Imperial College London

PHP ASPIS: USING PARTIAL TAINT TRACKING TO PROTECT AGAINST INJECTION ATTACKS

Ioannis Papagiannis, Matteo Miglivacca, Peter Pietzuch

Department of Computing, Imperial College London
{ip108, migliava, prp}@doc.ic.ac.uk

OVERVIEW

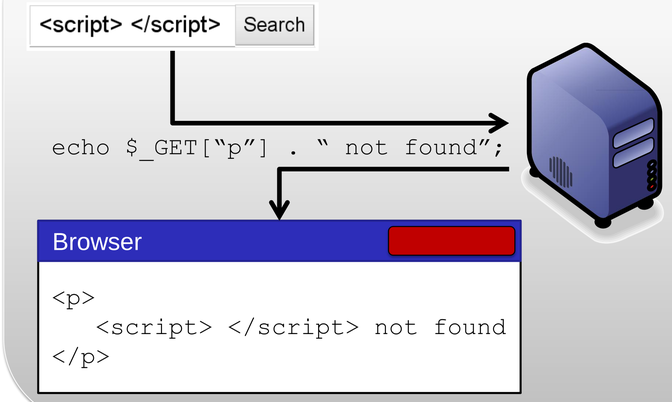
Web applications are increasingly popular victims of security attacks. Injection attacks, such as Cross Site Scripting (XSS) or SQL Injection, (SQLI), are a persistent problem.

PHP Aspis

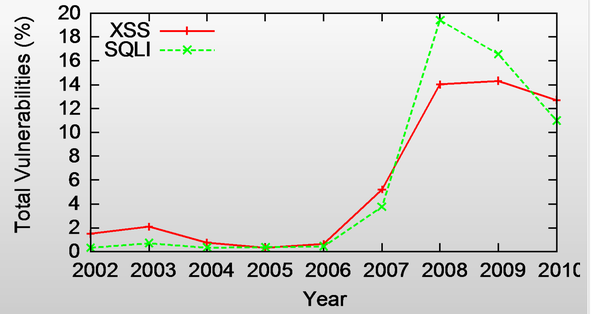
1. applies taint tracking at the language level using **source code transformations**.
2. augments values with taint meta-data to track their origin and uses such meta-data to detect injection attacks as they occur.
3. carries out **partial taint propagation** only in an application's most vulnerable parts.

We evaluate PHP Aspis with Wordpress, a popular open source weblog platform, and show that it prevents all code injection exploits that were found in Wordpress plugins in 2010.

XSS ATTACK EXAMPLE



POPULARITY



XSS and SQLI across all CVE vulnerabilities

PARTIAL TAINT TRACKING

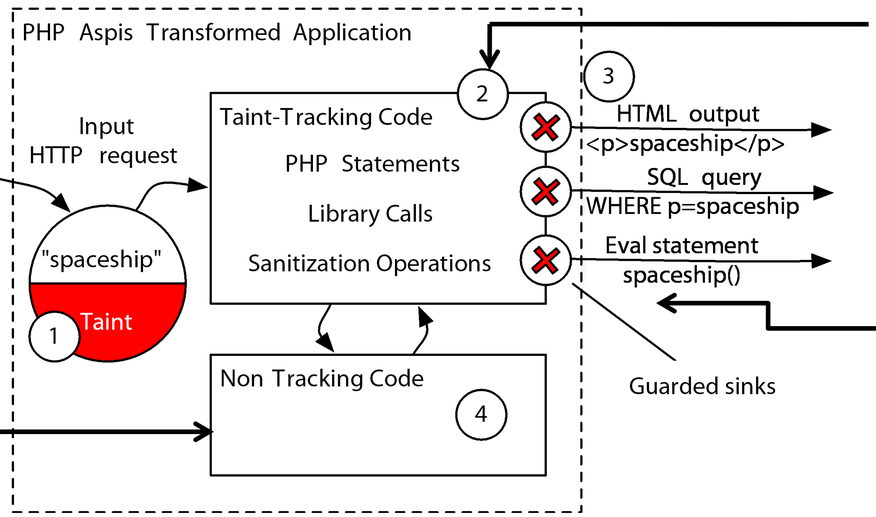
The application is separated in two parts: **tracking** and **non tracking** code.

1. Data entering the application (e.g. HTTP request) are augmented with **taint meta-data**.

Original Code	Code transformed to apply taint tracking
`$_GET["p"] = "spaceship";`	`$_GET["p"] = array( "spaceship", 0=>true );`

4. Non tracking code is transformed to **operate in presence of tracking code.**

Original Code	Code transformed to apply taint tracking
`say_hello( "hey " );`	`say_hello( array( "hey ", 0=>false ) );`



2. PHP source code marked as tracking is transformed to **propagate taint meta-data** (e.g. when strings are concatenated)

Original Code	Code transformed to apply taint tracking
`$reply = $hi . $_GET("p");`	`$reply = AspisConcat( $hi, $_GET("p") ) ;`

3. Operations that can be used in an injection attack (e.g. `echo()`) are **inspected for control characters** that originate from the user. The operation may optionally be aborted.

Original Code	Code transformed to apply taint tracking
`echo( $reply );`	`echo( AspisAntiXSS( $reply ) );`

SECURING WORDPRESS PLUGINS

We use PHP Aspis to only secure Wordpress **plugins** because:

1. Wordpress plugins are responsible for **20 out of the total 24** injection vulnerabilities reported for the Wordpress platform in 2009 and 2010.
2. Focusing taint tracking in a small part of a web application provides a **significant speedup** compared to tracking taint everywhere in the web application.

Our results show that partial taint tracking focused in plugin code , when combined with appropriate **sanitisation/filtering functions**, successfully prevents all Wordpress plugin exploits reported in 2010.

RESULTS

App	Taint Tracking	Page generation	Penalty
Prime	Off	44.9 ms	-
Prime	On	466.8 ms	10.4x
DB	Off	0.4 ms	-
DB	On	1.3 ms	3.4x
Wordpress	Off	65.6 ms	-
Wordpress	On	394.4 ms	6.0x
Wordpress	Partial	144.3 ms	2.2x

CVE 2010-#	Type	Plugin	Prevented
4637	XSS	FeedList 2.61.01	Yes
4630	XSS	WP Survey And Quiz Tool1.2.1	Yes
4518	XSS	Safe Search 0.7	Yes
4402	XSS	Register Plus 3.5.1	Yes
4277	XSS	Embedded Video 4.1	Yes
3977	XSS	cforms 11.5	Yes
2924	SQLI	myLinksDump 1.2	Yes
1186	XSS	NextGEN Gallery 1.5.1	Yes
0673	SQLI	Copperleaf Photolog 0.16	Not Available