qualifying broker-dealers and their institutional clients, subject to basic credit and regulatory requirements.

For more information, visit www.bidstrading.com.