**Bài 5: Security Testing – Cross Site Scripting (XSS)**

Cross Site Scripting (XSS) (mà sao lại là XSS, sao không phải là CSS, bởi lẻ CSS là bạn nghĩ tới …CSS dùng thiết giao diện, cho nên đã dùng chữ X thay cho cái giá trị nào đó, như một số bài hướng dẩn “bạn hãy thay xxxx với giá trị của bạn”) xãy ra bất kỳ nơi nào một ứng dụng lấy dữ liệu không tin cậy và gửi nó tới client (browser) mà không được xác nhận, kiễm tra.

Vì thế cho phép attacker thực hiện các script nguy hiễm trong browser của nạn nhân có thể dẫn đến việc chiếm quyền điều khiển các trang người dùng, đánh lừa các trang web hoặc chuyển hướng người dùng tới các trang web độc hại.

**Các Kiểu của XSS**

* **Stored XSS – Stored XSS**được biết như persistent XSS xãy ra khi người dùng nhập dữ liệu được lưu trữ trên server như database/message/forum/comment. Sau đó nạn nhân có thể rút trích dữ liệu lưu trử từ ứng dụng web.
* **Reflected XSS – Reflected XSS**cũng được biết như non persistent XSS xãy ra khi đầu vào của người dùng được trả về ngay lập tức trả về bằng một ứng dụng web trong một kết quả tìm kiếm lổi hoặc đầu vào do người dùng cung cấp như là một phần của yêu cầu và không lưu trữ dữ liệu do người dùng cung cấp.
* **DOM Based XSS –**DOM Based XSS là một form của XSS khi source của dữ liệu trong DOM, tức luồng dữ liệu không bao giờ rời khỏi trình duyệt.

Lấy ví dụ cho dể hiểu nè, đôi khi lý thuyết là ác cho mộng cho người đọc nếu tác giả quá giỏi văn kaka.

EX: ứng dụng sử dụng dữ liệu không tin cậy trong việc khởi tạo mà không kiễm duyệt. Các ký tự đặc biệt đã bị lọt sổ.

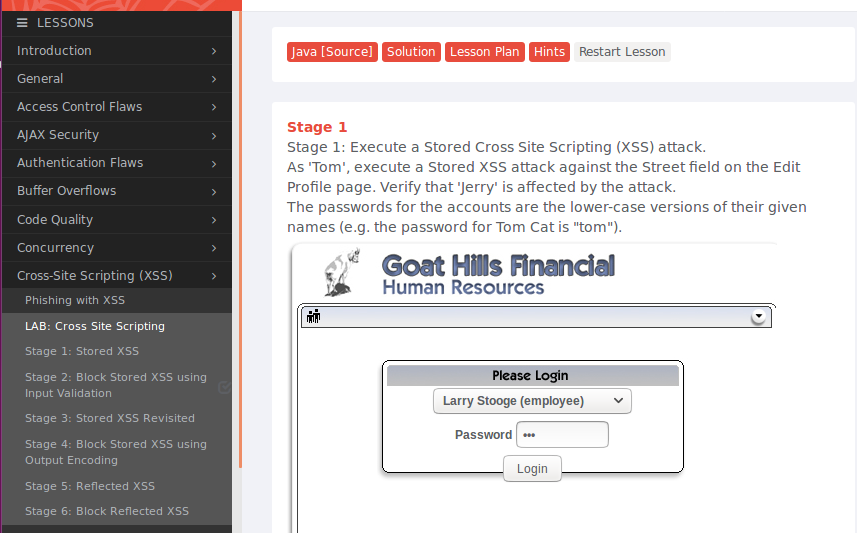
http://www.mysite.com/task/rule?query=do\_something

Bây giờ attacker thay thế đối số cho query trên trình duyệt của họ như sau:

[http://www.mysite.com/task/rule?query=<h3> Xin](http://www.mysite.com/task/rule?query=%3ch3%3e%20Xin) chao em iu”</h3>

**Thực Hành**

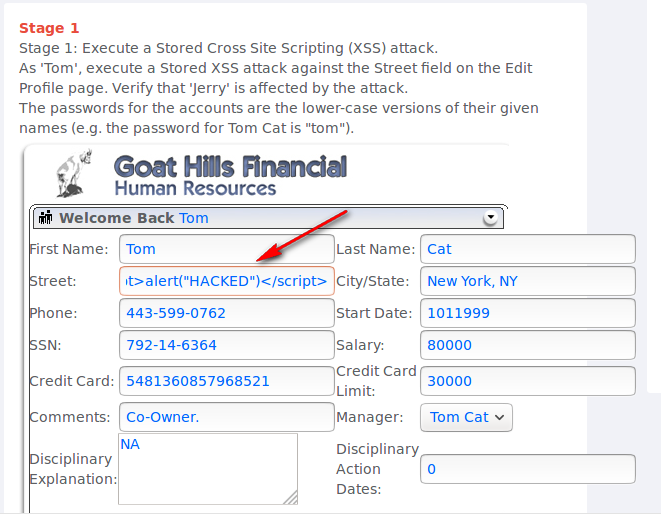
1. Login vào WebGoat và tìm đến phần Cross Site Scripting. Cho chúng ta thực hiện một tấn công XSS.



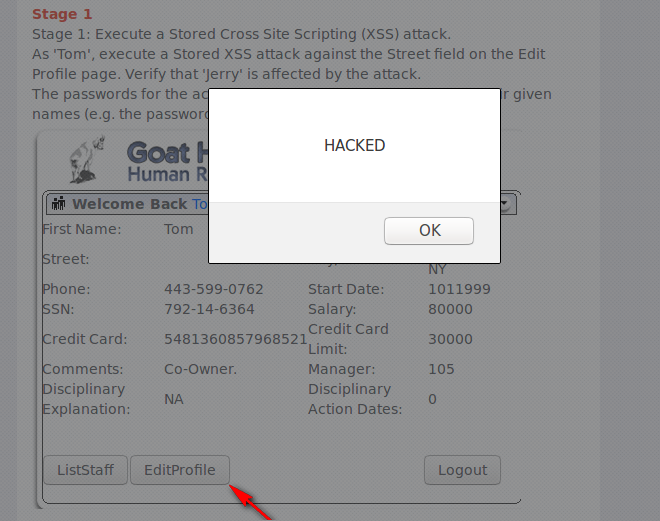
1. Giờ chúng ta login Tom với pass là ‘tom’ luôn. Click ‘view profile’ để vào trang edit. Khi đó tom là người tấn công, chúng ta tiêm vào một java script vào box edit đó.

              <script> alert(“HACKED”)</script>

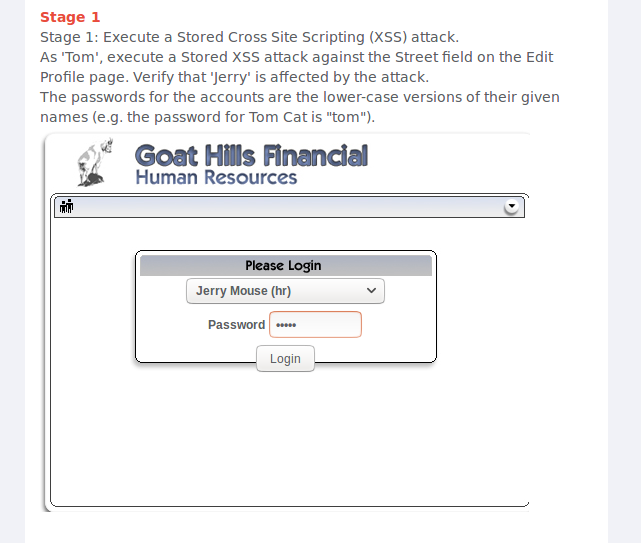
Có một số bạn phàn nàn là copy/paste code của AD rồi lổi tùm lum, mà thực là lổi mấy cái dấu nháy đó các bạn. Do AD soạn bài bằng word. Để AD tìm cách khác, còn giờ thông cảm tí. À mà biết đâu các bạn sữa tới sữa lui để chạy thì lại có thời gian đọc kỹ hơn thì sao keke.



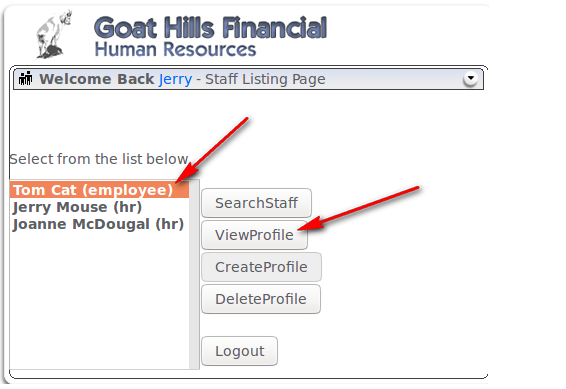
1. Ngay khi update thì bạn sẽ thấy một box alert với message “HACKED”. Điều này cũng có nghĩa là app có lổ hổng.



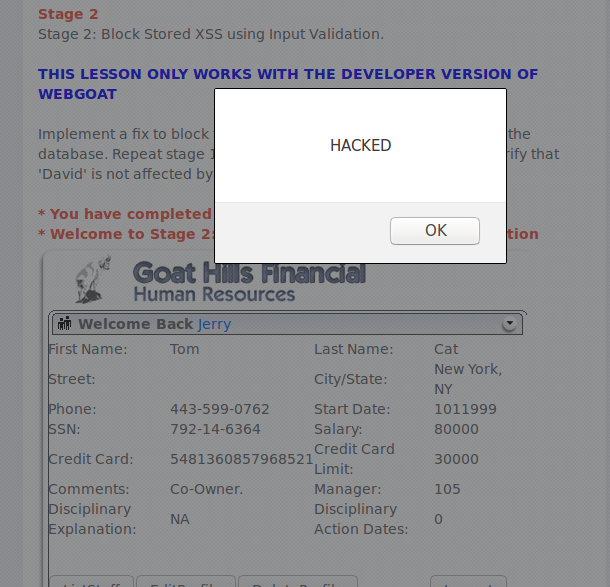
1. Bây giờ , chúng ta login và jerry( đây là giám đốc, hoặc nhân sự …) và kiếm tra nếu jerry bị nhiễm script trên.



1. Sau khi logging vào Jerry, chọn “Tom” và click xem profile



1. Trong xem profile của Tom từ tài khoản Jerry , anh ấy có thể nhận một message box:



1. một message box. AD chỉ đi tới đây thôi, còn làm cái gì khác thì các bạn có thể nghiên cứu thêm.

***Xin nhắc lại đây là serial giúp các bạn biết các lổ hổng của ứng dụng mà khi xây dựng tránh bị tấn công, chứ ko phải nghiên cứu để đi quậy phá, AD không chịu trách nhiệm đấu nhé.***

**Các Cơ Chế Ngăn Ngừa**

Developer phải bảo đảm rằng họ escape tất cả các dữ liệu không đáng tin cậy dựa vào ngữ cảnh HTML như body, thuộc tính, JavaScript, CSS, hoặc URL và dữ liệu sẽ được đặt vào.

Đối với các ứng dụng cần các ký tự đặc biệt như đầu vào, cần có cơ chế xác nhận mạnh mẽ tại chổ trước khi chấp nhận chúng như các đầu vào hợp lệ.

Một số lập trình viên chỉ validate bằng javascript mà quên cần thực hiện validate thêm 1 lớp nữa ở server site. Nếu người dùng vô hiệu hóa script trình duyệt thì lấy cái j mà check.

Rồi ok, các bạn có thắc mắc gì thì comment hoặc gửi mail cho AD nhé.

Chúc học vui vẻ!