Security in knowledge

APPLICATION SECURITY: ONE SIZE DOESN'T FIT ALL

Charles Henderson

Trustwave SpiderLabs

RSACONFERENCE2013

Session ID: SPO2-W25

Session Classification: Intermediate

AGENDA

- One size rarely fits all
- Sizing up an application
- Testing choices
- More than just testing
- Conclusions
- Q & A

ONE SIZE RARELY FITS ALL

- EJB for simple web app
- Using vi to develop an enterprise application
- Using a mainframe session for surfing the Internet
- Single server to run twitter
- Cracking passwords on an 80386

INFLEXIBLE APP-SEC SOLUTIONS

- Point and click web scanners
  - No ability to detect complex vulnerabilities
  - Low accuracy rate
- Purely automated source code review
  - Only support common languages
  - Low accuracy rate
- Fixed manual hour projects
  - Fails to account for complexity of application

SIZING UP AN APPLICATION

WAYS TO MEASURE

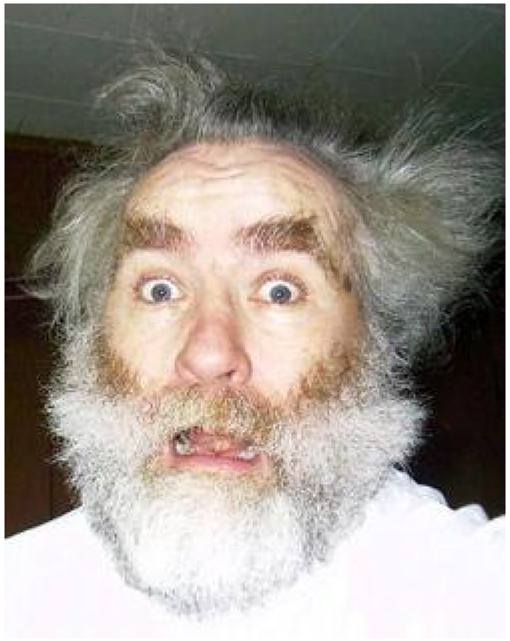
- Platform & framework
- Age of application
- Time since last test
- Level of maintenance
- Application criticality
- Skill of developers
- Goals of testing
- User base
- Internet accessibility

PLATFORM & FRAMEWORK

- Only applications using specific web-based technologies have extensive automated scanning tools
- Some Rich Internet Application (RIA) technologies are not easily scanned – Flash, Silverlight, etc.
- Legacy platforms and frameworks are more vulnerable to skilled attackers
- Embedded applications typically require a physical testing

AGE OF APPLICATION

- Older code tends to be more vulnerable, especially for web-based technology
- Older applications will often use vulnerable 3rd-party libraries and frameworks, or vulnerable versions of currently secure ones
- Older applications tend to have architectural mistakes that have since been recognized and fixed – passing parameters to command-line tools, building SQL with concatenated strings, for instance.



TIME SINCE LAST TEST

- Frequently tested applications may not need a comprehensive test if there are no major changes since last test
- Only incremental testing may be required, along with basic automated scanning

LEVEL OF MAINTENANCE

- Unmaintained code is typically at a high risk of compromise
- Original developers may no longer work for company
- Third-party components (libraries, controls, etc.) are unlikely to have been patched

APPLICATION CRITICALITY

- Sensitivity of data handled
  - Financial
  - Medical
  - PII
  - Business
- Revenue generated by application
- Disruption to business processes if application fails
- Compliance or regulatory mandates

SKILL OF DEVELOPERS

- Security staff usually have an idea about the skills of in-house developers
- Software vendors or open source projects that are popular may have a reputation about their security skills; niche products are usually a mystery
- Outsourced developers are usually a mystery unless there is a high degree of visibility into the provider. Provider training, turnover rates, and skill are often difficult to assess.

USER BASE & INTERNET ACCESS

- Applications used by only 10 trusted administrators may not need comprehensive testing beyond authentication enforcement
- Self-provisioned users that are offered extensive interfaces may pose a great threat
- Private applications may have a lower risk profile, especially if source IP filtering is being used

GOALS OF TESTING

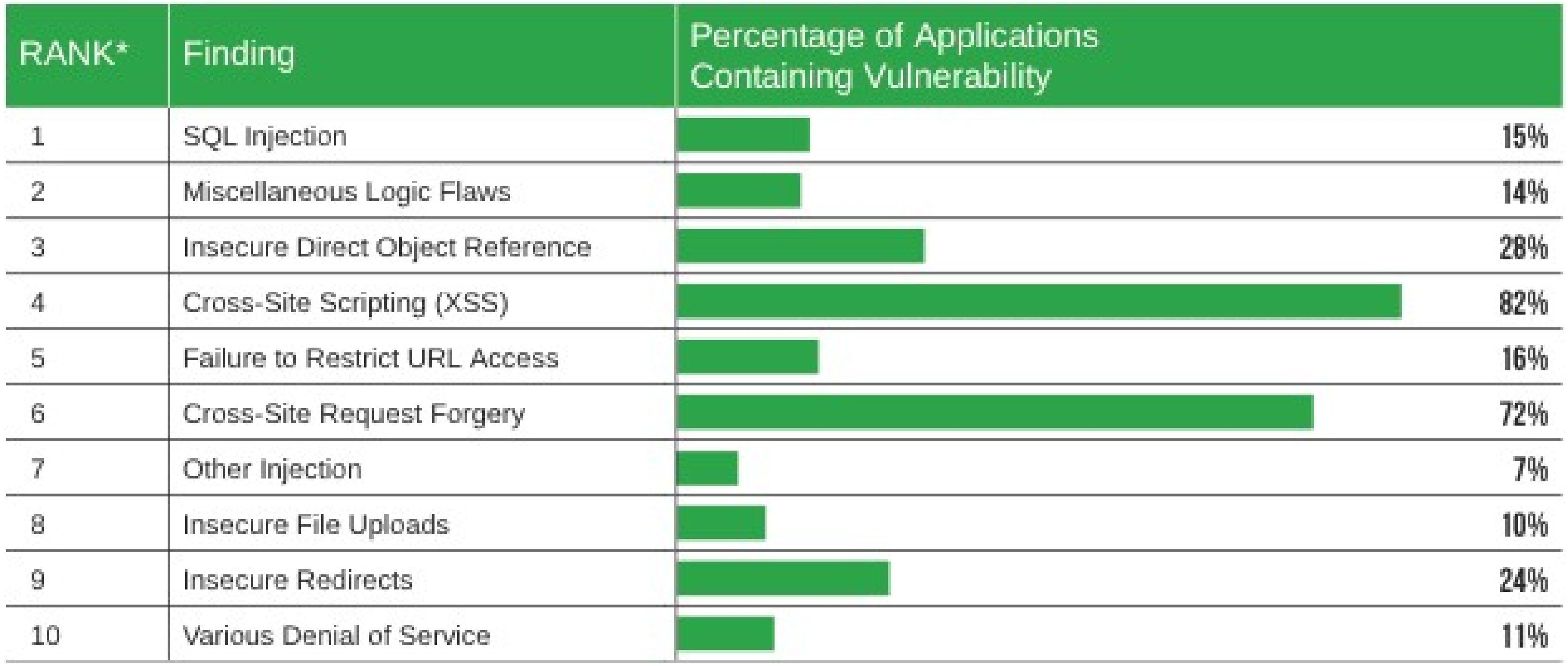
- Compliance mandate
- Internal policies
- Proactive security
- Post-breach security

TESTING CHOICES

TEST TYPES

- Fully automated scanning
- Manual pen testing
- Code review
- "Deep static analysis"

TOP 10 WEB APP VULNERABILITIES



RANK*	Finding	Percentage of Applications Containing Vulnerability
1	SQL Injection	15%
2	Miscellaneous Logic Flaws	14%
3	Insecure Direct Object Reference	28%
4	Cross-Site Scripting (XSS)	82%
5	Failure to Restrict URL Access	16%
6	Cross-Site Request Forgery	72%
7	Other Injection	7%
8	Insecure File Uploads	10%
9	Insecure Redirects	24%
10	Various Denial of Service	11%

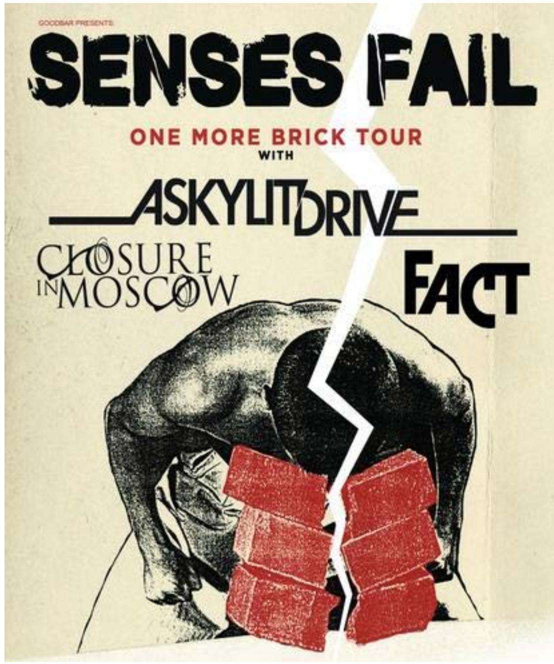
Source: 2013 Trustwave GSR

FULLY AUTOMATED SCANNING

- Appropriate only when accompanied with manual review of results
- Non-critical applications
- Relatively modern application & frameworks
- Only for web-based applications
- Impossible to discover complex vulnerabilities, including all logic flaws
- Will miss most authorization and authentication flaws

PRIVATE PERFORMANCES

- Online theater seat reservation system
- Put seats into a cart, then checkout later
- Once seats are in a cart, they are held so that seats are not overbooked
- Using multiple sessions
  1. Put the seats you want into a cart
  2. Put the remaining open seats into the second cart
  3. Complete the checkout of the first cart
  4. Never complete the checkout of the second cart.



SALAMI SLICING VARIANT

- Traditional Salami Slicing has been well known since at least the 1970's
- Office Space, Superman III...
- Stealing small amounts of money repeatedly can add up

SALAMI SLICING VARIANT

- From June 2007 to May 2008, Michael Largent obtained at least $60,000 from E-trade, Schwab.com, Google
- Brokerages will commonly deposit a few cents to confirm new bank accounts
- Largent programmatically opened thousands of accounts
- The transfers were legal, the phony checking accounts were not
- 11,385 Schwab accounts were opened as "Speed Apex" from only five AT&T IP addresses

MANUAL PENETRATION TESTING

- Quantity of manual testing important to establish
  - Size of application
    - Number of pages
    - Number of inputs
  - Complexity of functionality
    - Workflows
    - User groups / classifications
    - Permission levels
- Does not have "internal" visibility into application

CODE REVIEW

- Allows for thorough internal analysis of application
- Requires extensive development and security skills
- Can only be performed when source code available
- Better when combined with manual penetration testing

```
function removeBackdrop() {
  this.$backdrop.remove()
  this.$backdrop = null
}

function escape() {
  var that = this
  if (this.isShown && this.options.keyboard) {
    $(document).on('keyup.dismiss.modal', function ( e ) {
      e.which == 27 && that.hide()
    })
  } else if (!this.isShown) {
    $(document).off('keyup.dismiss.modal')
  }
}
```

"DEEP STATIC ANALYSIS"

- Effective when source code is not available
- Allows for internal analysis of application internals
- Best when combined with manual penetration testing
- Enterprise-wide use of a single library or framework could be a single point of failure

ACTUAL FRAMEWORK FLAWS

- Apache MyFaces
  - Insecure use of client-side view state
- Sun Mojarra
  - Insecure use of client-side view state
- Oracle Application Framework
  - Profoundly flawed diagnostic mode
- Unnamed game engine
  - Systemic buffer overflows

MORE THAN JUST TESTING

Security in knowledge

AFTER THE TEST

- Testing only defines the risk; does nothing to solve it
- Remediation options:
  - For web applications, virtual patching is an option
  - Risk reduction by restrictive firewall rules, bolt-on IPSec
  - Third-party assistance
  - Deferment
- Lessons learned
  - Developer training is critical
  - Examples from their own apps are ideal for developers
  - Learning how to "hack" an application can get devs excited about security

TACTICS VS. STRATEGY

- Application security programs are more than single tests strung together
  - Inventory applications
  - Prioritization of applications
  - Clear SDLC mandates
  - Application vendor review
- Commonly used frameworks and libraries must receive extensive attention, regardless of their source
- May vary testing on application; one year pen test, next year code review

WRAPPING UP

Security in knowledge

RSA CONFERENCE 2013

CONCLUSIONS

- Unless you are only securing one application, AppSec requires a diverse arsenal
- Multiple factors must be considered when determining what approach is appropriate for a specific target
- Budgets are not unlimited and it is important to make dollars spent count
- Understanding what you have is the first step to understanding what you need to protect

QUESTIONS & ANSWERS