**LFI to RCE via phpinfo**

Trong quá trình kiểm thử, dựa vào cấu hình của máy chủ mà bạn có thể thực thi lệnh từ xa qua những kĩ thuật sau:

* /proc/self/environ
* /proc/self/fd/…
* /var/log
* /var/lib/php/session/
* /tmp/
* php://input wrapper
* php://filter wrapper
* data: wrapper

Cách khai thác này mở rộng dựa trên “PHP LFI to RCE qua upload tập tin tạm”

**PHP LFI to RCE qua upload tập tin tạm**

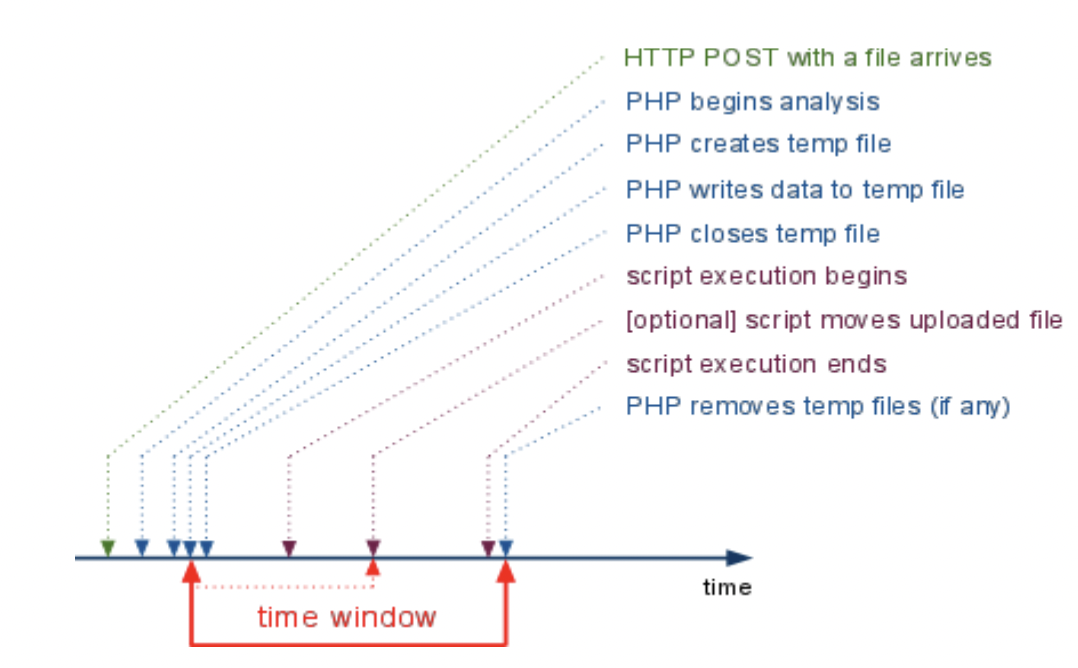
Một vài phương pháp thường được dùng để chứng minh có thể thực thi mã:

* including uploaded files: điều này yêu cầu chức năng upload trong trang web, truy cập chức năng upload và nơi lưu trữ các tập tin được upload có thể truy cập
* include data:// hay php://input các giao thức giả: các giao thức phải được bật và có thể truy cập qua include (allow\_url\_include set to on); tương tự giao thức giả php://filter rất hữu ích trong một số trường hợp
* including logs: điều này yêu cầu PHP script có thể truy cập các loại logs nhất định như httpd server error logs hay access logs; tương tự kích thước các file logs có thể làm cho tấn công trở nên khó hơn (nếu error log có 2GB).
* including /proc/self/environ: điều này yêu cầu PHP được chạy như CGI trên hệ thống có hệ thống tệp giả /proc và PHP script có thể truy cập pseudo-file.
* include session files: điều này yêu cầu kẻ tấn công có thể gây ảnh hưởng cho giá trị trong 1 phiên (để tiêm code), các phiên phải được lưu trữ trong 1 serialized session file (như x|s:19: "<?php phpinfo();?>"; - đây là cài đặt mặc định của PHP) và PHP script có thể truy cập session file (thường là /tmp/sess\_SESSIONID)
* include các file khác được tạo bởi ứng dụng PHP như database files, cache files, application-level logs,…

**Temporary upload file inclusion**

Một lựa chọn khác là lợi dụng cách Php xử lí file upload qua http. Phương pháp này ít được biết đến và chỉ khả thi trên các nền tảng như Windows. (xem các khai thác trên Linux)

PHP engine khi nhận được 1 gói tin POST, tạo 1 hay nhiều file tạm để lưu trữ các file được upload. PHP script muốn xử lí file upload yêu cầu dùng hàm move\_uploaded\_file để di chuyển các file tạm đến 1 vị trí mong muốn. Khi script kết thúc, Php engine xóa tất cả file tạm là các file đã upload.



Điều quan trọng ở đây là PHP engine tạo các file tạm mặc dù PHP script không cần chúng. Do đó, có thể gửi 1 file với đoạn mã bất kỳ và include file tạm mà PHP engine đã tạo.

**Lợi và Hại:**

Tin tốt là PHP script có thể truy cập thư mục chứa các file tạm (xem upload\_tmp\_dir trong php.ini). Trên cài đặt mặc định của PHP, upload\_tmp\_dir không được thiết lập - trong trường hợp này là /tmp của Linux hay C:\Window\Temp của Windows.

Tin xấu là tên của các file tạm là ngẫu nhiên, làm cho phương pháp này không thể sử dụng trong hầu hết các trường hợp - nó hoàn toàn có thể khai thác trên Windows và có thể khai thác trên hệ thống khác trong một số trường hợp.

**Khai thác trên Windows**

Để tạo 1 tên ngẫu nhiên trên Windows PHP dùng hàm GetTempFileName. Nhìn vào tài liệu chúng ta có thể thấy lời giải thích sau:

Hàm GetTempFileName tạo 1 file tạm tên của file theo định dạng sau: <path>\<pre><uuuu>.TMP

Trong trường hợp Php <path> là upload\_tmp\_dir (thường là C:\Windows\Temp) và <pre> là "php". Phần cuối được mô tả như sau:

<uuuu> Hexadecimal value of uUnique

uUnique là 1 đối số của GetTempFileName và trong trường hợp của PHP, nó đặt là 0, yêu cầu hàm dùng hệ thống thời gian. Điều quan trọng có thể được tìm thấy ở đây:

Chỉ dưới 16 bits của tham số uUnique được dùng. Điều này giới hạn GetTempFileName có tối đa 65535 tên file khác nhau.

Chỉ 65k tên khác nhau làm bruteforce khả thi, và dùng hệ thống thời gian sẽ trở nên dễ dàng hơn.

Tuy nhiên, brute force không cần thiết ở đây, nhờ vào FindFirstFile quirk cho phép dùng các masks (<< như \* và > như ?) trong đường dẫn LFI trên Windows. Nhờ vào nó, có thể include đường dẫn như sau:

http://site/vuln.php?inc=c:\windows\temp\php<<

Thường có những file khác với tiền tố php trong thư mục này vì vậy thu hẹp mask có thể cần thiết. Trong trường hợp này, tốt nhất là lựa chọn mask rộng nhất mà nó không bao gồm các file trên như php1<< hay phpA<<, tệ hơn là php11<< và gửi các gói tin upload cho đến khi PHP engine sẽ tạo 1 file tạm phù hợp với mask.

**Khai thác trên GNU/Linux**

Để tạo tên của file tạm PHP engine trên GNU/Linux dùng mkstemp từ GNU libc. Hàm này dựa vào cách glibc được biên dịch dùng:

1. random\_value = (seed += time() ^ PID)
2. random\_value = (seed += (gettimeofday().sec << 32 | gettimeofday().usec)^ PID)
3. random\_value = (seed += rdtsc ^ PID)

Giá trị ngẫu nhiên là 6 kí tự của k=62 (A-Za-z0-9) và nối vào tiền tố /tmp/php. Ví dụ: /tmp/phpUs7MxA.

Trong các phương pháp tạo giá trị ngẫu nhiên, cái thứ 3 là được dùng phổ biến nhất. Giá trị ngẫu nhiên không thể dự đoán được và có thể được xem là đủ an toàn với những kẻ tấn công.

Dùng khai thác đường dẫn này chỉ khả thi trong những trường hợp sau:

* Khi tester có thể liệt kê các tập tin trong thư mục upload (/tmp) - điều này có thể yêu cầu phương pháp race condition khi tester upload 1 file bằng 1 slow script (1 script chạy mất nhiều thời gian) và liệt kê các tập tin trong thư mục upload bằng 1 script khác, lấy tên của file tạm và nhanh chóng khai thác LFI.
* Khi tester có thể liệt kê được mảng $\_FILES trong 1 script - điều này yêu cầu 1 phương pháp race condition chặt chẽ hơn và thường không khả thi vì kẻ tấn công bắt buộc phải quan sát mảng $\_FILES trong quá trình upload và gửi gói tin khác trước khi script trước kết thúc và file tạm bị xóa. Điều này cũng không khả thi trong trường hợp buffering đầu ra của PHP (nếu mod\_gzip được bật trong Apache).

Do đó, phương pháp này chỉ được dùng trong 1 số trường hợp và không nên được xem là 1 phương pháp chung.

Quay lại với bài viết này nào.

Trong bài viết trên, tác giả nhắc đến cách PHP file upload hoạt động. Anh ấy lưu ý là file\_uploads=on phải được đặt trong cấu hình PHP, sau đó PHP sẽ chấp nhận 1 file upload post đến bất cứ file php nào. Anh ấy cũng lưu ý file upload sẽ được lưu trữ trong thư mục tmp cho đến khi toàn bộ trang PHP yêu cầu được xử lí xong.

FindFirstFile quirk không ảnh hưởng đến PHP engine trong GNU/Linux; tuy nhiên trong điều kiện nhất định, khai thác tính năng php file upload vẫn khả thi. Bài viết này sẽ chi tiết một trong những điều kiện đó, trở nên khả thi khi truy cập 1 tập lệnh mà kết quả là phpinfo() có sẵn trên máy chủ mục tiêu.

**LFI với sự hỗ trợ của PHPInfo()**

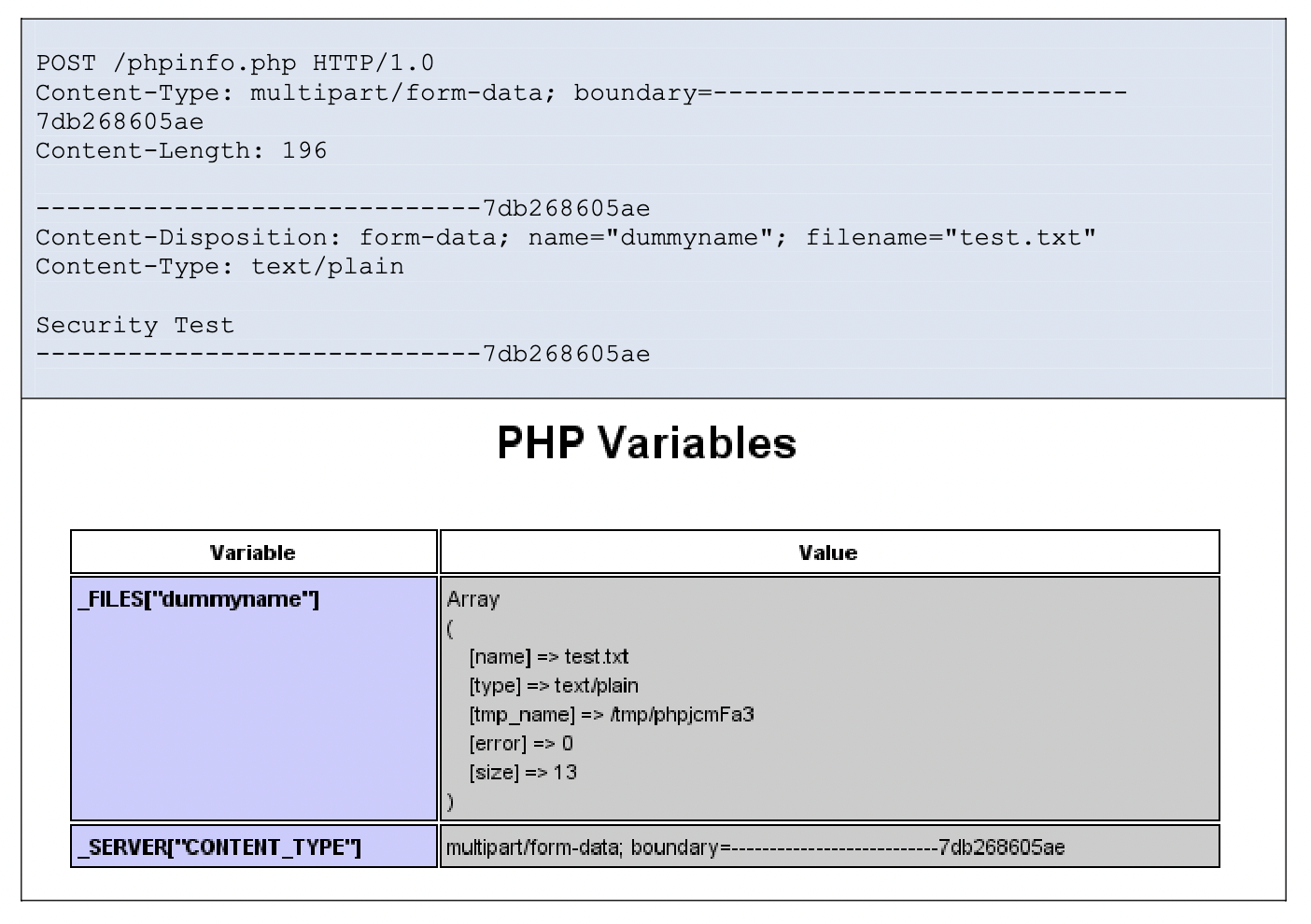
Các thành phần ở phía máy chủ cần thiết để thực hiện khai thác:

* Lỗ hổng LFI
* PHPInfo() script

**Tại sao lại là PHPInfo()?**

Đầu ra của script PHPInfo() chứa giá trị của các biến PHP, bao gồm bất cứ giá trị nào được đặt qua \_GET, \_POST, \_FILES.

Yêu cầu sau và đầu ra cho thấy cách script PHPInfo() có thể được dùng để khám phá tên của các file tạm.



**Winning The Race**

Mình không hiểu script này lắm, nhưng để race condition được tên của file tạm là không dễ. Họ phải dùng đến kĩ thuật output buffering trong PHP, tức là chèn thêm padding để đầu ra của script phpinfo phải lớn hơn yêu cầu của nó.

Bằng cách tạo nhiều post upload đến script phpinfo và lọc dữ liệu cẩn thận, có thể truy suất được tên của file tạm và gửi 1 yêu cầu đến LFI script trước khi nó bị xóa.

Có 1 challenge sẽ giúp bạn hiểu hơn về cách dùng script:

https://www.youtube.com/watch?v=rs4zEwONzzk&t=600s