TAKING THE MODULAR VIEW

Extracting security from the application

Chenxi Wang, Ph.D.
Forrester Research
SANS Application Security Summit, May, 2012
Application security remains an elusive goal

- 2012 Breach study (Verizon)
  - More than 50% breaches involve web apps

![Bar chart showing breach types and their percentages.]

- Web applications: 54%
- Backdoor control: 34%
- Remote desktop services: 20%
- Unknown: 17%
Mobile apps are taking off

Mobile apps: $6 billion Market today
Will hit $ 55.7 billion by 2015

March 2012 “Mobile App is the new fact of engagement”
State of application security: Abysmal and not getting better any time soon
Two unfortunate facts

**Software Complexity**

Software is getting more complex with modern development frameworks

**Security Technology**

Software security products are difficult to use. Dev adoption remains low,
Today’s software security processes and technologies

• Too complex
  – “6 months to stabilize a static analysis product on a single code project” – CISO of a large financial firm
So how do we change the status quo?
Let’s look at the problem in a different light
Extract security from the application

Authentication (credential handling, Risk-based auth, ...)

Secure Communication (protocols, encryption)

Crypto-related functions

Security log capturing & analysis

Core application business logic

App-level DDoS protection

Security-specific logic (e.g., jailbreak detection)

App dev: focus on this

Copyright © 2012 Forrester Research. Inc
Extract security from the application

**Security domain**
- Developed by engineers with "know how"
- Separately verified
- Separately hardened
- Separately developed

- Security-specific logic (e.g., jailbreak detection)
- Secure Communication (protocols, encryption)
- Crypto-related functions
- Security log capturing & analysis
- DDoS protection
Separation of duty between Security & Business Logic
This is a powerful architecture

- Authentication (credential handling, Risk-based auth, ...)
- Secure Communication (protocols, encryption)
- Security log capturing & analysis
- App-level DDoS protection
- Core application business logic
- Crypto-related functions
- Security-specific logic (e.g., jailbreak detection)
Taking it several notches up ...

- Authentication
- Threat protection (DDoS, WAF)
- Security analytics
- Core application business logic
- Security-specific logic
- Security log & analysis
- Crypto-related functions
- Secure Communication
The modular view

Core application business logic

Implemented separately

Come together at run time

Security functions

Implemented separately
Security is a case for modularity

- All programs need security
- Not all developers are trained in security
- Security has commonality from program to program
  - Authentication logic
  - Attack detection
  - Security monitoring
  - ...
Think about it

• Some are good at developing security functions
  – Authentication, app hardening, threat protection
  – May be unique, may be reusable

• Others
  – Business logics create security use cases
  – More efficient to rely on others to supply security functions
What makes this vision work?

• Standard APIs
• Modular way of programming
• Zero trust assumptions
Re-examine your trust assumptions

• Do not assume the network is trusted
  – No data is communicated in the clear
• Do not assume that the component on the other end is trusted
  – Always engage in authentication
• Trust is derived, verified, not assumed
Distributing security tasks presents a host of opportunities.

- modularity
- agility
- efficiency
- security
But we can’t outsource all security ...
For the core business logic

• Ensure code level integrity
• Eliminate security vulnerabilities from the code
• Reduce unintended side-effects

You still have to test and review the core program!
But at least ...

Your responsibility to secure

- Authentication (credential handling, risk-based auth, ...)
- Security-specific logic (e.g., jailbreak detection)
- Secure Communication (protocols, encryption)
- Crypto-related functions
- App-level DDoS protection
- Security log capturing & analysis
- Application business logic

Your responsibility to secure
And because you have standardized the interfaces

- Reduced attack surface
- Reduced the scope to monitor and verify
Real world examples – mobile application wrapping

- Separate Login (PIN Code Access)
- Geo fencing
- VPN Enforcement
- Encryption
- Data Containment
Next 2-3 years and beyond
How far are we from this modular vision?

• Not very close
  – Interfaces are not standardized
  – Third party functions are not necessarily trustworthy or easily comparable
  – Fine-grained control over data/content is not mature
  – Modular programming for security is not widely practiced
Cloud services rely on APIs heavily

35% of our enterprise applications include API calls to Amazon web services
- CIO of a large health care company

API-traffic is a major category of Internet traffic

75% API > web traffic
OAuth is gaining momentum

**OAuth API**: “single keychain for authenticating to” Twitter, salesforce.com, and Chatter APIs

**Google Apps and Gmail binding**: avoiding the password antipattern

**SecureSpan Gateway**: tool kit approach to STS for flexibility in the face of spec change

**WIF and Azure ACS**: client and server support for .NET development

**PingFederate**: STS that “allows apps to add identity information to their API calls” with OAuth

**Force.com**: “complete PaaS with brandable OAuth 2”

Source: June 3, 2011, “The ‘Venn’ Of Federated Identity” Forrester report
The Cathedral vs. Bazaar Model

Top down
Monolithic
Available to the elite

Bottom up
Agile
Developer-centric
Modular security is a journey

- Isolated experiments: 12 to 24 months
- API-based programming becomes prevalent: 2 to 5 years
- Modular by design: Establishing ecosystem: 5 to 8 years
- Secure by design: 5 to 8 years
Recommendations

• Take a modular view: extract security from applications and infrastructure
• Adopt the API-centric model
• Review your trust assumptions for building applications
• Prepare to distribute security tasks
Questions?

Chenxi Wang
E: cwang@forrester.com
T: @chenxiwang