



BẢO MẬT NHẬP MÔN

Bảo mật cơ bản cho developer



Phạm Huy Hoàng - Tôi đi code dạo

Lời tựa

Bảo mật là một vấn đề rất tốn kém và phức tạp. Gần như hệ thống nào cũng có lỗ hổng (cả phần mềm lẫn phần cứng), các hacker có thể thông qua các lỗ hổng này để tấn công hệ thống.

Việc đảm bảo hệ thống bảo mật là trách nhiệm của rất nhiều bên: Sysadmin, network, manager và developer. Trong phạm vi sách, mình sẽ cùng các bạn tiếp cận khía cạnh bảo mật dưới góc nhìn của một developer.

Những kiến thức trong ebook này cô cùng cơ bản, dễ học, nhưng chúng sẽ vô cùng hữu ích, giúp bạn tránh phải những sai lầm bảo mật “ngớ ngẩn, cơ bản” khi code. Dù cho bạn code C hay C++, Java C# hay PHP, bạn cũng sẽ học được vài điều bổ ích qua series này.

Trách nhiệm của developer là phải đảm bảo rằng code mình viết ra sẽ không có lỗi bảo mật. Trong ebook này, chúng ta đóng vai hacker để tấn công hệ thống mình viết. Thông qua đó, chúng ta sẽ cùng tìm hiểu về những lỗ hổng bảo mật thường thấy khi code và tìm cách vá lỗi.

Đa phần các lỗi bảo mật cơ bản đã được ngăn chặn trong các framework. Tuy vậy, nhiều trang web vẫn bị dính một số lỗi vì sự ... ngớ ngẩn hoặc sơ suất của chính developer. Do đó, hãy đọc kĩ ebook và cố gắng áp dụng những kiến thức này vào code để tránh dính các lỗi này nhé.

Đây là series hướng dẫn bảo mật cho developer, không phải là hướng dẫn làm hacker. Kiến thức trong ebook giúp bạn code, giúp bạn vá lỗi chứ không giúp bạn tấn công hệ thống khác hay lừa đảo người dùng. Bạn nào nghiêm túc muốn tầm sư học đạo về bảo mật có thể tìm thánh bảo mật Juno_okyo nhé.

Cảnh báo

Trước khi dạy võ, sư phụ luôn dặn các đồ đệ rằng: Học võ là để cường thân kiện thể, hành hiệp giúp đời, không phải để đi bắt nạt kẻ yếu. Trước khi bắt đầu sách, mình cũng muốn khuyên các bạn điều tương tự: Học về security để xây dựng hệ thống bảo mật tốt hơn, để giúp đỡ hệ thống khác, chứ không phải để đi hack hay phá hoại.

Vì lý do đạo đức, nếu phát hiện lỗi trong các hệ thống khác, các bạn nên thông báo cho quản trị chứ đừng nên phá hoại. Ranh giới giữa “tìm hiểu lỗ hổng” và “phá hoại hệ thống” nó mong manh lắm. Với các hệ thống quan trọng, bạn có thể bị truy tố để vào tù bóc lịch cho lỗ ass nở hoa chứ chẳng chơi.

Mục lục

PHẦN 1 – BẢO MẬT NHẬP MÔN	4
GIAO THỨC HTTP “BẢO MẬT” ĐẾN MỨC NÀO?	5
Ôn lại về HTTP	5
Sơ lược về Man-in-the-middle attack	5
Cách phòng chống	6
Lưu ý	7
Tổng kết	10
LỖ HỔNG BẢO MẬT XSS NGUY HIỂM ĐẾN MỨC NÀO?	11
Giới thiệu về XSS	11
Những dạng XSS	11
Cách phòng tránh	13
Lời kết	14
LƯU TRỮ COOKIE – TƯỚNG KHÔNG HẠI AI NGỜ HẠI KHÔNG TƯỚNG	15
Cookie – Chiếc “bánh qui” vô hại?	15
Bánh qui nho nhỏ, đầy những lỗ to to	15
Cách phòng chống	16
SQL INJECTION – LỖ HỔNG BẢO MẬT THẦN THÁNH	17
Tại sao SQL Injection lại “thần thánh”?	17
Hậu quả của SQL Injection	17
Tấn công SQL Injection như thế nào?	18
Cách phòng chống	18
Kết luận	19
INSECURE DIRECT OBJECT REFERENCES – GIẤU ĐẦU LỜI ĐUÔI	20
Lỗi gì mà tên dài rứa??	20
Cách lợi dụng lỗ hổng	20
Cách phòng chống	22
CSRF – NHỮNG CÚ LỪA NGOẠN MỤC	23
Cơ bản về CSRF	23
Các kiểu tấn công thường gặp	23
Lưu ý	25
Phòng chống cho website	26
Tổng kết	26
ẨN GIẤU THÔNG TIN HỆ THỐNG – TRÁNH CON MẮT NGƯỜI ĐỜI VÀ KẺ XẤU	27
Thông tin hệ thống là gì?	27
Chúng ta để thông tin hệ thống “hớ hênh” như thế nào?	27
Những hậu quả của việc “lộ hàng”	29
Giấu như thế nào cho đúng?	29
QUẢN LÝ NGƯỜI DÙNG – TƯỚNG DỄ ĂN MÀ KHÔNG ĐƠN GIẢN	30
Úi giờ! Đăng kí đăng nhập có gì khó?	30
Quan trọng nhất – Không lưu mật khẩu!	30
Làm thế nào khi người dùng quên mật khẩu?	31
Chống việc đoán mò mật khẩu	31

Những biện pháp nho nhỏ tăng cường bảo mật	32
PHẦN 2 – CASE STUDY.....	33
LỖ HỔNG BẢO MẬT KHỦNG KHIẾP CỦA LOTTE CINEMA.....	34
Đăng nhập hả? Chỉ cần một bảng User, hai cột Username và Password là xong.....	34
Vậy mã hóa là được chứ gì, lằm trò!!	34
Ồi giờ phức tạp thế, cùng lằm thì lộ password trên trang của mình thôi mà	35
Lỗ hổng bảo mật khủng khiếp của Lotte Cinema	36
TÔI ĐÃ HACK “TƠI TẢ” WEB SITE CỦA LOTTE CINEMA NHƯ THẾ NÀO?	37
Giới thiệu	37
Bắt đầu “câu cá”	37
Câu nhảm ... “cá mập”	39
Bonus thêm “cá voi”	39
Kết luận	41
Update (30/08/2016).....	42
LOZI.VN ĐÃ “VÔ Ý” ĐỂ LỘ DỮ LIỆU 2 TRIỆU NGƯỜI DÙNG NHƯ THẾ NÀO?.....	44
Dò tìm từ web	44
Đến app mobile.....	45
Quá trình xử lý lỗi.....	47
Nhận xét.....	47
Thay lời kết	49
Về tác giả.....	50
Thông tin liên lạc:	50

PHẦN 1 – BẢO MẬT NHẬP MÔN

Kiến thức cơ bản về bảo mật và một số lỗi hỏng bảo mật thường gặp

GIAO THỨC HTTP “BẢO MẬT” ĐẾN MỨC NÀO?

Ôn lại về HTTP

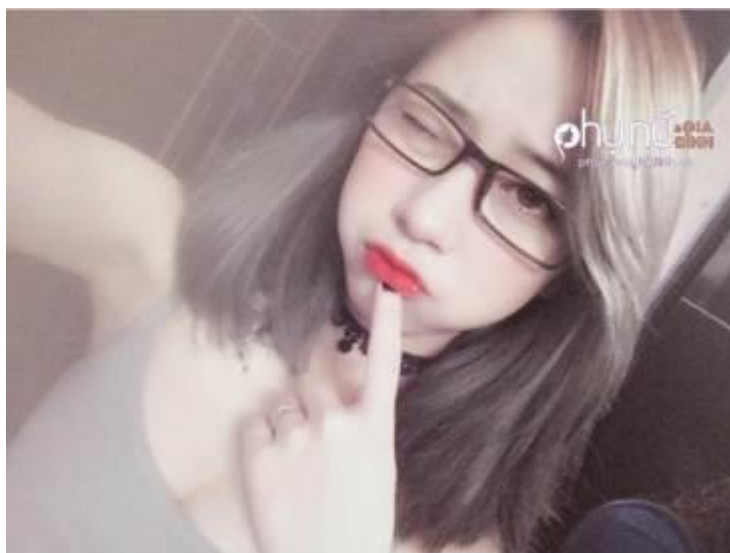
HTTP là một giao thức dùng để truyền nhận dữ liệu (Xem thêm ở đây). Hiện tại, phần lớn dữ liệu trên Internet đều được truyền thông qua giao thức HTTP. Các ứng dụng Web hoặc Mobile cũng gọi Restful API thông qua giao thức HTTP.

Tuy nhiên, nhược điểm của HTTP là dữ liệu được truyền dưới dạng plain text, không hề được mã hoá hay bảo mật. Điều này dẫn đến việc hacker có thể dễ dàng nghe lén, chôm cắp và chỉnh sửa dữ liệu. Người ta gọi kiểu tấn công này là Man-in-the-middle attack, viết tắt là MITM.

Sơ lược về Man-in-the-middle attack

Hãy tưởng tượng bạn đang tán tỉnh một em gái dễ thương mặt cute ngực to đánh khủng tên Linh. Để tăng tính lãng mạn, bạn không nhắn tin mà trực tiếp viết thư gửi cho nàng. Lúc này, bạn là client, bé Linh là server, việc gửi thư là giao thức HTTP.

Đương nhiên, hoa đẹp thì lắm ruồi bu. Có một thằng hacker xấu xa bí ối tìm cách phá rối bạn, ta tạm gọi thằng này là Hoàng cờ hó.



Search Linh phát nó ra con bé Linh lộ clip 18+ luôn....

Thằng Hoàng cờ hó có thể phá rối bạn bằng những cách sau:

1. Sniff packet để đọc lén dữ liệu

Bạn hí hửng bỏ thư vào hòm thư, chờ bức thư bay đến chỗ Linh. Thư đang trên đường tới, thằng Hoàng bắt được, mở bức thư ra xem, biết được hết những lời tâm tình ủ ê mà bạn dốc cạn tấm lòng ra viết.

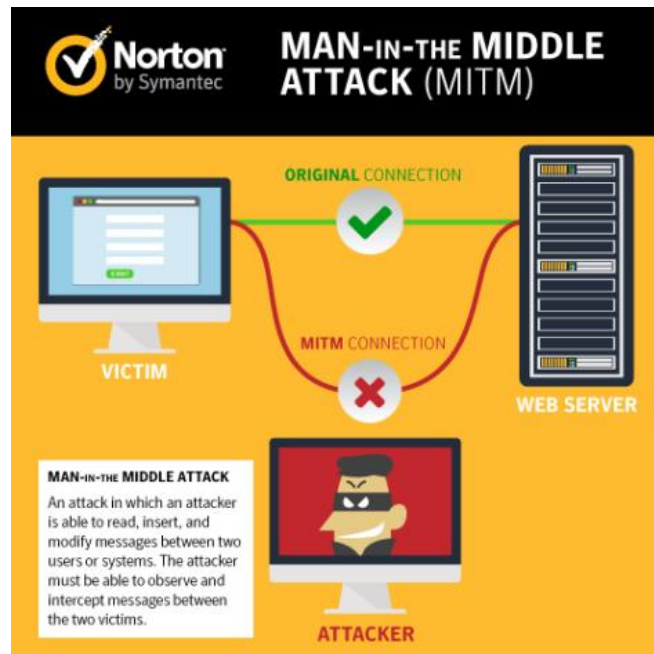
Trong thực tế, khi bạn gửi username, password qua HTTP, hacker có thể dễ dàng chôm username, password này bằng cách đọc lén các packet trong mạng. (Bạn gửi clip 18+ thì nó cũng chôm được nốt).

2. Sửa đổi packet

Không chỉ đọc trộm, thằng Hoàng cờ hó kia còn có thể sửa thư của bạn. Bạn khen Linh đẹp như Maria Ozawa thì nó sửa thành Happy Polla. Linh reply lại, hẹn bạn đi nhà nghỉ lúc 5h thì nó sửa thành 5h15.

Bạn vẫn không hay biết thư đã bị tráo gò cả. Đến lúc đọc xong, 5h15 ra nhà nghỉ thì đã thấy thằng cờ hó và Linh tay trong tay dắt nhau ra. (Thằng H yếu sinh lý nên 15p là xong, các bạn nên thông cảm cho nó).

Trong thực tế, hacker có thể thay đổi nội dung bạn nhận được từ server, làm thay đổi thông tin hiển thị trên máy bạn. Cả 2 trường hợp này đều khá nguy hiểm vì bạn không hề biết mình bị tấn công.



Kiến thức này thuộc dạng vô cùng cơ bản, nhiều người đã nói rồi nên mình sẽ không giải thích kĩ về khía cạnh kĩ thuật. Các bạn có thể tự tìm Google tìm hiểu thêm.

Cách phòng chống

Các giải pháp chống MITM trong mạng LAN thường do SysAdmin hoặc các bạn chuyên bảo mật lo, thông qua việc cài đặt thiết lập hệ thống. Là developer, cách phòng chống cơ bản nhất chúng ta có thể làm là sử dụng giao thức HTTPS cho ứng dụng, bằng cách thêm SSL Certificate.

Dữ liệu giao tiếp qua HTTPS đã được mã hoá nên người ngoài không thể đọc trộm hay chỉnh sửa được. Cách này tương tự như việc bạn và Linh viết mail cho nhau bằng teencode, thằng Hoàng cờ hó kia có đọc trộm mail cũng không hiểu hay sửa thư được.

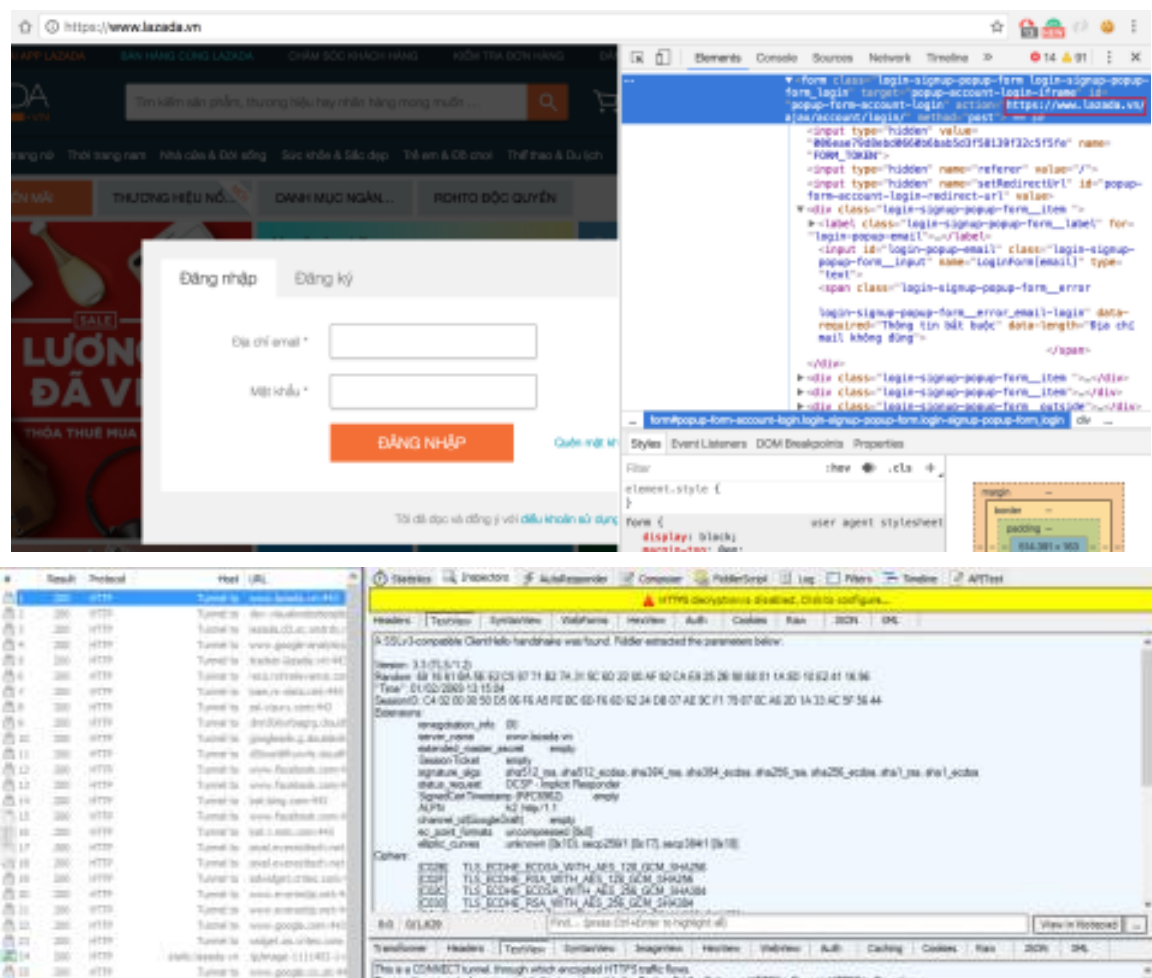
Tuy độ bảo mật của HTTPS vẫn chưa phải là tuyệt đối, nó vẫn cao hơn nhiều so với chỉ dùng HTTP thuần. Ngoài ra, nếu trang web của bạn chưa thể tích hợp https, bạn có thể tích hợp chức năng đăng nhập thông qua Facebook, Google. Tuy hacker vẫn có thể chôm cookie của người dùng, nhưng ít ra họ không bị lộ username và password.

Lưu ý

Hiện tại nhiều trang web vẫn sử dụng “https giả cầy” – chỉ sử dụng https ở những trang login và những trang có dữ liệu nhạy cảm. Cách làm này vẫn tồn tại khá nhiều nguy hiểm. Hiện tại, mình sử dụng Fiddler để demo ở local. Tuy nhiên, hacker có thể làm các trò này khi dùng chung LAN/WLAN với bạn. Do đó, cần hết sức cẩn thận khi dùng wifi chùa/wifi công cộng nhé.

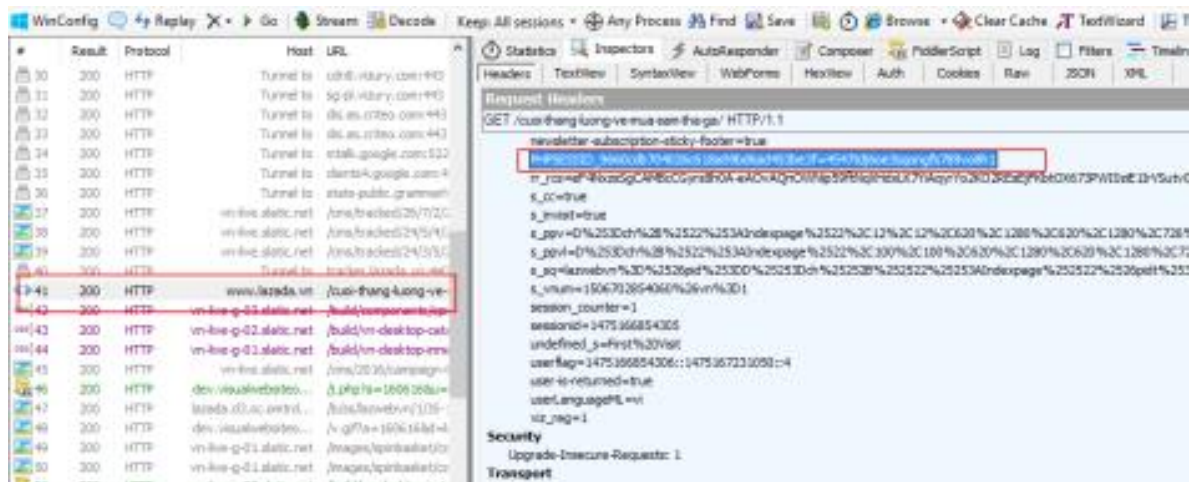
Ví dụ 1 – Lazada

Phần đăng nhập của trang này dùng https, do vậy mình không thể sniff được username, password.



Dữ liệu truyền qua SSL đã bị mã hoá nên không thể “đọc lên” được

Tuy nhiên, các trang khác của lazada vẫn dùng http. Khi người dùng vào các trang này mình có thể chôm được cookie, sử dụng cookie này để đăng nhập như thường.



Dùng Fiddler đọc lên cookie



Dùng EditThisCookie để dump cookie và đăng nhập như thường



Ngày xưa, khi Facebook chưa dùng https, tụi mình cũng dùng cách này để sniff và đăng nhập account facebook của người khác.

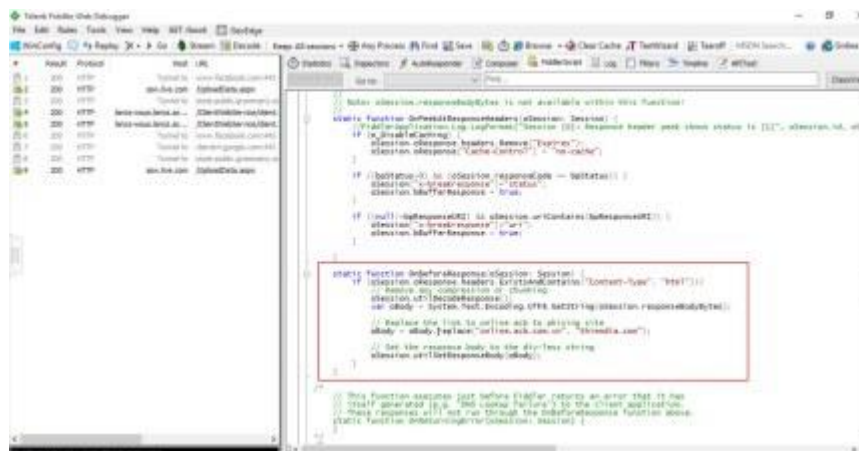
Ví dụ 2 – Ngân hàng ACB

Lần này mình sẽ lấy trang web của Ngân hàng ACB ra làm ví dụ. Trang này có sử dụng HTTPS cho trang giao dịch, nhưng trang chủ vẫn là HTTP.



Link ngân hàng trực tuyến dẫn đến online.acb.com.vn

Mình có thể sửa packet để dẫn người dùng tới trang lừa đảo.



Đoạn code này đổi nội dung HTML mà client nhận được

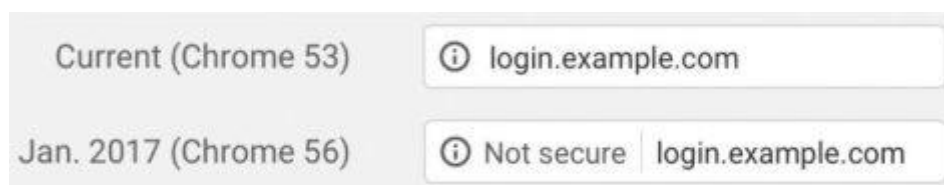


Đường link đã bị đánh tráo mà client không hay biết gì

Một số trường hợp khác, trang web dùng HTTPS nhưng vẫn tải hình ảnh, javascript, css qua http. Hacker vẫn có thể dễ dàng sửa nội dung javascript, trộm cookie như thường. Do đó, Google khuyến cáo sử dụng https cho toàn bộ các trang và các link chứ đừng để kiểu giả cầy như thế này nhé.

Tổng kết

Hiện tại Chrome cũng đang có kế hoạch thị các trang HTTP là không an toàn để cảnh báo cho người dùng. Ở những phiên bản sau, bạn sẽ thấy chữ “Not secure” trên thanh địa chỉ nếu trang web chỉ sử dụng HTTP.



Hai điều quan trọng nhất về HTTP rút ra từ bài viết:

- HTTP không an toàn hay bảo mật. Tuyệt đối không bao giờ submit thông tin quan trọng (mật khẩu, số thẻ ngân hàng) qua HTTP!
- Sử dụng http để duyệt web cũng giống như nện gáo mà không cần BCS. Nhiều khi dính bệnh chết lúc nào chẳng biết đấy!

LỖ HỒNG BẢO MẬT XSS NGUY HIỂM ĐẾN MỨC NÀO?

Giới thiệu về XSS

XSS (Cross Site Scripting) là một lỗi bảo mật cho phép hacker nhúng mã độc (javascript) vào một trang web khác. Hacker có thể lợi dụng mã độc này để deface trang web, cài keylog, chiếm quyền điều khiển của người dùng, dụ dỗ người dùng tải virus về máy. Các bạn có thể xem thêm demo trong vụ hack Lotte Cinema trước đây.

Đây là một trong những lỗi bảo mật thường gặp nhất trên các trang Web. Các hệ thống từ lớn đến nhỏ như Facebook, Twitter, một số forum Việt Nam, ... đều từng dính phải lỗi này. Do mức độ phổ biến và độ nguy hiểm của nó, XSS luôn được vinh dự được nằm trong top 10 lỗi bảo mật nghiêm trọng nhất trên OWASP (Open Web Application Security Project).



Để tóm tắt, xin trích dẫn vài câu của thánh bảo mật Juno_okyo, người vừa hack 3 triệu tài khoản của server X nào đó.

"Ờ thì nghe cũng có vẻ nguy hiểm đấy, nhưng sao tôi thấy ông hay viết về XSS thế? Rảnh quá hả!?"

À... một lỗi vừa phổ biến, nằm top 10 OWASP, lại vừa nguy hiểm, có thể kết hợp tốt với các lỗi khác. Nhưng dễ tìm, dễ fix, đã thế còn được tính bug bounty nữa.

Những dạng XSS

Trước đây, XSS thường nhắm vào code render HTML từ phía Server, ta gọi là Server XSS. Hai dạng Server XSS thường gặp là Persistent XSS và Reflected XSS. Ở đây, mình sẽ lấy một thanh niên tên Khoa ra làm ví dụ. Khoa là một sinh viên ĐH FPT, là fan của blog [tôi đi code dạo](http://toidicodedao.com/), thích lên thi*ndia để tìm địa điểm mátxa.

1. Persistent XSS

Trên forum thi*ndia, khi bạn post một comment vào topic, server sẽ lưu comment bạn post và hiển thị dưới dạng HTML. Khi Khoa post "Em muốn tìm JAV", server sẽ lưu lại và hiển thị như sau:

```
<div class="comment">
  <p>Em muốn tìm JAV</p>
</div>
```

Tuy nhiên, Khoa lại không hiển thị như thế. Do mới học về XSS, Khoa không nhập text mà nhập nguyên đoạn script `alert('XXX')` vào khung comment. Lúc này, HTML của trang web sẽ trở thành:

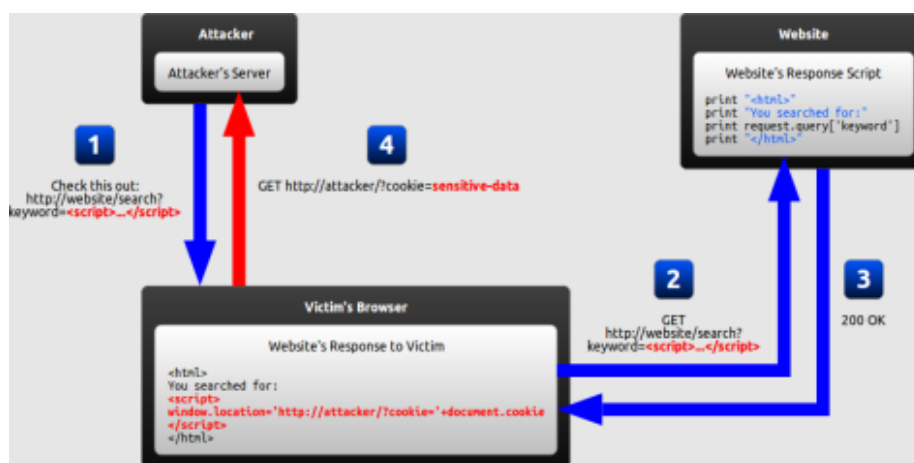
```
<div class="comment">
  <p><script>alert('XXX')</script></p>
</div>
```

Trình duyệt sẽ chạy đoạn script này, hiển thị cửa sổ alert lên. Khoa đã chèn được mã độc vào `thi*ndia`, thực hiện tấn công XSS thành công. (Lưu ý: Mình chỉ ví dụ thôi, `thi*ndia` không bị lỗi XSS đâu nhé, các bạn không nên thử).

Trong kiểu tấn công này, mã độc được lưu trong database trên server, hiển thị ra với toàn bộ người dùng, do đó ta gọi nó là Persistence XSS. Bất kì ai thấy comment của Khoa đều bị dính mã độc này, do đó kiểu tấn công này có tầm ảnh hưởng lớn, khá nguy hiểm.

2. Reflected XSS

Với cách tấn công này, hacker chèn mã độc vào URL dưới dạng query string. Khi người dùng ngáo ngơ nhấp vào URL này, trang web sẽ đọc query string, render mã độc vào HTML và người dùng “dính bẫy”.



Quay lại với Khoa. Do xin địa chỉ mất xa hoài nhưng không được share, Khoa cay cú, quyết định trả thù các đàn anh. Khoa bèn gửi đường một đường link giả JAV vào mail các đàn anh. Nội dung đường link: [http://thi*ndia.com?q=<script>deleteAccount\(\);</script>](http://thi*ndia.com?q=<script>deleteAccount();</script>). Khi các đàn anh click link này, họ sẽ vào trang thiendia. Sau đó server sẽ render `<script>deleteAccount();</script>`, gọi hàm `deleteAccount` trong JavaScript để xóa account của họ.

Tầm ảnh hưởng của ReflectedXSS không rộng bằng Persistence XSS, nhưng mức độ nguy hiểm là tương đương. Hacker thường gửi link có mã độc qua email, tin nhắn, ... và dụ dỗ người dùng click vào. Do đó các bạn đừng vì ham JAV mà click link bẫy bả nhé,

3. Client XSS

Gần đây, khi JavaScript dần được sử dụng nhiều hơn, các lỗi Client XSS cũng bị lợi dụng nhiều hơn. Do JavaScript được sử dụng để xử lý DOM, mã độc được chèn thẳng vào trong JavaScript.

Các lỗ hổng dạng này khó tìm và phát hiện hơn Server XSS nhiều (Xem ví dụ: <http://kipalog.com/posts/To-da-hack-trang-SinhVienIT-net-nhu-the-nao>).

Where untrusted data is used			
	XSS	Server	Client
Data Persistence	Stored	Stored Server XSS	Stored Client XSS
	Reflected	Reflected Server XSS	Reflected Client XSS

- ☐ DOM Based XSS is a subset of Client XSS (where the data source is from the DOM only)
- ☐ Stored vs. Reflected only affects the likelihood of successful attack, not the nature of vulnerability or the most effective defense

Cách phòng tránh

Tôn chỉ của series “Bảo mật nhập môn” là: Hack để học, chứ đừng học để hack. Mục tiêu của mình không phải là hướng dẫn các bạn đi hack và quấy phá các site khác, mà là dạy bạn biết và phòng chống những đòn tấn công này.

Vì XSS là một dạng tấn công hay gặp, dễ gây hậu quả cao nên hầu như các Web Framework nổi tiếng (Spring, Django, ASP.NET MVC) đều tích hợp sẵn cách phòng chống. Dù bạn là dân ngoại đạo, không biết gì về XSS, chỉ cần sử dụng framework bản mới nhất là đã đề phòng được khá khá rồi.



Lỗi XSS này cũng khá dễ fix, quan trọng là lỗi này thường gặp ở nhiều trang, dễ sót, do đó sau khi fix ta phải verify cẩn thận. Có 3 phương pháp thường dùng để fix lỗi này:

1. Encoding

Không được tin tưởng bất kì thứ gì người dùng nhập vào!! Hãy sử dụng hàm encode có sẵn trong ngôn ngữ/framework để chuyển các kí tự < > thành < %gt;.

2. Validation/Sanitize

Một cách chống XSS khác là validation: loại bỏ hoàn toàn các kí tự khả nghi trong input của người dùng, hoặc thông báo lỗi nếu trong input có các kí tự này.

Ngoài ra, nếu muốn cho phép người dùng nhập vào HTML, hãy sử dụng các thư viện sanitize. Các thư viện này sẽ lọc các thẻ HTML, CSS, JS nguy hiểm để chống XSS. Người dùng vẫn có thể sử dụng các thẻ <p>, , để trình bày văn bản.

Làm ơn, xin nhắc lại, làm ơn dùng các thư viện sẵn có chứ đừng “hồ báo” viết lại để thể hiện trình độ. Đã có rất nhiều trường hợp dính lỗi XSS vì developer tự tin và tự viết code để loại bỏ kí tự đặc biệt và... để sót.

3. CSP (Content Security Policy)

Hiện tại, ta có thể dùng chuẩn CSP để chống XSS. Với CSP, trình duyệt chỉ chạy JavaScript từ những domain được chỉ định. Giả sử thiendia.com có sử dụng CSP, chỉ chạy JavaScript có nguồn gốc thiendia.com. Vì Khoa để mã độc trên khoatran.com nên đoạn JavaScript sau sẽ không được thực thi.

```
<div class="comment">
  <p><script src="//khoatran.com/madoc.js"></script></p>
</div>
```

Để sử dụng CSP, server chỉ cần thêm header Content-Security-Policy vào mỗi response. Nội dung header chứa những domain mà ta tin tưởng.

```
Content-Security-Policy: script-src 'self' https://apis.google.com
```

Lời kết

Nói hơi chủ quan tí (do mình ko ưa PHP), số lượng trang web xây dựng bằng PHP bị lỗi XSS là nhiều nhất. Lí do thứ nhất là do số lượng web viết bằng PHP cực nhiều. Lí do thứ hai là mặc định PHP không encode các kí tự lạ. Các CMS của PHP như WordPress, Joomla rất mạnh với vô số plug-in. Tuy nhiên nhiều plug-in viết ẩu là nguyên nhân dẫn đến lỗi bảo mật này.

Hiện tại, số lượng website bị lỗi XSS là khá nhiều, các bạn chỉ cần lang thang trên mạng là sẽ gặp. Như mình đã nói, XSS là một lỗi rất cơ bản, hầu như hacker nào cũng biết. Trang web bị lỗi này rất dễ thành mồi ngon cho hacker. Do vậy, các bạn developer nhớ cẩn thận, đừng để web của mình bị dính lỗi này.

Một số link tham khảo:

- <http://excess-xss.com/>
- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

LƯU TRỮ COOKIE – TƯỜNG KHÔNG HẠI AI NGỜ HẠI KHÔNG TƯỜNG

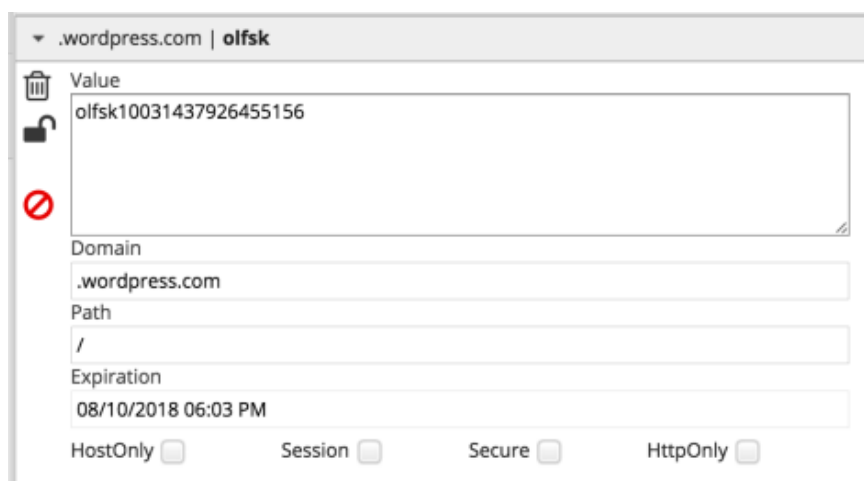
Cookie là một khái niệm hết sức cơ bản mà ta được học khi mới lập trình web. Tuy nhiên, nếu sử dụng không đúng cách, nó sẽ thành “mồi ngon” cho vô số hacker. Bài viết này sẽ đề cập đến những cách hacker mà có thể lợi dụng cookie để chiếm quyền người dùng, tấn công hệ thống, cùng với phương pháp sử dụng cookie đúng cách để ngăn chặn những lỗ hổng này nhé.

Cookie – Chiếc “bánh qui” vô hại?

Server và client giao tiếp với nhau thông qua giao thức HTTP. Đặc điểm của giao thức này là stateless. Server không thể biết được 2 request có tới từ cùng 1 client hay không. Vì đặt điểm này, cookie ra đời. Về bản chất, cookie là một file text nhỏ được server gửi về client, sau đó browser lưu vào máy người dùng. Khi client gửi request tới server, nó sẽ gửi kèm cookie. Server dựa vào cookie này để nhận ra người dùng.

Cookie thường có name, value, domain và expiration:

- Name, đi kèm với value: Tên cookie và giá trị của cookie đó
- Domain: Domain mà cookie được gửi lên. Như ở hình dưới, cookies chỉ được gửi khi client truy cập wordpress.com.
- Expiration: Thời gian cookie tồn tại ở máy client. Quá thời gian này, cookie sẽ bị xoá.



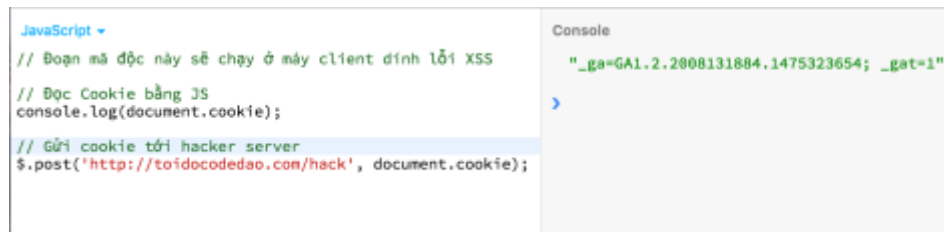
Bánh qui nho nhỏ, đầy những lỗ to to

Sau khi tìm hiểu cơ bản về cookie, ta sẽ tìm hiểu những lỗi bảo mật mà cookie có thể gây ra nhé. Vì cookie được gửi kèm theo mỗi request lên server. Server dựa theo cookie để nhận dạng người dùng. Do vậy, nếu có thể “chôm cookie” của người khác, ta có thể mạo danh người đó.

Cookie có thể bị chôm theo các con đường sau:

Sniff cookie qua mạng: Sử dụng 1 số tool đơn giản để sniff như Fiddler, Wireshark, ta có thể chôm cookie của người dùng ở cùng mạng. Sau đó, sử dụng EditThisCookie để dump cookie này vào trình duyệt để mạo danh người dùng. (Xem demo phần HTTP).

Chôm cookie (Cookie thief) bằng XSS: Với lỗ hổng XSS, hacker có thể chạy mã độc (JavaScript) ở phía người dùng. JS có thể đọc giá trị từ cookie với hàm `document.cookie`. Hacker có thể gửi cookie này tới server của mình. Cookie này sẽ được dùng để mạo danh người dùng.



Thực hiện tấn công kiểu CSRF (Cross-site request forgery). Hacker có thể post một link ảnh như sau:

```

```

Trình duyệt sẽ tự động load link trong ảnh, dĩ nhiên là có kèm theo cookie. Đường link trong ảnh sẽ đọc cookie từ request, xác nhận người dùng, rút sạch tiền mà người dùng không hề hay biết. Cách tấn công này có rất nhiều biến thể, mình sẽ nói rõ ở phần sau.

Cách phòng chống

Có thể áp dụng một số phương pháp sau:

- **Set Expired và Max-Age:** Để giảm thiểu thiệt hại khi cookie bị trộm, ta không nên để cookie sống quá lâu. Nên set thời gian sống của cookie trong khoảng 1 ngày tới 3 tháng, tùy theo yêu cầu của application.
- **Sử dụng Flag HTTP Only:** Cookie có flag này sẽ không thể truy cập thông qua hàm `document.cookie`. Do đó, dù web có bị lỗi XSS thì hacker không thể đánh cắp được nó.
- **Sử dụng Flag Secure:** Cookie có flag này chỉ được gửi qua giao thức HTTPS, hacker sẽ không thể sniff được.

Vì cookie dễ bị tấn công, tuyệt đối không chứa những thông tin quan trọng trong cookie (Mật khẩu, số tài khoản, ...). Nếu bắt buộc phải lưu thì cần mã hoá cẩn thận.

Lưu ý: Nếu website của bạn sử dụng RESTful API, đừng sử dụng cookie để authorize người dùng mà hãy dùng OAuth hoặc WebToken. Token này được vào Header của mỗi request nên sẽ không bị dính lỗi CSRF.

Các bạn có thể tìm hiểu thêm về cookie và các lỗi bảo mật liên quan ở đây:

- <http://resources.infosecinstitute.com/securing-cookies-httponly-secure-flags/>
- http://www.ibm.com/support/knowledgecenter/SSZLC2_7.0.0/com.ibm.commerce.admin.doc/concepts/csesmsession_mgmt.htm
- <https://www.nczonline.net/blog/2009/05/05/http-cookies-explained/>
- https://en.wikipedia.org/wiki/HTTP_cookie#Secure_and_HttpOnly
- <http://programmers.stackexchange.com/questions/298973/rest-api-security-stored-token-vs-jwt-vs-oauth>

SQL INJECTION – LỖ HỔNG BẢO MẬT THẦN THÁNH

Trong chương này, các bạn sẽ được tìm hiểu thực hư về lỗ hổng bảo mật SQL Injection “thần thánh”, một trong những lỗ hổng bảo mật phổ biến và nguy hiểm nhất mọi thời đại.

Tại sao SQL Injection lại “thần thánh”?

Những lý do sau đã tạo nên tên tuổi lẫy lừng của SQL Injection:

- Cực kỳ nguy hiểm – Có thể gây ra những thiệt hại khổng lồ. Với SQL Injection, hacker có thể truy cập một phần hoặc toàn bộ dữ liệu trong hệ thống.
- Rất phổ biến và dễ thực hiện – Lỗ hổng này rất nổi tiếng, từ developer đến hacker gần như ai cũng biết. Ngoài ra, còn có 1 số tool tấn công SQL Injection cho dân “ngoại đạo”, những người không biết gì về lập trình.
- Rất nhiều ông lớn từng bị dính – Sony, Microsoft UK. Mọi vụ lùm xùm liên quan tới “lộ dữ liệu người dùng” ít nhiều đều dính dáng tới SQL Injection.

Dễ tấn công, phổ biến, gây ra hậu quả nghiêm trọng, đó là lý do Inject (Không chỉ SQL mà OS và LDAP) nằm chễm chệ ở vị trí đầu bảng trong top 10 lỗ hổng bảo mật của OWASP. Tất nhiên là XSS, CSRF, và không mã hoá dữ liệu cũng nằm trong list này nốt.



Hậu quả của SQL Injection

Hậu quả lớn nhất mà SQL Injection gây ra là: Làm lộ dữ liệu trong database. Tùy vào tầm quan trọng của dữ liệu mà hậu quả dao động ở mức nhẹ cho đến vô cùng nghiêm trọng. Nếu lộ dữ liệu credit card, hacker có thể dùng credit card để “mua sắm hộ” hoặc chôm tiền của người dùng.

Hàng triệu Credit Card chôn tồn tại trên mạng, do hacker chôm từ các trang bán hàng thông qua SQL Injection. Lộ dữ liệu khách hàng có thể ảnh hưởng rất nghiêm trọng đến công ty. Hình ảnh công ty có thể bị ảnh hưởng, khách hàng chuyển qua sử dụng dịch vụ khác, dẫn đến phá sản v...v

Lỗ hổng này cũng ảnh hưởng lớn đến khách hàng. Do họ thường dùng chung một mật khẩu cho nhiều tài khoản, chỉ cần lộ mật khẩu một tài khoản thì các tài khoản khác cũng lộ theo. Đây cũng là lý do mình nhắc nhở phải mã hoá mật khẩu, nếu database có bị tấn công thì người

dùng cũng không bị mất mật khẩu. (Đây là lý do vừa rồi vietnamwork bị ăn chửi vì không mã hoá mật khẩu).

Trong nhiều trường hợp, hacker không chỉ đọc được dữ liệu mà còn có thể chỉnh sửa dữ liệu. Lúc này hacker có thể đăng nhập dưới vai trò admin, lợi dụng hệ thống, hoặc xóa toàn bộ dữ liệu để hệ thống ngừng hoạt động.

Tấn công SQL Injection như thế nào?

Cơ chế SQL Injection vô cùng đơn giản. Ta thường sử dụng câu lệnh SQL để truy cập dữ liệu. Giả sử, muốn tìm đăng nhập user, ta thường viết code như sau:

```
var username = request.username; // hoangcute

var password = request.password; // 123456

var sql = "SELECT * FROM Users WHERE Username = '" + username + "' AND Password = '" + password + "'";
// SELECT * FROM Users WHERE Username = 'Hoangcute' AND Password = '123456'
```

Đoạn code trên đọc thông tin nhập vào từ user và cộng chuỗi để thành câu lệnh SQL. Để thực hiện tấn công, Hacker có thể thay đổi thông tin nhập vào, từ đó thay đổi câu lệnh SQL.

```
var password = request.password; // ' OR '' = ''

var sql = "SELECT * FROM Users WHERE Username = '" + username + "' AND Password = '" + password + "'";
// SELECT * FROM Users WHERE Username = 'Hoangcute' AND Password = '' OR '' = ''
// Câu SQL này luôn cho kết quả true
```

Hoặc nếu ghét, hacker có thể drop luôn table Users, xóa toàn bộ người dùng trong database. Đáng sợ chưa nào?



Đến các mẹ bầm sữa còn biết cách dùng SQL Injection

Hacker có thể thông qua SQL Injection để dò tìm cấu trúc dữ liệu (Gồm những table nào, có những column gì), sau đó bắt đầu khai thác dữ liệu bằng cách sử dụng các câu lệnh như UNION, SELECT TOP 1...

Như mình đã nói SQL Injection rất phổ biến, bạn có thể dễ dàng google để tìm kiếm những bài viết liên quan tới nó. Do vậy, mình chỉ tóm tắt sơ về cơ chế tấn công. Các bạn tự tìm hiểu thêm qua các ví dụ ở bài viết này nhé: <http://expressmagazine.net/development/1512/tan-cong-kieu-sql-injection-va-cac-phong-chong-trong-aspnet>.

Cách phòng chống

May thay, mặc dù SQL rất nguy hại nhưng cũng dễ phòng chống. Gần đây, hầu như chúng ta ít viết SQL thuần mà toàn sử dụng ORM (Object-Relational Mapping) framework. Các framework web này sẽ tự tạo câu lệnh SQL nên hacker cũng khó tấn công hơn.

Tuy nhiên, có rất nhiều site vẫn sử dụng SQL thuần để truy cập dữ liệu. Đây chính là mồi ngon cho hacker. Để bảo vệ bản thân trước SQL Injection, ta có thể thực hiện các biện pháp sau.

- **Lọc dữ liệu từ người dùng:** Cách phòng chống này tương tự như XSS. Ta sử dụng filter để lọc các kí tự đặc biệt (; ' ') hoặc các từ khoá (SELECT, UNION) do người dùng nhập vào. Nên sử dụng thư viện/function được cung cấp bởi framework. Viết lại từ đầu vừa tốn thời gian vừa dễ sơ sót.
- **Không cộng chuỗi để tạo SQL:** Sử dụng parameter thay vì cộng chuỗi. Nếu dữ liệu truyền vào không hợp pháp, SQL Engine sẽ tự động báo lỗi, ta không cần dùng code để check.
- **Không hiển thị exception, message lỗi:** Hacker dựa vào message lỗi để tìm ra cấu trúc database. Khi có lỗi, ta chỉ hiện thông báo lỗi chứ đừng hiển thị đầy đủ thông tin về lỗi, tránh hacker lợi dụng.
- **Phân quyền rõ ràng trong DB:** Nếu chỉ truy cập dữ liệu từ một số bảng, hãy tạo một account trong DB, gán quyền truy cập cho account đó chứ đừng dùng account root hay sa. Lúc này, dù hacker có inject được sql cũng không thể đọc dữ liệu từ các bảng chính, sửa hay xóa dữ liệu.
- **Backup dữ liệu thường xuyên:** Các cụ có câu “cẩn tắc vô áy náy”. Dữ liệu phải thường xuyên được backup để nếu có bị hacker xóa thì ta vẫn có thể khôi phục được. Còn nếu cả dữ liệu backup cũng bị xóa luôn thì ... chúc mừng bạn, update CV rồi tìm cách chuyển công ty thôi!



Kết luận

Dữ liệu là một trong những thứ “đáng tiền” nhất trong website của bạn. Sau khi đọc xong chương này, hãy kiểm tra lại xem trang của mình có thể bị tấn công SQL Injection hay không, sau đó áp dụng những phương pháp mình đã hướng dẫn để fix.

Nguồn tham khảo thêm

- http://www.w3schools.com/sql/sql_injection.asp
- <http://expressmagazine.net/development/1512/tan-cong-kieu-sql-injection-va-cac-phong-chong-trong-aspnet>
- <http://freetuts.net/ky-thuat-tan-cong-sql-injection-va-cach-phong-chong-trong-php-107.html>
- <http://kienthucweb.net/sql-injection-la-gi.html>

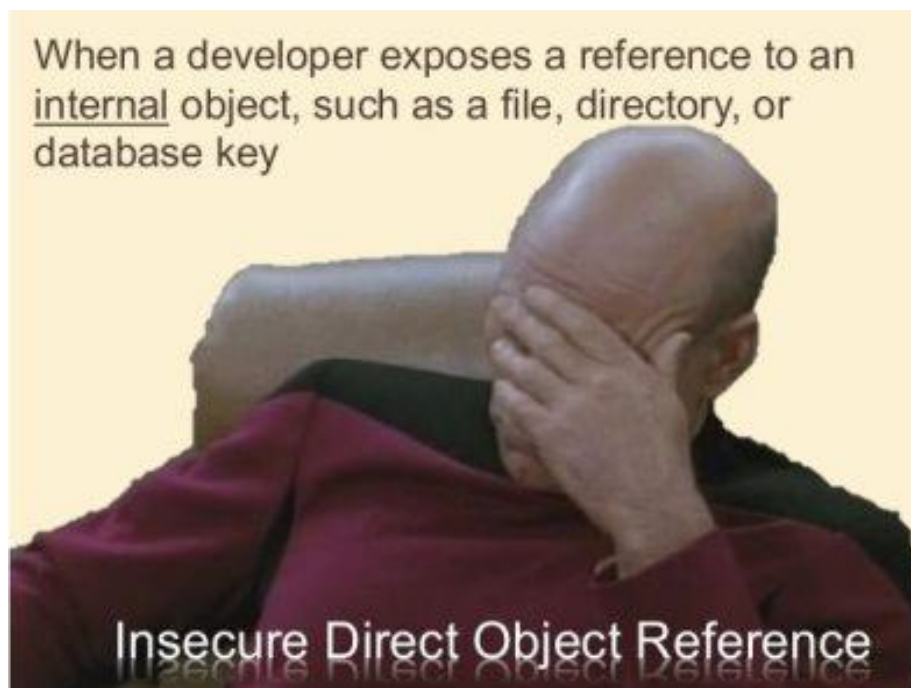
INSECURE DIRECT OBJECT REFERENCES – GIẤU ĐẦU LÒI ĐUÔI

Ở chương này, mình sẽ giới thiệu một lỗ hổng bảo mật khá “lạ” mang cái tên dài loằng ngoằng khó đọc: Insecure Direct Object References.

Lỗi gì mà tên dài rứa??

Lỗi này “lạ” ở chỗ nó nằm trong top 4 OWASP nhưng lại có rất ít tài liệu về nó. Nó cũng không nổi tiếng như XSS hay CSRF hay SQL Injection (Dù rank OWASP của nó cao hơn XSS hay CSRF nhiều). Bản thân mình trước đây cũng chưa hề nghe báo chí hay tin tức gì nhắc tới lỗi này. Có thể là do chưa có vụ án nổi tiếng nào liên quan đến nó, hoặc do lỗi này có nhiều biến thể phức tạp chẳng?

Nguyên nhân chính gây ra lỗ hổng này là sự bất cẩn của developer hoặc sysadmin (Gặp lỗi này là phải lỗi thằng dev ra chém trước, sau đó chém tester). Lỗ hổng này xảy ra khi chương trình cho phép người dùng truy cập tài nguyên (dữ liệu, file, thư mục, database) một cách bất hợp pháp, thông qua dữ liệu do người dùng cung cấp. Để dễ hiểu hơn, hãy đọc ví dụ phía dưới nhé.



Cách lợi dụng lỗ hổng

Rất tình cờ, mình phát hiện lỗi này khi đang giúp một thằng em test đồ án web bán hàng. Trong mục “Quản lý đơn hàng”, URL của một đơn hàng sẽ có dạng như sau: `http://shop.com/user/order/1230`. Server sẽ đọc ID 1230 từ URL, sau đó tìm đơn hàng có ID 1230 trong database và đổ dữ liệu vào HTML.

Bắt chước hacker, mình “nghịch ngợm” một tí, thay 1230 bằng các giá trị từ 1 tới 2000. Hệ thống cứ thế mà đọc và hiển thị cho mình toàn bộ các đơn hàng có ID từ 1 tới 2000 (kể cả đơn hàng của các khách hàng khác). Tai hại chưa!

Lỗi hổng ở đây chính là: chương trình cho phép mình truy cập tài nguyên (đơn hàng của người khác) bất hợp pháp, thông qua dữ liệu (ID) mà mình cung cấp qua URL. Lẽ ra, chương trình phải check xem mình có quyền truy cập các dữ liệu này hay không.

Trong thực tế, hacker có thể dùng nhiều chiêu trò như: thay đổi URL, thay đổi param trong API, sử dụng tool để scan những tài nguyên không được bảo mật. Chiêu hack lotte cinema ngày xưa của mình cũng na ná như thế, thay id trong URL bằng username trong cookie.

Cách đây khoảng 1-2 tháng, có 1 vụ lùm xùm liên quan tới công ty X (Hình như là CGV), lộ tài khoản của 3 triệu người dùng. Chính lỗi hổng Insecure Direct Object References này đã giúp hacker (ở đây là thánh bảo mật Juno_okyo) lợi dụng và dò ra thông tin của 3 triệu người dùng đó.

Viết mã khai thác lỗi hổng tự động

Như đã phân tích, hiện tại chúng ta có thể truy vấn thông tin của người dùng bất kỳ dựa theo ID và thực hiện brute force tài khoản dựa theo địa chỉ Email. Tôi tiến hành viết mã khai thác cho chúng bằng Python.

Vết cặn thông tin người dùng của X

Chỉ việc tạo một hàm truy vấn tới `https://x-server.com/api/customer/profile/id/[USER_ID]` rồi sau đó dùng vòng lặp để quét từ Min tới Max. Vấn đề bây giờ là cần tìm giá trị của Min và Max. Nếu một hacker có thời gian rảnh, anh ta có thể quét từ số 1 cho tới một số ID nào đó thật lớn. Nhưng như thế thì thật lãng phí thời gian cho những truy vấn vào ID nào đó không tồn tại.

(Bài viết gốc khá hay ở đây: <https://junookyo.blogspot.com/2016/10/ro-ri-3-trieu-thong-tin-ca-nhan.html>).

Có một vài vụ việc hi hữu, hacker scan được git repository nằm trên server. Truy cập git, hẳn lấy được username, password của database và các thông tin quan trọng khác. Bạn đừng nghĩ là mình... hư cấu. Cách đây vài hôm, git của vietnamwork vẫn nằm public chễm chệ tại `vietnamwork.com/.git/`, không bảo mật gì! May mà gặp hacker “có tâm” đi ngang qua nên chưa có gì đáng tiếc xảy ra.

Insecure Direct Object Reference

Realizing the impact



All resources accessible through direct object references could be exposed, including system files.

Unauthorized access to system files could compromise user accounts, leading to lack of accountability.



Sensitive end-user (customer) data could be stolen, leading to reputational damage and revenue loss.

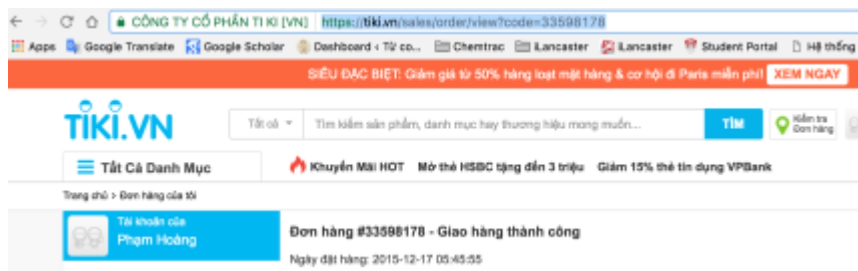
Cách phòng chống

Một số biện pháp phòng chống:

Test cẩn thận – Nguyên nhân gây ra lỗi thường là do sự bất cẩn của developer. Tuy nhiên, nếu để sản phẩm bị lỗi thì đây là lỗi của tester. Đây là lỗi nằm trong code, do đó tester phải chịu trách nhiệm nếu để lỗi này xảy đến với người dùng.

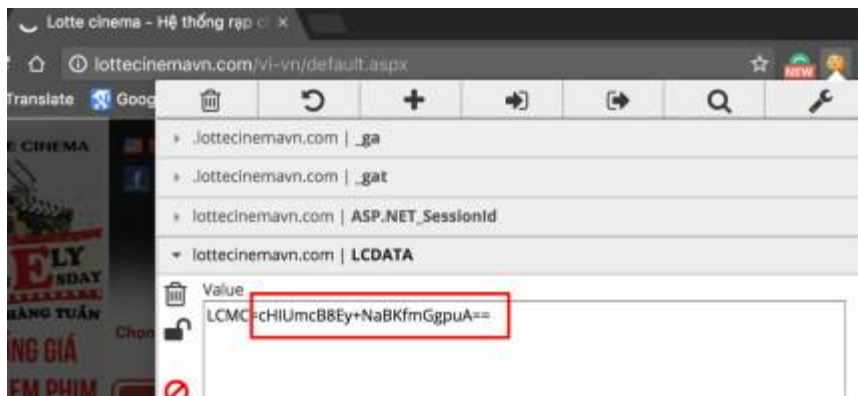
Bảo vệ dữ liệu “nhạy cảm” – Với những dữ liệu “nhạy cảm” như source code, config, database key, cần hạn chế truy cập. Cách tốt nhất là chỉ cho phép các IP nội bộ truy cập các dữ liệu này, hacker khỏi táy máy.

Kiểm tra chặt chẽ quyền truy cập của user – Hãy thử xem tiki giải quyết vấn đề này như thế nào? Đơn hàng trên tiki.vn có dạng: <https://tiki.vn/sales/order/view?code=33598178>



Để thấy, tiki để id của đơn hàng trong URL. tuy nhiên, khi mình thử thay đổi id của đơn hàng thì tiki redirect mình lại trang <https://tiki.vn/sales/order/history>. Bảo mật có tâm là phải như thế!

Tránh để lộ key của đối tượng – Trong các trường hợp đã nêu, id của đối tượng là số int, do đó hacker có thể đoán ra id của các đối tượng khác. Nhằm phòng tránh việc này, ta có thể mã hoá id, dùng GUID để làm id. Hacker không thể nào dò ra ID của đối tượng khác được.



Lotte Cinema giờ đã mã hoá username trong cookie, khỏi nghịch ngợm nhé

Một số nguồn tham khảo thêm:

- https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References
- <http://lockmedown.com/secure-from-insecure-direct-object-reference/>
- <https://codedx.com/insecure-direct-object-references/>

CSRF – NHỮNG CÚ LỪA NGOẠN MỤC

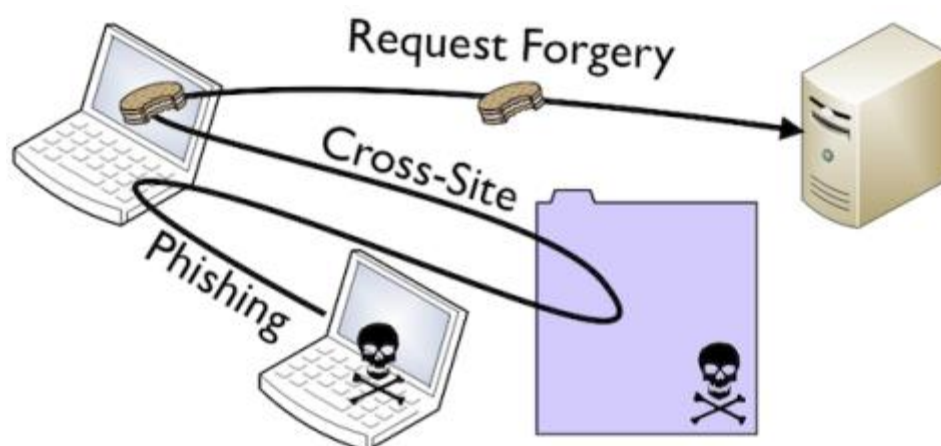
Trong Tam Quốc, các bậc quân sư tài năng có tài điều binh khiển tướng, ngồi trong trướng bỗng quyết thắng cách đó hàng ngàn dặm. Trong Tu Chân, các cao thủ có chiêu “Cách Không Thủ Vật” điều khiển đồ vật từ xa, hoặc “Ngự Kiếm Phi Hành”, dùng chân khí để điều động phi kiếm hay pháp bảo.

Ngày nay, hacker cũng có “chiêu thức” tương tự gọi là CSRF. Hacker có thể ngồi tại website A mà dụ dỗ người dùng tấn công site B và site C khác. Chương này sẽ giải thích cách hacker tấn công, đồng thời hướng dẫn cách phòng chống cho các bạn lập trình viên nhé.

Cơ bản về CSRF

CSRF có tên đầy đủ là Cross Site Request Forgery (Tên khác là XSRF). Lỗi hổng này khá phổ biến, Netflix và Youtube cũng từng là nạn nhân của lỗi hổng nó. Hậu quả do nó gây ra cũng “hơi” nghiêm trọng nên CSRF hân hạnh được nằm trong top 10 lỗi hổng bảo mật của OWASP.

Nguyên tắc hoạt động của CSRF rất đơn giản. Ở bài trước, chúng ta biết rằng server sẽ lưu trữ cookie ở phía người dùng để phân biệt người dùng. Mỗi khi người dùng gửi một request tới một domain nào đó, cookie sẽ được gửi kèm theo.



- Đầu tiên, người dùng phải đăng nhập vào trang mình cần (Tạm gọi là trang A).
- Để dụ dỗ người dùng, hacker sẽ tạo ra một trang web độc.
- Khi người dùng truy cập vào web độc này, một request sẽ được gửi đến trang A mà hacker muốn tấn công (thông qua form, img, ...).
- Do trong request này có đính kèm cookie của người dùng, trang web A đích sẽ nhầm rằng đây là request do người dùng thực hiện.
- Hacker có thể mạo danh người dùng để làm các hành động như đổi mật khẩu, chuyển tiền,

Để dễ hiểu hơn, bạn hãy đọc phần ví dụ phía dưới nhé.

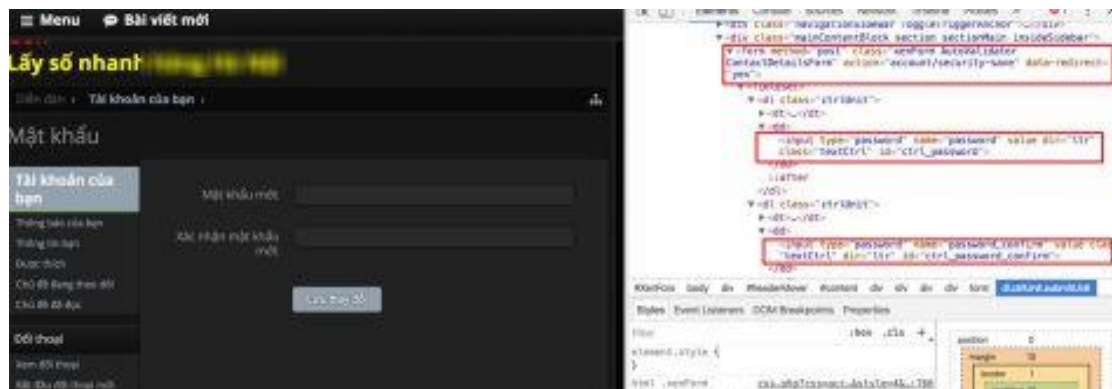
Các kiểu tấn công thường gặp

Kiểu 1. Dùng form

Ngày xưa ngày xưa, có hai anh em nhà nọ tên là Tưng và Tườn. Tưng, người anh, chăm lo học hành, chí thú làm ăn nuôi vợ con. Người em, Tườn thì người lại, suốt ngày lên thiên địa share hàng và tìm địa điểm mát xa.

Một hôm nọ, cãi nhau với vợ, Tung buồn quá muốn bỏ đi mát xa. Tiếc thay, lên thiendia hỏi địa chỉ không ai cho vì Tung tin dụng quá thấp. Biết Tườn là thành viên cộm cán, Tung bèn năn nỉ Tườn cho mượn acc nhưng vì sợ anh hư hỏng nên Tườn không cho. Đúng là anh em tốt!! Phần chí, Tung quyết định dùng lỗi CSRF để chôm account của Tườn.

Ta hãy quan sát HTML của form đổi mật khẩu thiendia. Form này gồm 2 field là password và password_confirm, submit tới http://thi*ndia.com/account/security-save



Tung làm giả một trang web JAV, giả vờ gửi cho thằng em xấu số. Trong trang web có một form ẩn với các giá trị tương tự form trên (Các đồng d*m vui lòng để ý form HTML bên trái và button bên phải).



Thanh niên Tườn ngây thơ mù IT, lỡ tay vào link và bấm vào button. Một request đổi password được gửi đến thiendia, kèm theo cookie account của Tườn. Thế là xong! Tung chỉ cần dùng email + mật khẩu mới là 123456 để đăng nhập vào account của thằng em xấu số.

Kiểu 2. Dùng thẻ img

Chuyện tới đây chưa hết. Có địa điểm mát xa, nhưng tiền bạc do vợ nắm cả, Tung không có tiền đi mát xa. Tung quyết định hack luôn tài khoản ngân hàng của Tườn. Tườn sử dụng JAVBank (Japan America Vietnam Bank).

Mỗi lần chuyển khoản, ngân hàng sẽ tạo 1 URL. Giả sử người A muốn chuyển 1000 cho người B, url được tạo ra sẽ có dạng <http://jav.bank?from=Person1&to=Person2&amount=1000>.

Tương bỏ url này vào 1 thẻ img. Khi Tườn truy cập trang, trình duyệt sẽ tự gọi GET request, gắn kèm với cookie trên JAVBank của Tườn. Thông qua cookie, ngân hàng xác nhận đó là Tườn, chuyển tiền qua cho Tườn.

```
<body>

<h1>Free JAV</h1>




<form method="post" action="http://thiendia.com/account/security-save"
  data-redirect="yes">
  <input type="hidden" name="password"
    value="123456" id="ctrl_password" />
  <input type="hidden" name="password_confirm"
    value="123456" id="ctrl_password_confirm"/>
  <input type="submit" name="save"
    value="Vào trang chủ để xem JAV" />
</form>

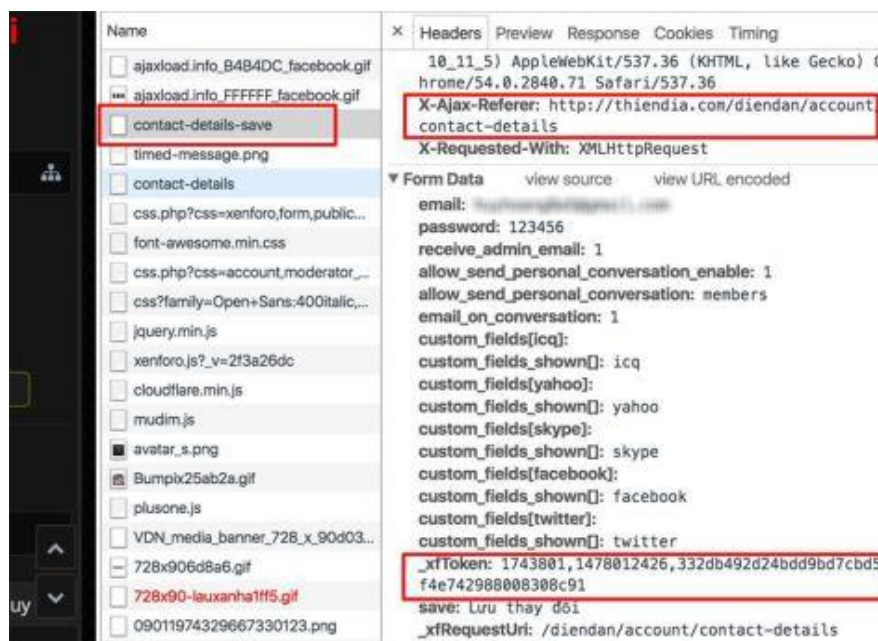
</body>
</html>
```

Có tiền lại có địa điểm, Tườn đối vợ lên đường mát xa. Chuyện về sau có nội dung 18+ nên mình không kể nữa....

Lưu ý

Tất nhiên, trong bài chỉ là ví dụ. Theo nguyên tắc, request GET chỉ được dùng để truy cập dữ liệu, không được dùng để thực hiện các hoạt động thay đổi dữ liệu như edit/delete. Các ngân hàng thường bảo mật rất kĩ bằng cách set cookie có thời gian sống khá ngắn, không cho phép chuyển tiền mà không có code OTP v...v.

Ngoài ra, thi*ndia cũng có các biện pháp bảo mật khá tốt (xem phía dưới) nên các bạn không dùng cách này để chôm account của bạn bè được đâu, đừng thử nhé!



Ảnh minh họa từ thiên địa, trang này có CSRF token

Tuy nhiên, ngày xưa, khi các lỗ hổng bảo mật còn chưa phổ biến thì đây là chính là cách mà hacker sử dụng. Chỉ cần post 1 tấm ảnh chứa đường dẫn như trên lên 1 forum nào đó, sẽ có vô số người dính bẫy khi truy cập vào forum đó.

Phòng chống cho website

Dưới đây là một số cách phòng chống CSRF cơ bản:

- **Sử dụng CSRF Token:** Trong mỗi form hay request, ta đính kèm một CSRF token. Token này được tạo ra dựa theo session của user. Khi gửi về server, ta kiểm tra độ xác thực của session này. Do token này được tạo ngẫu nhiên dựa theo session nên hacker không thể làm giả được (Các framework như RoR, CodeIgniter, ASP.NET MVC đều hỗ trợ CSRF token).
- **Kiểm tra giá trị Referer và Origin trong header:** Origin cho ta biết trang web gọi request này. Giá trị này được đính kèm trong mỗi request, hacker không chỉnh sửa được. Kiểm tra giá trị này, nếu nó là trang lạ thì không xử lý request.
- **Kiểm tra header X-Requested-With:** Request chứa header này là request an toàn, vì header này ngăn không cho ta gửi request đến domain khác (chi tiết).
- **Cần cẩn thận đề phòng lỗi XSS:** Với XSS, hacker có thể cài mã độc trên chính trang web cần tấn công. Lúc này, mọi phương pháp phòng chống CSRF như token, referrer đều bị vô hiệu hoá. Bản thân bác juno_okyo từng áp dụng lỗi XSS kết hợp CSRF để tấn công sinhvienit.net (Chi tiết).

Tổng kết

Ngày trước lỗi này khá nghiêm trọng và phổ biến. Gần đây các framework hầu như đều mặc định chống lỗi này nên tần suất gặp cũng ít đi. Tuy vậy ta vẫn phải đề phòng, nhất là các website tự code nhé (Đặc biệt là code bằng PHP, ahihi).

Ngoài ra, là một user, bạn cần biết tự bảo vệ mình theo nhiều cách sau:

- Đăng xuất khỏi account sau khi sử dụng để tránh lưu cookie.
- Không click quảng cáo hay button lung tung.
- Không ghé thăm các trang bày bạ, nguy hiểm. Như đã nói phía trên, nhiều khi ta không bấm nút gì, chỉ cần truy cập trang, trình duyệt cũng tự động post dựa trên javascript hoặc thẻ img. (Nếu bắt buộc, hãy sử dụng chế độ ẩn danh trong Chrome).

Nguồn để tham khảo thêm:

- https://en.wikipedia.org/wiki/Cross-site_request_forgery
- [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF))
- <http://stackoverflow.com/questions/17478731/whats-the-point-of-the-x-requested-with-header>
- <https://www.youtube.com/watch?v=m0EHLfTgGUU>

ẨN GIẤU THÔNG TIN HỆ THỐNG – TRÁNH CON MẮT NGƯỜI ĐỜI VÀ KẺ XẤU

Chương này đề cập tới một phương pháp bảo mật vô cùng đơn giản, hiệu quả nhưng lại được ít người biết và áp dụng. Đó là phương pháp: Ẩn giấu thông tin hệ thống.

Thông tin hệ thống là gì?

Có thể tạm hiểu thông tin hệ thống là những thông tin về cấu tạo và hoạt động của hệ thống đó. Lấy lottecinema ra làm ví dụ: Thông tin về hoạt động là file log, error page hiển thị khi bị lỗi. Thông tin về cấu tạo của trang này như sau (Lí do làm sao mình biết được thì các bạn theo dõi phần sau):

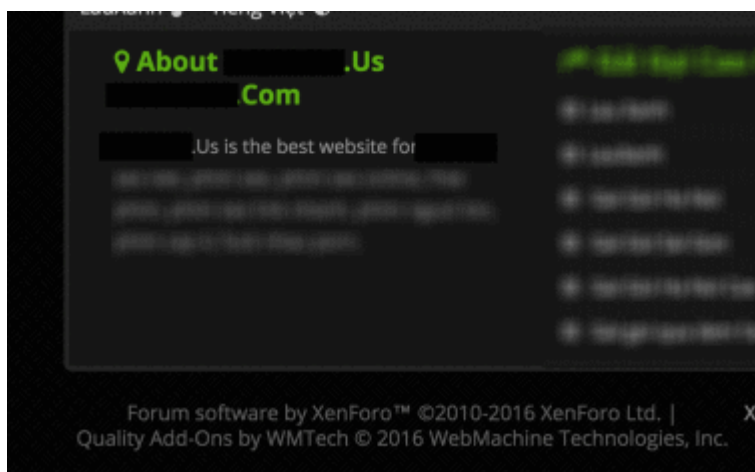
- Trang chính sử dụng KenticoCMS. Phiên bản sử dụng là ASP.NET WebForm 2.0
- Theo dự đoán thì do dùng ASP.NET nên database sẽ là MS SQL Server
- Trang web có sử dụng jQuery và jQueryUI
- Trang web được deploy trên Server ISS7

Chắc bạn đang tự hỏi: Ủa, nếu hệ thống của mình không làm gì mờ ám thì sao phải giấu? Ừ nhỉ! Hãy tưởng tượng nhà bạn là tiệm vàng có rất nhiều tiền và vàng bạc. Liệu bạn có treo biển “Góc dưới hàng rào có lỗ hổng, kết sắt nhà tao để ở lầu 3, kết hiệu Việt Tiến, mật khẩu 4 chữ số” không?

Dĩ nhiên, bạn không bao giờ để thông tin nhà cửa hở hênh cho ăn trộm biết. Điều này giống như mời trộm vào nhà vậy. Tuy nhiên, đa phần chúng ta lại để “hớ hênh” thông tin hệ thống cho hacker thấy. Thế có khác gì mời hacker tấn công không cơ chứ!!

Chúng ta để thông tin hệ thống “hớ hênh” như thế nào?

Chúng ta thường để lộ thông tin hệ thống một cách rất “hớ hênh”, không thua gì cách em Linh Miu khoe thân trong mấy bộ đồ thiếu vải (Điện hình là vụ lộ vếu nổi đình nổi đám gần đây). Dưới đây là một số kiểu lộ thông tin thường gặp:



Hiển thị chính mình trên trang sợ người khác không nhìn thấy

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the error has been written to the application log. Please review the log for more details. This error may be caused by a missing or misconfigured web resource located at /Resources/Scripts/ExtJS/default/

Stack Trace:

```
[SqlException (080131904): Cannot use a CONTAINS or FREETEXT p
System.Data.SqlClient.SqlConnection.OnError(SqlException exc
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(Tds
System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavi
System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() +61
System.Data.SqlClient.SqlDataReader.get_MetaData() +134
System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlData
System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(Command
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBeh
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBeh
System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavi
System.Data.SqlClient.SqlCommand.ExecuteDbDataReader(Command
System.Data.Common.DbCommand.System.Data.IDbCommand.Execut
System.Data.Common.DbDataAdapter.FillInternal(DataSet dataSe
System.Data.Common.DbDataAdapter.Fill(DataSet dataSet, Int32
System.Data.Common.DbDataAdapter.Fill(DataSet dataSet) +275
MBC2K2.mbcSearchAddressCodeClass.mbcSearchSectionByAll(Strin
MBC2K2.viewSearchAddressCodeForService.SearchDataBind() +843
System.Web.UI.WebControls.Button.RaisePostBackEvent(String e
System.Web.UI.Page.ProcessRequestMain(Boolean includeStages)
```

Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.36369

Hiển thị rõ phiên bản .NET, Exception khi bị lỗi. Đây là mồi ngon cho tấn công SQL Injection

▼ **Response Headers** [view source](#)

Cache-Control: private
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Sun, 16 Oct 2016 06:41:31 GMT
Server: Microsoft-IIS/7.0
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET

Để trong header trả về từ server

```
<?xml version='1.0' encoding='utf-8'>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<title>MBC 2K2 - Tra cứu mã bưu chính - PostNet</title>
<meta name="GENERATOR" content="Microsoft Visual Studio .NET 7.0">
<meta name="CODE_LANGUAGE" content="Visual Basic 7.0">
<meta name="vs_defaultClientScript" content="JavaScript">
<meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/...>
<style type="text/css">A { TEXT-DECORATION: none }
A:hover { TEXT-DECORATION: underline }
</style>
<link href="/_TopMenu.css" type="text/css" rel="stylesheet">
</HEAD>
<body MS_POSITIONING="GridLayout" bottomMargin="0" leftMargin="0" topMargin="0" rightMargin="0">
<form method="post" action="/" search.aspx id="Form1">

```

Để “hớ hênh” trong code

Tất cả những thông tin này đều có thể dễ dàng truy ra bằng buildwith.com. Trang này hoạt động trên nguyên lý đọc các header trả về từ server, xem HTML include các thư viện nào.

Những hậu quả của việc “lộ hàng”

Những thông tin “vô hại” này cũng vô tình “giúp đỡ” hacker tấn công hệ thống của bạn dễ dàng hơn bằng những cách sau:

- Biết được phiên bản thư viện/framework sử dụng, phiên bản server, phiên bản database, ... hacker có thể tìm ra lỗ hổng bảo mật (CVE) của hệ thống. Việc tra cứu rất dễ dàng, chỉ cần vào nvd.nist.gov. Dựa theo phiên bản framework/server/database, hacker có thể thấy được những lỗ hổng bảo mật của các phiên bản này. Từ các lỗ hổng này, hacker có thể tìm cách tấn công hệ thống.
- Ngoài ra, khi biết được framework đang sử dụng, hacker có thể mò ra đường dẫn tới trang admin (Với wordpress là /wp-admin, với joomla là /administrator, với phpmyadmin là /phpmyadmin). Tiếp theo, hacker có thể thử nhập username/password admin mặc định để đăng nhập vào hệ thống. Đáng sợ chưa??
- Với mobile app hoặc phần mềm, hacker có thể decompile để chôm API hoặc security key. Chẳng giấu gì các bạn, mình cũng từng decompile file API của Simsimi để lấy Key và API miễn phí gắn vào chat bot facebook ấy...

Giấu như thế nào cho đúng?

Cảnh giới cao nhất của việc giấu hàng là hacker không thể biết được hệ thống của bạn được viết ngôn ngữ/framework gì, dùng database gì, deploy ở đâu. Điều này làm công việc của hacker trở nên khó khăn hơn rất nhiều. Thật ra, việc giấu thông tin hệ thống cũng không quá khó khăn hay mất thời gian. Chỉ cần bạn để ý và cẩn trọng là được.

Một số phương pháp “giấu thông tin” hay dùng:

- Config server hoặc viết code để loại bỏ những HTTP header dư thừa.
- Khi deploy, ta obfuscate hoặc uglify code để code trở nên khó đọc. Để tránh việc hacker biết các thư viện JS sử dụng thì ta có thể bundle toàn bộ thư viện và code thành 1 file luôn.
- Khi hệ thống bị thống, hiển thị custom Error Page. Lỗi trong trang này nên giải thích rõ ràng cho người dùng hiểu. Nhưng tuyệt đối không hiển thị trực tiếp error/exception để tránh hacker tấn công.
- Thường xuyên cập nhật/nâng cấp framework lên phiên bản mới nhất để vá các lỗ hổng, tránh việc hacker lợi dụng những lỗ hổng đã phát hiện ở phiên bản cũ.

QUẢN LÝ NGƯỜI DÙNG – TƯƠNG ĐẼ ĂN MÀ KHÔNG ĐƠN GIẢN

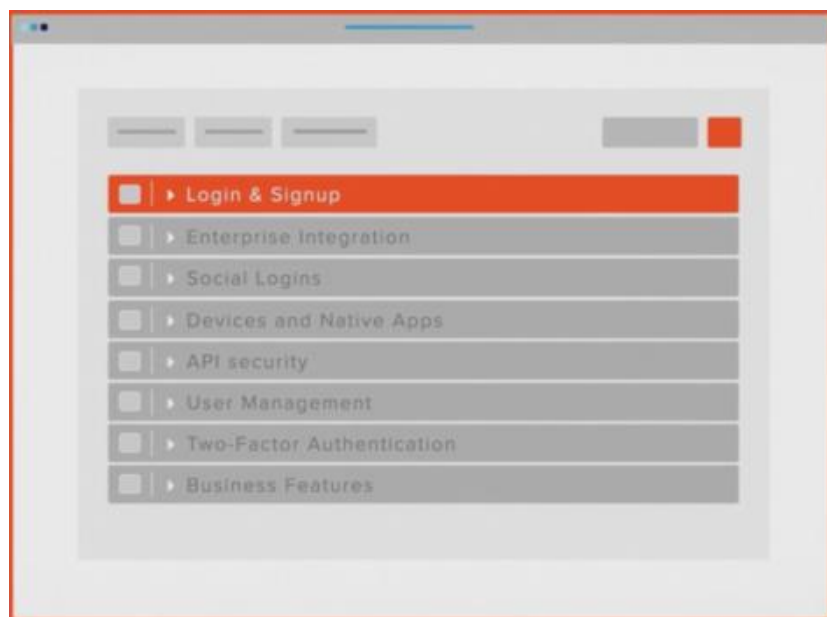
Website được tạo ra là để phục vụ người dùng. Có người sử dụng thì website và doanh nghiệp mới có thu nhập. Một trong những việc rắc rối nhất chính là quản lý và bảo mật thông tin người dùng.

Trong bài này, mình chia sẻ những điều cần lưu ý khi thực hiện tính năng này. Khá nhiều khó và phức tạp đấy, các bạn chịu khó đọc kĩ nhé!

Úi giời! Đăng kí đăng nhập có gì khó?

Không như bạn tưởng tượng, việc đăng kí/dăng nhập và quản lý người dùng thật ra không hề đơn giản. Nó có thể trở nên khá lằng nhằng với những tính năng sau:

- Cho phép người dùng đăng kí, đăng nhập bằng email
- Phân quyền người dùng
- Tích hợp với Gmail, Facebook
- Tích hợp với hệ thống người dùng có sẵn trong doanh nghiệp
- Reset mật khẩu khi người dùng quên
- Block account khi người dùng nhập sai pass nhiều lần
- Bảo mật cho API với app di động
- Bảo mật 2 lớp (Two factor authentication) với các account quan trọng
- Quản lý: Thêm bớt xóa sửa người dùng



Khi tính năng này hoạt động ổn định, không ai khen nó lấy một câu. Tuy nhiên, chỉ cần nó gặp phải chút vấn đề, cam đoan bạn sẽ hứng chịu vô số cơn thịnh nộ từ khách hàng.

Quan trọng nhất – Không lưu mật khẩu!

Developer phải thuộc nằm lòng câu nói sau: Tuyệt đối không bao giờ lưu mật khẩu khách hàng, dù sếp có nói gì đi nữa! Là một developer có tâm, bạn không bao giờ được lưu mật khẩu của khách hàng vào database (nhắc lại lần thứ ba cho nhớ).

Đến cả web lớn là vietnamwork mà cũng tắc trách đến mức không dùng https khi đăng nhập, không bảo mật dữ liệu đến nỗi làm lộ mật khẩu của người dùng: <http://nghehinvietnam.vn/tin-tuc/web-tim-viec-vietnamwork-bi-tan-cong-23535.html>. Hậu quả của việc này cũng không có gì nghiêm trọng, cùng lắm là mất mặt công ty, mất account khách hàng và làm khách hàng chuyển qua dùng dịch vụ khác thôi.

Làm thế nào khi người dùng quên mật khẩu?

Do không lưu mật khẩu trong database, ta không thể gửi mật khẩu về mail cho người dùng khi họ quên mật khẩu. Ở đây ta có 2 cách giải quyết.

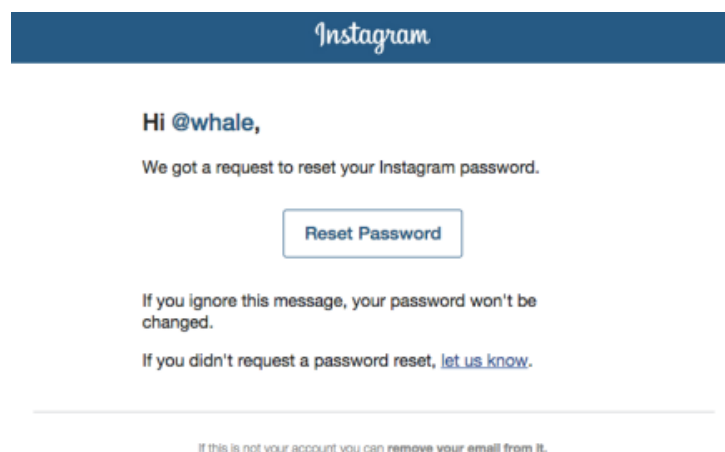
Cách 1: Reset mật khẩu mới ngẫu nhiên rồi gửi cho người dùng

Cách này có thể làm lộ mật khẩu vì email có thể bị đọc trộm. Ngoài ra, nếu như biết địa chỉ mail, hacker có thể lợi dụng nó để reset mật khẩu hàng loạt người dùng, nhằm ngăn cản họ đăng nhập vào hệ thống.

Cách 2: Gửi link để người dùng reset

Dựa theo tài khoản, ta tạo reset token rồi gắn nó vào link: <http://shop.com/resetpass?token=32343>, gửi link này vào mail cho người dùng.

Người dùng sử dụng link này để reset mật khẩu. Với cách này, dù hacker có request reset thì mật khẩu người dùng vẫn giữ nguyên, không bị ảnh hưởng. Như đã nói phía trên, do email không an toàn nên token này nên được expired ngay sau khi dùng, hoặc sau 24-48 tiếng đồng hồ sau khi email được gửi đi.



Gửi email có link Reset Password về cho người dùng

Chống việc đoán mò mật khẩu

Để dò mật khẩu, hacker có thể viết một con bot, lần lượt submit username và password cho tới khi đăng nhập được. Để phòng tránh việc này, ta áp dụng những phương pháp sau:

- Khi người dùng đăng nhập sai, đừng báo là sai username hay sai password. Chỉ cần báo username hay password không match, hacker sẽ gặp khó khăn hơn.
- Hacker lợi dụng chức năng reset mật khẩu để dò xem người dùng có email trên trang đó hay không. Dù account có tồn tại hay không, ta vẫn chỉ hiện thông báo: đã gửi message.

- Hạn chế số lần đăng nhập khi nhập mật khẩu sai. Ví dụ sau 3 lần nhập pass sai thì khoá account trong 10 phút. Hacker có thể dùng cách này để khoá tài khoản người dùng, nên cẩn thận. Có thể kết hợp thêm captcha.

Lưu ý: Những cách này có thể gây khó chịu cho người dùng, nếu dữ liệu không quá quan trọng (game, web hỏi đáp, giao lưu, giải trí ...) thì có thể nói lỏng một số yếu tố.

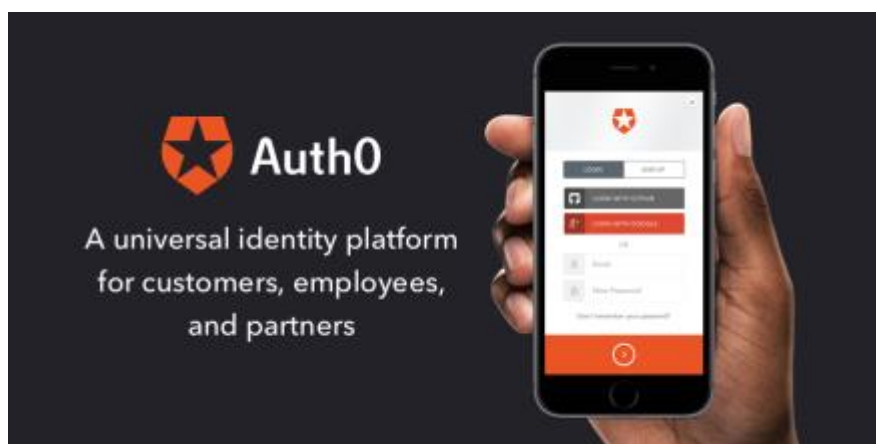


Facebook tạm khoá tài khoản khi hacker cố tình đăng nhập nhiều lần

Những biện pháp nho nhỏ tăng cường bảo mật

Một số điểm cần lưu ý khác:

- Với các thao tác quan trọng như đổi email, đổi pass, xoá nick, cần bắt người dùng nhập lại password. Lý do là đôi khi người dùng bị chôm cookie, hoặc lơ là quên khoá máy. Hãy nhìn Facebook và Google, cả 2 trang này đều bắt ta phải nhập lại mật khẩu khi muốn đổi pass.
- Với các ứng dụng cần bảo mật cao, phải có Two-factor verification (Gửi tin nhắn, device tạo authentication token). Mình hiện tại cũng đang dùng nó, dù các bạn có biết mật khẩu Gmail hay WordPress của mình cũng “éo” thể nào đăng nhập được.
- Nên khuyến khích (hoặc bắt buộc) người dùng sử dụng mật khẩu dài, đi kèm chữ và số, viết hoa viết thường và kí tự đặc biệt. Máy móc rất hiện đại khi crack mật khẩu, có thể vào đây để test xem máy mất bao lâu để mò ra mật khẩu của bạn.
- Nếu site của bạn không có HTTPS, hoặc team không có kinh nghiệm làm bảo mật, cứ để cho bọn khác lo. Bạn có thể dùng OAuth, cho phép người dùng đăng nhập từ Google, Facebook.
- Lúc này Google, Facebook sẽ chịu trách nhiệm quản lý mật khẩu và dữ liệu của người dùng. Người dùng thì không cần phải đăng kí nhiều tài khoản, một công đôi việc. Tìm hiểu thêm tại <https://oauth.io/> hoặc <https://auth0.com/>.



PHẦN 2 – CASE STUDY

Một số lỗi bảo mật của các trang web lớn tại Việt Nam

LỖ HỔNG BẢO MẬT KHỦNG KHIẾP CỦA LOTTE CINEMA

Đăng nhập là một chức năng đơn giản nhất mà hơn 90% các ứng dụng web cần phải có. Tuy nhiên, đôi khi ta lại không được hướng dẫn cách thực hiện chức năng “Đăng nhập” một cách đúng đắn, bài bản, dẫn đến những lỗi dở khóc dở cười, hoặc những lỗ hổng bảo mật khủng khiếp. Đến cả Lotte Cinema, một trang web được khá nhiều người dùng còn mắc lỗi sơ đẳng này.

Đăng nhập hả? Chỉ cần một bảng User, hai cột Username và Password là xong

Kể cũng buồn cười. Ngày xưa khi đi học, mình được hướng dẫn cách làm chức năng đăng nhập như thế này:

1. Người dùng nhập tên tài khoản (email) và mật khẩu.
2. So sánh tên tài khoản và mật khẩu với thông tin trong database.
3. Nếu đúng, cho người dùng đăng nhập, lưu thông tin vào session hoặc cookies.

Bước 1 và 3 không có gì đáng bàn, nhưng bước 2 mới là điều đáng nói. Đa phần tụi mình đều lưu trực tiếp tên tài khoản và mật khẩu vào database, sau đó đem ra so sánh.

```
public void Register(string username, string password)
{
    Database.SaveUser(username, password);
}

public bool Login(string username, string password)
{
    string pw = Database.GetPasswordByUsername(username);
    return pw == password;
}
```

Đây là cách củ chuối nhất và ngu nhất. Database là một trong những nơi hay bị tấn công, dễ làm thất thoát dữ liệu. Trong quá khứ, lỗi SQL Injection từng làm thất thoát hàng triệu thông tin khách hàng và thông tin credit card. Chưa tính đến chuyện hacker bên ngoài, nhiều khi thằng Database Admin hứng lên, nó có thể mò được mật khẩu của khách hàng, lớn chuyện chưa?

Cách lưu trữ mật khẩu đúng phải là làm sao để chỉ người dùng mới biết được mật khẩu của họ. Làm sao ư? Hãy đọc phần dưới nhé.

Vậy mã hóa là được chứ gì, lằm trò!!

Ừ, cách giải quyết cũng khá đơn giản. Bạn có thể dùng hàm hash để mã hóa mật khẩu như sau:

1. Sử dụng hàm hash (hàm băm) để mã hóa mật khẩu của người dùng.
2. Lưu trữ mật khẩu này dưới database.
3. Khi người dùng đăng nhập, hash mật khẩu đã nhập, so sánh với mật khẩu đã lưu dưới database.

Hàm hash này phải là hàm hash một chiều, không thể dựa theo mật khẩu đã hash để suy ngược ra đầu vào.

```
public void Register(string username, string password)
{
    string hashedPassword = HashHelper.Hash(password);
    Database.SaveUser(username, hashedPassword);
}

public bool Login(string username, string password)
{
    string pw = Database.GetHashedPasswordByUsername(username);
    return pw == HashHelper.Hash(password);
}
```

Cách này đảm bảo chỉ người dùng biết mật khẩu của họ, dù là lập trình viên hay database admin, có nắm được cả code lẫn database cũng không tài nào mò ra mật khẩu. Tuy nhiên, cách này có một vấn đề: Hai mật khẩu giống nhau khi hash sẽ có kết quả giống nhau. Hacker có thể mò ra mật khẩu bằng cách dùng dictionary attack – hash toàn bộ các mật khẩu có thể trong từ điển, rồi so sánh kết quả với mật khẩu đã hash dưới database.

Thế nhưng, vỏ quýt dày có móng tay nhọn. Đây là cách lưu trữ mật khẩu đúng mà hiện nay các framework đều áp dụng:

1. Khi tạo mật khẩu, tạo random một chuỗi kí tự gọi là salt.
2. Salt sẽ được cộng vào sau mật khẩu, toàn bộ chuỗi mật khẩu và salt sẽ bị băm (hash).
3. Lưu salt và giá trị đã băm xuống database (Một người dùng sẽ có 1 salt riêng).
4. Khi người dùng đăng nhập, lấy salt của người dùng, cộng nó với mật khẩu họ nhập vào, hash ra rồi so với giá trị trong database.

```
public void Register(string username, string password)
{
    string salt = SaltHelper.getRandomSalt();
    string hashedPassword = HashHelper.Hash(password + salt);
    Database.SaveUser(username, hashedPassword, salt);
}

public bool Login(string username, string password)
{
    string salt = Database.getSaltByUsername(username);
    string pw = Database.GetHashedPasswordFromUsername(username);
    return pw == hash(password + salt);
}
```

Với cách này, khi người dùng quên mật khẩu, hệ thống không tài nào mò ra mật khẩu để gửi cho họ. Cách giải quyết duy nhất là reset mật khẩu, random ra một mật khẩu mới rồi gửi cho người dùng.

Ồi giờ phức tạp thế, cùng lắm thì lộ password trên trang của mình thôi mà

Nói nhỏ một bí mật (mà chắc ai cũng biết) cho các bạn nghe nè: Hầu như người dùng chỉ sử dụng 1 username/mật khẩu duy nhất cho toàn bộ các tài khoản trên mạng. Nếu hacker tìm được mật khẩu từ trang của bạn, chúng sẽ thử với các account facebook, gmail, tài khoản ngân hàng, ... của người đó.

Mất 1 account là xem như mất sạch sành sanh. Kinh khủng chưa! Không tin à, bạn thử ngẫm lại xem, bạn có dùng chung 1 email/mật khẩu cho Gmail, Facebook, Evernote, ... và nhiều trang khác không?

Lỗi hổng bảo mật khủng khiếp của Lotte Cinema

Một ngày đẹp trời nọ, mình định dẫn gấu đi xem phim, ăn uống rồi *beep*. Định đặt vé online mà quên mất mật khẩu lottecinema.com, mình mò mẫm phần đăng nhập, tìm hoài mới thấy mục “Quên mật khẩu”. Nhập địa chỉ mail và chứng minh nhân dân, mình mau chóng nhận được một email gửi từ lottecinema, trong đó có cả username và mật khẩu của mình.



Thật là tiện quá đi mất, khỏi phải reset mật khẩu. Khoan, có cái gì sai sai ở đây!! Vậy là bọn lotte lưu thẳng mật khẩu của mình thẳng dưới database à. Lỡ database bị thất thoát dữ liệu là toàn bộ các tài khoản khác của mình (Và các thành viên lotte cinema khác) cũng đi tong theo.

Thật là đáng sợ!! Lỗi này mình phát hiện năm ngoái, đến cách đây mấy ngày vẫn còn y nguyên. Thế mới biết bộ phận IT của lottecinema giỏi giang thế nào. Các bạn có tài khoản lotte cinema thì nhớ cẩn thận nghe.

TÔI ĐÃ HACK “TÔI TẢ” WEB SITE CỦA LOTTE CINEMA NHƯ THẾ NÀO?

Làm một developer “có tâm”, chúng ta không chỉ phải đảm bảo code chạy được, mà còn phải bảo đảm về bảo mật (với các hệ thống quan trọng). Có nhiều lúc, lỗi bảo mật đến từ chính sự ẩu tả của developer. Trong chương này mình sẽ lôi trang web của Lotte Cinema ra làm mẫu để giải thích cho các bạn.

Lưu ý: Bài viết mang tính chất học thuật, bình luận về kĩ thuật. Mình không ủng hộ, cũng không chịu trách nhiệm nếu bạn mang kiến thức đề cập trong bài ra làm chuyện trái pháp luật! Thân.

Giới thiệu

Tại sao mình lại chọn Lotte Cinema? Đơn giản là cách đây mấy tháng, khi nhắc đến mật khẩu, mình đã nêu ra một lỗ hổng bảo mật khủng khiếp của Lotte Cinema: Lưu mật khẩu dưới dạng text.

Đến nay, lỗ hổng này vẫn chưa được sửa, điều này chứng tỏ hai chuyện: Đội ngũ lập trình web lotte cinema thiếu kiến thức cơ bản về lập trình và cũng không thèm quan tâm gì đến việc bảo trì sửa lỗi. Điều này đồng nghĩa với việc website sẽ có nhiều lỗ hổng để khai thác.

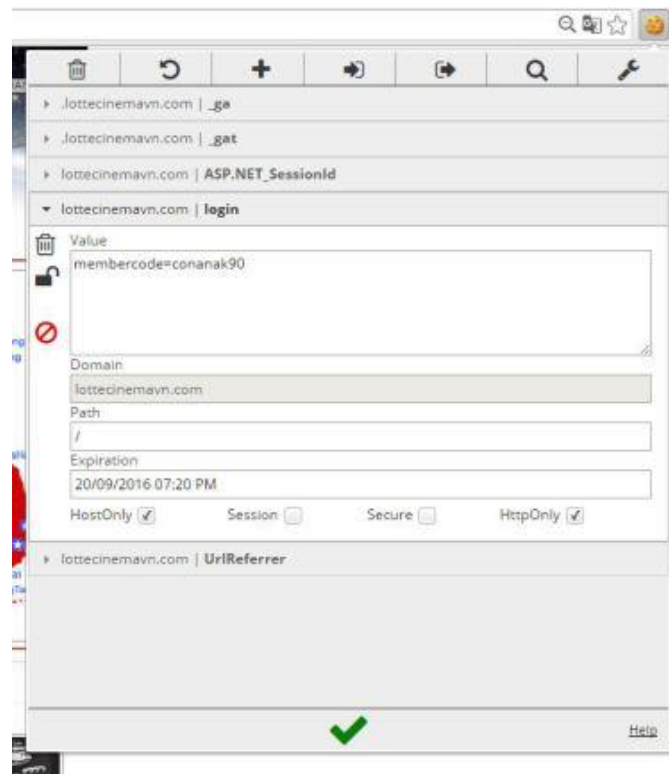
Với logic đó, mình bắt đầu tìm lỗ hổng của lotte với tâm thế học hỏi. Thật không ngờ, mình tìm được không chỉ một, mà đến tận vài lỗ hổng... cực kì chết người, có thể làm toàn bộ hệ thống ngừng hoạt động.

Bắt đầu “câu cá”

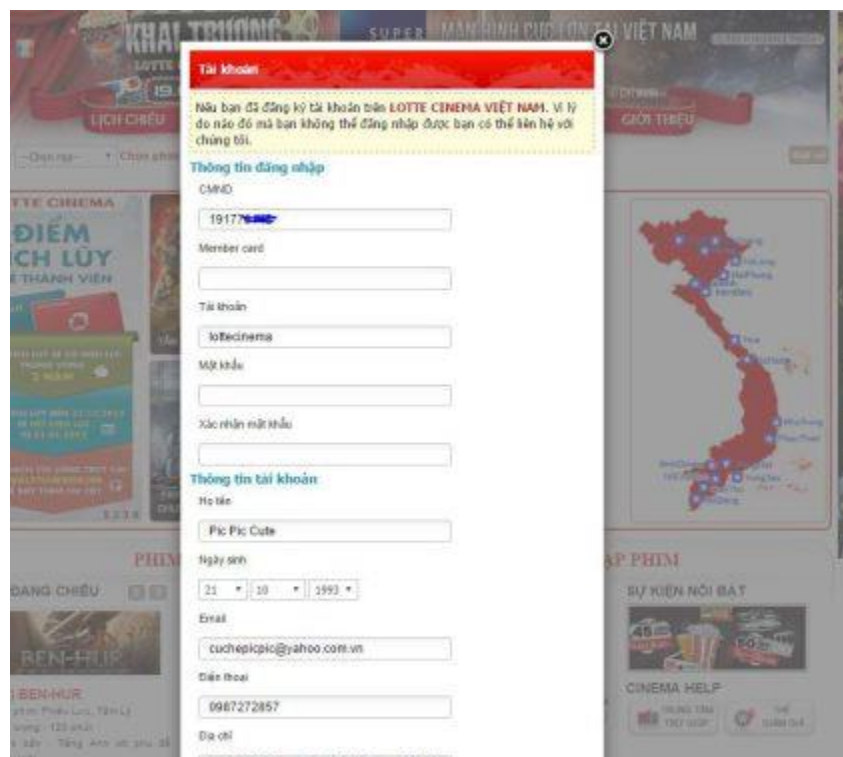
Đầu tiên, hãy nhìn góc trên bên trái trình duyệt. Một website không có https, đồng nghĩa với việc toàn bộ thông tin bạn điền vào (username, password) hoàn toàn có thể bị hacker trộm nếu bạn dùng chung đường dây mạng/chung wifi với hacker đó (Xem thêm về sniffing). Đó là lý do các trang ngân hàng, facebook, gmail, thanh toán điện tử đều đòi hỏi phải dùng https.



Tiếp theo, ta bắt đầu với việc kiểm tra cookie. Các bạn tải addon EditThisCookie về để làm việc nhé. Thử đăng nhập và xem lotte cinema lưu gì trong cookie nào.



Các bạn không nhìn lầm đâu, chính là username của các bạn đấy? Thôi, chúng ta cứ cầu trời là họ lưu username để nhắc bạn khi bạn cần đăng nhập lại thôi hạ. Thử đổi sang giá trị khác rồi refresh trang xem nào.



CÁI LẼ GÌ THỐN!!! Mình bị chuyển sang nick khác mất rồi. Thật không thể tin nổi. Một lỗi bảo mật to như bánh xe bò đã bị lộ chỉ sau 5p nghiên cứu. 1-0 cho Lotte Cinema. (Lỗi này có tên gọi là impersonation).

Câu nhảm ... “cá mập”

Nhờ đổi cookie, mình đã hack được vào tài khoản người khác. Ok ngon, có thông tin người dùng luôn! Giờ mình thử đổi thông tin xem nào, được luôn. Thử đặt vé xem nào, cũng được nốt!

THÔNG TIN ĐẶT VÉ



KUBO AND THE TWO STRINGS
Rạp: Landmark
Màn Hình: Scm04
Ngày chiếu: 25-08-2016
Giờ chiếu: 19:55
Ghế: D7-D8-D9
Giá:
D7 : 80.000
D8 : 80.000
D9 : 80.000
Tổng cộng: 240.000

KIỂM TRA NGAY THÔNG TIN TÀI KHOẢN !!!

Họ tên: Pic Pic Cute
Email: cuchepicpic@yahoo.com.vn
Điểm:

Có thể dự dõ Lotte Cinema gửi mật khẩu cho mình không nhỉ? Thử xem nào, đổi email của user này sang sang email mình, sau đó báo mất mật khẩu. Vào email xem sao?



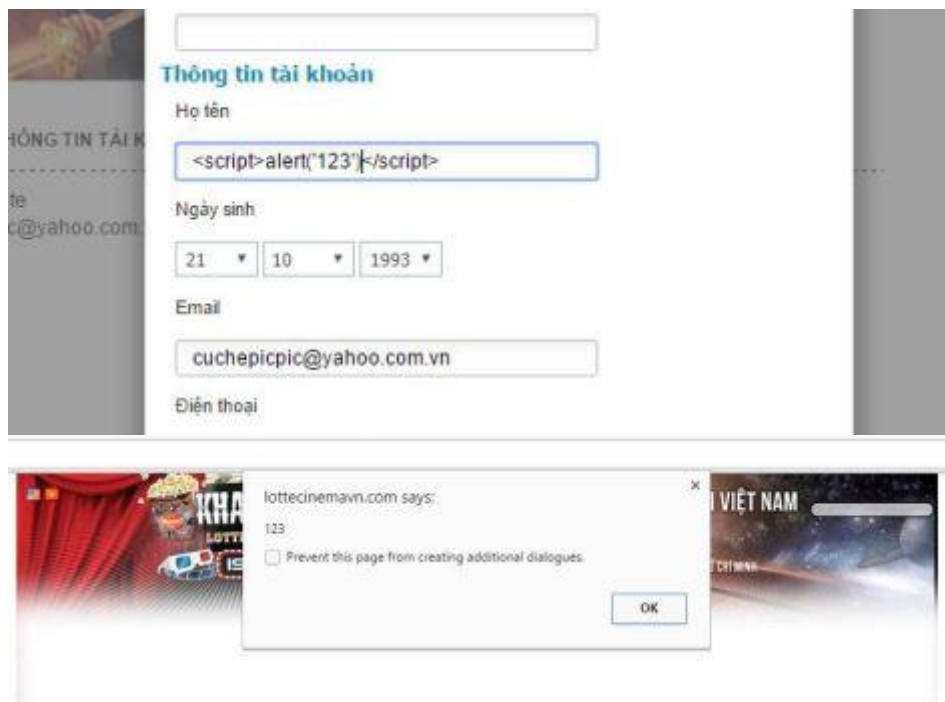
Ồ, nhận được mật khẩu hiện tại luôn, mail của Lotte nhanh thật! Vì một user thường tái sử dụng mật khẩu ở nhiều trang, mình có thể thử dùng username và mật khẩu này ở một số trang khác để mò account. Thấy chết người chưa??

Thử đổi mật khẩu hiện tại xem, được luôn. Giờ mình đã có thể đăng nhập với mật khẩu mới đổi. Đây là lỗi thứ 2: Khi thay đổi mật khẩu, bắt buộc người dùng phải đổi mật khẩu cũ. Tỉ số giờ đã là 2-0 cho Lotte Cinema.

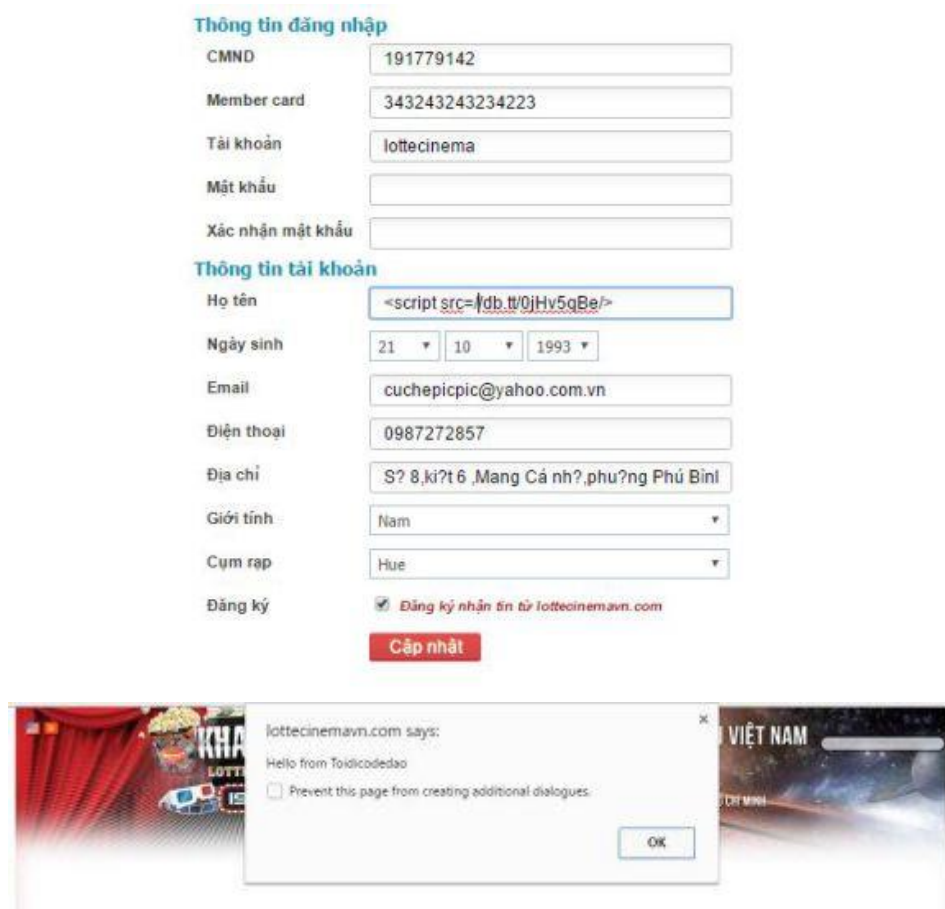
Bonus thêm “cá voi”

Hai lỗi trên đã đủ làm tơi tả toàn bộ hệ thống. Chỉ cần viết một con bot nho nhỏ, lần lượt thay giá trị membername trong cookie (từ a tới zzzzzzz) là có thể lấy gần như toàn bộ thông tin khách hàng, hoặc đổi toàn bộ password làm người dùng không đăng nhập được. (Các bạn khác dùng username dài quá thì chịu).

Thế nhưng mọi chuyện chưa dừng ở đây. Mình tiếp tục thử nghiệm điền tiên vào khung “Họ tên”. Lotte tiếp tục lòi ra lỗi XSS (Tấn công bằng cách chèn script vào trang chính).



Khá may mắn Lotte là đã cắt chuỗi thành 30 kí tự nên không thể điền JavaScript dài. Tuy nhiên, điều này vẫn không thể làm khó được mình khi viết file javascript ở nơi khác, sau đó embed script vào (Up file js lên dropbox rồi lấy shortlink là xong).



Lỗi XSS này chỉ hiện ra ở mỗi trang của user nên không thể dùng để deface website. Tuy nhiên, mình vẫn có thể hiện pop-up giả mạo người dùng tải virus như hình dưới. Dùng JS, mình có thể lấy số thẻ, số CMND để người dùng tin tưởng rằng message là của lotte.



Kết hợp với con bot đã nói phía trên, mình hoàn toàn có thể dụ dỗ rất nhiều người dùng Lotte tải virus khi họ đăng nhập vào hệ thống. Không còn lời nào để nói, 3-0 cho Lotte Cinema!

Kết luận

Những lỗi bảo mật mình chỉ ra không có gì cao siêu! Do mình không phải dân chuyên về bảo mật nên những kỹ thuật tấn công của mình cũng chỉ dừng ở mức vô cùng cơ bản. Vấn đề của Lotte Cinema là ở chỗ họ “không biết tí gì về bảo mật”, dẫn đến chuyện hệ thống bảo mật quá kém.

Như các bạn đã thấy, hành vi này là sự thiếu tôn trọng khách hàng và còn có thể gây nguy hại cho người dùng. Tuy nhiên, có vẻ Lotte Cinema đã rất khôn ngoan trong khâu pháp lý khi rũ bỏ mọi trách nhiệm trong phần “Thỏa Thuận”. Tuy thua 3-0 nhưng vẫn không phải chịu trách nhiệm gì, hoan hô Lotte Cinema.



Vì lý do đạo đức, mình đã gửi nội dung bài viết cho những người có trách nhiệm trên Lotte Cinema một khoảng thời gian khá lâu trước khi công bố. Tuy vậy, họ vẫn làm ngơ và không thèm quan tâm. RIP các bạn và các khách hàng của Lotte Cinema. Dù vậy, mình vẫn khuyên các bạn không nên thử phá hoại hệ thống. Mình không muốn ngày mai lên Mương 14 lại thấy tin: [Hacker trẻ tuổi bị Lotte Cinema bắt. “Tất cả là do em lỡ xem Tôi đi code dạo”] đâu.

Lời khuyên cuối cùng: các bạn vẫn có thể xem phim ở Lotte, nhưng đừng điền bất kì thông tin cá nhân gì vào cái hệ thống trời đánh của nó nhé! Thân chào.

Các bạn có thể xem video tóm tắt bài viết ở đây: <https://www.youtube.com/watch?v=CtnfOZmKR3A>. Nhớ like và subscribe trong link này nhé: https://www.youtube.com/c/toidicodedaoblog?sub_confirmation=1. Mình đang cần 100 sub để xin Custom URL cho Channel Tôi Đi Code dạo.

Update (30/08/2016)

Sau khi bài viết được công bố rộng rãi trên MXH thì bên chịu trách nhiệm xây dựng website cho Lotte Cinema đã liên hệ trực tiếp với mình mình. Đến ngày 1/9/2016 thì các lỗi bảo mật trong bài đã tạm được fix rồi nhé.



The screenshot shows a Facebook chat thread. At the top, a user with a rainbow profile picture (likely the author) sends a message at 14:40: "chào bạn". Another user responds at 14:40, mentioning they read an article about a security issue and providing a link: <https://toidicodedao.com/2016/08/30/hack-lotte-cinema/>. A preview card for the article is shown, with the title "Tôi đã hack 'tơi tả' Web Site của Lotte Cinema như thế nào?" and a snippet of text. The author then responds at 14:41, explaining they are responsible for building the website and that the team is working on fixing the issues. A user named "Hien Vo Thai" thanks them at 15:05. The author, "Huy Hoàng Phạm", responds at 15:16, asking for the hash password. Another user responds at 15:19, saying "okie bạn". The author then explains at 15:20 that they thought they fixed it but found an XSS vulnerability. The user responds at 15:20, saying they are not sure about the situation. The author responds at 15:20, asking to validate on the server side. The user responds at 15:21, saying they will fix it today.

chào bạn 30/08/2016 14:40

mình có đọc bài về lỗi bảo mật
<https://toidicodedao.com/2016/08/30/hack-lotte-cinema/> 30/08/2016 14:40

Tôi đã hack "tơi tả" Web Site của Lotte Cinema như thế nào?
Là một developer "có tâm", chúng ta không chỉ phải đảm bảo...
toidicodedao.com

bên mình chịu trách nhiệm xây dựng website cho Lotte, thực ra website này đã làm từ lâu, thời điểm đó còn rất nhiều khó khăn và kinh nghiệm để test và fix các lỗi bảo mật
team đang và sẽ fix các lỗi bạn nêu trong blog 30/08/2016 14:41

Hien Vo Thai 30/08/2016 15:05
nhân tiện rất cảm ơn bạn đã chia sẻ và cung cấp thông tin

Huy Hoàng Phạm 30/08/2016 15:16
Ok bạn 😊
À nhớ hash password dưới db luôn nhé 😊

okie bạn 😊 30/08/2016 15:19

Huy Hoàng Phạm 30/08/2016 15:20
Mình nghĩ mấy cái đây fix khoảng 1 buổi thôi bạn 😊
Lỗi XSS thì chỉ việc format lại dữ liệu lúc display
Chặn kí tự lạ

đúng rồi bạn, wan trọng là team ko nắm được tình hình 30/08/2016 15:20

Huy Hoàng Phạm 30/08/2016 15:20
Nhớ validate ở server side nữa nha
Ừm
Mình thấy validate có vẻ chỉ ở client side

ơ bản là phải fix xong trong hôm nay 30/08/2016 15:21



Huy Hoàng Phạm
Mình có gửi file sẵn cho họ luôn
Chứ lúc đó ko public post
Còn vụ ko hash password thì mình đăng cả năm trước cơ

30/08/2016 15:23



Huy Hoàng Phạm


30/08/2016 15:26



Huy Hoàng Phạm
hix, hiện tại team mới fix xong các lỗi
các vấn đề sâu bên trong sẽ có lộ trình update
chắc có gì mai bạn update lại bài viết giúp nhé
thanks bạn

30/08/2016 23:41

Thứ Ba

LOZI.VN ĐÃ “VÔ Ý” ĐỂ LỘ DỮ LIỆU 2 TRIỆU NGƯỜI DÙNG NHƯ THẾ NÀO?

Trong quá trình viết series Bảo mật nhập môn, mình vẫn hay đi nghịch đạo, tìm lỗi bảo mật đạo theo tinh thần “code đạo” của blog. Lẽ tất nhiên, đã tìm lỗi thì phải tìm các trang to to, nhiều người dùng một tí, chứ trang nho nhỏ thì ai quan tâm.

Là developer, mình không đủ giỏi về mạng hay hạ tầng để có thể tấn công server hay DDOS gì đó. Vì vậy, mình quyết định chỉ kiểm tra web và app, hai thứ mình rành nhất. Việt Nam nói là làm, mình bắt đầu truy cập website của app của 1 số ông lớn như tiki, lazada, foody....

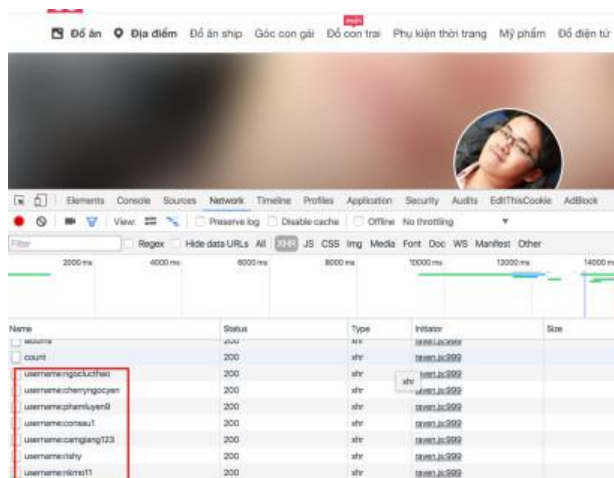
📁 Foody.apk	Nov 16, 2016, 8:01 PM
📁 Tiki vn Shopping Ha....1_apkpure.com.apk	Nov 16, 2016, 7:53 PM
📁 Sendo vn Mua sắm...16_apkpure.com.apk	Nov 17, 2016, 6:18 AM
📁 Lazada.apk	Nov 16, 2016, 7:37 PM
📁 Lozi Đăng và bán_v....0_apkpure.com.apk	Nov 16, 2016, 7:28 PM
📁 NhạcCuaTui.apk	Nov 16, 2016, 7:40 PM
📁 ZingMP3.apk	Nov 16, 2016, 7:42 PM
📁 ZALORA.apk	Nov 16, 2016, 7:59 PM
📁 Zingvn.apk	Nov 16, 2016, 7:41 PM
▶ 📁 Foody	Nov 16, 2016, 2:39 PM
▶ 📁 Lazada	Nov 16, 2016, 11:08 PM
▶ 📁 Lozi	Nov 16, 2016, 9:05 PM
▶ 📁 NhạcCuaTui	Nov 17, 2016, 3:10 PM
▶ 📁 ZingMP3	Nov 17, 2016, 3:16 PM
▶ 📁 Zingvn	Nov 17, 2016, 3:20 PM

Việc dò lỗi cũng giống như câu cá vậy, đôi khi câu được cá bự, đôi khi câu cả buổi không được con nào. Kì này, mình câu được một con cá nho nhỏ mà... nguy hiểm của lozi.vn.

Dò tìm từ web

Khi vào giao diện lozi.vn, đập vào mắt mình là lỗi bự nhất: không có HTTPS! Nói đơn giản, lướt web có thông tin quan trọng mà không có HTTP cũng giống như các bạn đi mát xa, nhâm, đi chịch mà không dùng BCS vậy. Hacker có thể chôm dữ liệu của bạn trong nháy mắt khi bạn không hay biết gì. (Xem thêm về độ bảo mật của giao thức HTTP).

Tiếp theo, mình bắt đầu nghịch ngợm bằng cách ... mở Chrome Developer Tool. Đừng coi thường nó nhé, công cụ này “bá đạo” lắm đấy. Chà, thử xem ta có gì nào?



Mình biết mình đẹp trai, nhưng các bạn đừng nhìn mình mà hãy nhìn vùng khoanh đỏ

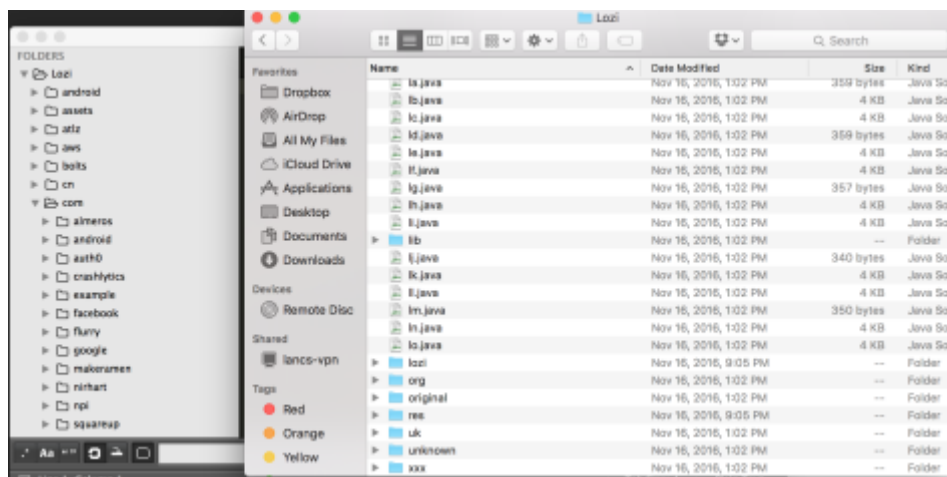
Một loạt hàm AJAX dạng get, truyền vào username và lấy thông tin user. Đặc biệt hơn, trong JSON hàm này trả về bao gồm cả thông tin nhạy cảm như địa chỉ cá nhân, ngày tháng năm sinh, e-mail.

Hàm GET này không có authentication, nên mình hoàn toàn có thể lần lượt thay username vào và lấy thông tin của toàn bộ user. Tuy nhiên, việc test lần lượt từng username khá lâu, nên cách này không khả thi lắm.

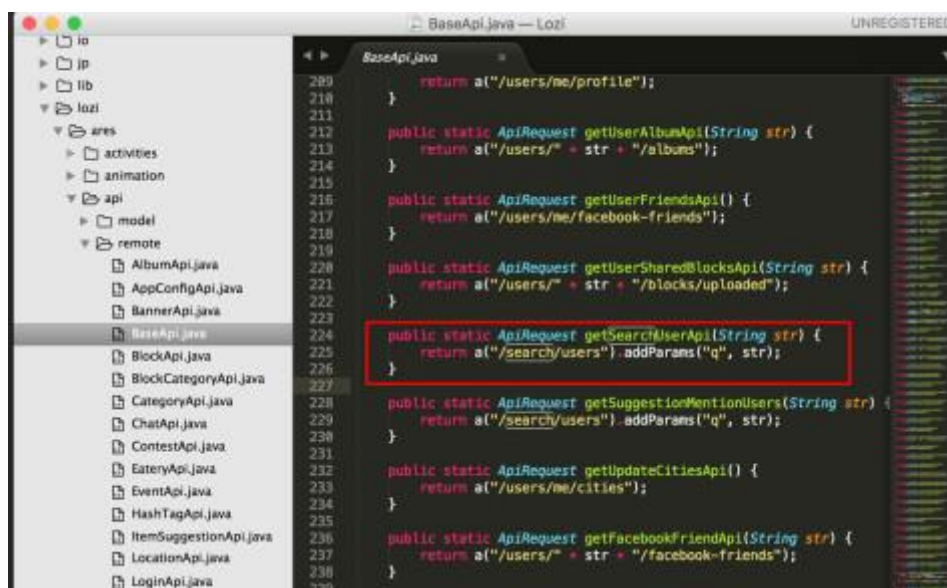
Làm sao tiếp tục? Mình bắt đầu chuyển qua nghịch... ứng dụng mobile của lozi.

Đến app mobile

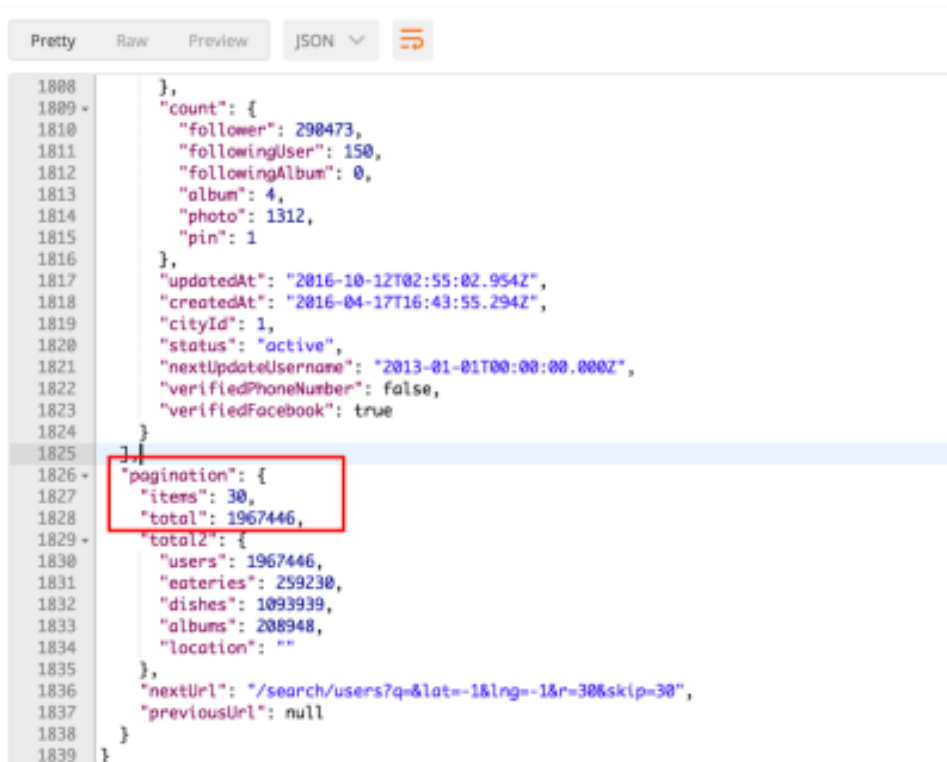
Có một sự thật “nhỏ nhỏ” mà ít bạn biết là: Mặc dù mình hay viết bài về C# và JavaScript nhưng thật ra mình cũng khá rành Java và Android đấy nhé. Thôi không khoe nữa, quay lại chủ đề chính nào. Việc nghịch ứng dụng cũng không quá phức tạp. Mình chỉ cần lên apkpure.com tải file apk, sau đó dùng tool decompile là đã có source code ứng dụng android của lozi rồi.



Có vẻ lúc publish, team lozi chưa obfuscate code nên code vẫn y nguyên. Do team code rất đúng chuẩn OOP và SOLID nên cũng không quá khó khăn để mình lục tìm đoạn code gọi API của lozi. Đoạn code khiến mình chú ý chính là đoạn gọi API SearchUser.

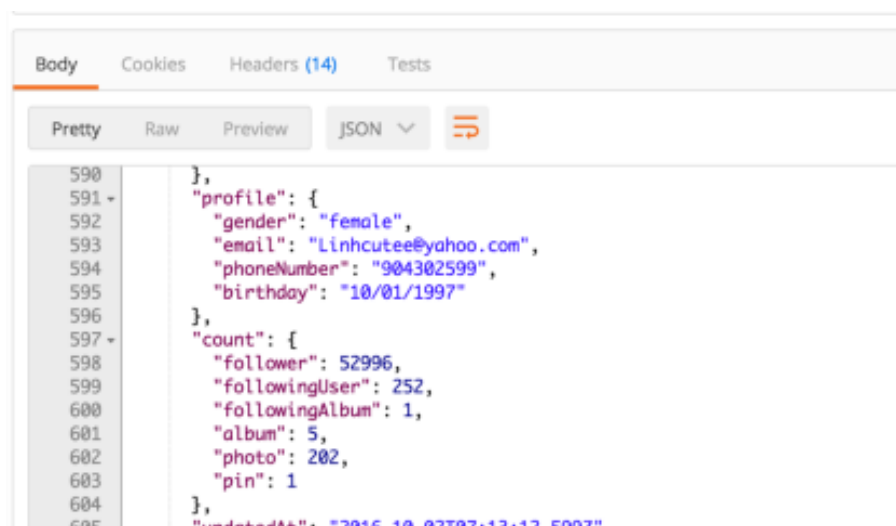


API này có dạng GET nên mình không cần thêm thông tin gì. Bật Postman lên, nhập url của API vào và ... bingo. Thông tin 2 triệu người dùng đây rồi.



```
1808      },
1809      "count": {
1810        "follower": 290473,
1811        "followingUser": 150,
1812        "followingAlbum": 0,
1813        "album": 4,
1814        "photo": 1312,
1815        "pin": 1
1816      },
1817      "updatedAt": "2016-10-12T02:55:02.954Z",
1818      "createdAt": "2016-04-17T16:43:55.294Z",
1819      "cityId": 1,
1820      "status": "active",
1821      "nextUpdateUsername": "2013-01-01T00:00:00.000Z",
1822      "verifiedPhoneNumber": false,
1823      "verifiedFacebook": true
1824    },
1825    "pagination": {
1826      "items": 30,
1827      "total": 1967446,
1828      "total2": {
1829        "users": 1967446,
1830        "eateries": 259230,
1831        "dishes": 1093939,
1832        "albums": 208948,
1833        "location": ""
1834      },
1835      "nextUrl": "/search/users?q=&lat=-1&lng=-1&r=30&skip=30",
1836      "previousUrl": null
1837    },
1838  },
1839 }
```

Có cả link để paging nhé



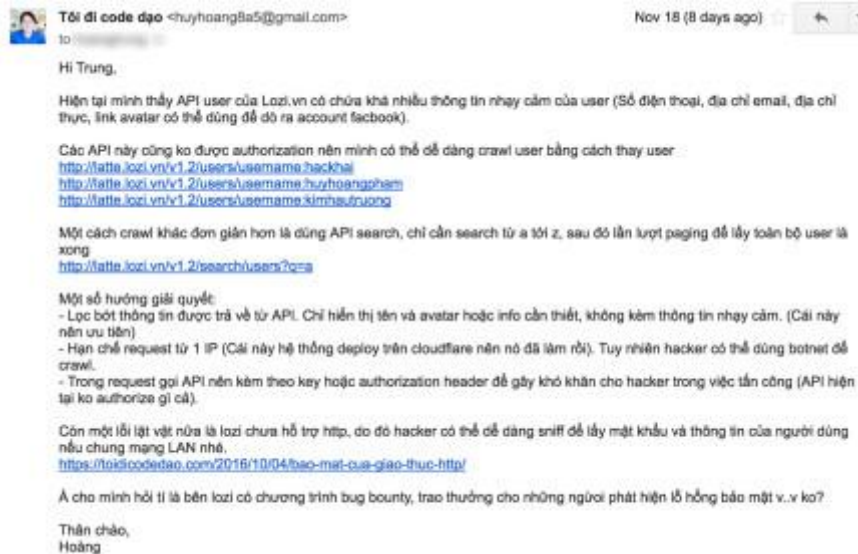
```
590      },
591      "profile": {
592        "gender": "female",
593        "email": "Linhcutee@yahoo.com",
594        "phoneNumber": "904302599",
595        "birthday": "10/01/1997"
596      },
597      "count": {
598        "follower": 52996,
599        "followingUser": 252,
600        "followingAlbum": 1,
601        "album": 5,
602        "photo": 202,
603        "pin": 1
604      },
605      "updatedAt": "2016-10-02T07:13:12.500Z"
```

Bao gồm các thông tin nhạy cảm như email, số điện thoại, ngày sinh...

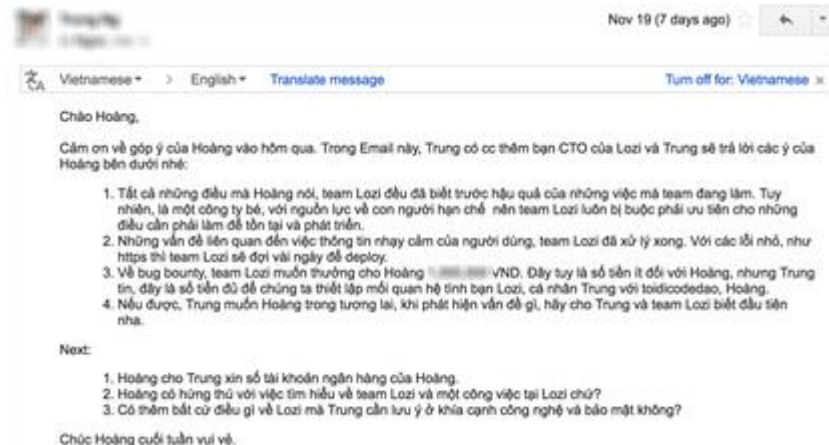
Quá trình xử lý lỗi

Tối thứ 4 ngày 16/11, mình tìm ra lỗi này, bắt đầu liên hệ với lozi.vn.

Chiều thứ 6 ngày 18/11, mình nhận được reply từ fanpage của lozi. Khoảng 5 phút sau khi mình gửi mail cho team lozi thì lỗi đã được fix ngay lập tức.



Ngay sáng thứ 7 ngày 19/11, mình đã nhận được mail reply rất tận tình của người chịu trách nhiệm dù đang là thứ 7. Hoan hô lozi. Thái độ làm việc khác hẳn với bên lotte cinema, bỏ mình hơn nửa tháng trời.



Khoảng 4,5 ngày sau khi mình báo cáo lỗi thì lozi cũng đã cập nhật https và thêm token cho các API rồi nhé.

Nhận xét

Trong suy nghĩ chung của developer, các RestAPI này thường bị ẩn đi, người dùng không thấy nên không thể nghịch được. Tiếc thay, developer và hacker có thể dễ dàng decompiler app và “nghịch ngợm” các API này.

Thật ra, không chỉ có team lozi mà đa phần các team khác cũng khá thiếu cảnh giác về việc bảo mật API. Điển hình là vụ CGV lộ 3 triệu người dùng cũng do API mobile. Tuy nhiên, team Foody và Lozi đã bảo mật API khá tốt, mình nghịch thử mà không thu được kết quả gì.

Điều đáng nói qua sự việc này là: các lỗ hổng bảo mật này thuộc loại vô cùng cơ bản, mình là dân “tay ngang” có thể khai thác mà không cần tool chuyên dụng (Kali Linux, Tool Penetration Test), chỉ cần Chrome và SublimeText.

Các hệ thống lớn mà đôi khi còn lỏng lẻo kiểu này, liệu dữ liệu của chúng ta có an toàn khi các hacker “chuyên nghiệp” ra tay??

Với cái “tâm” của người developer, mình liên lạc ngay với bên lozi để xử lý. Điều gì sẽ xảy ra nếu người tìm ra lỗi này không phải là mình mà là một hacker “có tâm”, sẵn sàng cào hết dữ liệu về rồi chia sẻ hoặc bán cho các công ty khác, hoặc các công ty đối thủ cạnh tranh với lozi?

Thay lời kết

Đây cũng là phần cuối cùng của cuốn sách. Chân thành cảm ơn các bạn đã bỏ thời gian đọc và ủng hộ!

Một điều mình sẽ nhắc đi nhắc lại trong suốt series là: Đừng bao giờ tin tưởng người dùng!! Đừng bao giờ tin tưởng những thứ người dùng nhập vào, đừng nghĩ rằng người dùng không biết sửa javascript, không biết nghịch lung tung. Dưới danh nghĩa người dùng, hacker có để mọi phương cách để tấn công hệ thống. Nhớ đấy nhé!

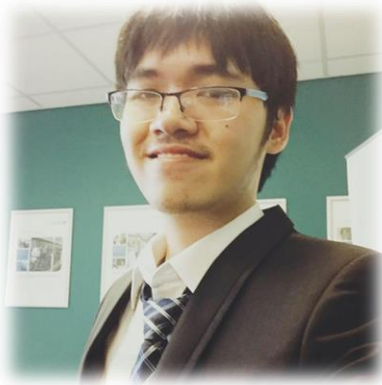
Việc post bài của mình cũng chỉ mang tính chất và cảnh tỉnh chứ không có ý khoe khoang hầy gì khác. Với các hacker "có tâm", họ phải lên kế hoạch tấn công, hoặc phải tốn công sức nghiên cứu để tìm được lỗ hổng chưa ai tìm ra. Hành động của mình chỉ là đi mày mò, nghịch ngợm các lỗi sơ đẳng của developer, tính ra cũng chẳng có gì tự hào để phải khoe cả ;)). Bất kì một hành động tấn công, phá hoại hệ thống nào nhằm "thể hiện" đều là những hành động trẻ trâu, thiếu suy nghĩ, có thể dẫn đến "tình tiền tù tội". Các bạn nhớ suy nghĩ cẩn thận trước khi hành động.

Mình chỉ có một hi vọng “nhỏ nhoi” là **cuốn ebook này được nhiều người biết tới hơn**. Nếu lập trình viên nào cũng biết những lỗi bảo mật cơ bản thế này, ta sẽ không phải gặp những lỗ hổng “ngớ ngẩn” kiểu lottecinema hay vietnamwork nữa. Cùng giúp mình chia sẻ nó tới nhiều bạn đọc hơn nhé!

Hãy nhớ rằng, bảo mật là một chuyên ngành rất lớn, thế giới bảo mật rất bao la. Những lỗi bảo mật mới xuất hiện từng ngày, không thua gì công nghệ mới trong lập trình. Quyển ebook nhập môn này chỉ cover được một phần rất nhỏ trong đấy (Còn vô số điều hay ho như: social engineering, row hammering ... không được nhắc tới trong sách). Do vậy, đừng nghĩ rằng đọc xong series là mình đã biết “tuốt tuồn tuột” những điều cần biết về bảo mật. Hãy tự trau dồi thêm kiến thức bảo mật, áp dụng vào code và thiết kế nhé.

Nội dung sách tham khảo theo course Hack Yourself First, Web Security OWASP Top 10 trên pluralsight và một số nguồn khác. Series này có phụ đề nên khá dễ học, các bạn khá tiếng Anh có thể học thử.

Về tác giả



Anh Phạm Huy Hoàng hiện đang theo học tại Thạc sĩ về Khoa Học Máy Tính (Computer Science) tại Đại học Lancaster, Anh. Tại Anh, Hoàng cũng làm Full-stack Developer cho trường. Anh từng phát hiện và công bố lỗ hổng bảo mật của Lotte Cinema và Lozi.vn

Hoàng cũng là chủ blog [Tôi Đi Code Đạo](https://toidicodedao.com) khá nổi tiếng tại Việt Nam. Anh có hơn 4 năm kinh nghiệm trong lĩnh vực phần mềm và rất đam mê nghiên cứu về bảo mật, công nghệ web, các công nghệ mới như Machine Learning, Cognitive.

Giới thiệu sách mới



Trong thời gian sắp tới, mình dự định sẽ **xuất bản một cuốn sách** mang tên: Code đạo ký sự - Lập trình viên đâu phải chỉ biết code. Sách viết về những kĩ năng mềm và cứng mà lập trình viên nào cũng cần biết, độ dài khoảng 200-250 trang.

Liệu bạn có thể bỏ chút thời gian cho vào link này và cho mình chút nhận xét về sách được không: <https://goo.gl/forms/z56ptOE7RZXL6cFU2>. Các bạn sẽ được hưởng được **một số ưu đãi nho nhỏ** (giảm giá, kí tặng, v...v) khi sách ra mắt đấy.

Thông tin liên lạc:

Email: huyhoang8a5@gmail.com

Blog: <https://toidicodedao.com>

Linkedin: <https://www.linkedin.com/in/huyhoangpham92>

CV: <http://cv.toidicodedao.com>