May 2012

Hacker Intelligence Initiative, Monthly Trend Report #10

Dissecting a Hacktivist Attack

1. Executive Summary

At the end of March 2012, the Lulzsec hackers had attacked http://www.militarysingles.com/ and disclosed sensitive information on more than 170,000 members.[1]

The dating site for single soldiers...
MilitarySingles.com
and those interested in meeting them.

This report analyzes the anatomy of the attack methods deployed by the "new" Lulzsec. Overall, the attack, using Remote File Inclusion, is nothing new. But it underscores how today's hackers adhere to Sun-Tzu's maxim: "Strike where your enemy is most vulnerable." RFI vulnerabilities are prevalent in PHP applications which comprise 77% of total applications on the web.

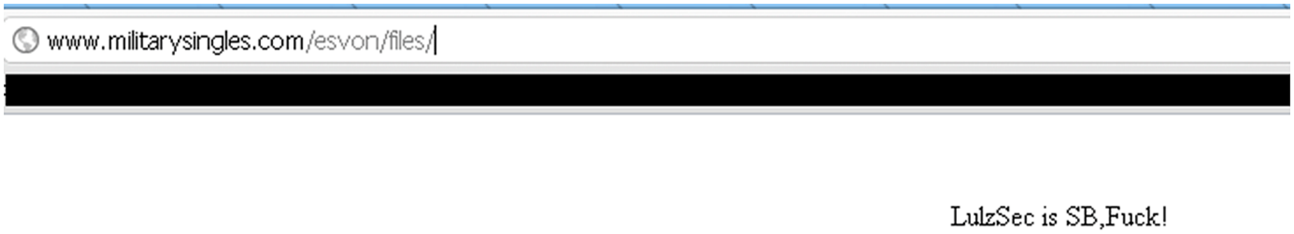
This attack also underscores the need for proper password encryption. In this case, archaic methods of password encryption meant hackers could decrypt the full list of passwords in just 9 hours.

Finally, military service professionals need to recognize that social networks are dangerous and that a different standard of involvement applies to men and women in uniform. As a special target for hacktivists and foreign hackers, the military should set policies in place for servicemen participating in social networks.

[1] http://www.zdnet.com/blog/security/lulzsec-hacks-military-singles/11088

2. Detailed Analysis

Lulzsec was able to load and execute a file on the server as shown in this screenshot:



www.militarysingles.com/esvon/files/

[redacted]

LulzSec is SB,Fuck!

2.1 File upload

The fundamental tenet of Web 2.0, user-generated content, is also the Achilles Heel from a security standpoint. Why? Allowing the upload of user-generated content to the website can be extremely dangerous as the server which is usually considered by other users and the application itself as "trusted" now hosts content that can be generated by a malicious source.

However, in Web 2.0 applications, the upload of user-generated content cannot be avoided. For example, imagine social networks without pictures and webmail without attachments. Very dull user, indeed.

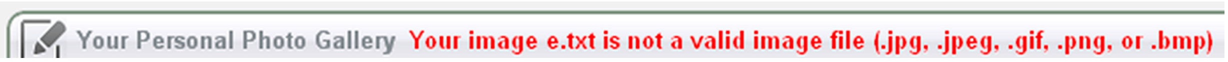
Any uploaded malicious content can be involved in numerous attacks such as:

- File execution – where the uploaded file can be executed on the server. The server trusts the file and allows it to run.
- Local File Inclusion – where the file contains some PHP script. In Local File Inclusion, the application trusts the code hosted on the server and allows it to run.
- Malware hosting – where the file is a malware or infected with malware. The user trusts the web application and downloads the content.
- Cross-site scripting – where the file contains a script, served from the victim's environment application context. The browser trusts that code to access the user data related to the application (such as cookies).
- Phishing – where the file is an HTML file that masquerades as genuine application functionality (such as login) of the victim application. The user trusts that the UI is genuine.

2.1.1 File upload vulnerability

The militarysingles.com had an upload functionality to support the upload of user's profile pictures.

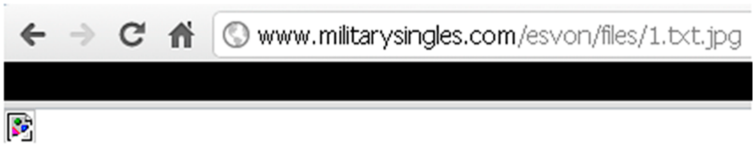
The application designer was apparently aware of the perils of hosting user generated content. In order to prevent rogue uploads, a filter functionality was implemented in order to restrict the upload of files only to picture files.



Your Personal Photo Gallery Your image e.txt is not a valid image file (.jpg, .jpeg, .gif, .png, or .bmp)

But the filter had two major flaws:

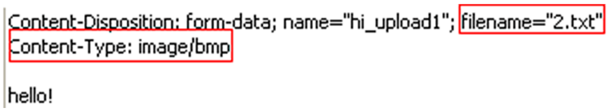
- It validates picture format by extension only, so hackers can upload 1.txt.jpg – that is not a picture and is, in fact, a textual file.



www.militarysingles.com/esvon/files/1.txt.jpg

[redacted]

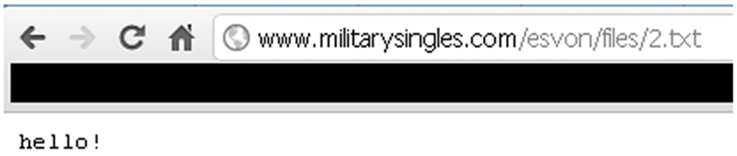
- The filter seems to trust the content type as specified by a browser – which is a client-side control instead of properly checking files on server side. So an attacker, using a proxy to fiddle with traffic after it had passed client-side security implemented on the browser, was able to change the filename without changing the "image" content-type.



```
Content-Disposition: form-data; name="hi_upload1"; filename="2.txt"
Content-Type: image/bmp

hello!
```

And the arbitrary file gets uploaded:

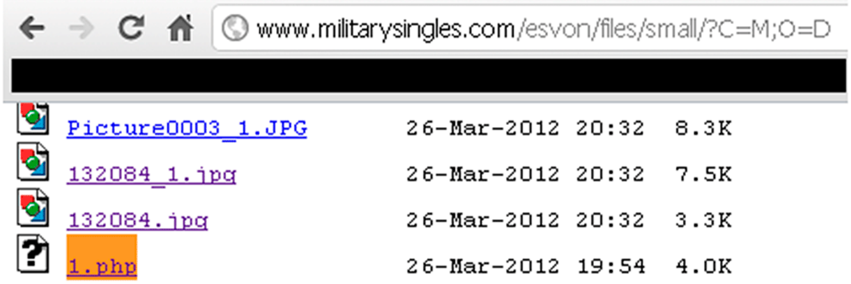


[redacted]

```
hello!
```

We assume the attackers performed this process and changed the uploaded file extension to be PHP – and therefore executable on victim's server.

We were able to find record of such uploads:



[redacted]

```
Picture0003_1.JPG      26-Mar-2012 20:32  8.3K
132084_1.jpg           26-Mar-2012 20:32  7.5K
132084.jpg             26-Mar-2012 20:32  3.3K
1.php                  26-Mar-2012 19:54  4.0K
```

That's probably how the LulzSec attacker has obtained control over the server.

2.1.2 File upload: Getting it right

In order to refrain from allowing malicious user-untrusted content to abuse the trust that's related with the server's trusted content, the server needs to isolate the original trusted content and the new, user-generated, untrusted content. Doing so requires:

- Assigning minimal permissions to the uploaded content – especially not giving the file executable permissions.
- Hosting user-generated content on a different domain. That way, even if the code is malicious it's not evaluated in the context of your site. For example - Facebook stores user uploads on fbcdn.net domain.
- Hosting user-generated content on a different machine – that way, even if the code gets executed, it's not executing on the machine that stores sensitive data and resources. That machine needs to be considered as untrusted and have minimal permissions. For example, Facebook stores user uploads on fbcdn.net domain which is hosted on Akamai machines.

Another key action – filter the uploaded content. This can be done in several ways:

- Whitelisting – make sure the content is a valid instance of the file type that the application expects – e.g., be certain that the picture is a valid jpeg file.
- Blacklisting – scan the file for malicious content, using a relevant scanner, e.g. antivirus to detect malware; HTML scanner to detect XSS.
- All security checks need to be implemented on the server-side and not trust client, as the client cannot be trusted.

2.2 Storing passwords

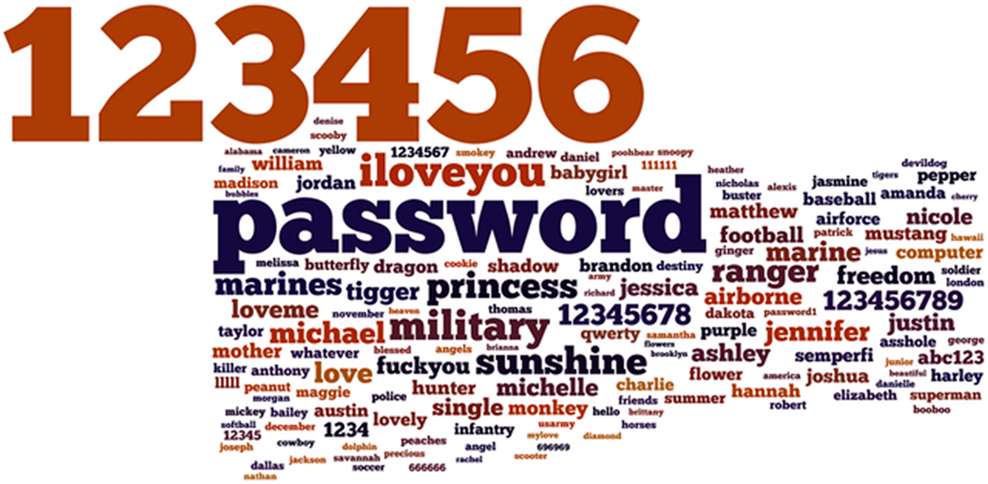
As a result of the code execution, the attacker has gained both full control over the server and the data stored within it. The data included very sensitive details (PII) about the users such as full name, address, phone number, email address, and more.

Even when all users details on the application are exposed, passwords are still considered to be sensitive, as users often reuse the same passwords over many applications. had the passwords been properly stored, it would be difficult for the attacker to find out the customer's passwords.

2.2.1 Cracking passwords

In the militarysingles.com hack, passwords were not stored in plaintext but were hashed. How were the passwords breached? The problem was that the hash was created using the MD5 algorithm and no additional salting[2] was involved. According to one blog,[3] the vast majority (more than 90%) of the disclosed hashes were leaked were cracked in 9 hours.

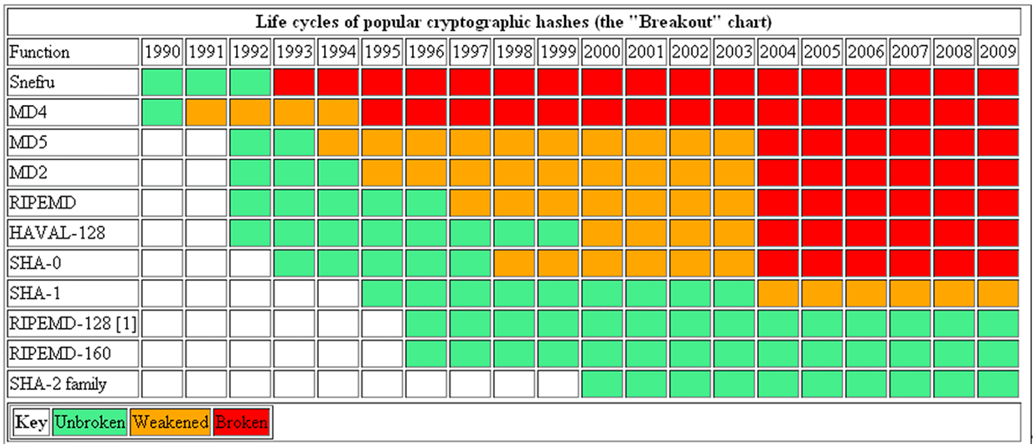
The blog also provided a nice visualization of the common passwords, which is very consistent with Imperva's ADC findings in similar cases.[4]



2.2.2 Storing passwords - Getting it right

What is the correct way to store passwords?

- First, use a modern strong hash function – MD5 is not suitable for modern applications – as shown in the table below.[5]



Life cycles of popular cryptographic hashes (the "Breakout" chart)

Function	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Snefru	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD4	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD5			🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥
MD2			🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥
RIPEMD			🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥
HAVAL-128			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥
SHA-0				🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥
SHA-1						🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧
RIPEMD-128 [1]							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
RIPEMD-160							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
SHA-2 family											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Key 🟩 Unbroken 🟧 Weakened 🟥 Broken

[2] http://en.wikipedia.org/wiki/Salt_(cryptography)
[3] http://iqsecur.blogspot.ca/2012/04/analysis-of-leaked-militarysinglesorg.html
[4] http://www.imperva.com/docs/WP_Consumer_Password_Worst_Practices.pdf
[5] http://valerieaurora.org/hash.html

A key best-practice involves using cryptographic hashes from the SHA-2 family, such as SHA-256. In fact, today, NIST[6] specifications "Federal agencies should stop using SHA-1 for digital signatures, digital time stamping, and other applications that require collision resistance as soon as practical and must use the SHA-2 family of hash functions for these applications after 2010."

- Second, add salt to the hash function – this would make the cracker work much harder as it disables the use of rainbow tables.
- Third, enforce a password policy that would force users create less predictable passwords. When users are allowed to choose their password, they tend to chose very predictable passwords (such as '123456'.)

3. Summary and Conclusions

The militarysingles.com episode illustrates:

- **Web 2.0 has come home to roost**. As we point out, the main driver of today's Internet is also its Achilles Heel: user-generated content – Worse, the current web landscape won't make things better anytime soon. Approximately 77% of Web applications utilize PHP, which is especially vulnerable to Remote File Inclusion (RFI) attacks. In the case of militarysingles.com, users upload photos. This episode highlights how savvy hackers bypass simple checks and filters to execute malicious code on servers, in turn infecting other machines or stealing data. Many in the security community have yet to notice.
- **Social networking and the public sector don't mix** – In addition to the Web application security concerns, the fact that hackers targeted MilitarySingles should not be overlooked. Consideration should be paid to the fact that hacktivists are increasingly using social engineering techniques to infiltrate targets. Imperva calls into question if military and government employees should be held to a higher standard when it comes to social networking.
- **The need for enterprises to pay closer attention to password encryption**. Strong password policies aren't enough. In addition, enterprises must use of a special form of encryption known has "salted digests." A salted value should increase the cost of guessing the password so that financially-motivated hackers will not make such an investment.

Hacker Intelligence Initiative Overview

The Imperva Hacker Intelligence Initiative goes inside the cyber-underground and provides analysis of the trending hacking techniques and interesting attack campaigns from the past month. A part of Imperva's Application Defense Center research arm, the Hacker Intelligence Initiative (HII), is focused on tracking the latest trends in attacks, Web application security and cyber-crime business models with the goal of improving security controls and risk management processes.

[6] http://csrc.nist.gov/groups/ST/hash/policy.html

Imperva
3400 Bridge Parkway, Suite 200
Redwood City, CA 94065

Tel: +1-650-345-9000
Fax: +1-650-345-9004
www.imperva.com