**Vulnerabilities 101**

An ninh mạng là ngành kinh doanh lớn trong thế giới hiện đại. Những cuộc tấn công chúng ta nghe trên báo là từ khai thác các lỗ hổng. Trong bài này, chúng ta sẽ giải thích chính xác lỗ hổng là gì, các loại lỗ hổng và cách chúng ta có thể khai thác chúng để thành công trong quá trình kiểm thử.

**Giới thiệu về các lỗ hổng:**

Một lỗ hổng trong an ninh mạng được xem là điểm yếu và lỗi trong thiết kế, thực thi hoặc những hành vi của hệ thống hay ứng dung. Kẻ tấn công có thể lợi dụng điều đó để truy cập thông tin và thực hiện những hành động trái phép.

Những lỗ hổng có nguồn gốc từ rất nhiều yếu tố, bao gồm cả thiết kế kém của ứng dụng hay giám sát các hành động khác thường của người dùng. Có 5 loại lỗ hổng chính:

* Hệ điều hành
* Sai cấu hình
* Thông tin đăng nhập yếu
* Tính logic của ứng dụng
* Yếu tố con người

Là 1 nhà nghiên cứu an ninh mạng, bạn sẽ đánh giá các ứng dụng và hệ thống - sử dụng các lỗ hổng để chống lại mục tiêu vì vậy quá trình khám phá và khai thác rất quan trọng.

**Đánh giá lỗ hổng (CVSS và VPR)**

Quản lí các lỗ hổng là quá trình đánh giá, phân loại và cuối cùng là khắc phục các lỗ hổng mà 1 tổ chức phải đối mặt.

Rốt cuộc, chỉ có khoảng 2% lỗ hổng bị khai thác. Thay vào đó, tập trung vào giải quyết những lỗ hổng nguy hiểm nhất và giảm khả năng của 1 cuộc tấn công

Đó là lí do đánh giá lỗ hổng ra đời. Nó được dùng để xác định rủi ro và ảnh hưởng của 1 lỗ hổng đến mạng và hệ thống. Ví dụ hệ thống đánh giá lỗ hổng phổ biến là CVSS phân loại lỗ hổng dựa vào đặc điểm, tính khả thi và khả năng tái sản xuất của nó.

**Common Vulnerability Scoring System (CVSS):**

Lần đầu được giới thiệu năm 2005, CVSS là 1 thư viện rất phổ biến để đánh giá lỗ hổng. Nó ghi điểm dựa trên những yếu tố sau:

* Khai thác lỗ hổng đó có dễ không?
* Những khai thác đó có còn không?
* Lỗ hổng đó can thiệp vào tam giác CIA như thế nào?

|  |  |
| --- | --- |
| **Rating** | **Score** |
| None | 0 |
| Low | 0.1 - 3.9 |
| Medium | 4.0 - 6.9 |
| High | 7.0 - 8.9 |
| Critical | 9.0 - 10.0 |

|  |  |
| --- | --- |
| **Advantages of CVSS** | **Disadvantages of CVSS** |
| CVSS has been around for a long time. | CVSS was never designed to help prioritise vulnerabilities, instead, just assign a value of severity. |
| CVSS is popular in organisations. | CVSS heavily assesses vulnerabilities on an exploit being available. However, only 20% of all vulnerabilities have an exploit available ([Tenable., 2020](https://www.tenable.com/research)) . |
| CVSS is a free framework to adopt and recommended by organisations such as NIST. | Vulnerabilities rarely change scoring after assessment despite the fact that new developments such as exploits may be found. |

**Vulnerability Priority Rating (VPR)**

VPR là 1 thư viện hiện đại hơn để quản lí các lỗ hổng - được phát triển bởi Tenable, 1 nhà cung cấp các biện pháp chung để quản lí lỗ hổng.

Các lỗ hổng được đánh giá chủ yếu dựa vào rủi ro mà nó gây ra cho tổ chức, thay vì các yếu tố như tác động.

Không như CVSS, VPR có tính đến mức độ liên quan của 1 lỗ hổng. Ví dụ sẽ không có rủi ro nào được xem xét nếu lỗ hổng đó không áp dụng cho tổ chức

|  |  |
| --- | --- |
| **Advantages of VPR** | **Disadvantages of VPR** |
| VPR is a modern framework that is real-world. | VPR is not open-source like some other vulnerability management frameworks. |
| VPR considers over 150 factors when calculating risk. | VPR can only be adopted apart of a commercial platform. |
| VPR is risk-driven and used by organisations to help prioritise patching vulnerabilities. | VPR does not consider the CIA triad to the extent that CVSS does; meaning that risk to the confidentiality, integrity and availability of data does not play a large factor in scoring vulnerabilities when using VPR. |
| Scorings are not final and are very dynamic, meaning the priority a vulnerability should be given can change as the vulnerability ages. | *Intentionally left blank.* |

**Những cơ sở dữ liệu về lỗ hổng**

Trong an ninh mạng, bạn sẽ thường đi qua 1 lượng lớn những ứng dụng và dịch vụ khác nhau. Ví dụ, CMS trong khi tất cả có chung mục đích thường có những thiết kế và hành vi khác nhau.

Thật may mắn cho chúng ta, có nhiều mã nguồn trên internet theo dõi những lỗ hổng của phần mềm, hệ điều hành,… Bài này sẽ chỉ ra hai cơ sở dữ liệu chúng ta có thể dùng để tra cứu những lỗ hổng hiện có cho các ứng dụng được khám phá trong hành trình infosec như:

1. NVD (National Vulnerability Database)

2. Exploit - DB

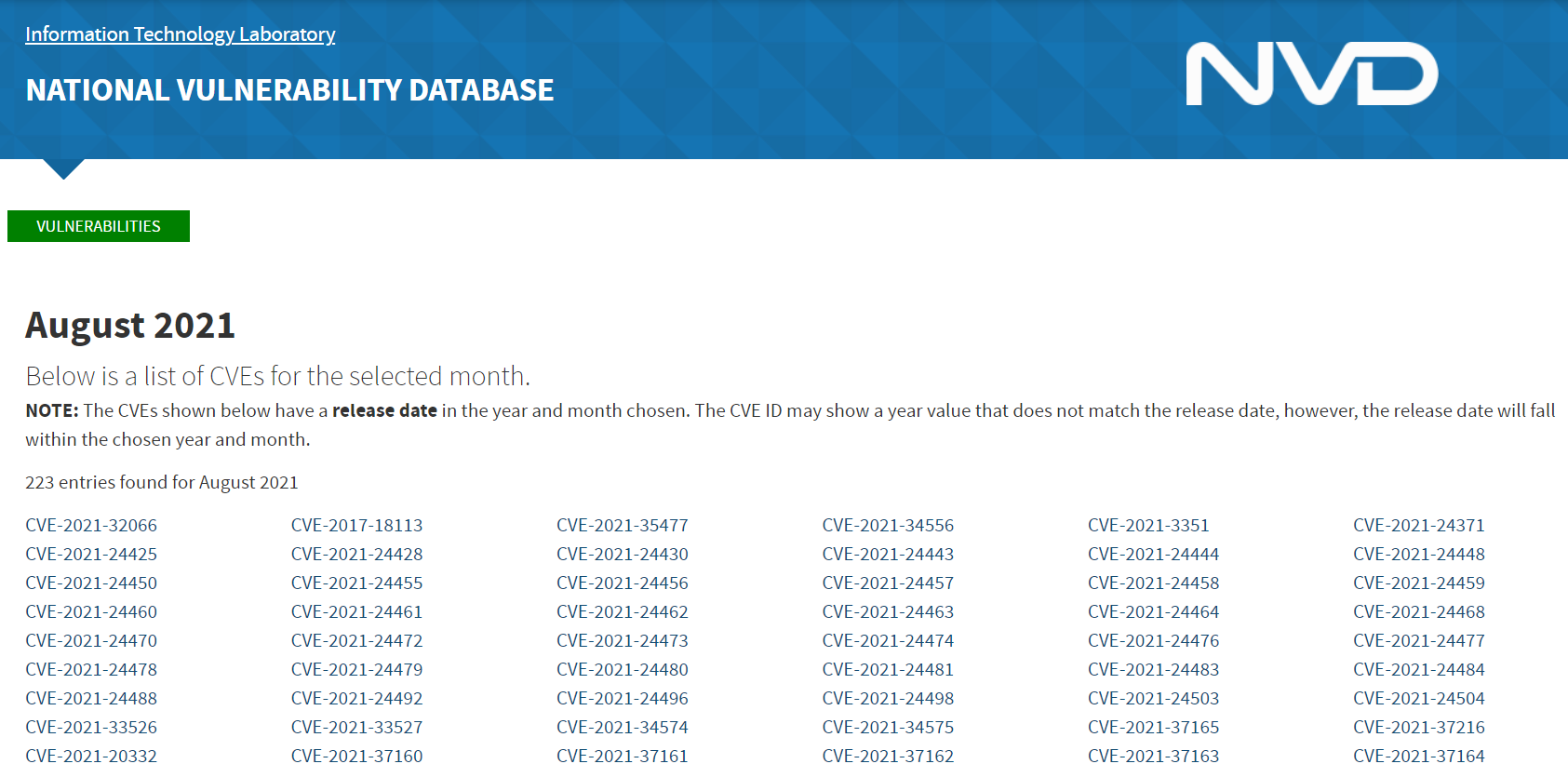
Trước khi chúng ta đào sâu vào hai mã nguồn này, hãy đảm bảo hiểu những thuật ngữ quan trọng sau đây:

|  |  |
| --- | --- |
| **Thuật ngữ** | **Mô tả** |
| Lỗ hổng | Điểm yếu hay lỗi thiết kế, thực thi hay những hành vi của hệ thống và ứng dụng |
| Khai thác | Khai thác là hành động sử dụng lỗ hổng trên hệ thống |
| Proof of Concept (PoC) | Kĩ thuật hay công cụ thường mô tả cách khai thác lỗ hổng |

**NVD - National Vulnerability Database**

NVD là 1 trang web liệt kê những lỗ hổng được phân loại trong an ninh mạng, các lỗ hổng bảo mật được phân loại theo “Common Vulnerabilities and Exposures” hay viết tắt là CVE.

Những CVE có định dạng sau: CVE-YEAR-IDNUMBER. Ví dụ, lỗ hổng mà mã độc nổi tiếng WannaCry được dùng là CVE-2017-0144.

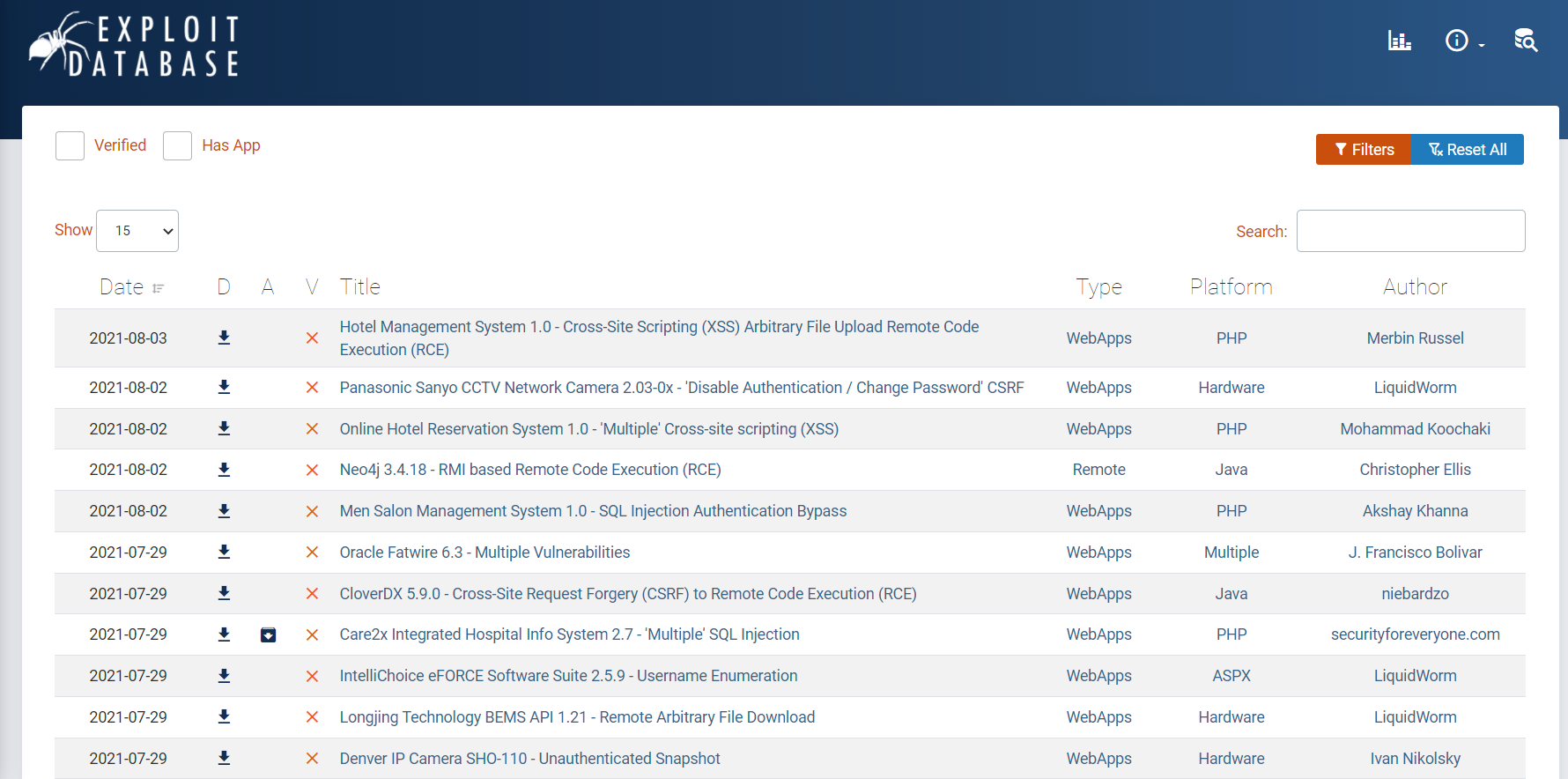


Trong khi trang web này giúp theo dõi những lỗ hổng mới, nó không tốt khi tìm kiếm những lỗ hổng cho 1 ứng dụng hay tình huống cụ thể.

**Exploit-DB:**

Là 1 mã nguồn mà chúng ta, những hackers sẽ thấy hữu ích hơn trong quá trình đánh giá. Exploit-DB chứa những khai thác cho phần mềm và ứng dụng được lưu trữ dưới tên, tác giả và phiên bản của chúng.

Chúng ta có thể dùng Exploit-DB để tìm kiếm các đoạn mã (được biết là PoC) được dùng để khai thác 1 lỗ hổng chỉ định.

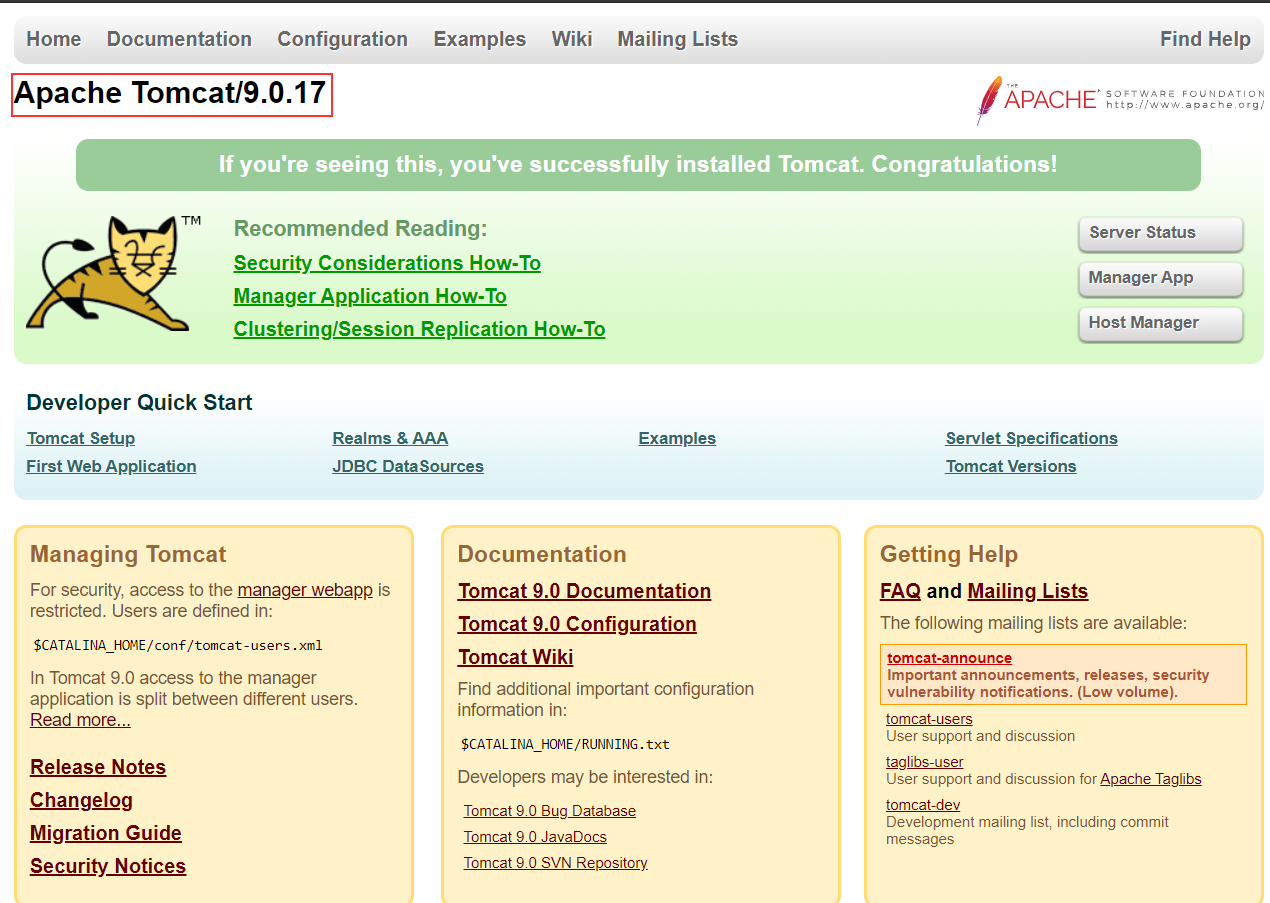


**Một ví dụ về tìm kiếm lỗ hổng:**

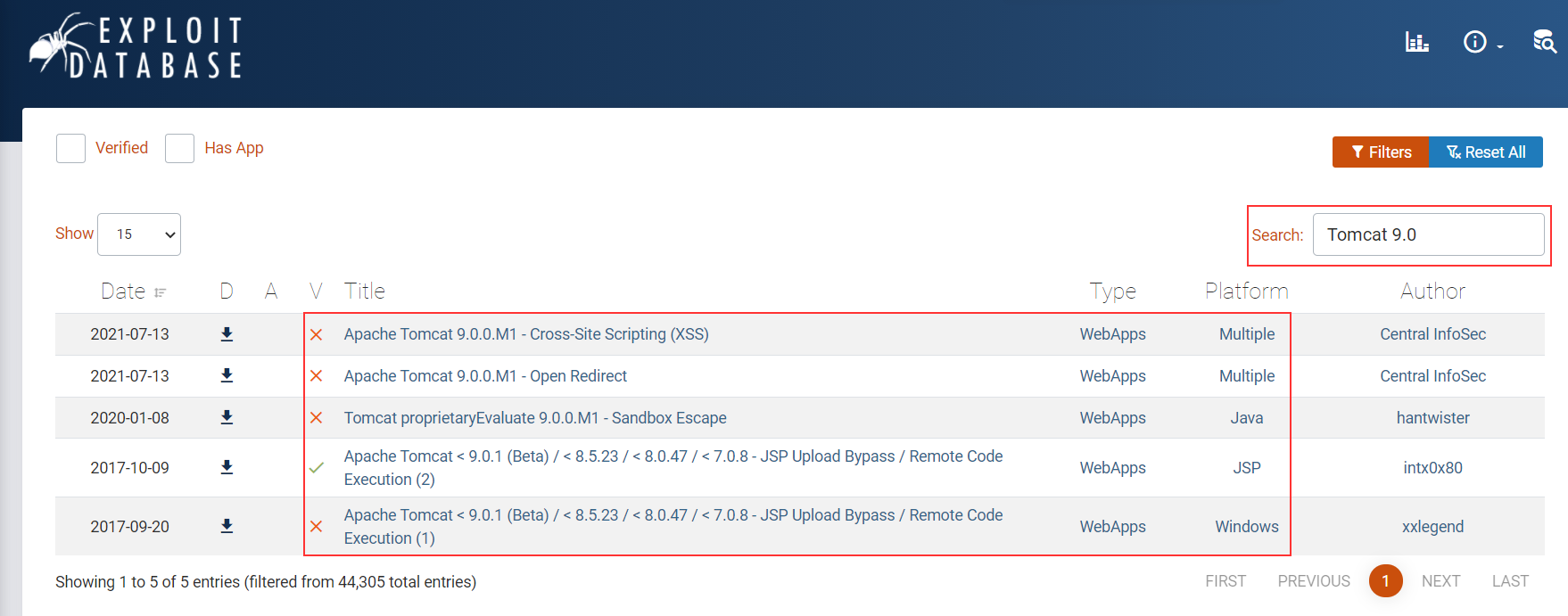
Trong bài này ta sẽ mô phỏng quá trình tìm kiếm 1 lỗ hổng nhỏ, kết hợp với một vài nghiên cứu của cơ sở dữ liệu các lỗ hổng.

Qua 1 bài đánh giá, bạn sẽ thường kết hợp nhiều lỗ hổng để có được kết quả. Ví dụ, chúng ta sẽ tận dụng lỗ hổng “Version Disclosure” để tìm kiếm phiên bản của ứng dụng. Sau đó dùng Exploit-DB để tìm kiếm bất cứ sự khai thác nào với phiên bản đó.

Trong hình dưới, chúng ta có thể thấy tên và phiên bản của ứng dụng này là Apache Tomcat 9.0.17



Tìm kiếm bất cứ khai thác nào có thể áp dụng cho Apache Tomcat 9.0.17 trên Exploit-DB.



Có 5 khai thác có thể thực hiện cho phiên bản này của ứng dụng.

**Exploit Vulnerabilities**

**Automated Vs. Manual Vulnerability Research**

Có rất nhiều công cụ và dịch vụ trong an ninh mạng để quét lỗ hổng. Từ thương mại đến mã nguồn mở, miễn phí, máy quét lỗ hổng là các phương tiện tiện lợi để nhanh chóng kiểm tra lỗi của ứng dụng.



Ví dụ, máy quét lỗ hổng Nessus có phiên bản miễn phí và thương mại. Bản thương mại tốn hàng ngàn bảng để có chính sách 1 năm được dùng trong những tổ chức cung cấp dịch vụ kiểm thử.

|  |  |
| --- | --- |
| **Advantage** | **Disadvantage** |
| Automated scans are easy to repeat, and the results can be shared within a team with ease. | People can often become reliant on these tools. |
| These scanners are quick and can test numerous applications efficiently. | They are extremely "loud" and produce a lot of traffic and logging. This is not good if you are trying to bypass firewalls and the likes. |
| Open-source solutions exist. | Open-source solutions are often basic and require expensive licenses to have useful features. |
| Automated scanners cover a wide range of different vulnerabilities that may be hard to manually search for. | They often do not find every vulnerability on an application. |

Những thư viện như Metasploit thường có các máy quét lỗ hổng cho một số mô-đun; Bạn sẽ được học trong bài tiếp theo.

Việc quét các lỗ hổng thủ công thường là vũ khí được người kiểm tra thâm nhập lựa chọn khi kiểm tra những ứng dụng hay chương trình cá nhân. Thực tế, quét thủ công sẽ liên quan đến việc tìm kiếm những lỗ hổng chung và sử dụng kĩ thuật tương tự như quét tự động.

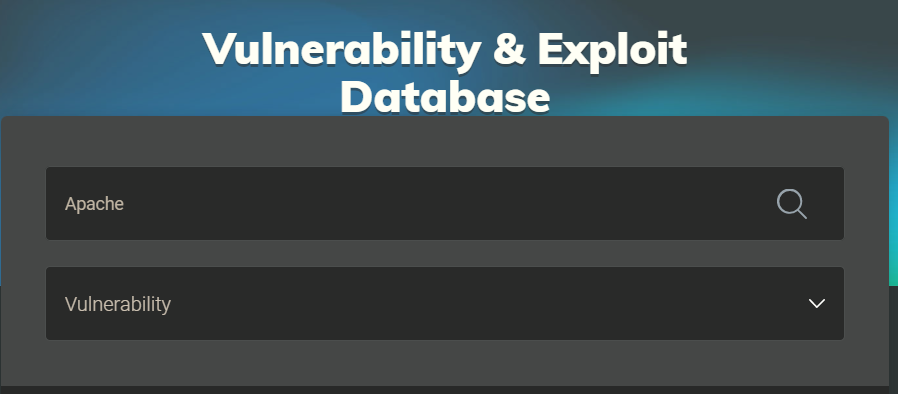
Cuối cùng, hai kĩ thuật liên quan đến việc kiểm tra lỗ hổng của ứng dụng hay chương trình. Những lỗ hổng đó là:

|  |  |
| --- | --- |
| **Lỗ hổng** | **Mô tả** |
| Cấu hình sai bảo mật | Cấu hình sai bảo mật liên quan đến các lỗ hổng do sự giám sát của nhà phát triển. Ví dụ, hiển thị thông tin máy chủ trong những tin nhắn giữa ứng dụng và kẻ tấn công. |
| Kiểm soát truy cập bị hỏng | Lỗ hổng này xảy ra khi kẻ tấn công có thể truy cập những phần của ứng dụng mà chúng không thể truy cập được. |
| Deserialization không an toàn | Đó là quá trình xử lí dữ liệu không an toàn khi gửi qua ứng dụng. Kẻ tấn công có thể gửi mã độc đến ứng dụng, nơi chúng sẽ được thực thi. |
| Tiêm | Lỗ hổng tiêm này tồn tại khi kẻ tấn công có thể đưa vào ứng dụng dữ liệu độc hại. Điều này là do không đảm bảo đầu vào không gây hại (santising) |

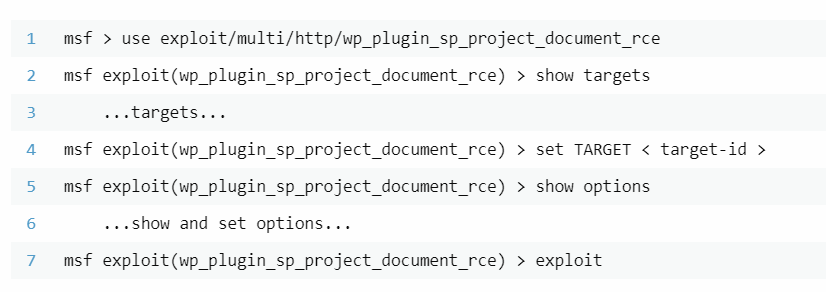
**Finding Manual Exploits:**

**Rapid 7**

Giống như nhiều dịch vụ khác như Exploit DB và NVE, Rapid 7 là 1 cơ sở dữ liệu nghiên cứu lỗ hổng. Bạn có thể lọc loại cơ sở dữ liệu (ứng dụng hay hệ điều hành)



Ngoài ra, nó duy trì những hướng dẫn để khai thác ứng dụng bằng việc dùng công cụ Metasploit. Ví dụ, bạn nhập trên Rapid 7 là “Wordpress Plugin SP Project & Document”, nơi bạn sẽ thấy những hướng dẫn về cách khai thác 1 lỗ hổng như vậy.

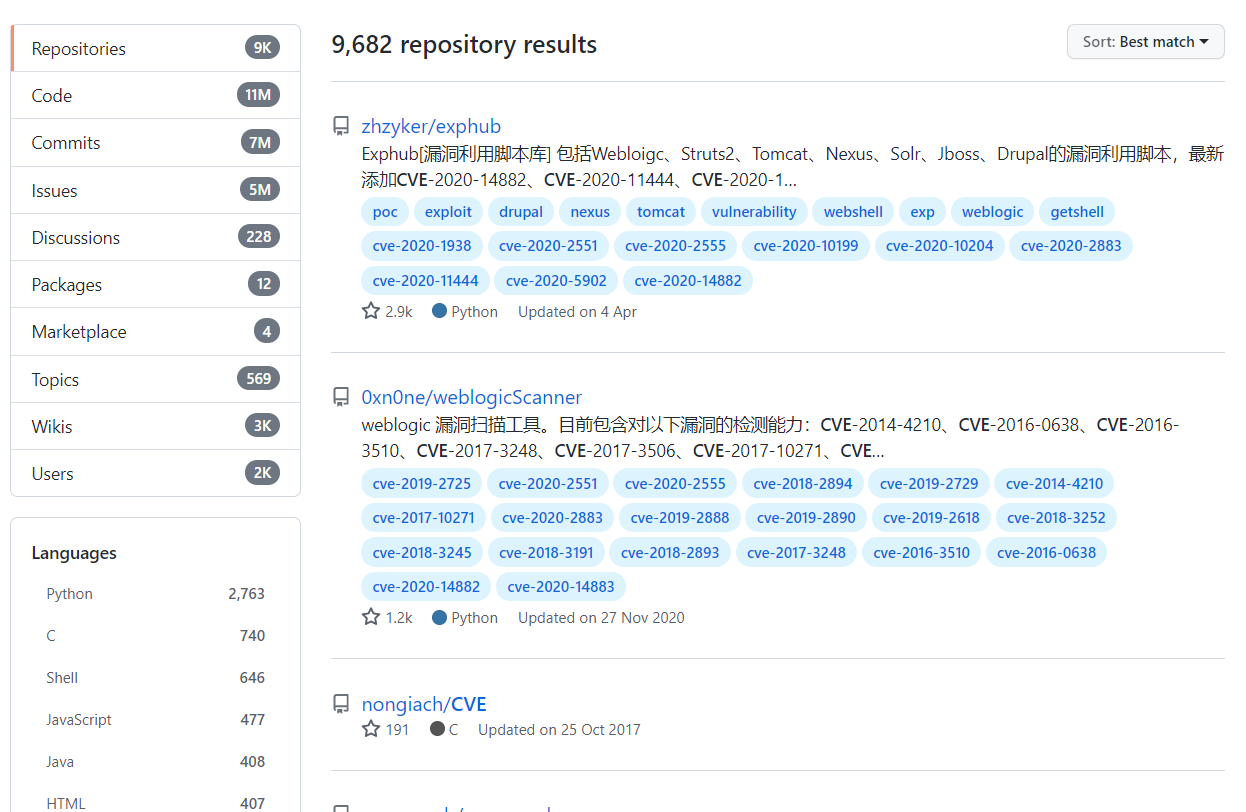


**Github**

Github là 1 dịch vụ web phổ biến được thiết kế cho các nhà phát triển phần mềm. Nó được dùng để lưu trữ và chia sẽ mã nguồn của các ứng dụng. Tuy vậy, những nhà nghiên cứu bảo mật đã chọn nền tảng này cũng vì những lí do trên. Họ lưu trữ và chia sẽ PoC trên Github, biến nó thành 1 cơ sở dữ liệu khai thác.

Github rất hữu dụng trong việc tìm các khai thác hiếm hoặc mới vì bất cứ ai cũng có thể tạo tài khoản và đăng tải- không có qui trình xác minh chính thức như với cơ sở dữ liệu khai thác khác.

Như đã nói, cũng có 1 nhược điểm là PoC có thể không hoạt động khi có ít hoặc không có hỗ trợ.

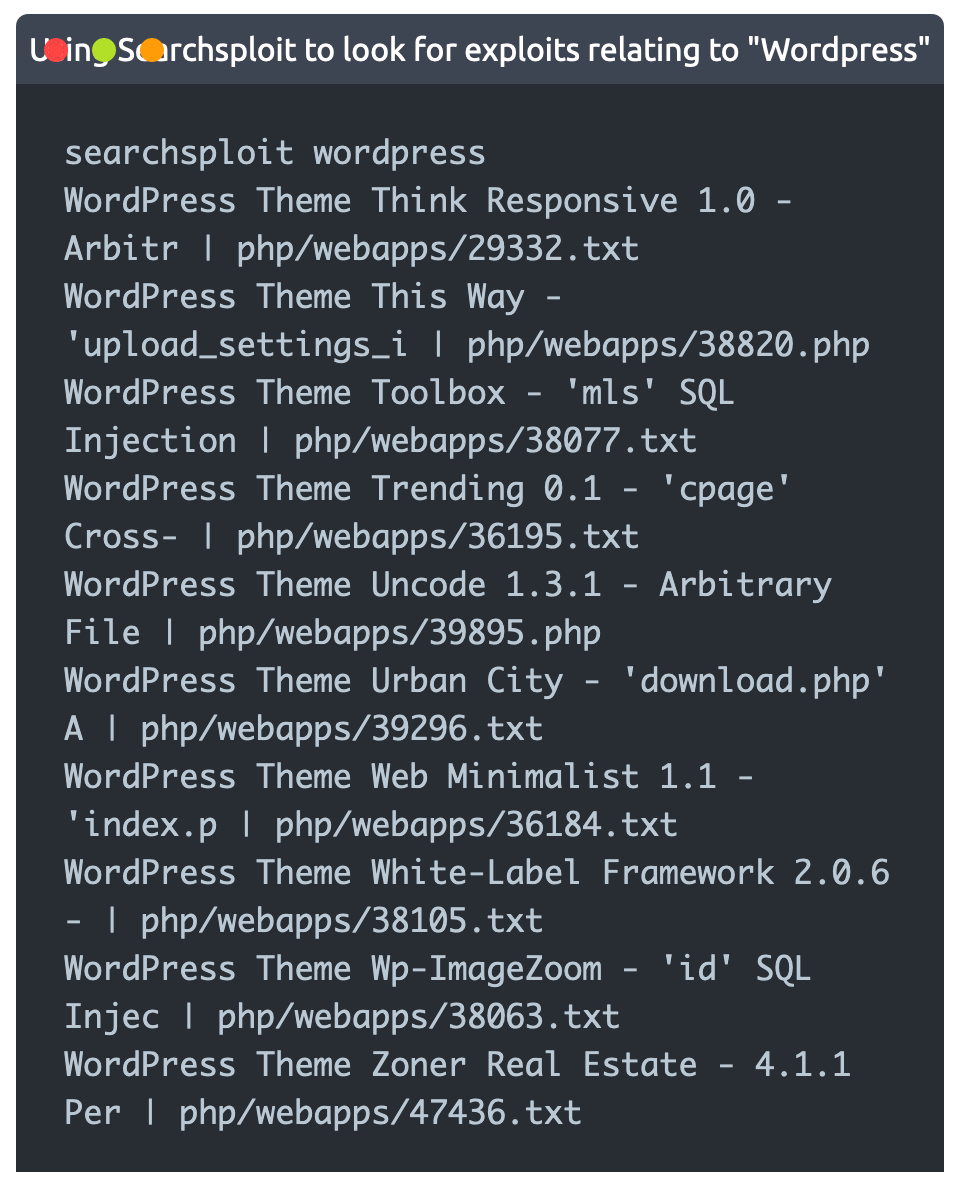


Github sử dụng 1 hệ thống đánh dấu và từ khóa. Chúng ta có thể tìm kiểm Github bằng từ khóa như PoC, vulnerability,….

**Searchsploit**

Searchsploit là 1 công cụ khả thi trên những bản phân phối kiểm thử như Kali Linux. Nó là 1 bản sao chép chính thức của Exploit-DB, chứa những khai thác trên hệ thống của bạn.

Bạn có thể tìm kiếm searchsploit bằng tên ứng dụng hay loại lỗ hổng.



**Example of Manual Exploitation**

Chúng ta có thể dùng thông tin thu thập được ở task 2 để khai thác lỗ hổng dịch vụ. Cuối cùng, một trong những lỗ hổng hiệu quả nhất mà chúng ta có thể khai thác là thực thi lệnh trên mục tiêu đang chạy ứng dụng hay dịch vụ đó.

Thực thi lệnh trên đối tượng sẽ cho phép chúng ta đọc tập tin hay thực thi những lệnh mà trước đây chúng ta không thể thực hiện chỉ bằng ứng dụng hay dịch vụ.

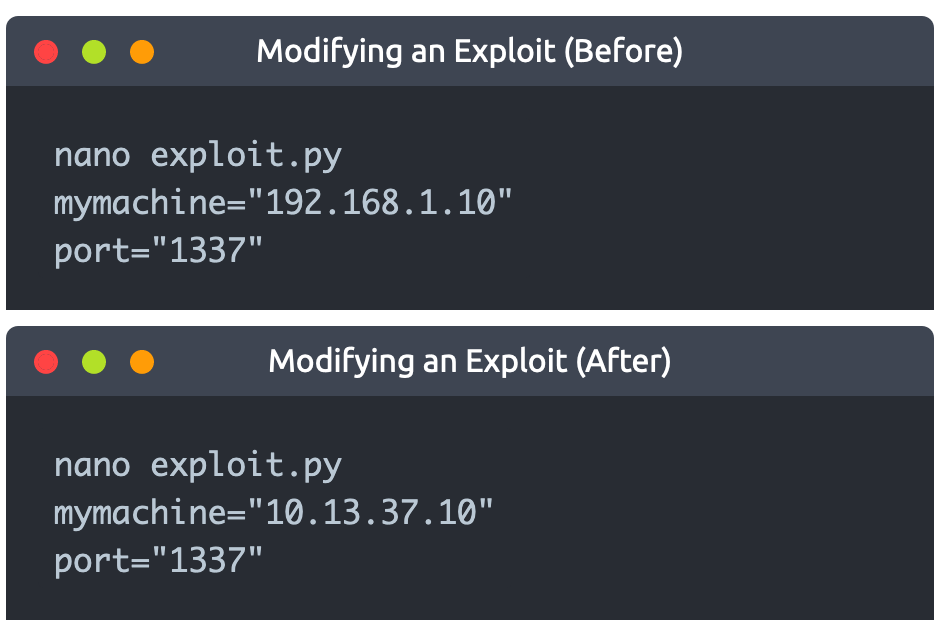
Ngoài ra, chúng ta có thể lạm dụng điều này để gain foothold ở máy ảo; foothold là sự truy cập vào console của máy ảo có lỗ hổng, nơi chúng ta có thể bắt đầu khai thác những ứng dụng hay máy ảo khác trong mạng.



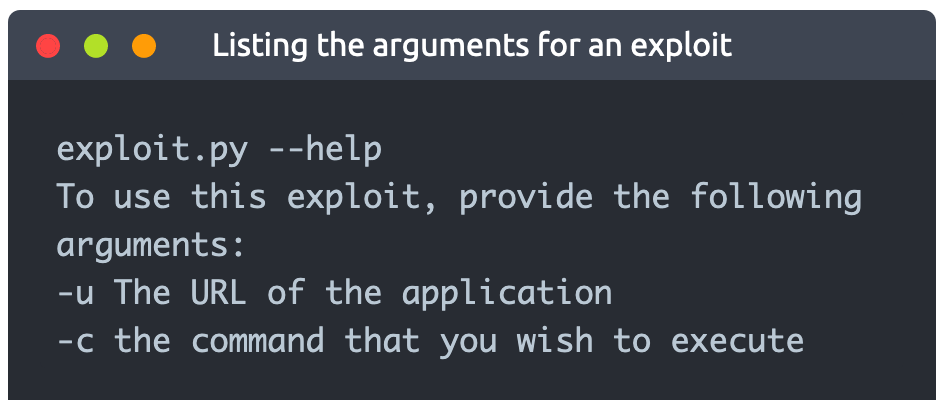
Chúng ta sẽ dùng 1 khai thác để thực thi code từ xa trên ứng dụng này.

Trước khi bắt đầu, điều đáng lưu ý là những khai thác hiếm khi ra khỏi hộp và sẵn sàng để sử dụng. Chúng thường yêu cầu 1 vài cấu hình trước khi sẽ hoạt động ở môi trường của chúng ta hay đối tượng. Mức độ cấu hình sẽ phụ thuộc và sự khai thác, vì vậy bạn sẽ thường tìm thấy nhiều khai thác cho cùng 1 lỗ hổng trên ứng dụng. Điều quan trọng là bạn phải tìm ra cách khai thác nào phù hợp hay hữu ích nhất với bạn.

Ví dụ, chúng ta có thể thấy 1 vài lựa chọn đã được thay đổi để phản ánh địa chỉ IP của máy ảo chúng ta đang tấn công.



Chỉ khi chúng ta cấu hình khai thác cho đúng, hãy đọc khai thác này để hiểu về cách sử dụng nó. Trong hình dưới, chúng ta cần cung cấp 2 đối số khi chạy khai thác:



Chúng ta sẽ làm như sau:

1. Sử dụng khai thác để đăng tải 1 tập tin độc hại đến ứng dụng có lỗ hổng chứa lệnh mà chúng ta muốn thực thi, nơi máy chủ web sẽ chạy nó.
2. Tập tin đầu tiên sẽ chứa 1 lệnh cơ bản để chúng ta kiểm tra xem khai thác có hoạt động không
3. Sau đó chúng ta sẽ đọc nội dung của tập tin đặt trên máy ảo.

