BAB 2
LANDASAN TEORI

2.1 HMAC-Based One-Time Password

HMAC-Based One Time Password atau **RFC 4226** adalah sebuah metode yang digunakan untuk membuat kata sandi sekali pakai berbasis *hash message based authentication code* (HMAC). Dasar dari metode ini adalah sebuah penghitung yang selalu bertambah dan sebuah *shared secret* yang hanya diketahui oleh kata sandi tersebut dan juga layanan yang memeriksa apakah kata sandi tersebut sahih atau tidak. Untuk membuat HOTP, digunakan sebuah metode yaitu HMAC-SHA-1 yang didefinisikan pada RFC 2104 [19].

Hasil dari HMAC-SHA-1 adalah 160 *bits*, maka dari itu nilainya harus dipotong agar menjadi sebuah nilai yang dapat digunakan oleh pengguna. Notasi formal HOTP adalah seperti pada Rumus 2.1:

$$HOTP(K,C) = Truncate(HMAC-SHA-1(K,C)) \tag{2.1}$$

Dimana *Truncate* adalah sebuah fungsi peubah yang mengubah bentuk HMAC-SHA-1 menjadi nilai HOTP. Kunci (K), penghitung (C), dan data harus diubah menjadi bentuk *bytes* agar bisa diproses lebih lanjut oleh algoritme ini. Untuk mendapatkan nilai HOTP, digunakan metode sebagai berikut:

1. Menghasilkan sebuah nilai dengan HMAC-SHA-1. Contohnya, anggap HS = HMAC-SHA-1(K, C).
2. Menghasilkan sebuah *string* yang berukuran 4 *bytes*. Proses ini disebut juga sebagai *dynamic truncation* (DT). Contohnya, Sbits = DT(HS).
3. Menghitung nilai HOTP. Contohnya, Snum = StToNum(Sbits), kemudian HOTP = Snum MOD 10^D, dimana D adalah jumlah *digits*.

Fungsi *Truncate* dijalankan pada tahap kedua dan ketiga. Kegunaan fungsi tersebut adalah untuk menjalankan *dynamic truncation* untuk mengekstrak hasil 4 *bytes* dari hasil 20 *bytes* HMAC-SHA-1. Hasil dari modulo untuk mendapatkan HOTP harus dilakukan *masking* agar menghilangkan segala ambiguitas, karena beberapa prosesor akan menjalankan operasi aritmatika tersebut secara berbeda.

Seluruh implementasi dari HOTP minimal mengekstrak 6 karakter sebagai kata sandi sekali pakai tersebut. Implementasi dengan mengekstrak 7 karakter atau lebih sangat direkomendasikan demi mendapatkan keamanan yang lebih baik [20].

2.2 Time-Based One-Time Password

Time-Based One Time Password atau **RFC 6238** yang memiliki singkatan TOTP adalah sebuah metode yang merupakan peningkatan dari versi *HMAC-based One Time Password* (HOTP). Perbedaan TOTP dengan HOTP adalah nilai penghitung diubah dengan T, dimana T adalah *time-step* dalam waktu UNIX dikurangi dengan *time-step* yang adalah lama OTP akan berlaku. K sendiri memiliki arti yaitu kunci atau *secret*. Secara formal, notasi TOTP secara matematis adalah seperti pada Rumus 2.2:

$$TOTP = HOTP(K,T) \tag{2.2}$$

Bentuk tersebut dapat dijabarkan secara matematis menjadi Rumus 2.3:

$$TOTP = Truncate(HMAC-SHA-1(K,T)) \tag{2.3}$$

Untuk mendapatkan nilai dari T, digunakan Rumus 2.4:

$$T = Floor((T_1 - T_0)/X) \tag{2.4}$$

T_1 adalah waktu UNIX sekarang. T_0 adalah *initial counter time* yang merupakan waktu UNIX. X adalah *time-step* yang didefinisikan oleh penerima dan pemberi kata sandi. Pembagian yang dilakukan merupakan pembagian yang dibulatkan ke bawah. Implementasi dari T wajib memiliki nilai *integer* yang di atas 32 *bits* ketika tahun sudah di atas 2038. Nilai T_0 dan X merupakan nilai yang sudah diinisialisasi dari awal pembuatan algoritme.

Algoritme TOTP tidak wajib menggunakan metode *hashing* HMAC-SHA-1. TOTP dapat menggunakan metode *hashing* HMAC-SHA-256 atau HMAC-SHA-512, tergantung dengan kebutuhan yang ada dalam kasus yang sedang dihadapi. Berdasarkan spesifikasi yang diberikan oleh RFC 6238, ada beberapa hal yang perlu dipertimbangkan apabila ingin menggunakan algoritme TOTP, antara lain adalah sebagai berikut:

1. Pemberi dan penerima kata sandi wajib mengetahui waktu UNIX.

2. Pemberi dan penerima kata sandi wajib memiliki *shared secret* yang sama atau metode untuk mendapatkan *shared secret*-nya.
3. Algoritme wajib menggunakan HOTP sebagai fondasi utama.
4. Pemberi dan penerima kata sandi wajib menggunakan waktu yang sama (*time-step* yang sama).
5. Setiap penerima kata sandi wajib memiliki kunci yang unik.
6. Kunci yang dihasilkan wajib menggunakan *secure random number generator* atau diturunkan menggunakan algoritme turunan kunci.
7. Kunci wajib disimpan dalam *device* yang *tamper resistant* dan seharusnya dilidungi dari akses yang tidak diinginkan (penyusup).

Untuk validasi di bagian penerima, algoritme TOTP wajib memberikan *time-step* yang masih bisa diberikan toleransi. Hal ini disebabkan karena latensi dari aktivitas pemberian TOTP dan penerimaan/pemrosesan TOTP bisa membuat sebuah kata sandi yang apabila dikirimkan di waktu akhir, dapat diterima di *time-step* selanjutnya, yang menyebabkan kata sandi menjadi tidak sahih. Sistem validasi wajib membandingkan kata sandi dengan tidak hanya waktu penerimaan, tetapi melainkan juga menggunakan waktu pengiriman yang mungkin ada karena latensi. Untuk TOTP, pada saat penerimaan, direkomendasikan memiliki satu lebih *time-step* sebagai kompensasi atas latensi yang ada. *Time-step* yang direkomendasikan untuk algoritme ini (untuk nilai T0) adalah 30 detik, karena merupakan keseimbangan antara penggunaan dan keamanan. TOTP dapat dikirimkan dengan berbagai macam media, misalnya SMS, Email, ataupun aplikasi Authenticator [7].

2.3 Encoding Base16, Base32, Base64

Encoding adalah sebuah metode untuk mengubah bentuk data dari sebuah jenis menjadi bentuk lain yang dapat dikembalikan ke bentuknya yang semula. Ada tiga buah metode yang dapat digunakan untuk melakukan *encoding*, yaitu adalah Base64, Base32, dan Base16. Ketiga jenis ini memiliki spesifikasi yang terdapat pada **RFC 4648** [21].

2.3.1 Base64

RFC 4648 mengemukakan bahwa Base64 adalah sebuah metode yang dirancang untuk merepresentasikan karakter sekuensial sembarang yang merupakan kombinasi antara huruf kapital dan huruf non-kapital. Hasil akhir dari metode Base64 tidak perlu bisa dibaca oleh manusia. Subset 65 karakter US-ASCII digunakan, yang memungkinkan karakter 6 bit menjadi diwakili per karakter yang dapat dicetak. Karakter ekstra ke-65, '=', digunakan untuk menandakan fungsi pemrosesan khusus atau *padding*.

Proses *encoding* merepresentasikan grup masukan 24-bit sebagai *output string* dari empat karakter yang *encoded*. Proses akan dijalankan dari kiri ke kanan, sebuah *string* 24-bit dibentuk dengan menggabungkan tiga kelompok masukan 8-bit. 24-bit ini kemudian ini kemudian diperlakukan sebagai empat grup 6-bit gabungan, masing-masing yang diterjemahkan menjadi satu karakter dalam bentuk Base64.

Ada sebuah proses khusus yang dapat terjadi apabila ada data yang lebih kecil dari 24-bit di belakang *string*. Kemudian, bit dengan nilai nol ditambahkan (di sebelah kanan) untuk membentuk bilangan dari grup 6-bit. Penyetaraan pada akhirnya dilakukan dengan menggunakan karakter '='.

2.3.2 Base32

RFC 4648 mengemukakan bahwa Base32 adalah metode yang mirip dengan Base64, tetapi perbedaannya adalah metode ini hanya menggunakan subset 33 karakter US-ASCII. Proses *encoding* dari metode ini merepresentasikan 5-bit per karakter.

Proses *encoding* merepresentasikan grup masukan 40-bit sebagai *output string* dari delapan karakter yang *encoded*. Proses akan dijalankan dari kiri ke kanan. Sebuah *string* 40-bit dibentuk dengan menggabungkan lima kelompok masukan 8-bit. Setiap 40 bit ini kemudian diperlakukan sebagai 8 grup 5-bit gabungan, yang masing-masing diterjemahkan menjadi satu karakter dalam bentuk Base32.

2.3.3 Base16

RFC 4648 mengemukakan bahwa Base16 adalah metode yang merupakan bentuk sederhana dari Base32 dan Base64. Metode ini memiliki nama lain

hex encoding. Dengan kata lain, Base16 hanya merupakan *encoding* yang direpresentasikan dalam bentuk heksadesimal. Metode ini menggunakan subset 16 karakter US-ASCII. Proses *encoding* dari metode ini merepresentasikan 4-bit per karakter dalam masukan.

Proses *encoding* merepresentasikan grup masukan 8-bit sebagai *output string* dari dua karakter yang *encoded*. Proses akan dijalankan dari kiri ke kanan. Sebuah *string* 8-bit diambil dari masukan. 8 bit ini kemudian diperlakukan sebagai 2 grup 4-bit gabungan, dimana masing-masing diterjemahkan menjadi satu karakter dalam bentuk Base16.

2.4 Basic Authentication

Basic Authentication atau **RFC 7617**, yang juga dapat disebut sebagai **Basic Authorization** adalah sebuah metode yang digunakan untuk mengirimkan autentikasi kepada penyedia layanan. Basic Authentication akan di-*encode* dengan menggunakan metode Base64, sesuai dengan RFC 4648 [21]. Skema ini bukan termasuk metode yang aman untuk digunakan pada autentikasi pengguna apabila digunakan tanpa adanya konjungsi atau komplemen dengan metode lain, seperti *secure socket layer* atau *transport layer security* (SSL/TLS). Hal ini dikarenakan kredensial (biasanya merupakan ID dan *password*, dimana *password* tidak harus merupakan *password* yang digunakan pengguna) seseorang dikirim sebagai *plaintext*. Perlu diperhatikan bahwa *identifier* dan *password* dapat bersifat *arbitrary*. Kredensial yang digunakan disebut sebagai *authorization string*.

Basic Authentication menggunakan pola khusus untuk membedakannya dengan autentikasi lain. Pola yang dimaksud adalah sebagai berikut:

- Nama skema autentikasi adalah ‘Basic’.
- Parameter ‘*realm*’ dalam autentikasi harus diisi.
- Parameter ‘*charset*’ dalam autentikasi merupakan opsional.
- Parameter lainnya dalam autentikasi tidak boleh dihiraukan oleh penyedia.

Ketika penyedia menerima sebuah permintaan yang berada di dalam *protection space* yang tidak mengandung *authorization string*, penyedia dapat menjawab dengan menggunakan kode HTTP 401. Untuk mendapatkan otorisasi, pemberi dapat membuat *authorization string* dengan mengikuti kaidah berikut:

1. Mendapatkan *identifier* dan *password* dari pengguna.
2. Membuat sebuah *string* dengan format yaitu '*identifier:password*', dimana '*identifier*' dan '*password*' digabungkan dengan simbol titik dua.
3. Melakukan *encoding* dalam sekuensi oktet.
4. Melakukan *encoding* dengan format 'Base64' menjadi karakter US-ASCII yang bersifat sekuensial.

Definisi skema autentikasi ini tidak melakukan spesifikasi pada jenis *encoding* yang digunakan untuk mengubah '*identifier*' dan '*password*' menjadi sekuensi oktet. Selebihnya, karakter-karakter tersebut juga tidak boleh mengandung *control characters*, atau karakter titik dua ':' (untuk *identifier*, *password* dapat memiliki karakter titik dua).

Contoh ketika ada pengguna yang memiliki id 'Aladdin' dengan kata sandi 'open sesame', alhasil *authorization string* yang diletakkan pada *network headers* untuk permintaan tersebut adalah sebagai berikut: "Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==". Dengan demikian, penerapan RFC 7617 dapat memiliki keamanan yang baik, asal data yang dikirimkan tidak berupa *cleartext* [1].

2.5 JSON Web Tokens

JSON Web Tokens atau **RFC 7519** yang biasanya disingkat sebagai JWT adalah sebuah konsep yang digunakan untuk membuat sebuah token yang digunakan untuk melakukan autentikasi. JWT sendiri dibagi menjadi beberapa bentuk, yaitu JWS (JSON Web Signature), JWE (JSON Web Encryption), JWK (JSON Web Keys), dan lain sebagainya. JWT dapat dikirimkan melalui beberapa metode, yaitu *uniform resource locator* (URL), *parameter* pada HTTP POST, dan/atau di dalam *Authorization header* pada sebuah *request*. JWT memiliki validasi unik yakni memiliki *default claims*, yang merupakan *claims* yang sudah diatur pada spesifikasinya. JWT juga dapat diisikan konten yaitu sebuah data *custom* sesuai kebutuhan sebuah sistem, dimana dapat diverifikasi dan divalidasi secara terpisah.

Claims yang terdapat secara *default* pada JWT adalah sebagai berikut:

- 'iss' atau *issuer*, yang berfungsi sebagai tanda siapakah yang memberikan JWT ini.

- ‘sub’ atau *subject*, yang berfungsi sebagai tanda untuk siapakah JWT ini.
- ‘aud’ atau *audience*, yang berfungsi sebagai penanda *realm* atau zona keamanan tempat token ini sahih.
- ‘exp’ atau *expiration time*, yang berfungsi sebagai penanda waktu kadaluarsa dari JWT.
- ‘nbf’ atau *not before*, yang berfungsi untuk memberikan penanda waktu kapan JWT ini akan dapat digunakan.
- ‘iat’ atau *issued at*, yang menandakan kapan JWT ini dibentuk oleh sistem.
- ‘jti’ atau JWT ID, merupakan *claim* yang dapat diabaikan. Apabila diisi, *claim* ini merupakan *identifier* unik dari sebuah JWT.

JWT sendiri memiliki *signature* yang dapat dilakukan *hashing* dengan metode simetris maupun asimetris. Metode *hashing* simetris yang didukung oleh JWT adalah SHA1, SHA256, dan SHA512, sedangkan metode *hashing* asimetris yang didukung oleh JWT adalah RSA, EdDSA, dan Ed25519. Karena unsur ini, hampir tidak mungkin untuk melakukan *forgery* karena apabila ada sebuah karakter yang diubah, pasti *hash* yang didapat untuk *signature* token tersebut akan berbeda. JWT memiliki struktur dalam format *xxx.yyy.zzz*, dimana *xxx* adalah *header*, *yyy* adalah *payload* atau data yang di-*encode* dalam token, dan *zzz* adalah *signature* [22].

2.6 Open Web Application Security Project

Open Web Application Security Project atau disebut OWASP adalah sebuah *foundation non-profit* yang bertujuan untuk meningkatkan keamanan perangkat lunak. OWASP sendiri bertugas untuk memberikan rekomendasi *framework* atau tata-cara untuk mengamankan perangkat lunak yang ada di dunia. Selain itu, OWASP juga sering mengadakan acara-acara seperti *conference*, *virtual meet up*, *virtual training courses*, dan lain sebagainya.

Adapun beberapa tata-cara yang dikeluarkan oleh OWASP untuk mengukur tingkat keamanan dari sebuah perangkat lunak, antara lain:

- *OWASP Top Ten*, yang menjelaskan tentang 10 masalah terbesar dalam perangkat lunak;

- *Dependency Track*, yang merupakan *component analysis form*;
- *Mobile Security Testing Guide*, yang merupakan panduan untuk mengamankan sebuah aplikasi berbasis *mobile*;
- *Software Assurance Maturity Model*, yang memberikan analisis dan cara meningkatkannya saat mengembangkan perangkat lunak;
- *Web Security Testing Guide*, yang memberikan panduan untuk mengamankan sebuah aplikasi berbasis web;
- *API Security Top 10 2019*, yang memberikan daftar dari keamanan API yang sering terlupakan;
- *Application Security Verification Standard*, yang memberikan standar-standar umum tentang aplikasi web yang bersifat *secure*;
- *Zed Attack Proxy*, yang merupakan *automatic penetration tester* untuk mengukur tingkat keamanan aplikasi.

Dengan adanya pengukuran tingkat keamanan menggunakan OWASP, pembuatan aplikasi dapat mencapai standar keamanan yang telah ditentukan dan menjadi bisa diukur secara kuantitatif [23].

2.7 Technology Acceptance Model

Technology Acceptance Model (TAM) adalah sebuah *framework* yang digunakan untuk mengukur tingkat keberhasilan suatu teknologi. TAM sendiri dapat dibilang sebagai teori sistem informasi yang menilai bagaimana pengguna dapat menerima dan menggunakan teknologi. Ada dua buah konsep utama yang dimiliki oleh TAM, antara lain *perceived usefulness* dan *perceived ease of use*. *Perceived usefulness* sendiri didefinisikan sebagai metrik yang mempengaruhi seseorang untuk berpendapat bahwa teknologi yang digunakan akan membantu kehidupannya atau tidak. *Perceived ease of use*, sesuai namanya merupakan sebuah metrik yang mempengaruhi seseorang untuk berpendapat bahwa teknologi yang sedang digunakan merupakan jenis teknologi yang mudah digunakan atau tidak. Dalam implementasinya, umumnya konsep ini didapatkan dengan cara melakukan wawancara, penyebaran kuesioner, atau evaluasi mendalam kepada orang lain yang menggunakan teknologi yang telah dibuat untuk kemudian menarik kesimpulan dari data yang telah didapatkan [18].