Security And Performance: A Tale Of Two Cities

Rekha Joshi
Principal Engineer, Intuit
@rekhajoshm
https://www.linkedin.com/in/rekhajoshm
An Insight Into Intuit

- Founded: 1983
- IPO: 1993
- Employees: 7,900
- Locations: 24
- Revenue: $4.7B
- Customers: 42M
Our Mission: Powering Prosperity Around World

- turbotax
- quickbooks
- proconnect
- mint
Is This Talk For You?

Is Your Application Online or Data Intensive?
The Joshi Hierarchy Of Application Needs

Maslow hierarchy of needs

Joshi hierarchy of application needs
The First Step In Defense Is To Know Your Weaknesses
Security Layers

Physical Security: Earthquake, Flood, Fire

Network Security: MIT Kerberos, TLS

Platform Security: Authentication, Authorization, Audit

Cloud Security: Regions, VPN, Subnet, VPC, Security Groups, ACL

Application Security: Secure Coding, No obscurity

Data Security: Data Governance, Encryption, Key Rotations
Constant Vigilance

- Fix Vulnerabilities
- Threat Modeling
- Penetration Testing

- Spam Emails
- Viruses, Trojans, Worms
- SQL Injection
- Cross Site Scripting
- Distributed Denial of Service attacks
- Spoofing
- Deprecation
- Social Engineering
Constant Vigilance

Fix Vulnerabilities
Threat Modeling
Penetration Testing

http://www.cvedetails.com/vulnerabilities-by-types.php

![Bar Chart showing various vulnerability types and their counts.]

- Denial of Service: 18962
- Execute Code: 25583
- Overflow: 12882
- XSS: 10681
- Directory Traversal: 3136
- Bypass Something: 4792
- Gain Information: 7676
- Gain Privilege: 4343
- SQL Injection: 6523
- File Inclusion: 2150
- Memory Corruption: 4167
- CSRF: 1463
- HTTP Response Splitting: 143
Constant Vigilance

- Fix Vulnerabilities
- Threat Modeling
- Penetration Testing

Open Source - Do Your Own Due Diligence

- heap buffer overflows
- global buffer overflows
- stack buffer overflows
- use after frees
- uninitialized memory
- stack overflows
- timeouts
- ooms
- leaks
- ubsan
- unknown crashes
- other (e.g. assertions)

https://security.googleblog.com/2017/05/oss-fuzz-five-months-later-and.html
## Constant Vigilance

Fix Vulnerabilities | Threat Modeling | Penetration Testing

### DREAD: Damage Potential, Reproducibility, Exploitability, Affected users, Discoverability

<table>
<thead>
<tr>
<th>Rating</th>
<th>High (3)</th>
<th>Medium (2)</th>
<th>Low (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Damage potential</td>
<td>The attacker can subvert the security system; get full trust authorization; run as administrator; upload content.</td>
<td>Leaking sensitive information</td>
<td>Leaking trivial information</td>
</tr>
<tr>
<td>R Reproduceibility</td>
<td>The attack can be reproduced every time and does not require a timing window.</td>
<td>The attack can be reproduced, but only with a timing window and a particular race situation.</td>
<td>The attack is very difficult to reproduce, even with knowledge of the security hole.</td>
</tr>
<tr>
<td>E Exploitability</td>
<td>A novice programmer could make the attack in a short time.</td>
<td>A skilled programmer could make the attack, then repeat the steps.</td>
<td>The attack requires an extremely skilled person and in-depth knowledge every time to exploit.</td>
</tr>
<tr>
<td>A Affected users</td>
<td>All users, default configuration, key customers</td>
<td>Some users, non-default configuration</td>
<td>Very small percentage of users, obscure feature affects anonymous users</td>
</tr>
<tr>
<td>D Discoverability</td>
<td>Published information explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable.</td>
<td>The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious use.</td>
<td>The bug is obscure, and it is unlikely that users will work out damage potential.</td>
</tr>
</tbody>
</table>
Constant Vigilance

- Fix Vulnerabilities
- Threat Modeling
- Penetration Testing
Secure By Design
Security Facets

- Authentication
- Authorization
- Auditing
- Isolation
- Encryption
- Data Governance
- Key And Secret Management
- Key Rotation
- Standby Cluster

Security In Practice


Deployment
• Regular AMI updates/patches
• Ensure no access Leaks
• Secure your configurations
• Heed internal security bulletin

Application
• Ensure no cross site scripting
• Ensure no memory leaks, buffer overrun
• Ensure thread safety
• Implement secure patterns
Is Your Tech Stack Secure?

Example Tech Stack
- Linux
- Hadoop 2.x
- AWS Services
- Spark
- Kafka
- Cassandra

For a service to be business viable, it needs to be secure!
But Secure Does Not Have To Mean Slow..
Performance Tuning

- **OS Optimizations:** CPU, Network, IO
- **Platform Optimizations:** Provider, Auto scaling, Tuning
- **Database Optimizations:** Indexes, tables, partitioning
- **Application Optimizations:** Code Quality, Language, Caching, Using Asynchronous API

Understanding workload
Leaner deployment patterns
Scalable components
Data structures, algorithms
Caching
Limit strings, objects created
Tune garbage collector
Use asynchronous processing
Code quality
Memory optimization
Performance Tuning

Hardware Optimizations:
Processor, cores, SSD, disks, file system

TEGRA X1
CPU CONFIGURATION

4 HIGH PERFORMANCE A57 BIG CORES
- 2MB L2 cache
- 48KB L1 instruction cache
- 32KB L1 data cache

4 HIGH EFFICIENCY A53 LITTLE CORES
- 512KB L2 cache
- 32KB L1 instruction cache
- 32KB L1 data cache

Tensor Processing Unit
Don’t Settle For A Single Test Run

Performance Tests:
Simulated Representative Load, Concurrent Users, Varying Variables, Environment, Key Vault Performance

Key Metrics
- Concurrent Users/Session
- Average Session Length (Minutes)
- Application CPU Utilization (%)
- Application Memory Utilization (%)
- IO usage
- Web Apache Throughput (TPS)
- Web Apache Busy Threads
- Web CPU Utilization (%)
- Web Memory Utilization (%)
- Network Packets
But Start With The End In Mind..

Intuit Open source AB Testing: https://github.com/intuit/wasabi
Be Proactive In Measuring Performance

- AWS resource alarms
- Custom process alerts
- Logging and alert

CloudWatch

#!/bin/bash
y=0
for x in $@
do

New Relic

APM

JVM and App Metrics

Custom App Metrics

ElasticSearch

Prometheus.io

Grafana

LogStash

Kibana

Slack

Pagerduty

Splunk
So What Should You Do Now?
Be Secure

Patch, Patch, Patch
  - AMI, OS, Software patches

Adopt Principle Of Least Privilege
  - Access Controls
  - Authentication, Authorization

Know And Manage Your Data
  - Data Governance
  - Legal And Compliance

Embrace Secure Coding Practices
  - Secure Configuration
  - Input Validation
  - No Memory Leak

Publish Internal Security Bulletins
  - Internal Security hack-a-thons
  - External Security blogs
Be Fast

Have Automated Fast Lean Stateless Deployment

Have Performing Infrastructure
  - Processor chips, cores
  - Type Of Instance
  - Auto Scaling

Embrace High Performance Coding Practices
  - Fewer Object Creation
  - Garbage Collection Tuning
  - Secure Configuration
Takeaways

Security Is Vital
Performance Is Critical
You Can Have Them Both
It’s a Journey
@rekhajoshm  https://www.linkedin.com/in/rekhajoshm