SAPHIRE: Sandboxing PHP Applications with Tailored System Call Allowlists

Alexander Bulekov  
Rasoul Jahanshahi  
Manuel Egele  
Boston University  
{alxndr, rasoulj, megele}@bu.edu
PHP is still the language of choice for web-servers...

**WordPress’ market share is 40% of all websites**

The web is powered in large part by scripting languages such as PHP
Remote Code Execution vulnerabilities are a major threat to web-apps

- RCE was the most common type of vulnerability identified in 2019 for Content Management Systems.
- A single RCE for a CMS, can lead to vast numbers of compromised hosts.

We're temporarily increasing our payouts for WordPress RCEs to $300,000 per exploit (usually $100K).

The exploit must work with latest WordPress, default install, no third-party plugins, no auth, no user interaction!

If you have this gem, contact us:
submit.zerodium.com/

Zerodium @Zerodium
Apr 9

ars

Hackers exploit critical vulnerability found in ~100,000 WordPress sites
ars.technica.com/informational/

HACKED

KashmirBlack botnet behind attacks on CMSs like WordPress, Joomla, Drupal, others | ZDNet
zdnets.com/article/kashmirblack...

KashmirBlack botnet behind attacks on CMSs like WordPress, Joomla, Drupal, others | ZDNet
26 Oct 2020

New KashmirBlack botnet is believed to have infected hundreds of thousands of websites since November 2019.

zdnets.com
Syscall-level RCE mitigations have gained widespread adoption for native software

- **seccomp** – Allows a Linux process to relinquish future access to system-calls that it does not need. Featured in Chromium, Docker, QEMU, Firefox, TOR…
- pledge() used to filter syscalls for ~70% of OpenBSD packages.
- Difficulties applying syscall-filters to interpreted PHP scripts.
  - We cannot filter syscalls by only looking at the script.
  - We cannot filter syscalls by only looking at the PHP interpreter
Mapping the Interpreter

The interpreter provides a set of APIs that scripts use to access system resources.

1. SAPHIRE performs a static analysis over the interpreter and its libraries to infer the system-calls required by each API.

2. We refine this mapping by running the PHP interpreter’s test-suite and observing which API caused each syscall to be invoked.

Result: PHP APIs → syscalls
② Mapping the PHP Application

For each PHP script, we identify the PHP APIs mentioned in the code, and the APIs used by the additional scripts that it includes.

1. **SAPHIRE** identifies the PHP API functions called within the code of each script.
2. **SAPHIRE** resolves dependencies between PHP scripts due to includes and classes.

**Result:** PHP Scripts → PHP APIs
Filtering Syscalls for a live web-app

Combine the mapping from Stage ①

SAPHIRE uses a PHP extension to apply the syscall filters to each PHP process, before it executes a script.

After the filter is applied, the OS handles the actual enforcement.

Result: Protection for the web-server
Evaluation Dataset

- Six popular PHP web applications
- 9 vulnerable WordPress plugins
- Chose vulnerabilities based on the 22 most-recently released exploit PoCs
On average SAPHIRE, filters 80% of dangerous* system-calls

* (Bernaschi et al. 2002):

  chmod, fchmod chown, fchown, lchown, execve, mount, rename, open, link, symlink, unlink, setuid, setresuid, setsuid, seteuid, setgroups, setgid, setfsgid, setresgid, setregid, create_module
SAPHIRE’s fully-automated approach blocks real-world exploits

<table>
<thead>
<tr>
<th>Application</th>
<th>Vulnerability</th>
<th>Exploits Blocked</th>
<th>False Positives</th>
<th>Dangerous System Calls Available to Exploits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drupal 7.0</td>
<td>CVE-2014-3704</td>
<td>y</td>
<td>0</td>
<td>openat, unlink</td>
</tr>
<tr>
<td>Drupal 7.5</td>
<td>drupal_restws_exec</td>
<td>y</td>
<td>0</td>
<td>chmod, openat, rename, symlink, unlink</td>
</tr>
<tr>
<td>Drupal 7.26</td>
<td>CVE-2014-3453</td>
<td>y</td>
<td>0</td>
<td>chmod, openat, rename, symlink, unlink</td>
</tr>
<tr>
<td>Drupal 7.57</td>
<td>CVE-2018-7600</td>
<td>y</td>
<td>0</td>
<td>chmod, openat, rename, symlink, unlink</td>
</tr>
<tr>
<td>Drupal 7.58</td>
<td>CVE-2018-7602</td>
<td>y</td>
<td>0</td>
<td>openat, unlink</td>
</tr>
<tr>
<td>Joomla 2.5.25</td>
<td>CVE-2014-7228</td>
<td>y</td>
<td>2</td>
<td>chmod, openat, rename, unlink</td>
</tr>
<tr>
<td>Joomla 3.7</td>
<td>CVE-2017-8917</td>
<td>y</td>
<td>1</td>
<td>chmod, openat, rename, unlink</td>
</tr>
<tr>
<td>Magento 2.0.5</td>
<td>CVE-2016-4010</td>
<td>y</td>
<td>0</td>
<td>chmod</td>
</tr>
<tr>
<td>Moodle 3.4</td>
<td>CVE-2013-3630</td>
<td>y</td>
<td>0</td>
<td>chmod, openat, rename, symlink, unlink</td>
</tr>
<tr>
<td>phpMyAdmin 3.3.10</td>
<td>CVE-2011-4107</td>
<td>y</td>
<td>0</td>
<td>chmod, openat, rename, symlink, unlink</td>
</tr>
<tr>
<td>phpMyAdmin 4.8.1</td>
<td>CVE-2018-12613</td>
<td>y</td>
<td>0</td>
<td>chmod, openat, rename, unlink</td>
</tr>
<tr>
<td>Wordpress 4.6</td>
<td>11 Vulnerabilities</td>
<td>y</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Exploits have no access to most dangerous system-calls
### Overhead

**SAPHIRE’s PHP extension is a lightweight wrapper around seccomp()**

<table>
<thead>
<tr>
<th>Concurrency</th>
<th>Wordpres</th>
<th>Protected</th>
<th>Default</th>
<th>Protected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default</td>
<td>(0.16%)</td>
<td>Default</td>
<td>(0.00%)</td>
</tr>
<tr>
<td>1</td>
<td>328.252</td>
<td>328.78</td>
<td>338.75</td>
<td>337.42</td>
</tr>
<tr>
<td>2</td>
<td>353.982</td>
<td>355.776</td>
<td>368.84</td>
<td>370.02</td>
</tr>
<tr>
<td>4</td>
<td>348.242</td>
<td>348.639</td>
<td>369.48</td>
<td>369.55</td>
</tr>
<tr>
<td>8</td>
<td>361.377</td>
<td>363.83</td>
<td>372.84</td>
<td>372.98</td>
</tr>
<tr>
<td>16</td>
<td>416.639</td>
<td>419.342</td>
<td>412.47</td>
<td>414.42</td>
</tr>
<tr>
<td>32</td>
<td>863.932</td>
<td>867.932</td>
<td>872.57</td>
<td>877.24</td>
</tr>
</tbody>
</table>

In our prototype implementation, SAPHIRE incurs less than 2% overhead for each request.
The syscall filtering mechanisms that have gained widespread adoption among critical compiled applications are also **applicable, and effective** for protecting script-based applications

**SAPHIRE:**
Fully automated
Protects real web-apps against real-world exploits
Few false-positives, even in the most restrictive configuration.
Built on mature filtering primitives – incurs low overhead.

[https://github.com/BUseclab/saphire](https://github.com/BUseclab/saphire)
Questions?

SAPHIRE: Sandboxing PHP Applications with Tailored System Call Allowlists

Alexander Bulekov     Rasoul Jahanshahi

Manuel Egele

Boston University

{alxndr, rasoulj, megele}@bu.edu