John Dickson

- Principal of Denim Group
- 15-year information security consultant background
- Ex-Air Force security analyst at AFCERT
- Trident Data Systems, KPMG, SecureLogix, and Denim Group information security consultant
- Works with CIO’ s and CSO’s to build successful software security initiatives
- Educates non-developer security professionals how to manage application risk

Denim Group Background

- Professional services firm that builds & secures enterprise applications
  - *External application assessments*
    - Web, mobile, and cloud
  - *Software development lifecycle development (SDLC) consulting*
- Classroom and e-Learning for PCI compliance
- Secure development services:
  - *Secure .NET and Java application development*
  - *Post-assessment remediation*
- Deep penetration in Financial Services, Banking, Insurance, Healthcare and Defense market sectors
- Customer base spans Fortune 500
- Contributes to industry best practices through the Open Web Application Security Project (OWASP)

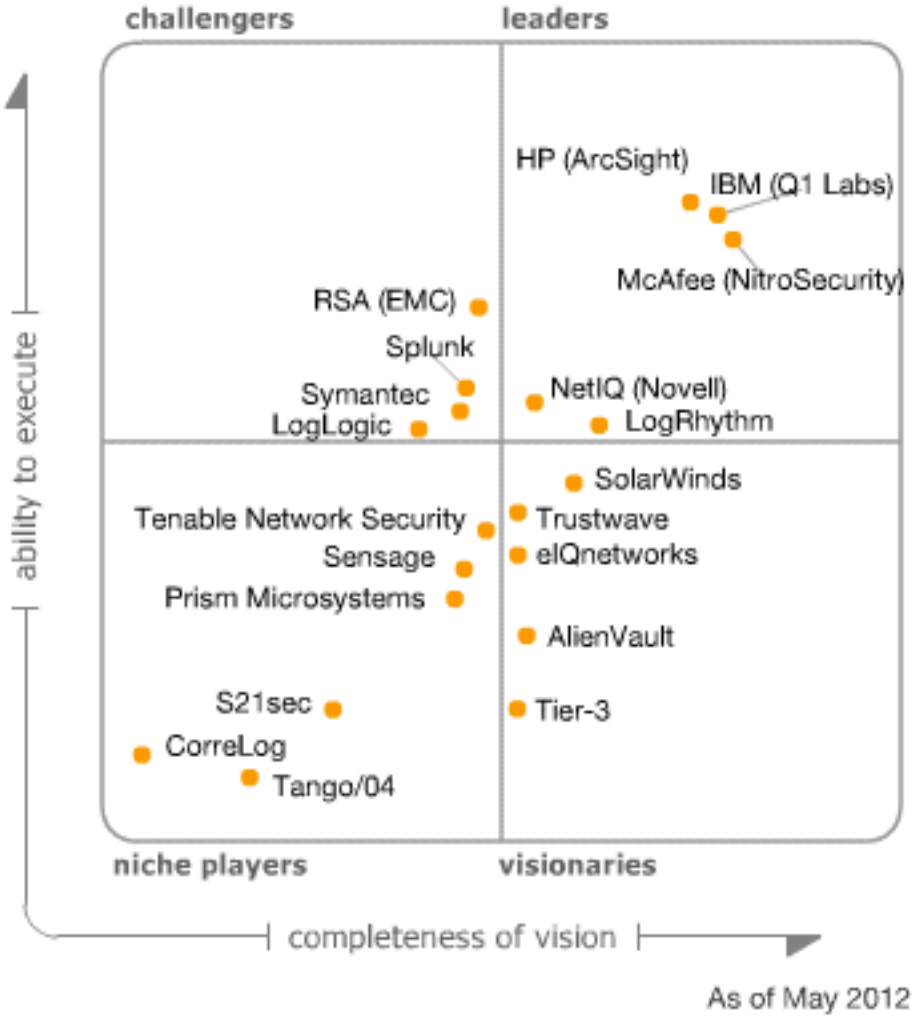
Overview

- Today's network security intelligence
- The target of choice – applications
- In application defenses
- Application logging blocking and tackling
- WAF's and application IDS
- Virtual patching for web applications
- Conclusion

Today's Network Security Intelligence

- Key information from security-related events in the organization can be collected
  - *The broader range the better*
  - *A variety of key security events on a multitude of devices*
    - Firewalls
    - Remote access servers
    - Critical servers (e.g., active directory)
- Correlation and analysis capabilities are also mature
  - *Context important to analysis*
- Network and server logging market maturing
  - *Marketplace includes Security Event Managers (SEM), Security Information and Event Managers (SIEM), and Advanced Logging Products*
  - *May address certain compliance requirements like Sarbanes-Oxley*

Today's Network Security Intelligence



The very crowded SIEM market

Source: Gartner Group, Magic Quadrant for Security Information and Management, May 2012

Today's Network Security Intelligence

- What do network defenders really need?
  - *What, when, where, and how an event occurred*
  - *In a format that is external to the system or the application that created it*
  - *In a predictable format that is straightforward to import to SIEMs*

The target of choice – applications

- App level breaches accounted for 10% of breaches overall, but 54% for large organizations
- Only 20% of all organizations were in compliance with PCI DSS Requirement 6
  - *Develop and maintain secure systems and applications*
- Only 57% of large organizations were compliant with the PCI DSS Requirement 6
- The average number of days a website was exposed to at least one serious vulnerability is 231 days
- XSS was found in 55% of websites in 2011

Sources: Verizon Business System, 2012 Breach Report and WhiteHat Security Report

Application Vulnerability – Injection

#1 in OWASP Top 10

- "Getting into" a file system or database used to imply establishing a root session or a direct SQL connection
- By exploiting applications, attackers can accomplish their goals without such direct access
  - *The whole system trusts the application*
  - *Attackers will try to leverage that trust*

With normal input

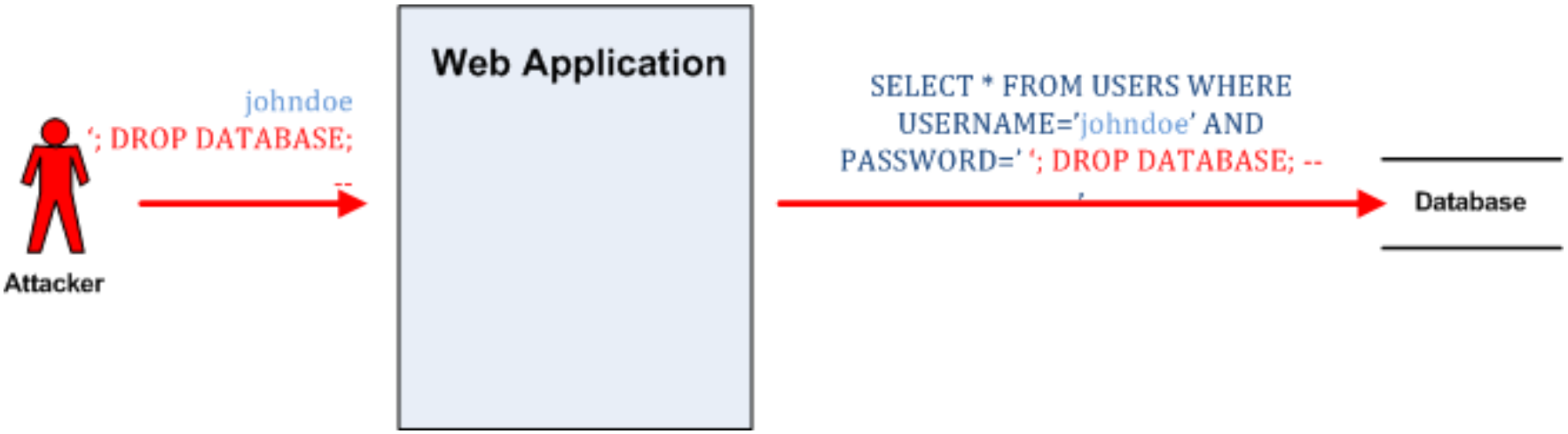
User Name: johndoe
Password: myBirthday

SELECT * FROM USERS WHERE USERNAME='johndoe' AND PASSWORD='myBirthday'

With malicious input

User Name: johndoe
Password: '; DROP DATABASE; --

SELECT * FROM USERS WHERE USERNAME='johndoe' AND PASSWORD=' '; DROP DATABASE; -- '



What Is It?

- Occurs when **unfiltered user inputs** are combined with **static text** and then **sent to an interpreter**.
- The interpreter then **executes commands of the attackers choosing** rather than the commands specified in the application.
- **Very common** application security flaw with potentially **disastrous security implication**s.
- We will focus on **SQL injection** because it is the most common.
- Other common injection flaws include OS Command, XML, and LDAP.

In application defenses

- A chasm exists between the development and operations/security communities
  - *DevOps and Rugged Software Development changing that gap*
- Most software developers don't build enterprise software with security in mind
  - *Outside the largest banks and financial institutions, the security of software is less important (vs. features and functionality)*
- Most defensive coding focused on filtering malicious inputs
  - *Very little focus on the fidelity of application logging to enhance security response*
- Most logging done to capture software debugging info
  - *Developers want to understand how an application failed*
  - *Enhanced security information rarely a requirement for logging*

The very real state of in application defenses

- Example application log #1

```
logger.warn("Caught exception " + e);
e.printStackTrace();
```

The very real state of in application defenses

- Example application log #2

```
logger.info("Failed login for user " + username + "
with password '" + password + "'");
```

The very real state of in application defenses

- Example application log #3

```
logger.info("Order placed with credit card number "
+ creditCardNumber);
```

The very real state of in application defenses

- What did we learn from these examples?
  - *Developers might log information that actually creates more security headaches*
  - *Developers might not log information that is needed to analyze an attack*
  - *Developers might not log key security events in a human readable format*
  - *Developers might log information that is not in a format that is easily consumable by a SIEM (i.e., structured data)*
  - *Developers rarely ask security operations analysts for input on the types of logging needed*
  - *Developers rarely worry about the need to conduct trusted logging*

Application logging blocking and tackling

- Security operators must inject themselves into the design phase of development projects to articulate security event logging requirement
- Security operators need to better understand application-layer information and how it can help them better identify security events
- Developers need to increase the fidelity of the security event information the do send to logs

Application logging blocking and tackling

- Increasing the fidelity of security event logging – HOW?
- Need to focus on:
  - *What*
  - *When*
  - *Where*
  - *How*
- Key events to log:
  - *Authentication*
  - *Authorization*
  - *Access*

Source: "How to Do Application Logging Right," Chuvakin, Anton, and Peterson, Gunnar,

Which events to log

- Input validation failures
- Output validation failures
- Authentication successes and failures
- Authorization failures
- Session management failures
- Application errors and system events
- Application and related systems start-ups and shut-downs
- Use of higher-risk functionality
- Legal and other opt-ins

Source: OWASP Application Logging Cheat Sheet

Which events attributes to log

- Log date and time
- Event date and time
- Interaction identifier
- Application identifier
- Application address
- Service
- Window/form/page
- Code location
- Source
- User ID
- Type of event

Source: OWASP Application Logging Cheat Sheet

Which events never to log

- Passwords
- Sensitive system attributes
- Source code
- Session identification values
- Sensitive business information
- Patient information (EPI)
- Bank account or payment card holder data
- HR, Payroll, M&A data or anything generally more sensitive than logs

Source: OWASP Application Logging Cheat Sheet

Application logging blocking and tackling

- Example application log #4

```
logger.debug("Failed login for user: " +
logEscape(username));
```

Application logging blocking and tackling

- Example application log #5

```
logger.warn("User " + logEscape(username) + "
attempted to access document id " +
logEscape(documentId) + " without sufficient
permissions");
```

WAF's and application IDS

- Broad set of technologies that enable enhanced application-layer logging
- Provide insight into Port 80/443 where most firewall don't have info
- Can block certain attack patterns at the application layer
- Most WAF's in production are not set in blocking mode
- Block obvious web application vulnerabilities like XSS & SQL Injection
  - *Less effective on business logic or authorization rules*

AppSensor

- A conceptual framework that offers guidance to implement intrusion detection capabilities into existing application
- Utilizes standard security controls and recommendations for automated response policies based upon detected behavior.
- Identifies malicious users within the application and eliminate the threat by taking response actions.

AppSensor

- An attacker often requires numerous probes and attack attempts in order to locate an exploitable vulnerability within the application.
- By using AppSensor, it is possible to identify and eliminate the threat of an attacker before they are able to successfully identify an exploitable flaw.

AppSensor

OWASP
The Open Web Application Security Project

- Behavior examples (~50)
  - *2 Detection Points 2.1 RequestException*
  - *2.1.1 RE1: Unexpected HTTP Command*
  - *2.1.2 RE2: Attempt to Invoke Unsupported HTTP Method*
  - *2.1.3 RE3: GET When Expecting POST*
  - *2.1.4 RE4: POST When Expecting GET*
  - *2.1.5 RE5: Additional/Duplicated Data in Request*
  - *2.1.6 RE6: Data Missing from Request*
  - *2.1.7 RE7: Unexpected Quantity of Characters in Parameter*
  - *2.1.8 RE8: Unexpected Type of Characters in Parameter*

Virtual patching for web applications

- Receives vulnerabilities from application vulnerability scanners
  - *Dynamic or static analysis (source code) reviews*
- Creates “virtual patches” that are sent to WAFs and block a URL
  - *Mod Security*
  - *F5*
  - *Imperva*
- Ecosystems being created to facilitate this process via certain open source tools
  - *ThreadFix – Application vulnerability aggregation and management system*
- Enable defenders to block a vulnerable web page/application while developers are remediating source code

References

- App Sensor Project, Open Web Application Security Project
  - https://www.owasp.org/index.php/OWASP_AppSensor_Project
- From White Hat Website Security Statistics Report
  - https://www.whitehatsec.com/assets/WPstats_summer12_12th.pdf
- "How to Do Application Logging Right," Chuvakin, Anton, and Peterson, Gunnar
  - http://arctecgroup.net/pdf/howtoapplogging.pdf
- "Magic Quadrant for Security Information and Event Management," Garter Group, May 2012
  - http://www.gartner.com/technology/reprints.do?id=1-1AOG9W9&ct=120529&st=sb&elq=51f9879c322f4bc8b964591857bdafa1
- OWASP Logging Cheat Sheet https://www.owasp.org/index.php/Logging_Cheat_Sheet
- 2012 Verizon Breach Report

Contact Info

John Dickson, CISSP & Principal
(210) 572-4400
john@denimgroup.com

Twitter: @johnbdickson