Nama : Leny Novita Sari
NIM : 09011181320027
Keamanan Jaringan Komputer

Methods of Manual Penetration Testing (Actual Exploit)

Penetration Testing adalah serangkaian kegiatan yang dilakukan untuk mengidentifikasi dan mengeksploitasi *vulnerability* (kerentanan keamanan). Ini akan membantu mengkonfirmasi efektivitas atau ketidakefektivan dari langkah-langkah keamanan yang telah dilaksanakan. *Penetration Testing* dapat dilakukan dengan dua cara, yaitu : secara manual dan secara otomatis. *Penetration Testing* Manual adalah pengujian yang dilakukan oleh manusia sedangkan *Penetration Testing* Otomatis adalah pengujian yang dilakukan oleh mesin, jadi testing ini tidak memerlukan orang yang ahli melainkan dapat dijalankan oleh setiap orang yang memiliki pengetahuan dibidang ini.

Umumnya, *Penetration Testing* Manual dilakukan dengan metode sebagai berikut :

- Data Collection

  Pengumpulan data memainkan peran penting untuk pengujian. Mengumpulkan data bisa dilakukan secara manual atau dapat menggunakan *tools* bebas yang tersedia secara *online*, seperti : teknik analisis *webpage source code* dan lain-lain. Tool-tool ini membantu untuk mengumpulkan informasi seperti nama table, versi DB, *database, software, hardware* atau bahkan tentang plugin pihak ketiga yang berbeda, dan lain sebagainya.

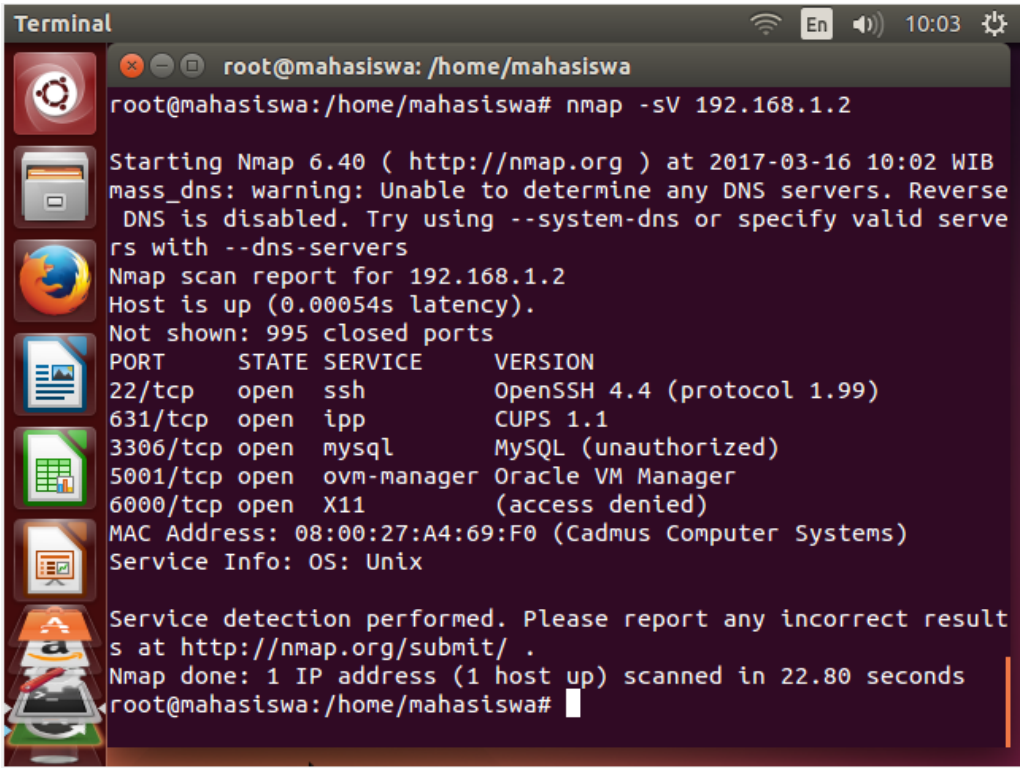
- Vulnerability Assessment

  Setelah data dikumpulkan, hal ini membantu para penguji untuk mengidentifikasi kelemahan keamanan dan mengambil langkah-langkah pencegahan yang sesuai.

- Actual Exploit

  Ini adalah metode khas yang menggunakan tester ahli untuk melakukan serangan pada sistem target dan juga mengurangi risiko serangan.

  Sebelum melakukan serangan, deteksi versi dari service yang sedang berjalan pada target menggunakan nmap. Nmap akan berusaha menentukan nama protokol layanan serta versi dari layananan tersebut dengan menggunakan option "-sV" seperti pada Gambar.1.

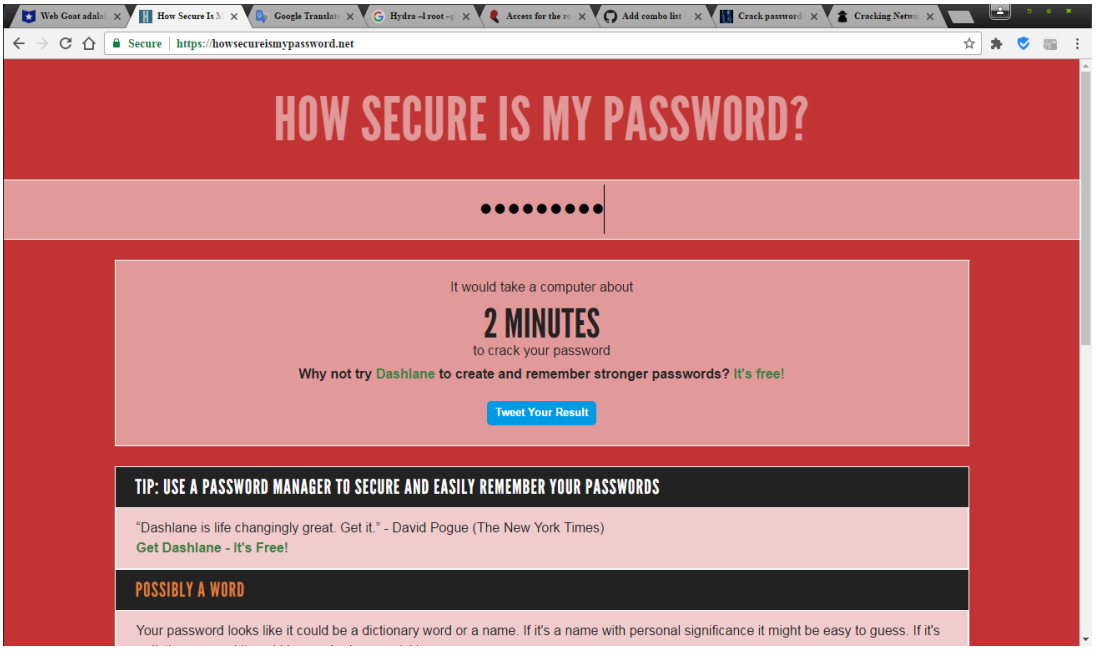


Gambar.1 Deteksi Versi dari Service

Penyerangan dibantu dengan tool hydra, untuk *brute force* hydra dibutuhkan password. Ada banyak password yang tersedia, dalam contoh ini digunakan password default yang disediakan *John the Ripper* yang merupakan tool *Password Cracking*. *Password list* yang diperoleh dari tool *Password Cracking* dapat diubah dengan menggunakan option : **`Mausepad password list,`** jika ingin menghitung jumlah baris pada file *password list* yang telah diperoleh dapat menggunakan option : **`Wc -l password list.`** Option -l akan mencetak jumlah baris pada file *password list*, output pertama sebagai *count* dan outuput kedua sebagai nama file. Setelah daftar password siap, selanjutnya daftar password yang telah ada akan digunakan untuk melakukan *brute force* server ssh untuk memecahkan password, gunakan option : **`Hydra -l root -p pasword lst 192.168.1.1 ssh, -l`** memberitahukan username atau login yang digunakan, –p yang menyediakan daftar kata yang digunakan. Dilanjutkan dengan option : **`Ssh root@192.168.1.2`** untuk melakukan login ssh.

Untuk memastikan seberapa kuat password yang kita gunakan dapat di cek pada website : https://howsecureismypassword.net/, contoh password acak yang saya inputkan

pada https://howsecureismypassword.net/ dapat di crack hanya dalam waktu 2 menit seperti pada Gambar.2.



Gambar.2 Cek Kekuatan Password

Terakhir melakukan *SQL Injection* menggunakan bantuan WebGoat. WebGoat adalah *Project Open Source* yang dapat digunakan agar orang lain bisa belajar *Web Hacking*, salah satunya adalah *SQL Injection*. WebGoat yang digunakan disini adalah WebGoat v5.1. WebGoat adalah suatu web aplikasi J2EE yang dibuat oleh OWASP sebagai pembelajaran *Web Application Security*. WebGoat memberikan pelatihan tentang isu lubang keamanan (*security hole*) pada web, yang dapat terjadi karena beberapa hal seperti :

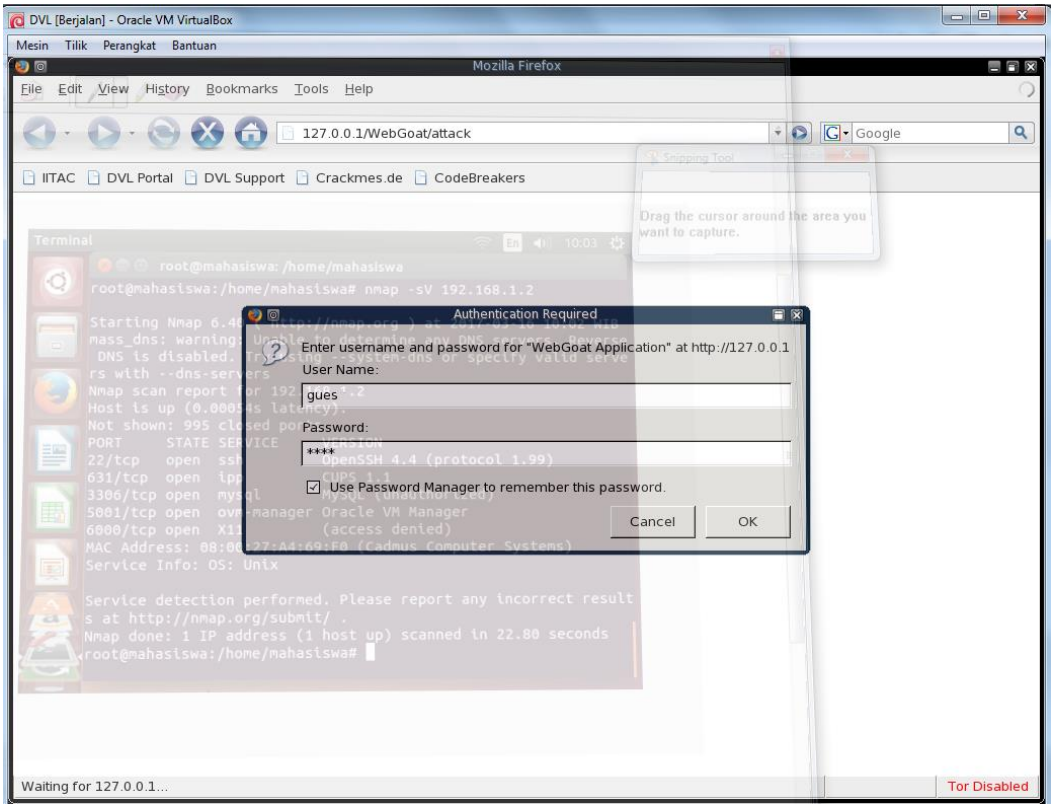
- Salah desain

  Contoh sistem yang lemah desainnya adalah *Algoritma Caesar Cipher*, dimana karakter digeser 13 huruf atau 3 huruf. Meskipun diimplementasikan dengan programming yang sangat teliti, siapapun yang mengetahui algoritmanya dapat memecahkan enskripsi tersebut.

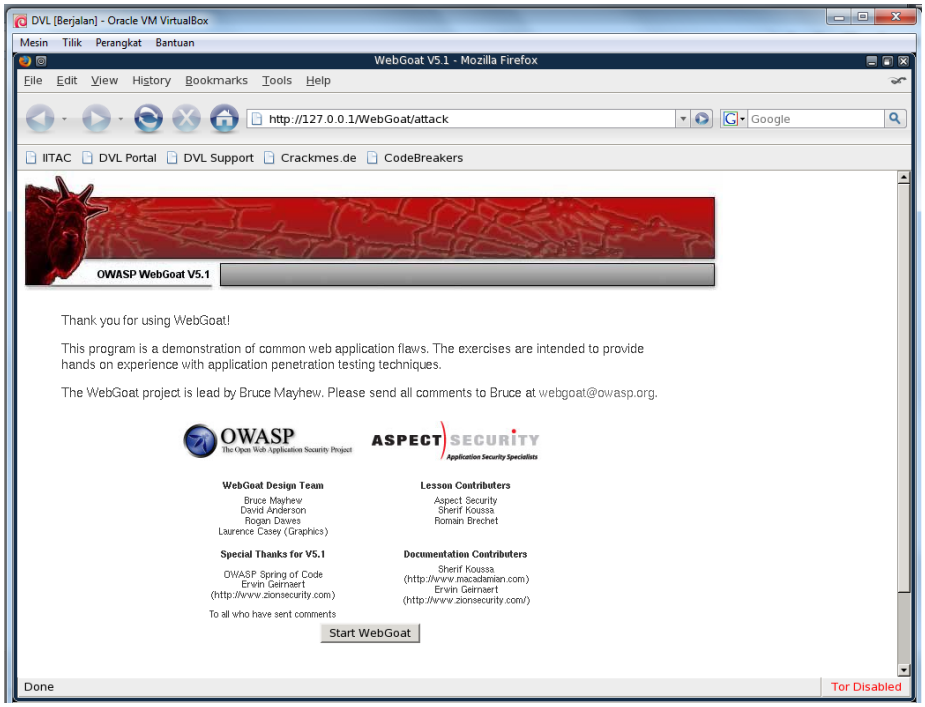
- Salah implementasi

Banyak program yang diimplementasikan secara terburu-buru sehingga kurang cermat dalam proses *coding*, akibatnya testing seringkali dilupakan. Contoh : programmer seringkali lupa memfilter karakter-karakter yang aneh-aneh yang dimasukkan sebagai input dari sebuah program, misalnya seperti kasus dibawah ini.

Mulai menjalankan WebGoat yang telah di download, masuklah ke dalam folder hasil ekstrak, jalankan WebGoat 80 kemudian buka browser dan isilah url dengan http://127.0.0.1/WebGoat/attack. Jika ada pesan untuk memasukkan *username* dan *password*, masukkan *username* : *gues* dengan *password: gues.*

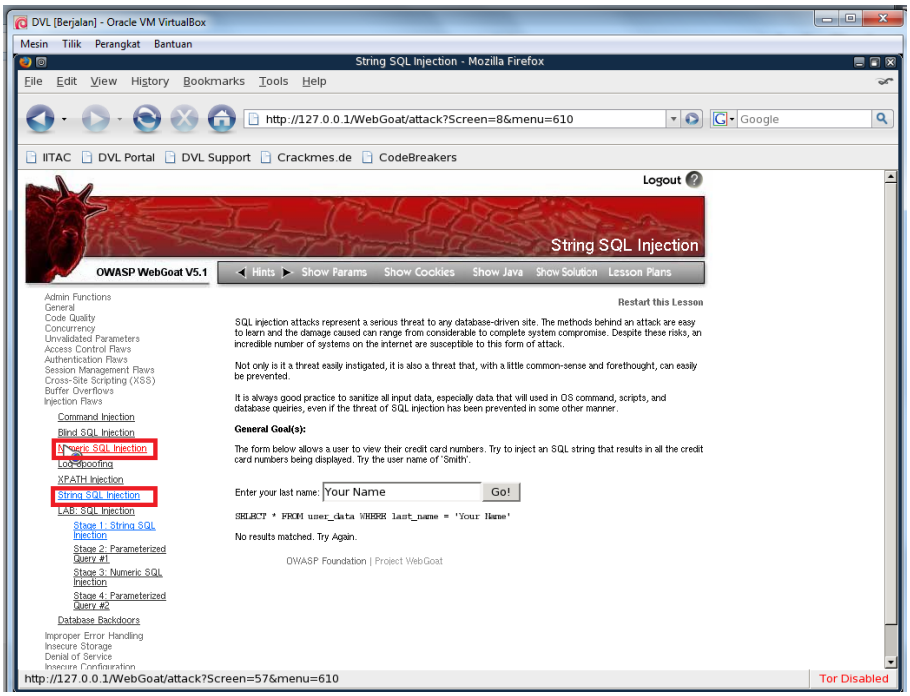


Gambar.3 Tampilan Login WebGoat

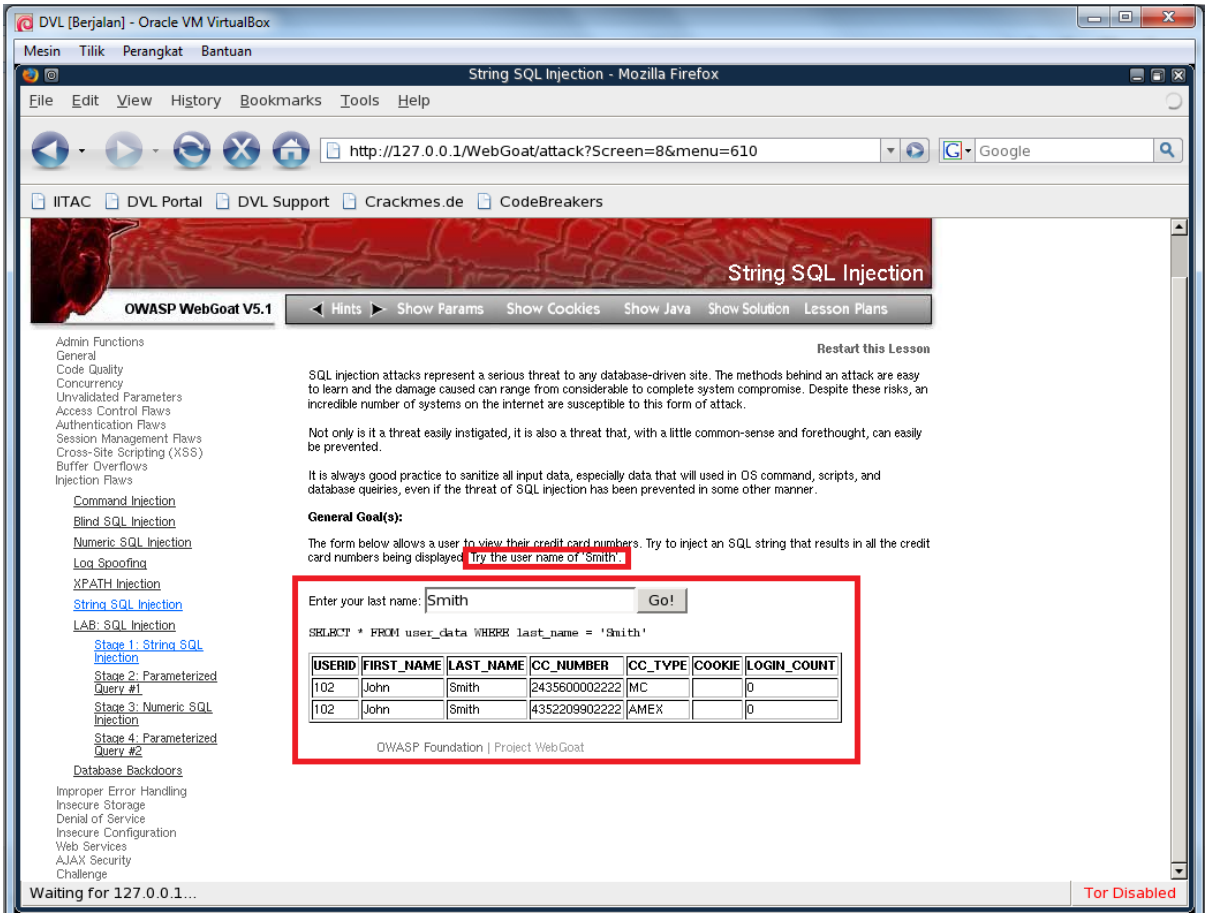


Gambar.4 Halaman Utama WebGoat

Setelah *Start* WebGoat, pilih *Injection Flaws* → *String SQL Injection*. Lakukan percobaan Injeksi dengan Nama User ‘Smith’ (sebagai contoh) seperti pada Gambar.6.



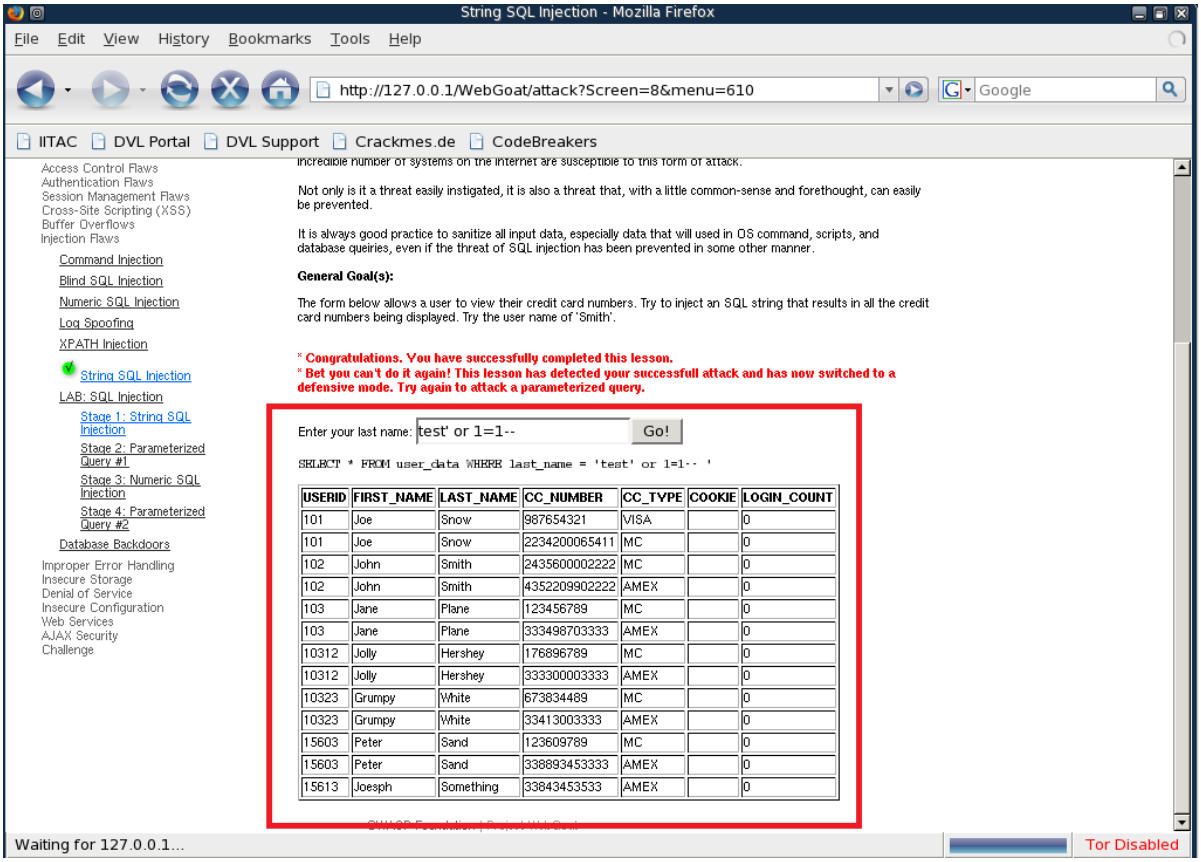
Gambar.5 Pilihan Injection Flaw pada WebGoat



Gambar.6 Percobaan dengan Nama User ‘Smith’

Lakukan percobaan Injeksi yang sebenarnya dengan Nama User ‘Test’ seperti pada Gambar.7 dengan *SQL Injection* berdasarkan 1=1 selalu benar atau 0=0 selalu benar maka hasilnya akan selalu benar. User name yang diinputkan : **`'test' or 1=1 - - (spasi)`**, single code ini ‘....’ adalah kesalahan pada SQL yang bermaksud menghentikan *query* yang diinputkan sedangkan **`or 1=1 - - (spasi)`** adalah *command* yang artinya membenarkan *query* yang diinputkan. Maka hasilnya akan tampak seperti pada Gambar.7 dengan informasi yang berisikan *userID, first_name, last_name,cc_number, cc_type, cookie, login_count*. Kita bisa saja mendapatkan akses kesemua nama pengguna dan *password* dalam *database* dengan hanya memasukkan **`'test' or 1=1 - - (spasi)`** ke kontak input.

Nama : Leny Novita Sari
NIM : 09011181320027
Keamanan Jaringan Komputer



Gambar.7 Percobaan dengan Nama User 'Test'

Bug yang biasanya ditimbulkan :

1. Cross Site Scripting (XSS), para *attackers* memanfaatkan web aplikasi untuk dijadikan sebagai mekanisme yang menghubungkan serangan pada *web browser* penggunanya. Seperti yang kita tahu *web browser* dapat menjalankan *code* yang dikirim dari *website server* seperti *javascript* atau *flash* sehingga memungkinkan untuk di *attack* seperti dengan mengirimkan isinya ke pemakai lain. Kata kuncinya adalah jangan membiarkan lubang untuk *attacker* mencuri isi web yang masih aktif karena saat pengguna web aplikasi dapat memasukkan data dan mengirimkan ke *web browser* tanpa harus melakukan validasi dan *encoding* terhadap isi data tersebut, sehingga mengakibatkan penyerang dapat menjalankan potongan kode *script* miliknya di *browser* target, dan memungkinkan untuk mencuri *user session* milik target, bahkan sampai menciptakan *Worm*.

2. Injection Flows, aplikasi web dapat melewatkan informasi dari pengguna ke sistem lain ataupun ke sistem operasi lokal. Jika penyusup dapat melewatkan perintah yang tidak diantisipasi oleh aplikasi web maka dia mungkin dapat masuk dalam sistem operasi lokal. Secara umum penggunaan parameter atau perintah dalam bentuk SQL jangan dilakukan dalam parameter konstan sehingga dapat diketahui oleh *attackers*. Karena celah injeksi umumnya injeksi terhadap SQL (*database*) dari suatu aplikasi web. Hal ini mungkin terjadi apabila pengguna memasukkan data sebagai bagian dari perintah (*query*) yang menipu interpreter untuk menjalankan perintah tersebut atau merubah suatu data.
3. Malicious File Execution, celah ini mengakibatkan penyerang dapat secara *remote* membuat file yang berisi kode dan data untuk di eksekusi, salah satunya adalah *Remote File Inclusion* (RFI).
4. Insecure Direct Object Reference, adalah suatu celah yang terjadi saat pembuat aplikasi web merekspos referensi internal penggunaan objek, seperti file, direktori, *database record*, dll
5. Cross Site Request Forgery (CSRF), celah ini akan memaksa *browser* target yang sudah *log-in* untuk mengirimkan "*pre-authenticated request*" terhadap aplikasi web yang diketahui memiliki celah, dan memaksa *browser* target untuk melakukan hal yang menguntungkan penyerang.
6. Information Leakage and Improper Error Handling, penyerang menggunakan informasi yang didapatkan dari celah yang diakibatkan oleh informasi yang diberikan oleh web aplikasi seperti pesan kesalahan (*error*) serta konfigurasi yang bisa di lihat. Kondisi *error* yang tidak ditangani secara benar dapat digunakan untuk memperoleh informasi dari sistem atau bahkan dapat membuat suatu sistem "*crash*". Jadi penanganan masalah harus dilakukan dengan benar sehingga tidak meimbulkan efek lainnya yang membahayakan seperti *Out of Memory, too many users, timeout, db failure, authentication failure, access control failure, bad input*. Kuncinya adalah perancangan skema penanganan *error* yang benar dan juga konfigurasi *server* yang tepat.
7. Broken Authentication and Session Management, *account* dan *token session* tidak terproteksi dengan baik. Celah ini merupakan akibat buruknya penanganan proses autentikasi dan manajemen sesi, sehingga penyerang bisa mendapatkan *password*, atau *key* yang di gunakan untuk autentikasi.

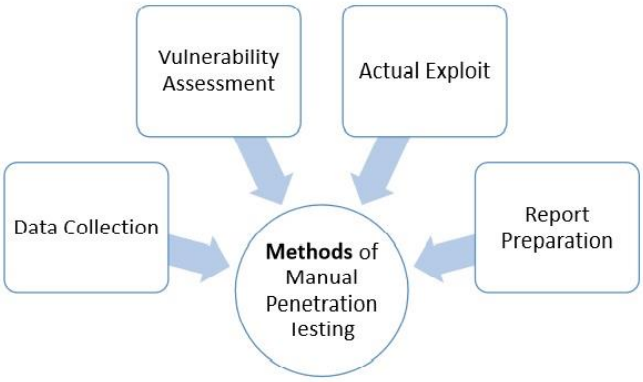
8. Insecure Cryptographic Storage, aplikasi web umumnya jarang menggunakan fungsi kriptografi untuk melindungi data penting yang dimiliki karena memiliki kelemahan. Pada umumnya fungsi tersebut dan juga bagian kode untuk mengintegrasikannya sulit untuk dituliskan secara benar, dan ini menghasilkan proteksi yang lemah. Sebaiknya menggunakan algoritma yang mudah digunakan dan terintegrasi dengan baik.
9. Insecure Communications, sedikit sekali aplikasi web yang mengamankan jalur komunikasinya, hal inilah yang dimanfaatkan oleh penyerang sebagai celah untuk mendapatkan informasi berharga.
10. Failure to Restrict URL Access, seringkali aplikasi web hanya menghilangkan tampilan *link* (URL) dari pengguna yang tidak berhak, tetapi hal ini dengan sangat mudah dilewati dengan mengakses URL tersebut secara langsung.

- Salah konfigurasi

  Meskipun program sudah diimplementasikan dengan baik, masih dapat terjadi lubang keamanan karena salah konfigurasi. Contoh : berkas yang semestinya tidak dapat diubah oleh pemakai secara tidak sengaja menjadi "*writeable*" atau adanya program yang secara tidak sengaja diset memiliki akses seperti *super user (root*) yang dapat melakukan apa saja.

- Report Preparation

Setelah penetrasi dilakukan, tester menyiapkan laporan akhir yang menjelakan segala sesuatu tentang sitem dan dianalisis untuk mengambil langkah-langkah korektif untuk melindungi sistem target.



Gambar.8 Metode pada Penetration Testing Manual