E-Journal Teknik Informatika Vol 12, No.1 (2017) ISSN: 2301-8364

KAJIAN IMPLEMENTASI *CROSS SITE REQUEST FORGERY* (*CSRF*) PADA CELAH KEAMANAN *WEBSITE*

Raydittо Makalalag[1)], Xaverius B. N. Najoan[2)], Agustinus Jacobus[3)]
[1,2,3] Program Studi Teknik Informatika, Fakultas Teknik, Universitas Sam Ratulangi
E-mail : 120216028@student.unsrat.ac.id[1)], xnajoan@unsrat.ac.id[2)], a.jacobus@unsrat.ac.id[3)]

CSRF (Cross Site Request Forgery) merupakan salah satu teknik penetrasi pada celah keamanan website. Teknik ini menggunakan metode untuk memasulkan data pengguna suatu situs. Hal ini tentunya sangat merugikan pihak pengguna aplikasi website karena data pribadi yang ada dalam suatu sistem aplikasi data diakses oleh pihak yang tidak bertanggung jawab. Untuk itu perlu diadakan suatu metode pencegahan untuk menangani serangan CSRF. Mekanisme penerapan token pada suatu form pengiriman data adalah dimulai dari inisialisasi nilai pada token, validasi nilai token, hingga otorisasi data pengguna. Pada penelitian kali ini token diklasifikan menjadi dua jenis; token yang bersifat statis (secret validation token), dan token yang bersifat dinamis (random validation token). Untuk meminimalisir serangan CSRF maka direkomendasikan untuk menggunakan random validation token karena dibutuhkan nilai token yang berbeda-beda pada setiap form pengiriman data.

Kata Kunci: Keamanan Aplikasi Web, CSRF, GET, POST, Token

I. PENDAHULUAN

Informasi via internet yang dipertukarkan melalui *website* bervariasi, baik itu dari segi jenis maupun tingkat kerahasiaannya. Tidak jarang data-data penting seperti data pribadi, data organisasi, sampai dengan data negara yang sangat dijaga kerahasiaannya dikirimkan via internet. Dengan meluasnya kemungkinan untuk mengakses internet, resiko ancaman terhadap informasi rahasia yang ada pada *website-website* tertentu juga mengalami peningkatan. Statistik terakhir pada *internetlivestats.com* memperlihatkan bahwa penetrasi internet pada tahun 2016 telah mencapai 46,1%. Itu artinya dari 3,4 miliar pengguna internet di dunia, ada sekitar 1,5 miliar kali percobaan penetrasi di internet, naik 2,7% dari tahun sebelumnya.

Resiko ancaman akan benar-benar menjadi masalah jika berhasil diekspolitasi dalam sebuah serangan. Adapun kategori serangan yang bertujuan untuk merusak data atau informasi yang tersimpan di dalam media perangkat teknologi informasi. Serangan yang dimaksud antara lain :

- Merubah isi sebuah situs *web* tanpa sepengetahuan pemiliknya;
- Mengambil kumpulan *password* atau informasi lengkap kartu kredit sekelompok individu untuk disalahgunakan atau diperjualbelikan;
- Merusak sistem basis data utama sehingga semua informasi didalamnya menjadi tidak dapat terbaca atau diakses secara normal; dan lain sebagainya.

CSRF (*Cross Site Request Forgery*) merupakan salah satu teknik penetrasi pada celah keamanan *website*. *CSRF* merupakan teknik pemalsuan identitas pengguna suatu situs. *CSRF* juga merupakan celah keaman yang cukup berbahaya. Contoh akibat dari teknik ini adalah mampu melakukan perubahan detail akun pada korban. Data pribadi seperti nama, alamat, bahkan sampai *password* korban bisa diubah dengan menggunakan teknik ini. Seperti yang terjadi pada kasus *paypal.me*. Pada tahun 2016, Florian Courtial seorang programmer asal Perancis menemukan *bug* pada situs *paypal.me*. Florian Courtial menemukan celah keamanan pada situs *paypal.me* yang memungkinkan untuk diterapkannya *CSRF*. Akibat dari celah tersebut ia dapat mengganti foto profil tanpa seizin pengguna yang bersangkutan. Situs pembayaran online tersebut memberikan hadiah $750 kepada Florian Courtial karena berhasil memperbaiki celah tersebut.

Berdasarkan uraian-uraian diatas, maka dilakukan penelitian mengenai *CSRF*, sehingga dapat diperoleh rekomendasi solusi untuk penanganan *CSRF*.

II. LANDASAN TEORI

A. World Wide Web (*WWW*)

Salah satu unsur yang paling umum digunakan dari Internet selain *e-mail* adalah *World Wide Web* (*WWW*). *WWW* atau juga sering disebut sebagai *web* merupakan salah satu aplikasi internet yang paling populer. *Web* adalah sebuah sistem dengan informasi yang disajikan dalam bentuk teks, gambar, suara, dan lain-lain yang tersimpan dalam sebuah *web server*.

Web dapat diakses oleh perangkat lunak pengguna *web* yang disebut *browser*. *Browser* membaca halaman-halaman *web* yang tersimpan dalam *server* melalui protokol yang disebut *HTTP* (*HyperText Transfer Protocol*). Ada dua komponen dasar di dalam arsitektur *web*, yaitu *web browser* dan *web server*. *Web browser* menawarkan antarmuka grafis untuk pengguna, dan bertanggung jawab untuk komunikasi dengan *web server*.

Pada umumnya antarmuka antara pengguna dan *web browser* menggunakan bahasa pemrograman *HTML* (*Hyper Text Markup Language*). Sedangkan komunikasi antara *browser* dan *server* menggunakan protokol *HTTP*. Dengan kata lain *web browser* dapat dikatakan sebagai klien dan *web server* dapat dikatakan sebagai *server*.

B. HyperText Transfer Protocol (HTTP)

HyperText Transfer Protocol atau *HTTP* adalah sebuah protokol yang memungkinkan *web browser* untuk berkomunikasi dengan *web server* dalam pertukaran informasi. *HTTP* menyediakan sebuah cara standar untuk berkomunikasi antara *browser* dan *server*, sehingga *browser* apapun dapat berkomunikasi dengan *server* manapun asalkan keduanya sesuai dengan spesifikasi *HTTP*. Saat *HTTP* diakses, klien memulainya dengan sebuah *request* dan direspon oleh *server*. Setiap *request* dan *response* memiliki 3 bagian yaitu *status line*, *header fields*/*section*, dan *entity body*.

- *Status Line*. Pada baris ini metode *request* yang berisi deskripsi pesan. Biasanya berupa lokasi dokumen yang ada pada *server*. Sedangkan pada metode *response* mendeskripsikan apa yang telah terjadi setelah *server* melakukan pesan *request*.
- *Header Fields*/*Section*. Serangkaian baris ini berisi *header HTTP* yang digunakan untuk menyampaikan informasi mengenai *request* dan klien kepada *server*.
- *Entity Body*. Pada bagian ini berisi data lainnya untuk diteruskan ke *server*. Biasanya informasi disini hanya ada ketika sebuah *form* dikirimkan.

Dalam penggunaannya *HTTP* memiliki beberapa metode yang memiliki fungsinya masing-masing, diantaranya :

Tabel 1. Metode pertukaran data pada HTTP

Method	Action
GET	Mengambil data dari *server* dengan menampilkan *URL*
POST	Merubah atau menambah data dari *server* tanpa menampilkan *URL*

GET dan POST sama-sama merupakan metode pertukaran data yang digunakan oleh protokol *HTTP*. Perbedaanya terletak pada variabel tempat penyimpanan data. Jika pada GET menggunakan variabel global $_GET dan pada POST menggunakan variabel $_POST.

Jika data yang dikirimkan bersifat sensitif dan rahasia, sebaiknya menggunakan metode POST, karena jika menggunakan metode GET data yang dikirimkan akan ditampilkan pada URL (alamat *website*).

C. Client & Server Side Scripting

Client Side Scripting dan *Server Side Scripting* merupakan istilah dalam pemrograman *web* yang digunakan untuk mengelompokan beberapa bahasa pemrograman *web* berdasarkan pihak mana atau siapa yang akan melakukan pengolahan data di *web*.

Client side scripting umumnya lebih mengacu pada suatu program *web* yang operasinya dijalankan di sisi pengguna melalui sebuah *web browser*. Jadi ketika pengguna tadi meminta informasi melalui sebuah *web server*, maka *server* akan menyediakan data berupa skrip yang kemudian diunduh oleh *web browser* dan kemudian komputer klien tersebut akan memprosesnya hingga informasi data yang diinginkan dapat ditampilkan pada *web browser*. Contoh bahasa *client side scripting* diantaranya adalah *HTML*, *CSS*, *JavaScript*, dan *XML*. *Client side scripting* biasanya digunakan untuk membantu sebuah *website* statis menjadi lebih dinamis. Berbeda dengan *server side scripting*, pada *client side scripting*, kode sumber dari program yang disediakan oleh *server* dapat dilihat oleh klien.

Server side scripting adalah sebuah teknik dalam perancangan desain *web* yang melibatkan *embedding script* dalam dokumen *HTML* yang diminta oleh klien dari sebuah *server*. *Server side scripting* biasanya digunakan untuk menyediakan antarmuka ke klien dan membatasi klien untuk mengakses *database*, atau sumber informasi yang sifatnya rahasia. Skrip ini dapat menyesuaikan respon berdasarkan karakteristik, kebutuhan pengguna, hak akses, dan lainnya. Selain itu, *server side scripting* juga memungkinkan pengelola *server* untuk membatasi akses ke kode sumber dari skrip yang dijalankan. Karena proses pengolahan informasi data berlangsung di sisi *server*, maka metode *server side scripting* banyak digunakan dalam pemrograman *web* yang memungkinkan *server* dapat menghasilkan halaman *web* dinamis. Contoh bahasa *scripting server-side* yang banyak digunakan dalam pemrograman *web* diantaranya adalah *ASP*, *PHP*, *Phyton*, *Perl*, dan *Java Server Pages*.

D. Cross Site Request Forgery (CSRF)

Cross Site Request Forgery (*CSRF* atau *XSRF*) adalah salah satu celah keamanan yang memanfaatkan program atau *task* yang digunakan oleh sebuah *website*[1]. Celah tersebut terjadi karena kurangnya pengamanan pada suatu file yang memproses *form* tertentu, sehingga kita dapat melakukan *request* ke *file action* tersebut, dengan *form* yang telah kita modifikasi sebelumnya.

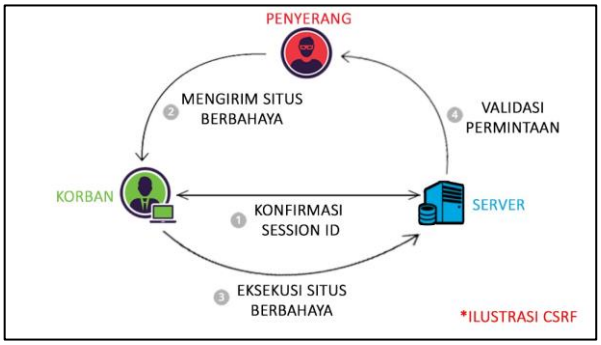
Daripada mengeksploitasi pengguna *web* (*user*) agar percaya terhadap *website* yang sedang diakses, *CSRF* mengeskpolitasi *website* agar *user* tetap percaya bahwa *website* yang digunakan memang benar *website* yang ingin diakses. *CSRF* juga dikenal dengan sebutan "*one link attack*", karena pada implementasinya sang penyerang hanya perlu menginjeksi sebuah link yang berisi suatu *web task URL* pada halaman tertentu untuk dibuka oleh calon korban, agar ketika korban membuka halaman tersebut, secara otomatis si korban akan mengeksekusi *link URL* yang telah diinjeksi sebelumnya.

Dalam dokumen OWASP Testing Guide v4 pada sub-bab *Testing for CSRF* (*OTG-SESS-005*), dikatakan bahwa *CSRF* adalah sebuah serangan yang memaksa pengguna untuk melakukan tindakan yang tidak diinginkan pada aplikasi web saat pengguna tersebut sudah terotentikasi. Dengan sedikit bantuan rekayasa sosial (seperti mengirim tautan melalui email atau chat), penyerang dapat memaksa pengguna aplikasi web untuk tindakan penyerang. Keberhasilan *CSRF* ketika mengeksploitasi pengguna biasa dapat membahayakan data dan pengoprasian pengguna. Jika yang ditargetkan untuk dieksploitasi adalah pengguna akun administrator, serangan *CSRF* dapat membahayakan keseluruhan aplikasi *web*.

Terdapat beberapa hal yang harus terjadi supaya *Cross-Site-Request-Forgery* berhasil, diantaranya : Penyerang harus menemukan formulir disitus target sesuatu yang tidak berguna untuk dia (misalnya, transfer uang, mengubah alamat *e-mail*, atau *password* korban); Penyerang harus menentukan nilai-nilai yang tepat untuk segala bentuk masukan[2]. Jika salah satunya diminta untuk menjadi otentikasi nilai rahasia atau ID yang penyerang

tidak dapat ditebak penyerang, serangan akan gagal; Para penyerang harus membujuk korban ke halaman Web dengan kode berbahaya saat korban masuk ke situs sasaran.

CSRF memiliki dua tipe serangan, diantaranya : *Stored CSRF, Reflected CSRF*[3]. *Stored CSRF* merupakan serangan dimana penyerang dapat menggunakan aplikasi itu sendiri untuk memberi korban tautan eksploitasi atau konten lainnya yang mengarahkan *browser* korban untuk melakukan tindakan yang dikendalikan oleh penyerang. *Reflected CSRF*, merupakan serangan yang memanfaatkan link atau konten diluar sistem aplikasi. Hal ini bisa dilakukan dengan menggunakan *email*, *blog*, pesan instan yang terdapat didalam aplikasi tersebut.



Gambar 1. Gambaran umum *CSRF*

Gambar 1 merupakan gambaran umum mengenai implementasi teknik *CSRF*.

- Tahap pertama : *client* dan *server* akan mengkonfirmasi *session id* dari suatu halaman *web* tertentu.
- Tahap kedua : penyerang mengirimkan pesan yang dimana pesan tersebut seolah-olah adalah pesan konfirmasi dari halaman web contoh.com. Pesan yang dikirimkan berupa *link* yang di dalamnya sudah diinjeksi perintah untuk mengubah *password* dari suatu akun pada *website* contoh.com
- Setelah pesan tersebut terkirim ke korban, selanjutnya korban melakukan klik terhadap *link* karena mengira pesan tersebut memang benar dari *website* contoh.com.
- *Password* dari akun korban pun secara otomatis berubah pada halaman *web* contoh.com. Ketika korban keluar (*logout*), maka korban sudah tidak bisa lagi *login* dengan *password* yang sebelumnya karena *password* tersebut sudah diubah oleh penyerang.

E. Social Engineering

Definisi *social engineering* adalah suatu teknik 'pencurian' atau pengambilan data atau informasi penting/krusial/rahasia dari seorang dengan cara menggunakan pendekatan manusiawi melalui mekanisme interaksi social[4]. Dengan kata lain *social engineering* adalah suatu teknik memperoleh data/informasi rahasia dengan cara mengeksploitasi keamanan manusia.

Pada dasarnya teknik *social engineering* ini dapat dibagi menjadi dua jenis, yaitu : berbasis interaksi sosial dan berbasis interaksi komputer. Salah satu contoh teknik *social engineering* menggunakan medium komputer antara lain adalah teknik *phising*.

Phising merupakan sebuah proses pra-serangan atau kerap dikatakan dengan *soft attack*, dimana sang penyerang berusaha mendapatkan informasi rahasia dari target dengan cara menyamar menjadi pihak yang dapat dipercaya – atau seolah-olah merupakan pihak yang sesungguhnya.

III. METODOLOGI PENELITIAN

Gambar 2. Diagram Alur Kerangka Berpikir Penelitian

A. Studi Literatur

Tahap awal penelitian dimana pada tahap ini penulis mencari referensi teori yang relefan dengan area penelitian.

B. Implementasi *CSRF*

Tahapan ini merupakan tahap untuk mengeksplotasi celah keamanan *web* yang ditargetkan. Dalam implementasi kali ini, penulis akan mensimulasikan teknik *CSRF* pada beberapa *web* yang sudah memiliki celah untuk dieksploitasi

C. Pengkajian Hasil Implementasi *CSRF*

Setelah mengeksploitasi *CSRF* pada beberapa web yang memiliki celah, selanjutnya penulis akan melakukan pengkajian terhadap hasil pengimplementasian tersebut, dengan tujuan untuk memberitahukan kenapa *CSRF* bisa tereksplotasi.

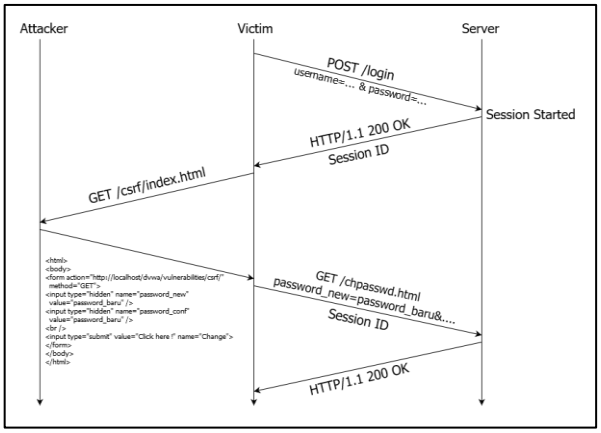
D. Rekomendasi Solusi

Tahap terakhir dari penelitian adalah berupa rekomendasi solusi dari hasil kajian pengimplementasian *CSRF*. Hal ini bertujuan sebagai bahan referensi bagi pengembang *web* agar dapat mengantisipasi *CSRF*.

IV. HASIL DAN PEMBAHASAN

A. Implementasi *CSRF* pada *Web Test Case*

1) Implementasi *CSRF* pada *DVWA*



Gambar 3. Mekanisme penerapan CSRF pada *DVWA*

Sesuai dengan diagram pada gambar 3, bisa terlihat mekanisme penerapan/implementasi *CSRF* pada *DVWA* yang dimulai dari korban *login* sampai dengan *password* korban berganti tanpa sepengetahuan korban.

Gambar 4. *DVWA login failed*

Hasil dari penerapan *CSRF* pada *DVWA* adalah *User* "admin" sudah tidak bisa lagi melakukan *login* dengan *password* "password" dikarenakan password tersebut telah berhasil diubah.

2) Implementasi *CSRF* pada *bWAPP*

Pada dasarnya mekanisme penerapan *CSRF* pada *bWAPP* sama seperti mekanisme penerapan *CSRF* pada *DVWA*, yang membedakan hanyalah skenario untuk penerapan *CSRF*.

Skenario dari simulasi kali ini adalah korban akan mengklik gambar yang sudah dikirimkan sebelumnya. Gambar tersebut berisi *link* yang berisikan skrip agar ketika *link* tersebut diklik, maka sejumlah uang pada akun korban akan dikirimkan pada akun milik penyerang.

Gambar 5. Tampilan laman pengimplementasian *CSRF*

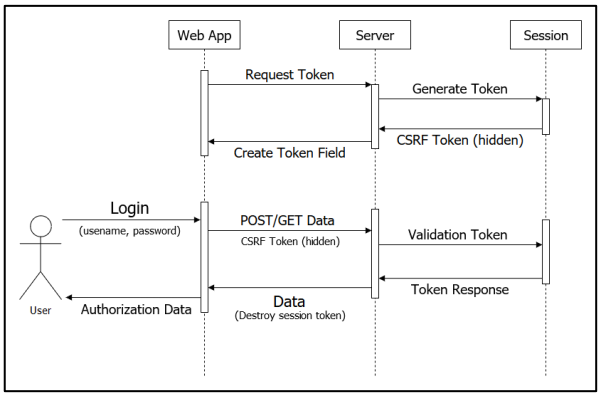
Ketika tombol "*click here*" diklik, maka sejumlah uang dengan nominal yang sudah ditentukan akan ditransfer pada akun penyerang.

B. Hasil Analisa Pengkajian *Cross Site Request Forgery*

Berdasarkan pengimplementasian *CSRF* pada *web-test-case* ditemukan beberapa hasil analisis. Hasil analisis dari pengkajian tersebut antara lain :

1. Dengan tidak diterapkannya token pada suatu form pengiriman data, maka pengeksplotasian *CSRF* pada form tersebut akan mudah untuk diimplementasikan.
2. Penerapan token yang bersifat dinamis akan membuat suatu aplikasi website akan lebih mudah untuk mengantisipasi serangan *CSRF*. Hal ini dikarenakan pada pengimplementasiannya, nilai pada token akan berubah-ubah jika halaman tersebut dimuat kembali.
3. Halaman web yang menggunakan metode pengiriman GET untuk data yang bersifat pribadi akan lebih mudah untuk dieksploitasi karena data-data tersebut akan tampilkan pada *URL* ketika data tersebut dikirim.
4. *Session* sangat penting dalam mengotentikasi *user* ketika berinteraksi dengan sistem.

Token merupakan salah satu mekanisme terbaik untuk mengantisipasi dari serangan CSRF[5]. Bagi penyerang akan sulit untuk memprediksi pola acak dari token karena nilai pada form yang dikirim berubah-ubah setiap saat.



Gambar 6. Mekanisme token pada aplikasi *web*

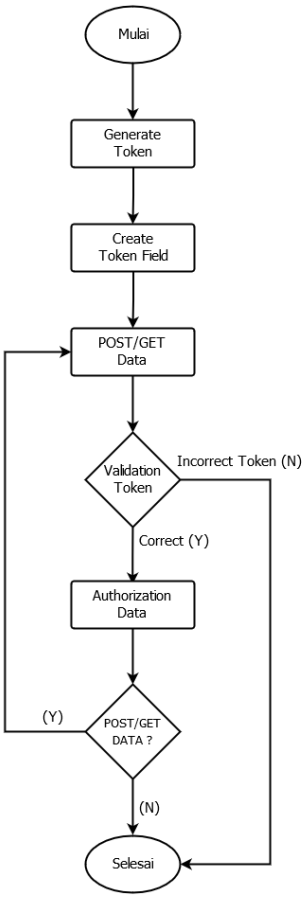
C. Teknik Pencegahan *Cross Site Request Forgery*

1) *Secret Validation Token*

Secret Validation Token atau validasi token rahasia merupakan teknik penanganan untuk serangan *CSRF* dengan cara mengirimkan informasi tambahan pada setiap pengiriman data ke server. Informasi tersebut berisi nilai acak yang sudah diatur sebelumnya berdasarkan suatu mekanisme yang dibuat sendiri oleh aplikasi web. Hal ini bertujuan untuk menentukan apakah permintaan tersebut

memang berasal dari pengguna yang mempunyai otorisasi atau tidak.

Validasi token rahasia ini sulit ditebak bagi orang yang tidak memiliki akses ke akun pengguna. Jadi, jika tidak terdapat nilai tersebut dalam sebuah pengiriman data atau nilai token tidak sesuai, maka server akan menolak permintaan tersebut. Untuk melihat mekanisme *secret validation token* dapat dilihat pada gambar 10.

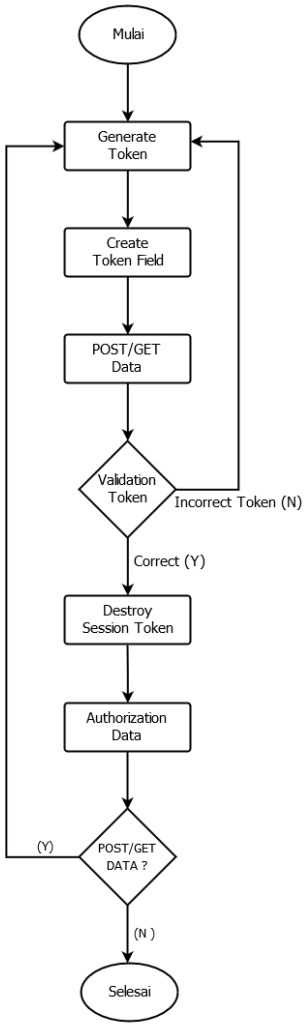


Gambar 7. Mekanisme *Secret Validation Token*

Jika nilai token sesuai pada saat validasi, maka pengiriman data berhasil dilakukan. Selanjutnya jika form pengiriman data kembali diakses, maka nilai pada token masih tetap sama seperti nilai token pada saat validasi sebelumnya. Hal ini juga berlaku pada saat validasi dan nilai token tidak cocok atau sesuai. Pada saat pengiriman data selanjutnya, nilai token pada form pengiriman data masih tetap sama seperti pada form pengiriman sebelumnya.

2) *Random Validation Token*

Secara umum teknik pencegahan *Random Validation Token* sama seperti *Secret Validation Token*. Namun pada teknik pencegahan ini, semua pengiriman form data memiliki nilai token yang berbeda untuk setiap permintaan. Untuk melihat mekanisme *random validation token* dapat dilihat pada gambar 11.



Gambar 8. Mekanisme *Random Validation Token*

Yang membedakan dari kedua metode ini adalah pada saat proses pengiriman data. Ketika proses pengiriman data sudah selesai (baik itu berhasil atau tidak), nilai token yang ada pada form pengiriman selanjutnya sudah berubah. Hal ini dikarenakan pada saat proses pengiriman selesai, nilai yang tersimpan pada *session* akan dihapus dan nilai pada token tersebut akan dimuat kembali. Dengan kata lain *random validation token* bersifat lebih dinamis dibandingkan *secret validation token*.

Sebagai perbandingan alternatif metode penanganan *CSRF*, dalam penelitian kali ini telah dirangkum beberapa metode penanganan *CSRF* yang bersumber dari beberapa penelitian yang relefan dengan penelitian kali ini.

Secara umum terdapat empat metode untuk menangangi *CSRF*. Keempat metode tersebut antara lain : Validasi Token, HTTP Referer, Custom HTTP Header, dan Origin Header[6]. Untuk mengetahui hasil analisa dari rangkuman mengenai metode penanganan *CSRF* dapat dilihat pada tabel 2.

Tabel 2. Rangkuman metode penanganan *CSRF*

	Metode Pengiriman Data		Rekomendasi Metode Penangan *CSRF*			
	GET	POST	Validasi Token	HTTP Referer	Custom HTTP Header	Origin Header
Adam Barth, dkk (2008)	discussed	discussed	recommended	recommended	recommended	recommended
Andrew Pakpahan (2009)	not discussed	discussed	recommended	recommended	not discussed	not discussed
Jesse Burns (2007)	not discussed	discussed	recommended	recommended	not discussed	not discussed
Jeremiah Grossman (2009)	not discussed	discussed	recommended	recommended	not discussed	not discussed
Nenad Jovanovic, dkk (2006)	discussed	discussed	recommended	recommended	not discussed	not discussed
Nikunj J. Tandel, dkk (2014)	not discussed	discussed	recommended	recommended	recommended	recommended
Purnima Khurana, dkk (2014)	discussed	discussed	recommended	not discussed	not discussed	not discussed
RadhaRani Sankuru, dkk (2013)	discussed	discussed	recommended	not discussed	not discussed	not discussed
Rupali D. Kombade, dkk (2012)	discussed	discussed	recommended	recommended	recommended	not discussed
Sentamilselvan. K, dkk (2014)	not discussed	discussed	recommended	recommended	recommended	recommended
William Zeller, dkk (2008)	not discussed	discussed	recommended	recommended	recommended	recommended

Dari rangkuman pada tabel 2 dapat terlihat bahwa keseluruhan penelitian merekomendasikan untuk menggunakan token sebagai metode untuk menangani serangan CSRF. Beberapa peniliti sudah tidak lagi merekomendasikan HTTP Referer, Custom HTTP header, dan Origin Header, karena pada penjelasannya, saat ini sudah banyak browser yang menonaktifkan fitur-fitur tersebut. Atas dasar itu, ketiga metode tersebut sudah tidak lagi disebutkan pada beberapa penelitian.

V. PENUTUP

A. Kesimpulan

CSRF merupakan teknik pemalsuan identitas pengguna suatu situs.. Untuk itu perlu diadakan suatu metode pencegahan untuk menangani serangan *CSRF*. Berdasarkan hasil penelitian, token merupakan salah satu metode pencegahan yang efektif untuk mengantisipasi serangan *CSRF*. Tujuan dari token sendiri adalah untuk memvalidasi nilai yang dikirimkan ke server. Jika nilai tersebut cocok atau sesuai, maka permintaan tersebut akan dilanjutkan. Namun jika nilai pada token tidak sesuai, maka permintaan tersebut akan ditolak.

Penerapan token yang bersifat dinamis dapat membantu meminimalisir pengeksploitasian menggunakan teknik *CSRF*. Nilai token yang bersifat dinamis akan berubah-ubah nilainya setiap form pengiriman tersebut dikirim. *Random Validation Token* merupakan metode yang menerapkan token yang bersifat dinamis. Maka dari itu direkomendasikan untuk menggunakan mekanisme *Random Validation Token* sebagai solusi untuk mengantisipasi serangan *CSRF*.

B. Saran

Bagi peminat yang ingin melanjutkan penelitian pada bidang penetrasi celah keamanan *website* menggunakan teknik *CSRF*, disarankan untuk meneliti lebih lanjut metode pencegahan lain untuk serangan *CSRF*, agar terciptanya lebih banyak solusi untuk penanganan serangan *CSRF*.

DAFTAR PUSTAKA

[1] Zaki, Ali dan tim. 2009. *"Teknik Hacking dan Overlay Friendster"*. PT Elex Media Komputindo. Jakarta. hal 46.

[2] Pakpahan, A. 2009. *"Kajian Keamanan Aplikasi Web Berbasis AIAX Terhadap Serangan Cross Site Request Forgery (CSRF)"*. Jurnal TeIka. Volume 2, Nomor 2.

[3] Tandel, N, J dkk. 2014. "*Defensive Mechanisms of CSRF Attack*". International Journal of Engineering Development and Research.

[4] Indrajit, R. E. 2014. *"Konsep dan Strategi Keamanan Informasi di Dunia Cyber Cyber"*. Cetakan ke-1. Graha Ilmu. Yogyakarta.

[5] K, Sentamilselvan dkk. 2014. "*Cross Site Request Forgery: Preventive Measures*". International Journal of Computer Applications (0975 – 8887).

[6] Barth, A, dkk. 2008. "*Robust Defenses for Cross-Site Request Forgery*". Proceeding - CCS '08 Proceedings of the 15th ACM conference on Computer and communications security. Pages 75-88.

[7] Burns, J. 2007. "*Cross Site Request Forgery – An Introduction to a Common Web Application Weakness*". Information Security Partners, LLC.

[8] Grossman, J. 2009. *"Cross-Site Request Forgery: The Sleeping Giant"*. A WhiteHat Security Whitepaper.

[9] Jovanovic, N dkk. 2006. "*Preventing Cross Site Request Forgery Attacks*". Securecomm and Workshops.

[10] Khurana, P dkk. 2014. "*Vulnerabilities and Defensive Mechanism of CSRF*". International Journal of Computer Trends and Technology (IJCTT) . Volume 13, Number 4.

[11] Kombade, R, D dkk. 2012, "*CSRF Vulnerabilities and Defensive Techniques*". International Journal of Computer Network and Information Security (IJCNIS).

[12] Sankuru, R, dkk. 2013. "*Web Application Security – Cross-Site Request Forgery*". International Journal of Computer Science & Engineering Technology (IJCSET).

[13] Zeller, W dkk. 2008. "*Cross-Site Request Forgeries: Exploitation and Prevention*". Technical Report, Princeton university.

[14] Auger, R. 2010. "*The Cross-Site Request Forgery (CSRF/XSRF) FAQ"*. Diakses : http://www.cgisecurity.com/csrf-faq.html. (21 Januari 2017).

[15] Dewhurst, R. 2015. "*Damn Vulnerable Web Application (DVWA)*". Diakses : https://dewhurstsecurity.com/#projects. (24 Mei 2017).

[16] Mesellem, M. 2014. "*bWAPP, a buggy web application!*". Diakses : http://itsecgames.blogspot.co.id/. (15 Mei 2017).

TENTANG PENULIS

Raydditto Makalalag, lahir di Manado pada tanggal 23 Juni 1995. Penulis menempuh Pendidikan secara berturut-turut di Taman Kanak-kanak Al-Mutazam Manado (1999 – 2000), Sekolah Dasar Negeri 20 Manado (2000 – 2006), Sekolah Menengah Pertama Negeri 8 Manado (2006 – 2009), Sekolah Menengah Atas Negeri 9 Manado (2009 – 2012).

Pada tahun 2012, penulis melanjutkan studi di Program Studi Informatika, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Sam Ratulangi. Selama masa kuliah, penulis telah menjalani kerja praktek di ELFAH Hotel Manado, serta mengikuti kegiatan Kuliah Kerja Terpadu di Desa Watuadambo II, Kecamatan Kauditan, Kabupaten Minahasa Utara. Selama kuliah penulis pernah tergabung dalam organisasi kemahasiswaan yaitu, Himpunan Mahasiswa Elektro Fakultas Teknik Universitas Sam Ratulangi, Badan Tadzkir Fakultas Teknik Universitas Sam Ratulangi, Unsrat IT Community (UNITY) dan Senat Mahasiswa Fakultas Teknik Universitas Sam Ratulangi.

Penulis menyelesaikan studi di Program Studi Informatika, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Sam Ratulangi pada 18 September 2017.