Multiple Open Source Web App Vulnerabilities Fixed

Jul 27, 2021 | 13 min read | Tod Beardsley (/blog/author/tod-beardsley/)

Last updated at Wed, 01 Sep 2021 17:01:11 GMT

Today, Rapid7 is disclosing 9 vulnerabilities that affect three open-source projects: EspoCRM (https://github.com/espocrm/espocrm), Pimcore (https://github.com/pimcore/pimcore), and Akaunting (https://github.com/akaunting/akaunting/). Right out of the gate, I'd like to give a special thanks to these 3 open-source project maintainers. While it's never great to learn of new vulnerabilities in your own product, all 3 project maintainers accepted, validated, and provided fixes for these vulnerabilities within **one day**, which is amazing when it comes to vulnerability disclosure. EspoCRM was notified on May 4, 2021 and patched source on May 5; Akaunting, on May 13 and turned it around on May 14; and Pimcore validated their vulnerabilities on April 29 after learning about them on April 28, 2021. Nice work, all around.

Now, I'm not sure why open source is just so much faster than the typical proprietary software vuln-patching pipeline, at least for the disclosures I've been involved in. It might be because, in open source, you're almost guaranteed to have your first communication with a hands-on-keyboard software engineer who is personally and emotionally invested in the software; whereas in proprietary land, first contact might be a lightly monitored support alias, staffed by a third-party provider. Rapid7's vulnerability disclosure process (https://www.rapid7.com/disclosure/) assumes a minimum of 60 days for remediation of any vulnerability we report to a vendor, and I'd say about half the time, we're looking at more like 90 to 120 days from report to disclosure — and, sometimes, we are left with the unhappy option of publishing without a fix in hand at all.

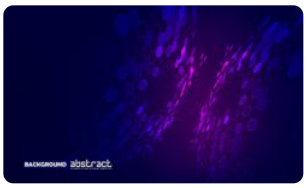
Of course, proprietary software occasionally offers fast turnaround times on validation and fixes to source, as well (SonicWall (https://www.rapid7.com/blog/post/2021/06/23/cve-2021-20025-sonicwall-email-security-appliance-backdoor-credential/) comes to mind), and proprietary vendors often have very good reason to take their time with acknowledging, fixing, testing, and releasing fixes; but the fact remains that what's normal in open source communities — hyperfast turnaround on fixing reported vulnerabilities — is a rarity in proprietary software.

By the way, these aren't one- or two-person passion projects. All 3 of these projects have real users, real customers of their attendant support services and cloud-hosted versions, and are undoubtedly the core applications supporting thousands of small to medium businesses running today. This popularity is the reason why Trevor and Wiktor took a look at them in the first place; they suspected these small-to-medium business applications haven't seen a ton of attention from the eye of a penetration tester, and this blog post is a result of testing that hypothesis.

With that, I'll stop picking on proprietary software vendors in general and switch gears to take a look at the specific vulnerabilities in these specific projects.

Common Vulnerability Classes

From this completely unscientific and statistically insignificant sampling of vulnerabilities, we can draw the deeply unsurprising conclusion that enterprise web applications tend to suffer from common web application vulnerabilities. 3 are



Topics

Metasploit (799) (/blog/tag/metasploit/)

Vulnerability Management (418) (/blog/tag/vulnerability-management/)

Detection and Response (388) (/blog/tag/detection-and-response/)

Research (277) (/blog/tag/research/)

Application Security (156) (/blog/tag/application-security/)

Cloud Security (110) (/blog/tag/cloud-security/)

Popular Tags

Search Tags

Metasploit (/blog/tag/metasploit/)

Logentries (/blog/tag/logentries/)

IT Ops (/blog/tag/it-ops/)

Vulnerability Management (/blog/tag/vulnerability-management/)

Detection and Response (/blog/tag/detection-and-response/)

Metasploit Weekly Wrapup (/blog/tag/metasploit-weekly-wrapup/)

Research (/blog/tag/research/)

Automation and Orchestration (/blog/tag/automation-and-orchestration/)

Nexpose (/blog/tag/nexpose/)

Incident Detection (/blog/tag/incident-detection/)

InsightIDR (/blog/tag/insightidr/)

Exploits (/blog/tag/exploits/)

Incident Response (/blog/tag/incident-response/)

examples of persistent cross-site scripting (XSS), where a malicious user can plant a bit of browser-executable code in the application, which is designed to lie in wait and trigger when someone else comes along and loads that code, and 2 are SQL injection (SQLi) vulnerabilities, where the attacker uses the web application as a convoluted portal to issue direct commands to the backing database, usually to steal data or create powerful web-app users.

SQL injection used to be a nice way to get a command injection path to the underlying operating system, but that's something of a rarity these days. But, 1 issue disclosed here is a command injection issue, which we rate as the highest critical vulnerability of the bunch, since it can allow the attacker to commandeer the operating system and do things like use it as a beachhead into the rest of the network, install a cryptominer or ransomware, or perform other nefarious lower-level actions.

The remaining vulnerabilities are: a denial-of-service vulnerability, where the attacker can crash the whole application with a naughty HTTP request; an authentication bypass, where the attacker can move from one logical group to another without authorization; and a weak password-reset vulnerability, where the attacker can abuse the "I forgot my password" function to source a phishing email from the application to a registered user.

The table below provides the salient information about the 9 vulnerabilities being disclosed today. Note that every vulnerability listed here was promptly fixed by the vendor in the typical open-source manner. In short: if you use any of these applications in your business and keep up on your updates, you already have the fixes. The rest of this post details the individual findings by Wiktor Sędkowski (https://www.linkedin.com/in/wiktorsedkowski/) of Nokia and Trevor Christiansen of Rapid7, who worked together on this project and disclosed these issues through Rapid7's vulnerability disclosure process (https://www.rapid7.com/disclosure/).

We're publishing these details today so other, similar web applications can be made aware of these vulnerability classes and take a look at their own codebases to make sure they're not making the same mistakes. Thanks, Wiktor and Trevor!

CVE	Affected Project	CWE	Base CVSS	Status
CVE-2021-3539	EspoCRM v6.1.6	CWE-79 (Persistent XSS)	6.3 (Medium)	Fixed in version 6.1.7
CVE-2021-31867	Pimcore Customer Data Framework v3.0.0	CWE-89 (SQL Injection)	6.5 (Medium)	Fixed in v3.0.2
CVE-2021-31869	Pimcore AdminBundle v6.8.0	CWE-89 (SQL Injection)	6.5 (Medium)	Fixed in v6.9.4
CVE-2021-36800	Akaunting v2.1.12	CWE-94 (Code injection)	8.7 (High)	Fixed in Akaunting v2.1.13
CVE-2021-36801	Akaunting v2.1.12	CWE-639 (Auth bypass)	8.5 (High)	Fixed in Akaunting v2.1.13
CVE-2021-36802	Akaunting v2.1.12	CWE-248 (Uncaught Exception DoS)	6.5 (Medium)	Fixed in Akaunting v2.1.13
CVE-2021-36803	Akaunting v2.1.12	CWE-79 (Persistent XSS)	6.3 (Medium)	Fixed in Akaunting v2.1.13
CVE-2021-36804	Akaunting v2.1.12	CWE-640 (Weak Password Reset)	5.4 (Medium)	Fixed in Akaunting v2.1.13
CVE-2021-36805	Akaunting v2.1.12	CWE-79 (Persistent XSS)	5.2 (Medium)	Fixed in Akaunting v2.1.13

Komand (/blog/tag/komand/)

Penetration Testing (/blog/tag/penetration-testing/)

Related Posts

CVE-2022-4261: Rapid7 Nexpose Update Validation Issue (FIXED) — READ MORE (/BLOG/POST/2022/12/07/CVE-2022-4261-RAPID7-NEXPOSE-UPDATE-VALIDATION-ISSUE-FIXED/)

CVE-2022-41622 and CVE-2022-41800 (FIXED): F5 BIG-IP and iControl REST Vulnerabilities and Exposures — READ MORE (/BLOG/POST/2022/11/16/CVE-2022-41622-AND-CVE-2022-41800-FIXED-F5-BIG-IP-AND-ICONTROL-REST-VULNERABILITIES-AND-EXPOSURES/)

FLEXlm and Citrix ADM Denial of Service Vulnerability — READ MORE (/BLOG/POST/2022/10/18/FLEXLM-AND-CITRIX-ADM-DENIAL-OF-SERVICE-VULNERABILITY/)

Baxter SIGMA Spectrum Infusion Pumps: Multiple Vulnerabilities (FIXED) — READ MORE (/BLOG/POST/2022/09/08/BAXTER-SIGMA-SPECTRUM-INFUSION-PUMPS-MULTIPLE-VULNERABILITIES-FIXED/)

EspoCRM v6.1.6 (1 issue)

EspoCRM is an open-source customer relationship management (CRM) application used in all sorts of industries, although it seems to enjoy special success in the real estate sector. More about EspoCRM can be found at the vendor's website (https://www.espocrm.com/).

CVE-2021-3539: EspoCRM Avatar Persistent XSS

Any user with default rights, which allows them to upload their own avatar, can abuse the API for this by providing executable Javascript code instead of an image. An example call to the API is detailed below:

```
PUT /api/v1/User/609108e6b123bb29d HTTP/1.1
Host: 10.0.0.10:8443
{redacted}
Content-Length: 43
Origin: https://10.0.0.10:8443
Connection: close
Referer: https://10.0.0.10:8443/
Cookie: {redacted}

{
"avatarId":"\" onerror=\"alert(0)\" "
}
```

This leads to rendering the avatar as:

resulting in triggering the `onerror` event:

Because EspoCRM allows administrators to install arbitrary, custom extensions, an attacker can leverage this XSS to silently coerce an administrator (who views the attacker's avatar) to install a malicious extension, thus retaining permanent control of the web application, as seen in the screenshot below.

Pimcore Customer Data Framework v3.0.0 (1 issue)

Pimcore CDF is a component of the Pimcore platform and is a CRM enterprise application. More about Pimcore CDF can be found at the vendor's website (https://pimcore.com/en/customer-data-platform).

CVE-2021-31867: Pimcore CDF 'SegmentAssignmentController.php' Blind SQL Injection

An SQL injection vulnerability exists in the Customer Management Framework Bundle, specifically in the SegmentAssignmentController.php component. The vulnerable code was introduced in commit 6fc8aff8f95fc168d173ef3b473760dd98d026c4 and is shown below.

```php
public function inheritableSegments(Request $request)
{
$id = $request->get('id') ?? '';
$type = $request->get('type') ?? '';
/* @var $db Connection */
$db = $this->get(Connection::class);
$parentIdStatement = sprintf('SELECT `%s` FROM `%s` WHERE `%s` = "%s"', $type
$parentId = $db->fetchOne($parentIdStatement);
$segments = $this->get(SegmentManagerInterface::class)->getSegmentsForElement
$data = array_map([$this, 'dehydrateSegment'], array_filter($segments));
return $this->adminJson(['data' => array_values($data)]);
}
```

`$id` is retrieved from the request parameters and then placed directly into the SQL query through the use of `sprintf` and then executed (as long as `$type` is something other than `object`). This allows a malicious actor to inject the SQL query through the use of a single quote `'`.

This vulnerability can be thought of as a Boolean-based Blind SQL Injection, as an exploit is unable to pull out data from the database directly, but has to piece together the information through a series of True/False requests.

This image below shows a request that tests if the integer 1 equals 1:

The response returns a `200 OK` along with the data that has an `id` of 137:

This second request tests if the integer 1 is equal to 2:

This time, the response is a `500 Internal Server Error` along with a stack trace.

Using these 2 queries, a malicious actor can automate the retrieval of information from the database. This last example shows a query to find the first character of the version from the database server.

Pimcore AdminBundle v6.8.0 (1 issue)

Pimcore AdminBundle is part of the core Pimcore platform, a Product Information Management (PIM) platform, which is closely related to the Enterprise Resource Planning (ERP) functions of a business. More about the Pimcore platform can be found at the vendor's website (https://pimcore.com/en/platform).

CVE-2021-31869: Pimcore AdminBundle 'specificID' SQL Injection

Requests sent to `/admin/object/grid-proxy` are handled by `Bundles/AdminBundle/Controller/Admin/DataObject/DataObjectController.php` file, starting on line 1568, as shown below:

This file collects all the parameters (line 1586), then includes the parameters in a call to `prepareListingForGrid` shown below:

`prepareListingForGrid` is found in the previously mentioned `Pimcore/Bundles/AdminBundle/Helper/GridHelperService.php` file and starts on line 489. This function builds the SQL query from the provided parameters. The parameter `specificID` is vulnerable to SQL injection, since the `specificId` parameter data is concatenated directly into the string and then added to the `$conditionFilters` array, as shown below:

A request to the `grid-proxy` url is shown below. In this query, the `specificId` field is set to `1+or+'a'='a'`. The response shows a content-length of 7546.

GET /admin/object/grid-proxy/classId=BS&[other params]&specificId=1+or+'a'='a'&query=[other params] HTTP/1.1

This next request sets the `specificId` parameter to `1+or'a'='b'`. As shown in the following image, the response length is now 47, and no records were returned.

By combining these 2 requests, a malicious actor can programmatically return data from the database by testing each character and monitoring the response. The image below shows an example of this by requesting the database version and checking if the first character of the version is equal to 8.

Akaunting v2.1.12 (6 issues)

Akaunting is an enterprise accounting system, providing a variety of services related to the normal day-to-day business operations, notably in the retail sector, such as invoicing and expense tracking. More about Akaunting can be found at the vendor's website (https://akaunting.com/).

CVE-2021-36800: Akaunting OS Command Injection

The Akaunting application allows for PHP code sent to the application to be executed by the web server. This can lead to a shell directly on the host operating system. The vulnerability was introduced upon the creation of the `Money.php` file in the first commit, 1c01d2120941d99f758cf23be20fe5931bdd4a36. To exploit this vulnerability, the attacker must first be authenticated and already have permissions to add or modify sales invoices.

A POST sent to `/{company_id}/sales/invoices/{invoice_id}` with an `items[0][price]` that includes a PHP callable function is executed directly. The image below shows the post body, including a `items[0][price]` set to `phpinfo`. The response on the right shows the response, which includes the results from the application executing `phpinfo()`:

This is due to a lack of input sanitization in the Money.php middleware component. The following is the code responsible for the execution; as shown, it checks to see if what is received is callable and, if so, executes it.

```
protected function parseAmountFromCallable($amount)
{
if (!is_callable($amount)) {
return $amount;
}
return $amount();
}
```

CVE-2021-36801: Akaunting Authentication Bypass in Company Selection

A user is able to change the company their account is associated with, allowing them to view/modify information from another company. This vulnerability was introduced in the first commit of the application. To exploit this vulnerability, the attacker must first be authenticated as any user.

The first image shows that the user `Test_company_1` is associated with the company named `My Company`:

While logged in as the user `Test_company_1` we click on `Profile` to change the user settings:

By clicking on the `Save` button and intercepting the request, we can modify the `companies[0]` field to the `id` of another company. The image below shows changing the company information from 1 to 2 while updating the profile information:

Once done, viewing the dashboard shows that the associated company has been changed:

CVE-2021-36802: Akaunting DoS via User-Controlled 'locale' Variable

Any user can crash the Akaunting platform by supplying an invalid 'locale' variable as part of an otherwise well-formed HTTP POST request. This vulnerability was introduced in the first commit of the application. To exploit this vulnerability, the attacker must first be authenticated as any user.

The image below shows a post to `/2/settings/settings` with an invalid locale that is successfully processed without error:

Visiting any page will result in a 500 response:

CVE-2021-36803: Akaunting Avatar Persistent XSS

A user can inject HTML into the avatar upload process and trigger an XSS for anyone who views it, including high-privilege administrators of the application. This vulnerability was introduced in the first commit of the application. To exploit this vulnerability, the attacker must first be authenticated as any user.

The image below shows a post to `/{company_id}/auth/users/{user_id}` with HTML embedded in the image upload field:

Example payload:

```

----------------------------110883763421077057633417501 65

Content-Disposition: form-data; name="picture"; filename="Screenshot_2021-05-02_05_11_16.png"

Content-Type: image/png

</pre><html><b>test</b><script>alert('xss')</script><pre>

```

The HTML is directly rendered on screen while accessing the avatar URL; e.g /{company_id}/uploads/{upload_id}:

CVE-2021-36804: Akaunting Password Reset Relay

Setting the host header while sending a Post to `/auth/forgot` endpoint changes the link generated by the application. An attacker can send a password-reset request for an existing user and modify the host header to point to a web server they control. If the user clicks on the password reset URL, the attacker will receive the password-reset token and can then set the password to something the attacker knows. This vulnerability was introduced in the first commit of the application. To exploit this vulnerability, the attacker must first know or guess the email address of a valid user.

The image below shows a post to the /auth/forgot endpoint, with the Host set to example.com:

The email sent by the application directs the user to the example.com domain with the password reset token.

Note that the root of this vulnerability is due to a design decision in the Laravel framework and how proxy headers are handled with respect to single instance and multi-tenant implementations. In other words, while CVE-2021-36804 is a (now fixed) vulnerability in Akaunting, other multi-tenant implementations involving Laravel should be aware that the default configuration of that framework is likely vulnerable to a similar issue. For more information on this design issue, please see Enlightn's Host Injection Analyzer (https://www.laravel-enlightn.com/docs/security/host-injection-analyzer.html), Daniel Coulbourne's tweet (https://twitter.com/DCoulbourne/status/1331317933675581442), and PR 5477 (https://github.com/laravel/laravel/pull/5477) in the Laravel GitHub repository.

CVE-2021-36805: Akaunting Invoice Footer Persistent XSS

The Akaunting application allows for HTML to be written to the footer of a sales invoice and relies on its built-in “firewall” to prevent malicious code, such as XSS, from being accepted. The following example shows how specially crafted HTML code can bypass the filtering. This vulnerability was introduced in the first commit of the application. To exploit this vulnerability, the attacker must have the permissions to add or modify sales invoices.

A POST sent to `/{company_id}/sales/invoices/{invoice_id}` with a `footer` that includes the following HTML will execute the javascript:

Proof of concept payload:

```
POST /1/sales/invoices/201 HTTP/1.1
...
----------------------------117666534612857833648279657 38
Content-Disposition: form-data; name="footer"
'\"<img class="/>" onerror=alert("Vulnerable+to+XSS") src="b.png"
----------------------------117666534612857833648279657 38
…
```

The results of viewing the sales invoice:

The payload bypasses the firewall restrictions because of the `>` placed in the class attribute. The image below shows how this string is not matched against the regex designed to prevent XSS:

Remediation

For all of these issues, updating to the latest versions of the affected applications will resolve them. If updating is difficult or impossible due to external factors or custom, local changes, users of these applications can limit their exposure by not presenting their production instances to the internet directly — instead, expose them only to trusted internal networks with trusted insiders.

Alternatively, since these applications are open source, users can contact these projects directly for any help needed to backport a fix to their own running version. One way to discover the exact code changes is simply to look at the git diffs between the fixed version and the most immediately prior version, and the fixes should be fairly obvious to anyone familiar with the languages in which these applications are written. In general, fixing bugs is fairly straightforward once you know what the vulnerabilities are. Finding and proving out the bugs is the hard part, so thanks again to Wiktor and Trevor for their work here.

POST TAGS

Vulnerability Disclosure (/blog/tag/vulnerability-disclosure/)

Open Source (/blog/tag/open-source/)

SHARING IS CARING

AUTHOR

Tod Beardsley (/blog/author/tod-beardsley/)

og/(author/t
rdsley/)

Director of Research at Rapid7, contributing author of several Rapid7 research papers, CVE Board member, and Metasploit collaborator.
https://keybase.io/todb

VIEW TOD'S POSTS

Related Posts

VULNERABILITY DISCLOSURE

CVE-2022-4261: Rapid7 Nexpose Update Validation Issue (FIXED)

VULNERABILITY DISCLOSURE

CVE-2022-41622 and CVE-2022-41800 (FIXED): F5 BIG-IP and iControl REST Vulnerabilities and Exposures

VULNERABILITY DISCLOSURE

FLEXlm and Citrix ADM Denial of Service Vulnerability

VULNERABILITY DISCLOSURE

Baxter SIGMA Spectrum Infusion Pumps: Multiple Vulnerabilities (FIXED)

VIEW ALL POSTS

Search all the things

BACK TO TOP

(/)

CUSTOMER SUPPORT

+1-866-390-8113 (Toll Free) (tel:1-866-390-8113)

SALES SUPPORT

+1-866-772-7437 (Toll Free) (tel:866-772-7437)

Need to report an Escalation or a Breach?

CLICK HERE (/services/incident-response-customer-escalation/)

SOLUTIONS

All Solutions (https://www.rapid7.com/solutions)

Industry Solutions (https://www.rapid7.com/solutions/industry)

Compliance Solutions (https://www.rapid7.com/solutions/compliance/)

SUPPORT & RESOURCES

Product Support (https://www.rapid7.com/for-customers)

Resource Library (https://www.rapid7.com/resources)

Our Customers (/customers/)

Events & Webcasts (https://www.rapid7.com/about/events-webcasts)

Training & Certification (https://www.rapid7.com/services/training-certification)

IT & Security Fundamentals (https://www.rapid7.com/fundamentals)

Vulnerability & Exploit Database (https://www.rapid7.com/db)

ABOUT US

Company (https://www.rapid7.com/about/company)

Diversity, Equity, and Inclusion (https://www.rapid7.com/about/diversity-equity-and-inclusion/)

Leadership (https://www.rapid7.com/about/leadership)

News & Press Releases (https://www.rapid7.com/about/news)

Public Policy (https://www.rapid7.com/about/public-policy)

Open Source (https://www.rapid7.com/open-source/)

Investors (https://investors.rapid7.com/)

CONNECT WITH US

Contact (https://www.rapid7.com/contact)

Blog (https://blog.rapid7.com/)

Support Login (https://support.rapid7.com/)

Careers (https://www.rapid7.com/careers)

(https://[illegible]/rapid7/)

(https://www.rapid7.com/about/rapid7-cybersecurity-partner-boston-bruins/)

 Legal Terms (/legal/) | Privacy Policy (/privacy-policy/) | Export Notice (/export-notice/) | Trust (/trust/)

Contact Us