Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer Vol. 14, No. 2, September 2019 58
e-ISSN 2597-4963 dan p-ISSN 1858-4853 DOI: http://dx.doi.org/10.30872/jim.v14i2.1798

Evaluasi Teknik Penyadapan Lalu Lintas Data Dengan Session Hijacking Pada Protokol HTTP

Rizal Munadi [1)], Mirza Purnandi [2)], Teuku Yuliar Arif [3)]
Magister Teknik Elektro, Jurusan Teknik Elektro dan Komputer, Fakultas Teknik, Universitas Syiah Kuala
Jalan Tengku Syech Abdurrauf No. 7, Darussalam, Banda Aceh 23111 Indonesia
E-Mail: rizal.munadi@unsyiah.ac.id [1)]; mirza.purnandi@gmail.com [2)]; yuliar@unsyiah.ac.id [3)];

ABSTRAK

Hyper Text Transfer Protocol (HTTP) merupakan protokol jaringan aplikasi yang digunakan pada aplikasi berbasis *web*. Kerentanan protokol ini yang mengekspoitasi web server secara tidak aman, memberikan peluang peretas untuk melakukan serangan, salah satunya dengan serangan *session hijacking*. Oleh karena itu, dalam penelitian ini dilakukan survei kerentanan terhadap session hijacking dengan menguji beberapa web. Untuk mendapatkan data, maka digunakan *Wi-fi UC browser* untuk membaca lalulintas data protokol jaringan HTTP yang sedang dibuka oleh pengguna. Dalam pengujian diperoleh *cookies* yang digunakan oleh pengguna. Informasi *session id* yang terekam dalam penyadapan, dapat disalahgunakan oleh *attacker* untuk melakukan tindakan illegal. Hasil penelitian menunjukkan kelemahan pada protokol jaringan HTTP yang rentan terhadap penyadapan lalulintas data dengan cara penyerangan *session hijacking*. Untuk mengatasi kelemahan ini, dapat digunakan metode pengacakan seperti algoritme *Fisher-Yates shuffle* atau pun algoritme *Linear Congruent* Method yang dapat meningkatkan keamanan.

Kata Kunci – *session hijacking, fisher-yates shuffle, linear congruent method,* HTTP

1. PENDAHULUAN

Era teknologi internet memainkan peran penting terjadinya penyebaran informasi secara cepat, efisien dan efektif. Media penyampaian informasi berbasiskan *website* merupakan salah satu alternatif yang paling umum dipilih oleh semua organisasi. Informasi yang disajikan dapat berupa informasi umum yang tersedia secara bebas dan tidak memerlukan hak akses khusus. Sebagian informasi dikelola secara ketat dan menyediakan akses dengan tingkat keamanan. Salah satunya adalah dengan mengimplementasikan sistem verifikasi dengan *login page.*

Namun integritas sistem informasi dan aplikasi yang menggunakan teknologi layanan *web*, tidak sepenuhnya bebas dari gangguan. Namun serangan terhadap aplikasi berbasis *web* terus meningkat dan menjadikannya target serangan oleh pihak yang tidak berkepentingan. Keamanan aplikasi berbasis *web* sangat rentan terhadap serangan dikarenakan kurangnya minat dan kesadaran untuk meningkatkan keamanan dalam pengembangan aplikasi berbasis *web*. Bahkan jika *firewall* tradisional digunakan untuk mencegah serangan pada lapisan jaringan dengan sukses, namun tidak efektif jika terjadi serangan berbasis web pada aplikasi web (Namit, Abakash, & Dheeraj, 2008).

Aplikasi berbasis *web* yang menggunakan *Hyper Text Transfer Protocol* (HTTP) banyak mendapatkan serangan yang memanfaatkan adanya celah protokol jaringan ini, diantaranya serangan *Replay*, *Man in the Middle*, *SQL Injection*, *DNS Poisoning*, *Brute Force*, *SYN flood*, *Smurfing*, *Port Scanning*, *Session Hijacking*, dan lain-lain. Ada banyak penelitian yang berkaitan dengan pendeteksian lalu lintas HTTP dan penggunaan aplikasi yang membaca lalu lintas permintaan dari *server*. Hal yang paling sederhana terletak pada protokol jaringan HTTP yang banyak di akses ke aplikasi berbasis *web* sudah semakin meluas (Mor, Riva, Nath, & Kubiatowicz, 2015). Hal ini disebabkan kinerja protokol jaringan HTTP lebih ringan dibandingkan protokol jaringan HTTPS menggunakan protokol SSL yang memiliki tiga tingkat pengamanan data. Tetapi SSL hanya mengenkripsi koneksi, bukan mengenkripsi data pada kedua pihak yang berkomunikasi (Shaikh, Bhat, & Moharir, 2017). Salah satu yang menjadi peluang serangan adalah pada *session hijacking.*

Aplikasi berbasis *web* yang kurang memperhatikan aspek keamanan sehingga peretas dapat menemukan celah kerentanan pada *webserver* secara langsung di internet, akan memberikan peluang serangan. Pada dasarnya, sistem keamanan yang baik sangat diperlukan untuk menjaga informasi yang dimiliki oleh perusahaan, namun pada kenyataannya belum ada sebuah keamanan aplikasi berbasis *web* sempurna dan mampu menjaga informasi dengan baik. Salah satunya disebabkan adanya kelemahan pada sisi *session id* sehingga sangat mudah peretas menembus keamanan sebuah aplikasi berbasis *web*. Hal ini dibuktikan oleh penelitian yang telah melakukan audit di jalur komunikasi HTTP dan menemukan celah serangan terhadap pencurian *cookie* pada jenis situs *web* tertentu, disebabkan *website* tidak menerapkan enkripsi *cookie* yang tersimpan pada ekstensi *browser* resmi, *history* yang tersimpan pada pencarian *browser* dan aplikasi seluler (Sivakorn, Polakis, & Keromytis, 2016).

Berikut ini gambaran umum dari situs *web* dan layanan yang diauditasi, kelayakan serangan *cookie*

e-ISSN 2597-4963 dan p-ISSN 1858-4853 DOI: http://dx.doi.org/10.30872/jim.v14i2.1798

hijacking, dan jenis informasi pengguna dan fungsionalitas akun yang diperoleh (Sivakorn et al., 2016):

Tabel 1. Jenis Layanan dan Gangguan Keamanan

Service	HTTPS Adoption	Cookie Hijacking	XSS Cookie Hijacking
Google	partial	√	×
Baidu	partial	√	√
Bing	partial	√	√
Yahoo	partial	√	√
Youtube	partial	√	×
Amazon	partial	√	√
Ebay	partial	√	√
MSN	partial	√	√
Walmart	partial	√	√
Target	partial	√	√
CNN	partial	√	√
New York Times	partial	√	√
Huffington Post	partial	√	partial
The Guardian	partial	√	√
Doubleclick	partial	√	√
Skype	partial*	×	×
LinkedIn	partial*	×	×
Craigslist	partial*	×	×
Chase Bank	partial*	×	×
Bank of America	partial*	×	×
Facebook	full	×	×
Twitter	full	×	×
Google+	full	×	×
Live (Hotmail)	full	×	×
Gmail	full	×	×
Paypal	full	×	×

* *Sementara ini layanan tidak memiliki HTTPS di mana-mana tidak ada personalisasi ditawarkan di atas halaman HTTP*

Hal diatas yang menjadikan alasan penelitian ini dilakukan untuk mengajukan sistem keamanan pada aplikasi berbasis *web* dengan pengusulan metode pengacakan *session id*.

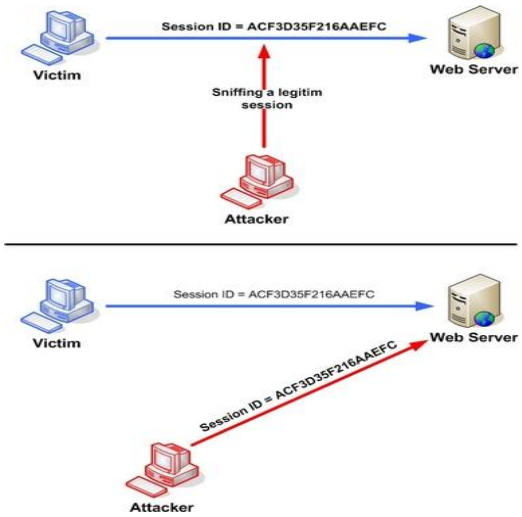
2. TINJAUAN PUSAKA

A. Session Hijacking

Serangan *Session Hijacking* (Chung, Yee, Singh, & Hassan, 2014) terdiri dari eksploitasi mekanisme kontrol sesi *web*, yang biasanya dikelola untuk token sesi. Oleh karena komunikasi HTTP menggunakan banyak koneksi TCP yang berbeda, *webserver* membutuhkan metode untuk mengenali koneksi setiap pengguna (Angsar, 2018). Metode yang paling berguna tergantung pada tanda bahwa *webserver* mengirimkan ke *browser client* setelah autentikasi *client* sukses. Token sesi biasanya terdiri dari serangkaian lebar variabel dan dapat digunakan dengan cara yang berbeda, seperti di URL, di *header* dari permintaan HTTP sebagai *cookie*, di bagian lain dari *header* permintaan HTTP, atau belum ada permintaan di dalam HTTP. Serangan *sesión hijacking* mencari titik temu token sesi dengan pencurian atau memprediksi token sesi valid untuk mendapatkan akses tidak sah ke *webserver*.

Session *hijacking attack*[6] yang juga dikenal dengan *cookie hijacking* ini digunakan *attacker* untuk mendapatkan akses tidak valid untuk mendapatkan informasi dari *user*. Dalam penyerangan ini *attacker* lebih memfokuskan ke paket IP korban sehingga dengan cara tersebut *attacker* dapat memasukkan *script* yang mendukung untuk *hacking* atau *script* kedalam jalur komunikasi antar *user* dan *server*. Selain itu *attacker* juga mencoba menemukan *cookie* korban dengan kelemahan yang ada pada sistem *blackboard learn. Attacker* mencoba mengirim email yang disertakan *script* seperti *virus* untuk membaca jalur lintasan dan dijalankan di latar belakang *browser* tersebut.

Seperti contoh pada Gambar 1, pertama penyerang menggunakan teknik *Sniffer* untuk mendapatkan sesi yang valid "*session id*", maka *attacker* mengunakan *session id* tersebut untuk dapat mengakses *webserver* secara tidak sah (Chung et al., 2014). *Attacker* akan menyadap komunikasi antara *user* dengan *server*, jika attacker mampu mendapatkan *session id* maka *attacker* dapat masuk kedalam aplikasi web tanpa melalui proses autentikasi *login* (Arifin, Bejo, & Najib, 2017). *Attacker* akan menyadap komunikasi antara *user* dengan *server*, jika *attacker* mampu mendapatkan *session id* maka *attacker* dapat masuk kedalam aplikasi *web* tanpa melalui proses autentikasi *login*.



Gambar 1 Penyerangan menggunakan teknik *session sniffing* (Chung et al., 2014).

B. Algoritme Fisher-Yates Shuffle

Fisher-Yates shuffle (Wedman, Tetmeyer, & Saiedian, 2013) merupakan algoritme yang berfungsi untuk menghasilkan permutasi acak dari suatu himpunan. Sebuah varian dari *Fisher-Yates shuffle*, yang dapat digunakan untuk menghasilkan siklus acak panjang ***n*** sebagai gantinya. Proses dasar dari

e-ISSN 2597-4963 dan p-ISSN 1858-4853 DOI: http://dx.doi.org/10.30872/jim.v14i2.1798

Fisher-Yates shuffle ini mirip dengan pemilihan secara acak dengan mengacak tiket bernomor keluar dari gelas/wadah yang nantinya akan dipilih satu persatu dan disusun kembali. Algoritme *Fisher-Yates Shuffle* (Wedman et al., 2013) (dinamai *Ronald Fisher* dan *Frank Yates*) digunakan untuk mengubah urutan acak diberikan *input* (daftar). Permutasi yang dihasilkan oleh algoritme ini terjadi dengan probabilitas yang sama.

Versi asli dari algoritme *Fisher-Yates shuffle*, yang diterbitkan pada tahun 1938, didasarkan pada yang berulang mencolok dari elemen daftar input dan menuliskannya ke daftar keluaran kedua (pendekatan ini dimaksudkan untuk dilakukan oleh manusia dengan kertas dan pensil). Meskipun Algoritme ini memiliki cara pengacakan secara dinamis, bentuk *modern* dari algoritme ini lebih efisien. Implementasi peningkatan kompleksitas algoritme menjadi ***0(n)*** dari ***0(n2)*** hanya membutuhkan waktu sebanding dengan jumlah item yang dikocok dan tidak ada ruang penyimpanan tambahan (D'silva, Vanajakshi, Manjunath, & Prabhu, 2017).

Penggunaan algoritme *Fisher-Yates shuffle* pada penelitian ini yaitu membantu pengacakan nilai *session id* yang dihasilkan dari proses autentikasi *login* oleh pengguna. Hasil proses *login* tersebut mengeluarkan nilai *session id* yang dapat dimanfaatkan oleh *attacker* untuk dapat masuk kedalam sistem tanpa melalui proses *login.*

Ilustrasi Algoritme Fisher-Yates shuffle

Pada *pseudocode* algoritme berikut ini menunjukkan bagaimana kinerja algoritme *Fisher-Yates shuffle* yang *modern* akan beroperasi pada bilangan bulat. Pada Gambar 2, diilustrasikan pengacakan dengan algoritme *Fisher-Yates shuffle.* Algoritme ini mengambil data yang sudah ada, lalu data tersebut diacak secara *random* dengan panjang data sama dengan data awal. Oleh karena itu, pengacakan yang menggunakan algoritme *Fisher-Yates shuffle* tidak akan memberatkan beban *webserver* dalam mengacak algoritme disebabkan data tidak mengambil dari nilai data yang lain melainkan data yang sudah tersedia oleh *webserver*.

```
Alur_Ilustrasi_algoritma
    {Ilustrasi Algoritma Fisher-Yates shuffle}

Kamus
Range    : jumlah angka yang belum terpilih,
Roll     : angka acak yang terpilih,
Scratch  : daftar angka yang belum terpilih,
           contoh nilai (7 9 1 3 5 8 4 2)
Result   : hasil permutasi yang akan didapatkan

Deskripsi
Read(Scratch)
FOR i=1 i<=Length(Scratch) i++
  Panjang = Length(Scratch) - i
  IF Range 1 - i <-- Untuk Pengacakan range
    Roll = Random(Panjang)
    Result = ambil_salah_satu_angka(Scratch(Roll))
    Output Result
  END IF
END FOR
```

Gambar 2 pseudocode algoritme *Fisher-Yates Shuffle*

Metode dasar yang digunakan oleh algoritme *Fisher-Yates Shuffle* untuk menghasilkan suatu permutasi acak untuk angka **1** sampai ***N*** adalah sebagai berikut (Utama & Asriningtias, 2017):

a. Tuliskan angka dari **1** sampai ***N***
b. pilih sebuah angka acak ***K*** diantara **1** sampai dengan jumlah angka yang belum dicoret.
c. dihitung dari bawah, coret angka ***K*** yang belum dicoret, dan tuliskan angka tersebut di lain tempat.
d. ulangi langkah **2** dan langkah **3** sampai semua angka sudah tercoret.
e. urutkan angka yang dituliskan pada langkah **3** adalah permutasi acak dari angka awal.

C. Algoritme Linear Congruent Method

Linear Congruent Method (LCM) adalah algoritme yang membangkitkan bilangan acak yang digunakan pada program komputer. Bilangan acak yang dibangkitkan tersebut komputer adalah bilangan semu, karena proses pembangkitannya menggunakan operasi-operasi aritmatika.

Metode yang digunakan algoritme *Linear Congruent Method* memanfaatkan metode linier untuk membangkitkan bilangan acak yang didefinisikan sebagai berikut (Utama & Asriningtias, 2017):

$$Z_i = (a\, Z_{i-1} + c)\, mod\ m \quad (1)$$

Dimana :

Z_i = bilangan acak ke-i
Z_{i-1} = bilangan acak sebelumnya
a = *factor* pengali
c = *increment*
m = *modulus*

Syarat-syarat untuk menentukan konstanta dalam algoritme *Linear Congruent Method* adalah sebagai berikut (Utama & Asriningtias, 2017):

a. konstanta ***A*** harus lebih besar dari $\sqrt{M}$
b. untuk konstanta ***C*** harus berangka ganjil apabila ***M*** bernilai pangkat 2. Tidak boleh nilai dari kelipatan ***M***
c. untuk ***M*** harus bilangan prima
d. untuk pertama ***Z0*** harus merupakan angka integer dan juga ganjil cukup besar.

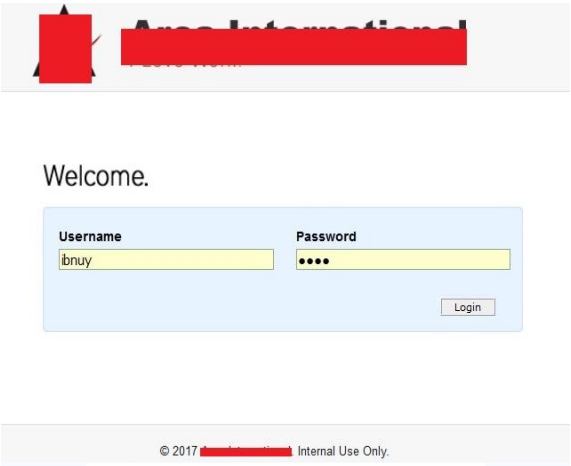
3. METODE PENELITIAN

Penelitian ini menggunakan metode penelitian kuantitatif dengan menganalisis serangan *session hijacking* terhadap protokol jaringan HTTP. Pengujian ini menggunakan aplikasi *UC browser* dan aplikasi *wireshark*. *UC browser* berguna untuk menjebak pengguna agar dapat masuk ke dalam *Wi-fi UC browser* untuk membaca lalulintas data protokol jaringan HTTP yang sedang dibuka oleh pengguna.

4. HASIL DAN PEMBAHASAN

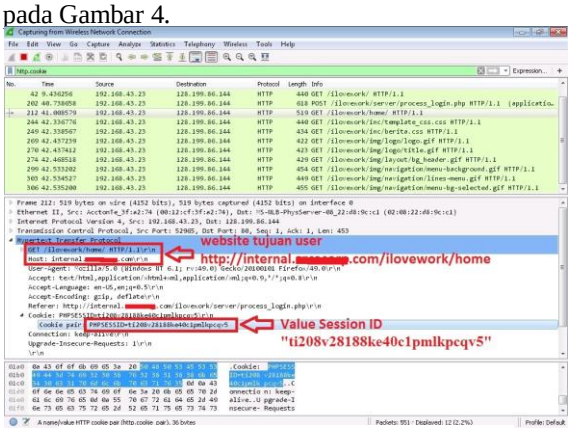
Bagian ini membahas mengenai cara melakukan penyadapan lalulintas data seperti pada Gambar 3 *user* melakukan *login* dan data tersebut masuk ke lalulintas pengiriman data yang dapat dilihat dibawah ini.

e-ISSN 2597-4963 dan p-ISSN 1858-4853 DOI: http://dx.doi.org/10.30872/jim.v14i2.1798



Gambar 3 Pengguna melakukan *login*

Dari proses *login* diatas dan berhasil *login* ke dalam sistem aplikasi ini dapat disadap lalulintas pengiriman data oleh aplikasi *wireshark* yang dilakukan *attacker* untuk mendapatkan *session id* yang digunakan *client* dan *server* agar dapat terhubung. Hasil penyadapan data yang dapat dilihat pada Gambar 4.

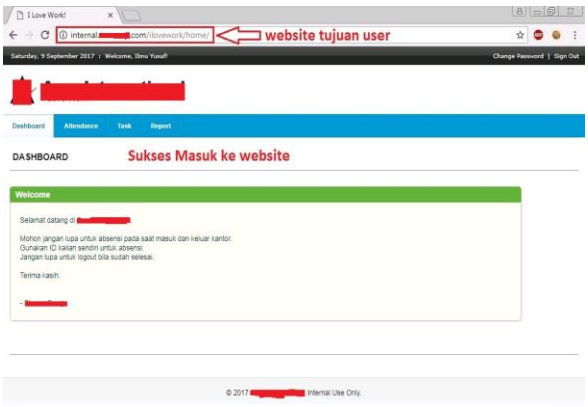


Gambar 4 Hasil Penyadapan lalulintas data

Hasil penyadapan Gambar 4 diatas terlihat *website* ini mengirim *cookie* PHPSESSID dengan *value* "ti208v28188ke40c1pmlkpcqv5". Nilai *cookie* ini dapat dimanfaatkan oleh *attacker* dengan menggantikan *cookie* yang ada disebabkan *cookie* ini sudah mengandung *value* untuk *login*. Pergantian *cookie* oleh *attacker* dapat dilihat pada Gambar 5.

Gambar 5 Pergantian *cookie* oleh *attacker*

Percobaan pergantian *cookie* seperti terlihat pada Gambar 5 ini menggunakan *add-ons* milik peramban *Chrome* yaitu "*Awesome Manager Cookie*" yang dapat bekerja menggantikan *cookie*. Jika pergantian *cookie* berhasil maka *website* tersebut dapat langsung masuk ke dalam sistem tanpa melalui proses *login* seperti yang dilakukan pengguna. Pengujian ini sukses masuk kedalam sistem seperti terlihat pada Gambar 6.



Gambar 6 Pengujian sukses masuk ke sistem tanpa proses *login*

5. KESIMPULAN

Hasil penelitian menunjukkan keamanan pada protokol jaringan HTTP rentan terhadap penyadapan lalulintas data dengan cara melakukan serangan *session hijacking*. Oleh karena itu perlu adanya peningkatan keamanan terhadap aplikasi berbasis *web*. Berdasarkan penelitian ini, diusulkan metode pengacakan *session id* menggunakan algoritme *Fisher-Yates shuffle* maupun algoritme *Linear Congruent Method* untuk diterapkan dalam pengacakan nilai dari *session id*.

6. DAFTAR PUSTAKA

Angsar, N. (2018). Pengujian Statis pada Sistem ServerWeb berbasis Cluster dengan Algoritma Never Queue. *JNTETI*, *7*(3), 297–301.

Arifin, M., Bejo, A., & Najib, W. (2017). Integrasi Login tanpa Mengetik Password pada WordPress. *JNTETI*, *6*(2), 162–167.

Chung, S. K., Yee, O. C., Singh, M. M., & Hassan, R. (2014). SQL injections attack and session hijacking on e-learning systems. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)* (pp. 338–342). Langkawi: IEEE. https://doi.org/10.1109/I4CT.2014.6914201

D'silva, K., Vanajakshi, J., Manjunath, K. N., & Prabhu, S. (2017). An effective method for preventing SQL injection attack and session hijacking. In *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)* (pp. 697–701). Bangalore: IEEE.

Mor, N., Riva, O., Nath, S., & Kubiatowicz, J. (2015). Bloom Cookies: Web Search Personalization without User Tracking. In *NDSS Symposium 2015* (pp. 1–15).

Namit, G., Abakash, S., & Dheeraj, S. (2008). *"Web Application Firewall". CS499: B. Tech Project Final Report*,.

e-ISSN 2597-4963 dan p-ISSN 1858-4853 DOI: http://dx.doi.org/10.30872/jim.v14i2.1798

Shaikh, K. A., Bhat, A. K., & Moharir, M. (2017). A Survey on SSL Packet Structure. In *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)* (pp. 1–5). Bangalore: IEE. https://doi.org/10.1109/CSITSS.2017.8447634

Sivakorn, S., Polakis, I., & Keromytis, A. D. (2016). The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information. In *IEEE*.

Utama, D. S., & Asriningtias, Y. (2017). Perbandingan waktu akses algoritma Fisher-Yates Shuffledan Linear Congruent Method pada soal Try-Out Berbasis web. *JISKA*, *2*(2), 93–102.

Wedman, S., Tetmeyer, A., & Saiedian, H. (2013). An Analytical Study of Web Application Session Management Mechanisms and HTTP Session Hijacking Attacks. *Information Security Journal: A Global Perspective*, *22*(2), 55–67. https://doi.org/https://doi.org/10.1080/19393555.2013.783952