Static and Dynamic Analysis for PHP Security

V. C. Sreedhar
IBM  TJ Watson Research Center
vugranam at us.ibm.com
Outline

- Security Issues
- Static Analysis
- Dynamic Analysis
- Future Challenges
Objective of the Talk

- Expose deep, fundamental, and state-of-the-art research and technology in static and dynamic analysis.
- Most existing tools for PHP only do shallow and structural analysis.
Our Security Goals

- Static and dynamic analysis of PHP scripts to detect vulnerabilities
- Best practices and coding guidelines for secure PHP and Web Applications
- Language extensions to improve security features of PHP
What is Security?

- Security is a capability that satisfy three classes of requirements
  - Confidentiality: Assets/Artifacts are accessed only according to well-defined policies.
  - Integrity: Assets/Artifacts are not undetectably corrupted, and altered only according to well-defined policies.
  - Availability: Assets/Artifacts are available when they are needed.
Goal of Security

- The goal of Security is the protection of assets/artifacts against threats to confidentiality, integrity, or availability, using appropriate systems or infrastructure, tools, methodologies, and processes.
Threats, Vulnerability, Attack, and Flaw

- A threat is an expression of an intention to inflict pain, injury, evil, or punishment.
- A vulnerability, is a means whereby a hostile entity can successfully violate a system’s security.
  - For example, a web application might be vulnerable to a “poisoned cookie” (a maliciously altered cookie, which the web app will trust without verification).
Threats, Vulnerability, Attack, and Flaw

- An “attack” refers to the tool or technique with which an attacker will attempt to detect and exploit a vulnerability.
- A flaw is a defect in a system which can result in a security violation.
  - Every vulnerability must be due to at least one flaw, but it is possible for a flaw not to cause any vulnerabilities
  - E.g., the flaw might be masked
Common Criteria Evaluation

- How many of you have heard of CCE?
- How many of you follow the CCE process?
- What is your Evaluation Assurance Level?
  - EAL1 to EAL7
Security Model According to Common Criteria

Owners → countermeasures

Impose that may possess
May be aware of

Threat agents → threats

Give rise to
Wish to abuse and/or may damage

Countermeasures → vulnerabilities

That may be reduced by
That increase

Vulnerabilities → risk

Leading to

Risk → assets

Wish to minimize

Value
Application Security

92% of reported vulnerabilities are in applications, not networks

SOURCE: NIST
Security Engineering

- Security Principle
  - One cannot just look at one software artifact and declare that the software is secure.

- Security Engineering is all about considering security across all phases of the software life cycle.
Secure Software Lifecycle

Iterative approach

- Security Models & Requirements
- Threat Model & Risk analysis
- Security Design Review
- Security Code Analysis
- Security Penetration Testing
- Requirements and use cases
- Design
- Code
- Testing
- Deployment and Field

Security Mechanism and Procedures

1. User Authentication
2. User Profile
3. Business Service
5. Application Programs
6. Data Center
7. Logs
8. Information Flow Controls
9. Application Library
10. Data Models and Management
11. Application Deployer
12. Application Developer
13. Auditor

Users

Security Manager

© 2006 IBM Corporation
Security Models

- Multilevel Security (MLS)
  - Bell-LaPadula
  - Biba

- Clark-Wilson Model
  - Chinese-Wall Model

- Role-based Model

- MAC/DAC
Information Flow

- High security information should not flow through low security ``channels''.
- A low security control should not influence the outcome of high security output.

- Using the term “channel” in a generic way

- E.g. Lowx = Highx // bad assignment
- If(LowX) then HighX // bad condition.

- In general, Policy certified information must not be leaked through channels that do not satisfy the policies
- Based on Multi-Level Security (Bell-LaPadula) Model
Reading and Writing Information

- Information flows up, not down
  - “Reads up” disallowed, “reads down” allowed
  - Sometimes called “no reads up” rule

- Information flows up, not down
  - “Writes up” allowed, “writes down” disallowed
  - Sometimes called “no writes down” rule
Implicit Flow of Information

- Information flows from \( x \) to \( y \) without an explicit assignment of the form \( y := f(x) \)
  - \( f(x) \) an arithmetic expression with variable \( x \)
- Example from previous slide:
  - if \( x = 1 \) then \( y := 0 \)
  - else \( y := 1; \)
- So must look for implicit flows of information to analyze program
Security Analysis

- Information Leak
- Tainted variables
- Permission programs
- Inserting Security Hooks and Sanity Checks
- Complete mediation
- Consistent Role Assignment
- Escape Analysis (generalized escape analysis)
- Security races and deadlocks
- Intrusion Detection (part of) using dynamic info flow
- Exploit analysis
- Confinement Analysis
- Covert Channel (part of) analysis
Figure 2: CERT Vulnerability Taxonomy (subset)

1. CERT is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
A Few Bad Ones

- Input/Output data not validated
  - Tainted Variables
  - Cross Site Scripting
  - SQL Injection Flaws

- Buffer Overflows

- Improper Error Handling

- Insecure Storage

- Formatting Errors

- Complete Mediation

- Insecure Default Configuration
Static and Dynamic Analysis

- Static analysis is a process for determining the relevant properties of a (PHP) program without actually executing the program.

- Dynamic analysis is a process for determining the relevant properties of a program by monitoring/observing the execution states of one or more runs/executions of the program.
Static Analysis

- At each program point (statement boundary) in a (PHP) program, determine properties or relations that may hold at that point during execution.
  - These properties or relations are abstract representation of the properties or relations that are true during some execution of the program.
  - E.g., uninitialized variable, whether two variables alias or not, etc.
  - Useful for finding security vulnerability
An Example: Register Global

```php
<?php
    if (CheckIAmGod($user)) {
        $god = true;
    }
    if ($god) {
        include '/bless/you/child.php';
    }
?>

Let us query ?god=1&user=satan
An Example

```php
<?php
$god = 1;
if (CheckIAmGod($user)) {
    $god = true;
}
if ($god) {
    include '/bless/you/child.php';
}
?>
```

Nothing Wrong with Register Global!
The problem is with the developer.
Static Analysis

```php
<?php
    $god=uninitialized
    if (CheckIAmGod($user)) {
        $god = true; $god=initialized
    } $god=initialized AND $god=initialized
    if ($god) { $god= uninitialized
        include '/bless/you/child.php';
    }
?>
```
Why Static Analysis?

- Manual code inspection is necessary but not sufficient
- Static Analysis helps developers and tester to find bugs
- Static analysis often has many false positive
- Some deep static analysis can help reduce the number of false positives.
- Combining static and dynamic analysis, along with testing and manual inspection is the best bet
Analysis Pandora Box

- Pessimistic Analysis
  - Typically slower and conservative
- Optimistic Analysis
  - Typically faster and more aggressive (and sometimes unsound)
- Sound Analysis
- Unsound Analysis
  - Useless for code generation, but useful for error analysis (bug finding)
  - Very fast
  - High false positives. Use other techniques to filter many false positives
- Static versus Dynamic analysis (and hybrid)
- Demand-driven, Incremental, and Exhaustive
- Monotonicity, Non-monotonicity, and Extensible
  - Monotonic/Extensible guarantees convergence
  - Non-monotonic requires a different approach
- Context/flow/path sensitive/insensitive analysis
- Type-based and type-state analysis
- Partial evaluation and semantic-based analysis
- ...

Threat Modeling and Abstraction

- To secure your application you must understand threats from your attackers’ point of view.

- Threat modeling is a process of assessing and documenting potential risks created by an application.

- Threat modeling is a necessary step before you can design your static and dynamic analysis for security.

- A flaw may be due to insufficient threat modeling.

- Threat modeling is a serious business and has to be done very early in the lifecycle of the application development.
<?php
    $uname = '';  
    if (isset($_GET['uname'])) {
        $uname = $_GET['username'];
    }
    echo $uname;
?>

In threat modeling we have to think like an attacker

Suppose $uname =
<script>alert('Hello World!');</script>
Filtering or Sanitizing

- Most attacks can be mitigated by filtering and sanitizing inputs and outputs.
- An unsanitized input or output is said to be tainted.
- A variable that contains unsanitized data is said to tainted variable.
- Given a PHP program, how can we detect variables and data that are tainted?
Taint Analysis

- Taint analysis essentially consists of determining variables and data that have not been sanitized
- Taint analysis relies on other kinds of analysis such as Alias Analysis, Data Dependence analysis, and Slicing
Alias Analysis

- In PHP whenever you use address reference operator & you are essentially creating an alias

```php
<?php
    $a = &$b  // $a and $b are aliases
    $bar = new fooclass();
    // $bar points to the new fooclass object
?>
```
Alias Analysis

```php
<?php
    $start= 0;
    $index =& $start;
    foreach (array(1,2,3) as $index) {
        ....
    }
    echo $start; // Can you guess the value of $start?
?>
```
Alias Analysis and Taint Analysis

```php
<?php
    $a = &$b;  // $a and $b are aliases
    echo $a ; // XSS
?>
```

If $b is tainted then $a is also tainted
Alias Analysis

- Precision versus scalability

- Scalability is the ability to deal with large programs >1 Million Lines of Code

- Not many MLOC in PHP
Alias Analysis

- Many different kinds of analysis techniques
  - Flow sensitive versus Flow insensitive
  - Context sensitive versus context insensitive

- Flow sensitive and context sensitive analysis is more precise, but very expensive

- Analysis done over call graph and control flow graph representations
Deep Analysis

- There are many different kinds of “deep analysis”
  - An example of a deep analysis is typestate analysis
  - Very expensive and does not scale well

- But can be used to find some nifty errors
  - Especially in OO and protocol related bugs

- Unfortunately there is a complex interaction between typestate analysis and alias analysis
Typestates

- Strom and Yemini from IBM introduced the concept of typestate as an extension to the notion of a type. It requires that a variable be in certain state before operations on the variables can be performed.
- In OO programs, a method that is invoked on an object \textit{o} typically has a partial view of the object \textit{o}. One can use typestates to define a consistent view of an object prior to an invocation of a method on the object.
- Very useful for finding flow-sensitive bugs.
Typestate Example

read()

close()

create()

open()

write()
Typestate Analysis

- Typestate and alias analysis interaction

```php
$x = new File() ; $y = new File() ;
$z = &$y ;
if($blah) {
    $y->close() ;
    $z = &$x ;
}
$z -> read() ; // is this ok?
```
Our Analysis Framework

- Based on an IBM tool called CAPA/DOMO
- Common architecture for static & dynamic program analysis technology
  - quickly create software lifecycle applications through composition.
  - foster sharing and collaboration between disparate research groups across the world.
  - speed technology transfer to our product groups.
Static Analysis for PHP

- PHP Scripts
- Structural Analysis
- Type Analysis
- Security Vulnerability Analysis
- Data Flow Analysis
- Control Flow Analysis
Typestate Analysis Tool

- Based on an IBM tool called SAFE
  - Currently for Java
  - Recognizes patterns and anti-patterns
PHP Static Analysis Tool

- PHP Program
  - PHP Parser
  - Class Hierarchy Analysis
  - Call Graph Construction

- User Spec (Query Language)
- Security Pattern database
- Query engine (XPATH/XQUERY)
- Error Report (XML)

- DOMO IR 2XMLModel
- Structural Information (XML)

- CAPA/DOMO
SWORD4J

- Yet another IBM tool for J2SE security
  - IDE based on Eclipse

- SWORD4J statically analyzes Java bytecode and detects:
  - Permission requirements
  - Recommended privileged code locations
  - Mutability and accessibility violations

- Plan to leverage SWORD4J for PHP
Dynamic Analysis

- Runtime detection of flaws and vulnerability
- How static analysis can help dynamic instrumentation and monitoring?
- Context Sensitive String Evaluation (CSSE)
CSSE Approach

- CSSE automatically applies the appropriate checks for syntactic content in user-provided input
- Ability to distinguish between user- and developer-provided parts
- Metadata assignment to user-provided input
  - Determine the appropriate checks on the user-provided parts
  - Context-sensitive string evaluation
CSSE Approach

- Metadata describes which string fragments are user-provided and which developer-provided
  - All user-provided input is untrusted
- User input can be
  - network input: e.g., HTTP headers
  - environment variables
  - stored input: e.g., db, XML
Dynamic Analysis: CSSE

Network Input:
GET, POST, cookie
$email = "alice@host";
$pincode = "1234 or 1=1";

Direct Input:
arguments, env, ...

Stored Input:
DB, XML, CSV, ...

Inputs
Metadata Assignment

Constants
SELECT * FROM users WHERE email=''
AND pincode=/usr/bin/mail

Textual Representations
SELECT * FROM users WHERE email='alice@host'
AND pincode=1234 or 1=1
/usr/bin/mail alice@host

Metadata-Preserving
String Operations

Metadata-Preserving
String Evaluation

Outputs
Execute:
shell, XSLT, ...

Query:
SQL, XPath, ...

Locate:
URL, path, ...

Render:
HTML, SVG, ...

Store:
DB, XML, ...

Static and Dynamic Analysis for PHP Security © 2006 IBM Corporation
**PHP String Analysis**

- PHP string analyzer is a static analyzer that checks the sanity of a PHP string using a context-free grammar.
  - Useful for detecting security errors, including flagging programmer sloppiness.
- [http://www.score.is.tsukuba.ac.jp/~minamide/phpsa/](http://www.score.is.tsukuba.ac.jp/~minamide/phpsa/)

- We are extending this to handle more complicated cases, including JavaScript, XML and Web Services strings.
Combining Static and Dynamic Analysis

- Use static analysis for finding potential properties and program points that you want to track at run-time.
AJAX and PHP Security

- Focus for next year
PHP and AJAX

- Security can become more challenging, especially with its rising popularity.
- Especially with support for Mashups.
  - A **mashup** application uses content from more than one source to create a completely new service.
  - Rich/Fat clients present more challenges for security.
Enterprise Mashups: An Industry Case Study

*PHP meets Web.20*

...a web of data sources, services for exploring & manipulating data, and ways that (end) users can connect them together *Instantly*

Tom Coates/Yahoo

New York PHP Conference & Expo 2006
THE PHP BUSINESS COMMUNITY

Rod Smith
VP Internet Emerging Technology, IBM

June 2006
IBM Web 2.0 Technologies Current Focus Areas

Enterprise Mashups: An Industry Case Study

- Web 2.0 Technologies converging on a few key value proposition
- Broad Collaboration
- Simplicity & rich(er) internet experiences
- Remixability
  - Enabling “applications” that can be created by non-professional programmers
  - APIs based on open (defacto) standards

PHP On Forefront of Opportunities
- It’s about instant results
- It’s about empowering line-of-business professionals
Web 2.0 - What technologies are we talking about?
Enterprise Mashups: An Industry Case Study

- Many Web 2.0 Technologies still in innovation stage

- Customer Interest High In:
  - AJAX is most tangible in terms of potential business value
  - RSS/Atom - RSS & Atom/APP being seen as potential approaches to simplify specific content centric application architectures
  - Programmable Web - potential seen in building/Extending business ecosystems
  - Web 2.0 “instant” applications
Future Effort

- Configuration analysis

- Feedback Analysis
  - Combining Static and Dynamic Analysis

- Concurrency and Security
Acknowledgement

- Dana Glasner
- Gabriela Cretu
- Ted Habeck
- Julian Dolby
- Larry Koved
- Wietse Venema
- Chris Vanden Berghe,
- Kouichi Ono
- David Boloker
- ...
Thank You!

For more information:
V. C. Sreedhar,
vugranam@us.ibm.com