Research of CVE-2015-8124 vulnerability

Verner Läll

Supervisor: Meelis Roos

Tartu 2017
Introduction

Symfony is a PHP framework to create websites and web applications. The CVE-2015-8124 vulnerability was discovered on 23. November 2015 by RedTeam Pentesting GmbH.

Session fixation is an attack that permits an attacker to hijack a valid user session.

CVE-2015-8124 is a session fixation vulnerability within the Symfony web application framework’s "Remember Me" login functionality, that allows an attacker to impersonate the victim in the web application if the victim’s session ID value is known or set by the attacker.

Details

CVE-2015-8124 vulnerability allows an attacker to access a Symfony web application with the attacked user’s permissions or essentially, log in to the web application with the victim’s user only by having their PHPSESSID cookie value.

The attack requires that the "Remember Me" login functionality is used by the application. The attacker needs either access to the PHPSESSID cookie value or to successfully set a new value in the user’s browser.

Because of these strict requirements and an easy workaround, this vulnerability poses a low risk. The risk estimation could be increased to medium or high, depending on the web application affected, or if the accessible data from the victim’s user is confidential or regulated by law.
Usage

I set up a local server to try this myself and installed the Symfony v2.3.0 demo application on it. I have marked the command line input I used with a grey background, and the output as blue text.

About the command line tool: “curl” is a tool to transfer data from or to a server. The option -i includes the HTTP-header in the output. Option -b passes data to the HTTP server as a cookie. I will use it to simulate the victim being authenticated with a valid REMEMBERME cookie and the attacker using that user’s session id to use that vulnerability.

I’ve truncated the html in square brackets, describing the contents inside them.

If a privileged page is requested with an unauthorised session ID, a Symfony application redirects to the login page:

```
$ curl -i 'http://localhost/en/admin/post/' -b 'PHPSESSID=attackerKnowsThisID'
```

HTTP/1.1 302 Found
Host: localhost:80
Location: http://localhost/en/login
[... login page html...]

But in the case that a valid REMEMBERME cookie is included in the HTTP request, which happens when the user has logged in with “Remember me” functionality enabled in the past, the user is successfully authenticated:

```
$ curl -i 'http://localhost/en/admin/post/' -b 'PHPSESSID=attackerKnowsThisID'
'REMEMBERME=someValidRememberMeCookie'
```

HTTP/1.1 200 OK
Host: localhost:80
[... symfony app logged in as the victim html ...]

After this HTTP request, the PHPSESSID value suffices to authenticate the user. In contrast to the regular login procedure, the web application did not assign a new value to the PHPSESSID cookie. If an attacker somehow got in possession of the PHPSESSID cookie’s value or has successfully set a given cookie value in the user’s browser at some point in the past, the attacker is now able to access the web application with the user’s permissions from his own computer:

```
$ curl -s -i 'http://localhost/en/admin/post/' -b 'PHPSESSID=attackerKnowsThisID'
```

HTTP/1.1 200 OK
Host: localhost:80
[... symfony app logged in as the victim html ...]
The solution is to migrate the session after a successful login via the "Remember Me" login feature, in the file which handles cookie based authentication.

```php
use Symfony\Component\Security\Http\Session\SessionAuthenticationStrategy;

// RememberMeListener implements authentication capabilities via a cookie.
private $authenticationManager;
private $logger;
private $dispatcher;

public function ___construct($securityContextInterface $securityContext, RememberMe $rememberMe)
    $this->authenticationManager = $authenticationManager;
    $this->logger = $logger;
    $this->dispatcher = $dispatcher;

    $this->sessionStrategy = new SessionAuthenticationStrategy(SessionAuthenticationStrategy::MIGRATE);
}

public function handle($event)
    try {
        $token = $this->authenticationManager->authenticate($token);

        if ($request->hasSession() && $request->getSession()->isStarted()) {
            $this->sessionStrategy->onAuthentication($request, $token);
        }

        $this->securityContext->setToken($token);
```
Sources


https://github.com/symfony/symfony/pull/16631/files#diff-44fbd2fa1a8f6875ebbc275a311a75e8

http://www.securityfocus.com/bid/77694