Backporting Security Patches of Web Applications:

A Prototype Design and Implementation on Injection Vulnerability Patches

Youkun Shi[1], Yuan Zhang[1], Tianhan Luo[1], Xiangyu Mao[1], Yinzhi Cao[2], Ziwen Wang[1], Yudi Zhao[1], Zongan Huang[1], Min Yang[1]

Fudan University[1]

Johns Hopkins University[2]

Unpatched Web Applications

A large number of websites are still running old vulnerable applications [1,2]

Critical Drupal Core Vulnerability: What You Need to Know

Josef Weiss | Cyber Exposure Alerts
March 29, 2018 | 2 Min Read

Drupal is popular, free and open-source content management software. On March 28, the Drupal security team released patches for CVE-2018-7600, an unauthenticated remote code execution vulnerability in Drupal core. The vulnerability affects Drupal versions 6, 7 and 8. Patches have been released for versions 7.x, 8.3.x, 8.4.x and 8.5.x.

CVE-2018-7600, a high-risk unauthenticated RCE vulnerability in Drupal core

Bad Packets ✓
@bad_packets · **Follow**

I've shared the list of 115,070 vulnerable Drupal sites with @USCERT_gov and @drupalsecurity. Due to the highly critical risk of CVE-2018-7600 being exploited, the list won't be shared publicly.

4:51 PM · Jun 5, 2018

After three months since the patch release, there are still about 115,000 unpatched websites

[1] https://threatpost.com/drupalgeddon-2-0-still-haunting-115k-sites/132518/
[2] https://www.tenable.com/blog/critical-drupal-core-vulnerability-what-you-need-to-know

Vulnerability Patching Practice

1. Patch command
   - Directly applies the official patch (for a specific version) to a vulnerable version
   - Limitations: Highly susceptible to code conflicts
     - In our dataset, 1,049/1,526 target versions report code conflicts when applying the patches
2. Auto-upgrade APIs
   - Uses auto-upgrade APIs provided by (some) web applications
   - Limitations: Requires significant developer efforts and is prone to compatibility issues
     - In our dataset, 624 / 1,526 target versions **do not have** auto-upgrade APIs

       863 / 1,526 target versions report compatibility issues

Running Example

- OpenEMR 5.0.0.5 and 5.0.0.6 are affected by CVE-2018-10572
  1. OpenEMR doesn't provide auto-upgrades API
  2. Directly apply patch command will fail on the old version due to code conflicts

```
1    <?php
2  + require_once("../globals.php");  // patch modification
3  + require_once($GLOBALS['srcdir'] . "/patient.inc"); // patch modification
4
5    use OpenEMR\Core\Header; // the anchor for patch modification location
6
7  - include_once("../globals.php"); // patch modification
8  - include_once($GLOBALS['srcdir'] . "/patient.inc"); // patch modification
9    $template_dir = $GLOBALS['OE_SITE_DIR'] . "/letter_templates";
10   ...
11 - $fh = fopen("$template_dir/".$_GET['template'], 'r'); // patch modification
12 + $fh = fopen("$template_dir/" .  // patch modification
                convert_safe_file_dir_name($_GET['template']), 'r');
13   ...
14   ?>
```

Official Patch for CVE-2018-10572 on **OpenEMR 5.0.0.6**

```
1    <?php
2    $sanitize_all_escapes  = true; // the anchor changes
3    $fake_register_globals = false; // the anchor changes
4
5    include_once("../globals.php");
6    include_once($GLOBALS['srcdir'] . "/patient.inc");
7    $template_dir = $GLOBALS['OE_SITE_DIR'] . "/letter_templates";
8    ...
9    $fh = fopen("$template_dir/".$_GET['template'], 'r');
10   ...
11   ?>
```

Code Snippet of **OpenEMR 5.0.0.5**

Code conflicts hinder patch apply!

Patch Backporting

- **Problem Definition:** Given a patch for a vulnerable version, backport the patch to fix the same vulnerability on another vulnerable version.
- **Challenges:** How to automatically backport patches to old versions with guaranteed compatibility and security?
  - Can the patch be compatible (not affecting normal functionality) with another vulnerable version?
  - Can the patch fix the vulnerability on another vulnerable version?
  - Can the patch be automatically applied to another vulnerable version?

Problem Understanding

- Three Mismatches among <vulnerability, patch, target>
  - <Patch, Vulnerability> Mismatch → break compatibility
    - The patch may contain vulnerability-irrelevant modifications, which may affect the functionalities of a web application.
  - <Target, Vulnerability> Mismatch → break security
    - The target version may have a different vulnerability logic to the one that the patch aims to fix, thus requiring a new patching logic.
  - <Patch, Target> Mismatch → break automation
    - The patch may not be easily applied to a target version due to cross-version code location changes

This Work: Patch Backporting

- **Scope**: Injection-based vulnerability patches
  - Key Insight: injection-based vulnerabilities are fixed by restricting the capability of the sink function
  - Sink Capability: all the user inputs that can go to the sink functions
- **Key Idea**: backport the safe sink capability across different vulnerable versions
  - Safely Backportable Patch (SBP): *the patch only restricts the capability of the sink function*
    - Filter our irrelevant patch modifications: ✗ ~~<Patch, Vulnerability> Mismatch~~ & ✓ Compatibility
  - Safely Backportable Version (SBV): *the target has the same sink capability as the pre-patch version*
    - Select only a part of backportable versions: ✗ ~~<Target, Vulnerability> Mismatch~~ & ✓ Security
  - Deploy SBP upon on SBV: *replace the vulnerable sink with the safe sink*
    - Only requires minimal source code modifications: ✗ ~~<Patch, Target> Mismatch~~ & ✓ Automation

Running Example: Patch Backporting

Official Patch

```
1   <?php
2  + require_once("../globals.php");
3  + require_once($GLOBALS['srcdir'] . "/patient.inc");
4    use OpenEMR\Core\Header;
5    include_once("../globals.php");
6    include_once($GLOBALS['srcdir'] . "/patient.inc");
7    $template_dir = $GLOBALS['OE_SITE_DIR'] . "/letter_templates";
8    ...
9  -  $fh = fopen("$template_dir/".$_GET['template'], 'r');
10 + $fh = fopen("$template_dir/" .
                  convert_safe_file_dir_name($_GET['template']), 'r');
11   ...
12   ?>
```

Official Patch for CVE-2018-10572 on OpenEMR 5.0.0.6

convert

SBP

```
1   <?php
2   function convert_safe_file_dir_name($label){
3      return preg_replace('/[^A-Za-z0-9_.-]/', '_', $label);
    }
4   function safe_fopen($bp_globals_oe_site_dir, $bp_get_template){
5      $template_dir = $bp_globals_oe_site_dir . "/letter_templates";
6      return fopen("$template_dir/" .
7            convert_safe_file_dir_name($bp_get_template), 'r');
    }
8   ?>
```

Safely Backportable Patch for CVE-2018-10572

deploy

SBV

```
1   <?php
2  + $bp_get_template = $_GET['template'];
3  + $bp_globals_oe_site_dir = $GLOBALS['OE_SITE_DIR'];
4    $sanitize_all_escapes  = true;
5    $fake_register_globals = false;
6
7    include_once("../globals.php");
8    include_once($GLOBALS['srcdir'] . "/patient.inc");
9    $template_dir = $GLOBALS['OE_SITE_DIR'] . "/letter_templates";
10   ...
11 -  $fh = fopen("$template_dir/".$_GET['template'], 'r');
12 + $fh = safe_fopen($bp_globals_oe_site_dir, $bp_get_template);
13   . ..
14   ?>
```

Deployed Safely Backportable Patch on OpenEMR 5.0.0.5

safely backport with guarantee

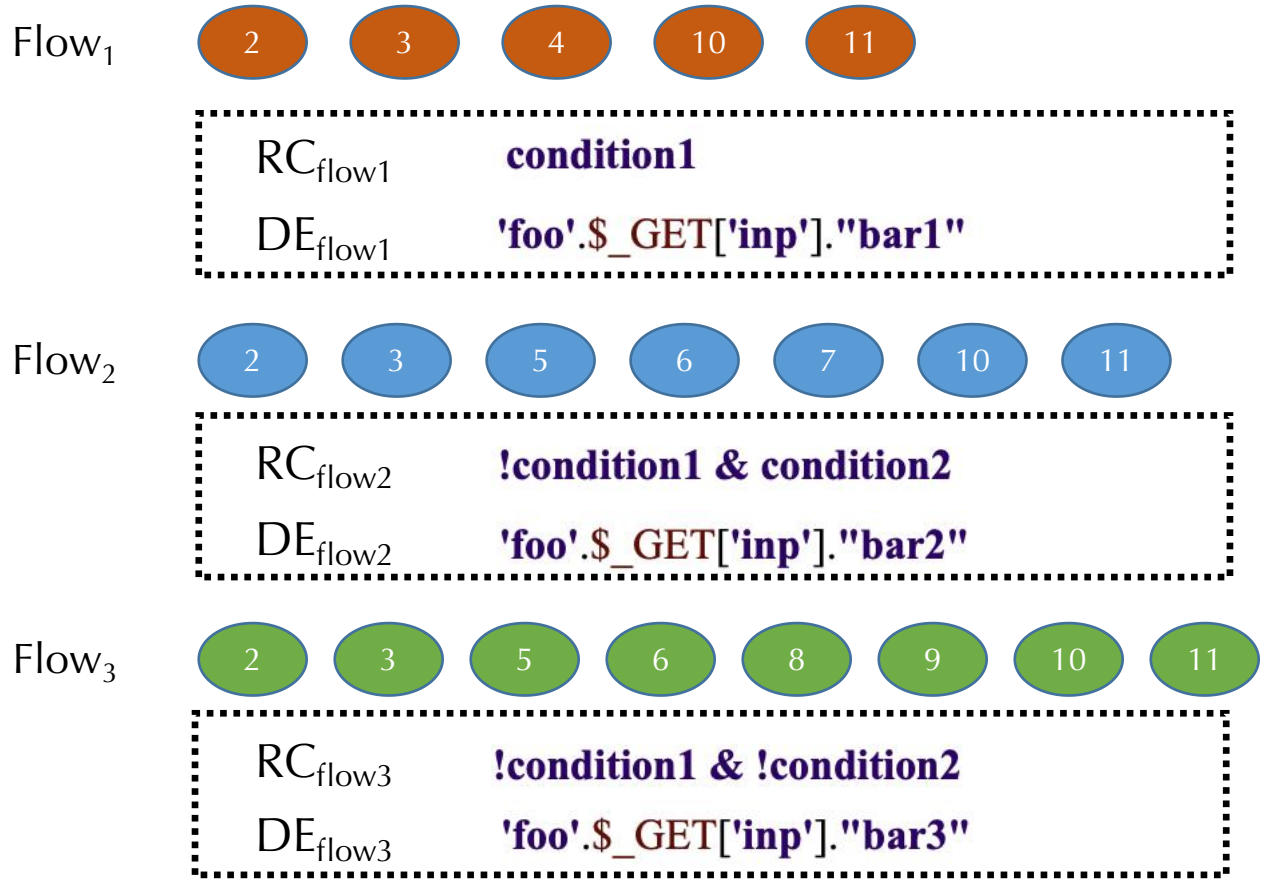
Sink Capability

- The Representation
  - Sink Flow: a control-flow path leading to the sink function
    - The inputs that reach the sink along each path are represented as $<flow_1, flow_2, ...>$
  - Each sink flow consists of $<RC_{flow}, DE_{flow}>$
    - Reaching Condition (RC_{flow}): a set of the control-flow conditions in the flow
    - Data Expression (DE_{flow}): the symbolic expression of the critical sink parameters in the flow
  - Thus, the sink capability can be represented as
    - $\{<RC_{flow1}, DE_{flow1}>, <RC_{flow2}, DE_{flow2}>, ...\}$

Sink Capability Example

- The Sink Capability (SC) can be represented as <$flow_1$, $flow_2$, $flow_3$>

```
1  <?php
2  $input = $_GET['inp'];
3  if ($condition1)
4      $value = $input."bar1";
5  else{
6      if($condition2){
7          $value = $input."bar2";
       }
8      else{
9          $value = $input."bar3";
       }
   }
10 $value = 'foo'.$value;
11 sink_func($value);
12 ?>
```



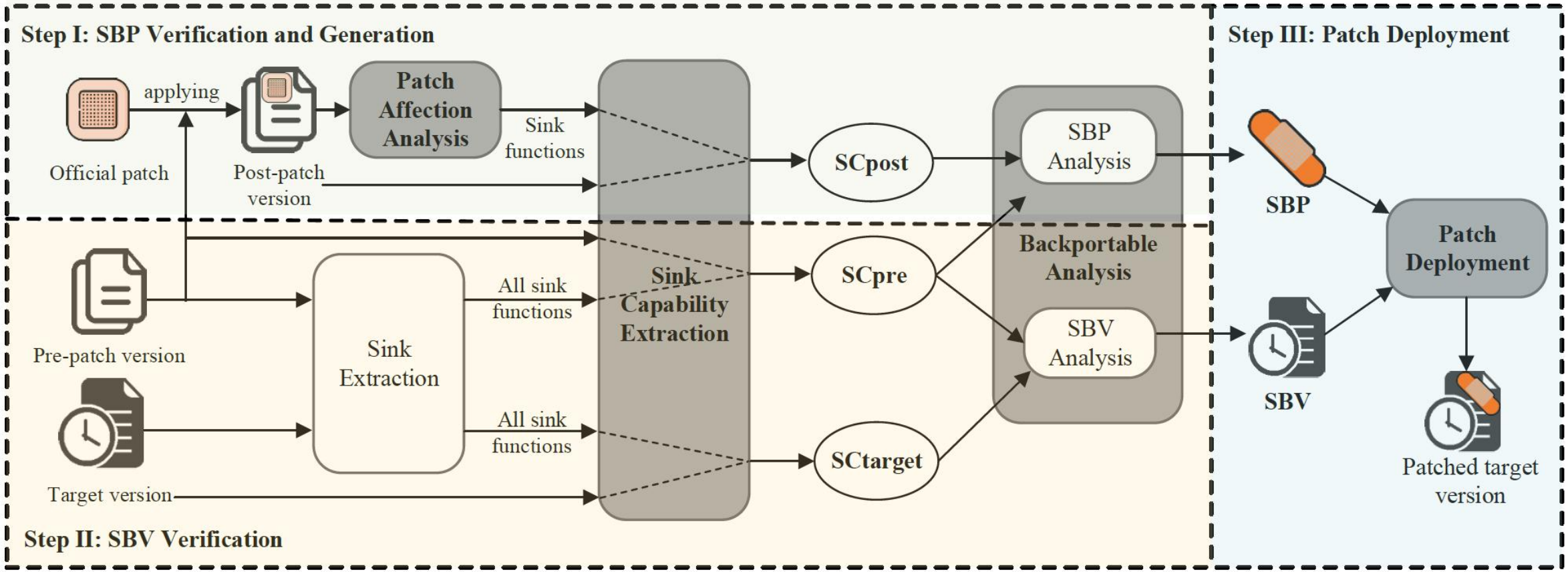
SBP & SBV

- Safely Backportable Patch (SBP) Properties: pre-patch vs post-patch
  1. $P_{SBP\text{-}a}$: $RC^{post}_{flow_k}$ is a subset of $RC^{pre}_{flow_k}$
  2. $P_{SBP\text{-}b}$: $DE^{post}_{flow_k}$ is a subset of $DE^{pre}_{flow_k}$
  3. $P_{SBP\text{-}c}$: $RC^{post}_{flow_k}$ and $DE^{post}_{flow_k}$ are deterministically computable for every $flow_k$
  - Compatibility Guarantee: SBP deployment will not affect the functionality of the target application
- Safely Backportable Version (SBV) Properties: pre-patch vs target
  1. $P_{SBV\text{-}a}$: $RC^{pre}_{flow_k}$ is same as $RC^{target}_{flow_k}$ for every $flow_k$
  2. $P_{SBV\text{-}b}$: $DE^{pre}_{flow_k}$ is same as $DE^{target}_{flow_k}$ for every $flow_k$
  - Security Guarantee: SBP deployment can fix the vulnerability of the target application

Approach Overview

- Three Steps
  1. SBP Identification & Generation
     - Analyze whether a patch is backportable and if so, transform it to SBP
  2. SBV Verification
     - Verify whether a target version is an SBV (aka, can apply the SBP)
  3. Patch Deployment
     - Automatically deploy SBP on an SBV
- Automatic Tool: SKYPORT, based on PHPJoern

SKYPORT Workflow



Four Modules

- Patch Affection Analysis (M1), Sink Capability Extraction (M2)
- Backportable Analysis (M3), Patch Deployment (M4)

Evaluation & Dataset

CMS Name	# CVEs	# Versions	# <CVE, Version>
WordPress	34	187	430
PHPMyAdmin	29	108	257
Prestashop	11	34	101
RoundcubeMail	8	48	76
MantisBT	24	74	198
Piwigo	11	37	108
OpenEMR	11	20	70
phpipam	3	6	13
MISP	9	55	118
LimeSurvey	15	82	155
Total	155	651	1,526

Selection Criteria

1. The Web application with more than 1k stars in the GitHub
2. Injection vulnerability patches
3. Patches that fix the vulnerability by restricting the sink functions

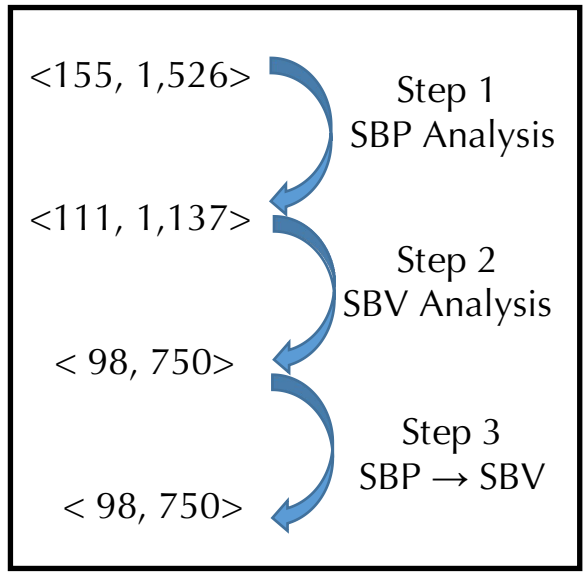
Evaluation & Dataset Statistics

- Patches
  - 98 / 155 security patches contain vulnerability-irrelevant modifications
    - E.g., functionality modifications, variable or function name modifications
    - May lead to backward compatibility or patch deployment issues
- Target versions
  - 563 / 1,526 target versions do not have the same vulnerable logic as the pre-patch
    - These versions are not SBVs, thus not being backportable (aka, requiring a new patch)
  - 1,071 / 1,526 target versions have code location changes around the patch
    - May lead to code conflicts when directly applying the original patch via patch command

These results show that patch backporting is non-trivial!

Evaluation & SKYPORT

1. Effectiveness: How effective is SKYPORT in patch backporting?
   - SKYPORT successfully backport 98 SBPs to 750 SBVs with 100% success rate
2. Efficiency: How efficient is SKYPORT in patch backporting?
   - SKYPORT takes 6459.75 seconds on average for an end-to-end case
3. Comparison: How does SKYPORT compare to existing practices?



	Patch Command	Auto-upgrade/Strict	Auto-upgrade/Relaxed	SKYPORT
Success	455	39	149	750

Evaluation & SKYPORT-patched Apps

- Evaluating SKYPORT-patched Apps involves significant human efforts
  - We evaluate a subset of SKYPORT-patched Apps, covering <11, 27> CVE-version pairs

1. Security: Can the SBPs defend against vulnerability-related attacks?
   - SKYPORT-patched apps successfully defended against all the collected exploits
2. Compatibility: Do the SBPs incur functionality issues?
   - SKYPORT-patched apps with 100% test pass ratio for compatibility with single or multiple SBPs
3. Performance: What is the performance overhead introduced by SBPs?
   - The SBPs introduce negligible overhead when compared with the official patches

Conclusion

- Methodology for automatic patch backporting with guaranteed compatibility and security.
- Formulation for safely backportable patches (SBP) and safely backportable versions (SBV), which enable safe patch backporting.
- Tool for automatically backporting injection-based PHP patches to old vulnerable versions.
- Evaluation results that demonstrate the effectiveness and efficiency of the proposed approach.

THANKS

Q&A

ykshi21@m.fudan.edu.cn