MEKANISME OTENTIKASI BERBASIS TOKEN UNTUK KOMUNIKASI *REST* PADA INTERNET *OF THINGS*

Arif Setiawan, I Wayan Mustika, Teguh Bharata Adji
Jurusan Teknik Elektro dan Teknologi Informasi
Universitas Gadjah Mada
Jl. Grafika No 2 Yogyakarta -55281
E-mail: arif.setiawan.mti13@mail.ugm.ac.id , wmustika@ugm.ac.id, adji@ugm.ac.id

ABSTRAK

Mudahnya sensor *Internet of Things* yang dapat diatur untuk mengumpulkan data setiap menit bahkan detik membuat penggunaan IoT menjadi pilihan utama untuk sistem yang membutuhkan monitoring secara terus-menerus. Salah satu metode komunikasi data yang paling umum diterapkan pada IoT adalah *Representational State Transfer* (REST). REST memiliki keunggulan mampu mendukung perangkat-perangkat yang berbeda standar. Namun aspek keamanan belum menjadi topik perhatian utama. Didalam penelitian ini, akan dikembangkan sistem otentikasi REST dengan menggunakan token. Mekanismenya dalam setiap request yang terjadi, klien harus menyertakan token yang terdaftar agar *request* dapat tersebut dilayani. Hasil dari penelitian ini, diharapkan sistem yang dibuat mampu meningkatkan aspek keamanan pada aplikasi IoT yang menggunakan metode komunikasi REST.

***Kata kunci** : REST, web service, internet of things, otentikasi*

ABSTRACT

Nowadays, The Internet of Things (IoT) sensor which can be set to collect data every minute and seconds make the IoT be the first choice for systems that require continuous monitoring. One of the most common data communication methods applied to IoT is Representational State Transfer (REST). REST has the advantage of being able to support different standard devices. But the security aspect has not been a major concern in this topic. In this research, REST authentication system will be developed by using token based mechanism. In every request that occurs, the client must include a registered token for the request can be served. Result of this study, the created system can improve the security aspects of IoT applications based on REST communication

***Keywords** : REST, web service, internet of things, authentication*

PENDAHULUAN

Internet Of Things (IoT) menjadi salah satu teknologi yang ramai diperbincangkan saat ini. Kemajuannya yang pesat dan penerapannya yang bisa digunakan di semua bidang menjadikan IoT menjadi salah satu teknologi yang paling berkembang. Menurut hasil studi dari Gartner[1] , perusahaan riset dan teknologi dari Amerika Serikat. Pada tahun 2017 ini akan ada 1,5 miliar perangkat baru yang terhubung ke internet. Jumlah tersebut akan meningkat hingga 20 miliar perangkat pada tahun 2020. Perangkat IoT sendiri dapat dibedakan menjadi 3 kategori, yaitu *Wearables*, Perangkat Smart Home dan Perangkat M2M *(Machine to Machine).*

Dengan semakin banyaknya perangkat IoT yang terhubung secara online, pekerjaan manusia tentu akan terbantu. Namun dilain pihak, juga akan menimbulkan masalah baru ketika perangkat-perangkat yang terhubung tersebut memiliki tingkat keamanan yang rendah. Perangkat IoT yang dikuasai hacker dapat diubah menjadi botnet. Botnet ini mampu dikendalikan secara jarak jauh oleh hacker untuk

melakukan serangan *Distributed Denial Of Service Attacks* (DDOS) ke jaringan tertentu. Pada akhir tahun 2016 kemarin, Dyn sebuah perusahaan provider Domain Operated System (DNS) mengalami serangan DDOS pada server mereka. Hal ini mengakibatkan situs-situs besar yang menggunakan layanan DSN Dyn seperti Amazon, Airbnb, CNN, Netflix dan Spotify tidak dapat diakses oleh pengguna. Setelah dilakukan investigasi menyeluruh, ditemukan bahwa serangan tersebut dilakukan lebih dari 100.000 perangkat IoT yang terkena malware Mirai botnet[2].

Berkaca dari hal diatas, sisi keamanan dari IoT perlu ditingkatkan agar kasus tersebut tidak terulang kembali. Baik dari sisi perangkat itu sendiri, komunikasi data ataupun dari sisi *Application Programming Interface* (API). API merupakan penghubung antara bagian *backend* dan *frontend.* Di pengembangan aplikasi berbasis IoT, API menjadi sistem penghubung penghubung antara sensor dengan basis data, ataupun basis data dengan antarmuka aplikasi. Penggunaan API diimplementasikan dalam bentuk *web service.* Generasi pertama *web service* yang diperkenalkan adalah *Simple Object Acces Protocol* (SOAP) namun karena perkembangan perangkat IoT yang semakin banyak dan SOAP tidak mampu menghandle perangkat yang berbeda standar maka penggunaan SOAP mulai ditinggalkan.

Representational State Transfer (REST) menjadi pengembangan selanjutnya dari web service. REST web service / RESTful memiliki keunggulan mampu mendukung perangkat-perangkat yang berbeda standar karena menggunakan basis *Resource Oriented Architecture* (ROA)[3]. REST sendiri memiliki 4 attribute yaitu :

1. *Addresssability*

Semua resource akan diimplementasikan menggunakan *Uniform Resource Identifiers* (URI). Setiap resource tersebut akan memiliki alamat URI sendiri. Ketika alamat URI dipanggil dia akan mengembalikan respon dalam bentuk JSON atau XML.

2. Connectedness

Resource yang ada dalam REST harus memiliki relasi dengan resource yang lain agar dapat dipresentasikan melalui URI.

3. *Homogeneus Interface*

Resource akan dipanggil menggunakan 4 metode HTTP yaitu GET, PUT, POST dan DELETE dengan 2 tambahan metode yaitu HEAD dan OPTION. HEAD digunakan untuk menunjukkan metadata sedang OPTION digunakan untuk memeriksa metode yang ada

4. *Statelessness*

Stateless menunjukkan bahwa server tidak menyimpan data dari klien dari setiap koneksi yang terbentuk

Karena menggunakan dasar HTTP sebagai komunikasinya, maka metode REST memiliki kerentanan yang sama dengan beberapa aplikasi web yang lain. Menurut Femke[4], beberapa serangan yang bisa dilakukan terhadap komunikasi REST, yaitu:

1. *SQL Injection*

Metode komunikasi REST sangat bergantung pada komunikasi HTTP. Jika tidak ada mekanisme untuk melakukan validasi pada data yang masuk, maka sistem akan rentan terhadap serangan SQL Injection

2. Serangan *Man-in-the-Middle* (MITM)

Serangan yang terjadi ketika penyerang membuat sebuah koneksi baru kepada user, membuat user seolah-seolah sedang berkomunikasi dengan server yang asli.

3. *Replay Attack*

Serangan yang terjadi pada jaringan dimana penyerang mengambil informasi yang bersifat rahasia seperti otentifikasi , lalu penyerang menggunakan informasi tersebut untuk pura-pura menjadi klien yang ter-otentifikasi

4. *Spoofing*

Serangan dimana penyerang berpura-pura menjadi host yang dapat dipercaya pada suatu jaringan. Teknik ini dapat digunakan oleh penyerang untuk memalsukan data yang diminta oleh klien

5. *Cross-Site Scripting* (XSS) dan *Cross-Site Request Forgery* (CSRF)

XSS dan CSRF merupakan 2 serangan yang sama-sama ditujukan melalui browser kepada klien. Serangan ini mampu mencuri otentikasi dari klien atau memanipulasi konten yang dikirim dari server

REST merupakan metode komunikasi yang *stateless* dimana setiap *request* yang terjadi bersifat independen sehingga server harus melalukan otentikasi klien setiap kali *request* terjadi. *Stateless* juga berarti tidak ada *session* yang disimpan selama otentikasi dilakukan. Hal ini mengakibatkan otentikasi berdasar protokol HTTP menjadi tidak memadai. Pada paper ini akan dijabarkan metode otentikasi berbasis token untuk meningkatkan keamanan pada aplikasi *internet of things*

PENELITIAN TERKAIT

Penelitian pertama menggunakan metode HTTP *Basic Authentication* [5]. Metode ini menggunakan server HTTP untuk melakukan otentikasi terhadap Web Browser. Ketika klien melakukan *request* terhadap *resource* maka server akan meminta identitas dari klien tersebut. Identitas ini berupa *username* dan *password*. Ketika klien memberikan identitas yang sesuai maka server akan memberikan respon berupa *resource* yang diminta. Namun metode ini memiliki kelemahan yaitu tidak ada enkripsi terhadap *request* yang dilakukan.

Metode HTTP *Basic Authentication* kemudian dikembangkan menjadi HTTP *Digest Authentication*. Proses otentikasi yang dilakukan sama dengan *Basic Authentication* namun mekanisme yang dilakukan lebih komplek. Ketika server meminta identitas dari klien, maka klien akan memberikan *username* dan *password* yang ditambah dengan hash string seperti MD5[6]. Proses otentikasi pada HTTP *Digest Authentication* seperti berikut. Pertama klien akan melakukan *request* kepada server dan server akan memberikan *nonce* (kata acak) kepada klien. Kemudian klien akan menggabungkan *username*, *pasword* dan *nonce* tersebut untuk membuat hash. Hash tersebut akan dikirim kembali dari klien ke server. Oleh server, hash dari akan dibandingkan dengan hash yang dibuat sendiri oleh server berdasar *username* dan *password* klien. Jika hash tersebut memiliki nilai yang sama maka klien akan diberikan akses terhadap *resource*. Metode HTTP *Digest Authentication* ini lebih aman jika dibandingkan dengan HTTP *Basic Authentication* namun memiliki kelemahan karena penyerang dapat melakukan serangan MITM [3].

Pengembangan selanjutnya yaitu penggunaan token sebagai otentikasi. Metode ini diklaim lebih aman karena tidak menggunakan *username* dan *password*. Proses otentikasi pada metode ini yaitu sebagai berikut. Pertama, klien melakukan *request* kepada server dengan menggunakan *username* dan *password*. Server akan memberikan respons berupa token. Token ini akan digunakan oleh klien setiap melakukan request *resource* kepada server. Token bersifat acak dan tidak berelasi apapun dengan data klien yang ada sehingga lebih aman.

Pengembangan metode ini dilanjutkan oleh Huang et al [7] dengan penambahan *timestamp* di token pada setiap request yang terjadi. Token yang ditambahkan *timestamp* disebut *disposable token*. Dengan penambahan timestamp, maka token yang digunakan hanya berlaku dalam waktu tertentu. Sehingga mengurangi terjadinya resiko serangan. Kelemahan dari metode ini yaitu klien dan server harus membuat *disposable token* setiap request baru sehingga akan membebani resource dari sistem. Metode ini tidak cocok diterapkan pada sistem IoT karena sebagian besar perangkat IoT memiliki spesifikasi yang rendah.

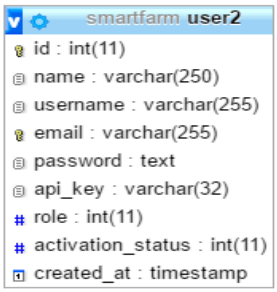
Metode lain dikembangkan oleh Lee et al [3]. Metode ini merupakan pengembangan token yang dipadukan dengan otentikasi berdasar *Identity-Based Encryption* yang dicetuskan oleh Boneh dan Franklin [8]. Pada metode ini terdapat 4 tahap yaitu *Setup, Extract, Encrypt* dan *Decrypt*. Semua otentikasi dan otorisasi klien akan dilakukan melalui *public* dan *private key* yang digenerate oleh *Public Key Generator* (PKG). Dengan metode ini maka server tidak memerlukan session ID atau *username* dan *password* dari klien. Namun metode ini masih sebatas konsep dan belum diimplementasikan dalam aplikasi IoT.

METODE

Pada bab ini akan dijelaskan metode yang dikembangkan. Ada 2 buah API yang dibuat pada sistem ini. API pertama digunakan untuk membuat fungsi *register*

dan *login* serta fungsi untuk membuat token secara acak. Sedangkan API kedua digunakan untuk melakukan *request* data pada *resource*.

Pada API Pertama, klien akan mengirim data berupa *name, username, email, password, role* dan *activation status* seperti desain tabel pada gambar 1. Data tersebut akan digunakan sebagai *registrasi klien* baru. Setelah data diterima dan tidak ada error maka server akan membuat token baru. Pembuatan token harus benar-benar unik dan sulit ditebak oleh penyerang, karena token ini berfungsi sebagai kunci akses yang digunakan user untuk melakukan *request* data.



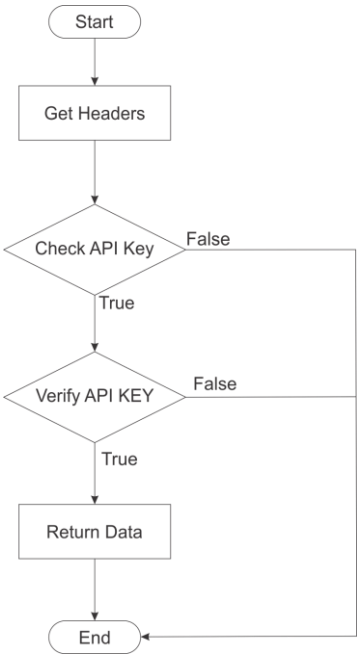
Gambar 1. Struktur tabel user

Untuk membuat token maka digunakan fungsi bin2hex. Potongan kode pembuatan token dapat dilihat pada gambar 2. Bin2hex merupakan bagian dari *Cryptographically Secure Pseudo-random Number Generator* (CSPRNG) [9] fungsi yang diperkenalkan sejak PHP versi 7. Fungsi ini digunakan untuk membuat kode kriptografi secara acak. Kode yang dihasilkan lebih aman jika dibandingkan dengan fungsi MD5 yang sudah *deprecated.*

```
private function generateApiKey() {
        return bin2hex(random_bytes(16));
    }
```

Gambar 2. Kode untuk membuat Token

API kedua berfungsi melakukan otentikasi setiap kali ada *request* pada resource. Setelah token berhasil dibuat dan disimpan maka setiap kali klien melakukan *request resource* harus menyertakan token tersebut pada *header* HTTP. Proses otentikasi token pada server dapat dilihat pada gambar 3. Ketika klien melakukan request data maka server akan melakukan pengecekan header pada request tersebut. Jika terdapat token pada header maka proses akan berlanjut. Sebaliknya, jika tidak terdapat token maka request akan ditolak. Proses selanjutnya adalah pengecekan apakah token tersebut ada pada database. Jika token ditemukan di database maka server akan memberikan data *resource* yang diminta oleh klien.

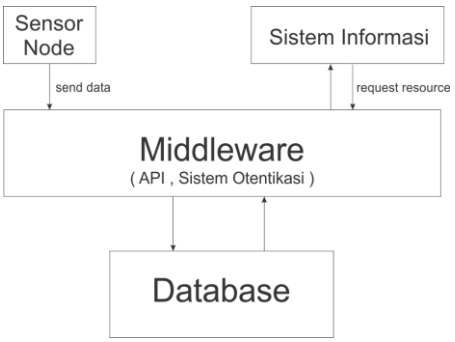


Gambar 3. Flowchart request data

HASIL

Sebagai penerapan dari metode yang diusulkan pada bagian sebelumnya, digunakan *framework* Slim untuk membantu pembuatan REST web service pada aplikasi IoT. Slim merupakan sebuah *micro framework*. Dikatakan micro karena Slim hanya fokus pada kebutuhan pokok yang diperlukan dalam suatu aplikasi web seperti : menerima HTTP request, mengirimkan request tersebut ke code yang sesuai dan mengembalikan HTTP response. Pada perancangan sistem ini Slim berperan sebagai *middleware* yang berfungsi menghubungkan antara sensor node dan layer aplikasi dengan layer database. Konsep sistem IoT yang dirancang dapat dilihat pada gambar 4.

Ada 4 bagian utama dalam sistem ini, *sensor node*, *middleware*, *database* dan sistem informasi. Sensor node berfungsi memonitoring kondisi lingkungan dengan sensor-sensor seperti sensor suhu udara, sensor suhu tanah, sensor kelembapan tanah, sensor kelembapan udara dll. Data-data yang telah diambil akan dikirim ke *middleware* yang berfungsi meneruskan data tersebut ke database. *Middleware* juga berfungsi sebagai penghubung bila ada request dari klien ke database.



Gambar 4. Konsep sistem IoT

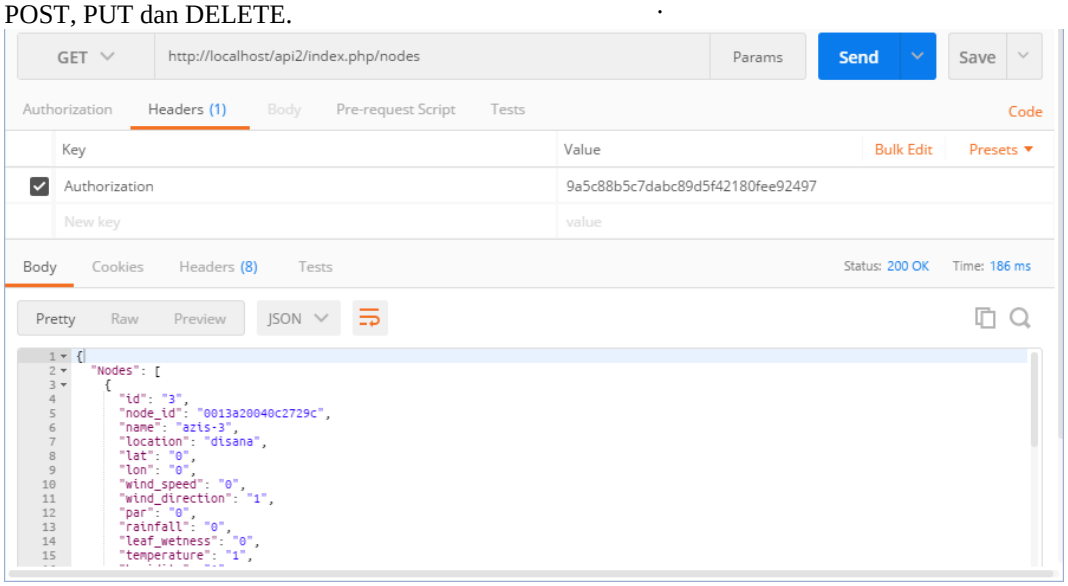
Gambar 5 merupakan potongan kode otentikasi token pada *middleware*. Kode tersebut akan berjalan setiap ada request terhadap URI yang telah ditentukan. Request yang dapat dilakukan antara lain GET, POST, PUT dan DELETE.

Untuk melakukan pengujian pada sistem yang dibuat akan dilakukan request terhadap URI yang telah ditentukan. URI yang akan dilakukan test yaitu URI untuk request data *sensor node* pada database. Untuk mempermudah dalam melakukan tes maka digunakan aplikasi Postman seperti pada gambar 6. Pada gambar tersebut klien berhasil melakukan request data dengan menyertakan token yang sesuai pada header

```
function authenticate(\Slim\Route $route) {

    $headers = apache_request_headers();
    $response = array();
    $app = \Slim\Slim::getInstance();

    if (isset($headers['Authorization'])) {
        $db = new UserDB();
        $api_key = $headers['Authorization'];

        if (!$db->isValidApiKey($api_key)) {

            $response["error"] = true;
            $response["message"] = "Api Key Salah";
            echoRespnse(401, $response);
            $app->stop();
        } else {
            global $user_id;
            $user_id = $db->getUserId($api_key);
        }
    } else {
        $response["error"] = true;
        $response["message"] = "Api Key tidak ditemukan";
        echoRespnse(400, $response);
        $app->stop();
    }
}
```

Gambar 5. Kode Otentikasi Token

.



Gambar 6. Contoh request resource dengan Token

KESIMPULAN

Aspek keamanan menjadi topik penting dalam pengembangan sistem IoT berbasis REST. Prinsip REST yang *stateless* membuat otentikasi berdasar HTTP seperti penggunaan username dan password tidak memadai lagi. Pada paper ini telah dilakukan pengembangan middleware pada sistem IoT dengan menggunakan keamanan berbasis token. *Request* pada *resource* bila tidak menggunakan token yang sesuai akan ditolak oleh *middleware*. Penelitian selanjutnya akan mencoba mengembangkan manajemen user dalam penggunaan token tersebut.

DAFTAR PUSTAKA

[1] "Gartner Says 6.4 Billion Connected "Things"; Will Be in Use in 2016, Up 30 Percent From 2015." [Online]. Available: http://www.gartner.com/newsroom/id/3165317. [Accessed: 20-Mar-2017].

[2] "DDoS attack that disrupted internet was largest of its kind in history, experts say | Technology | The Guardian." [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet. [Accessed: 20-Mar-2017].

[3] S. Lee, J. Y. Jo, and Y. Kim, "A Method for secure RESTful web service," *2015 IEEE/ACIS 14th Int. Conf. Comput. Inf. Sci. ICIS 2015 - Proc.*, pp. 77–81, 2015.

[4] F. De Backere, B. Hanssens, R. Heynssens, R. Houthooft, A. Zuliani, S. Verstichel, B. Dhoedt, and F. De Turck, "Design of a Security Mechanism for RESTful Web Service Communication through Mobile Clients."

[5] S. Lawrence and L. Stewart, *HTTP Authentication : Basic dan Digest Access Authentication*. 1999.

[6] D. Peng and C. Li, "An Extended UsernameToken-based Approach for REST-style Web Service Security Authentication," 2009.

[7] X. Huang, C. Hsieh, C. H. Wu, and Y. C. Cheng, "A token-based user authentication mechanism for data exchange in RESTful API," pp. 601–606, 2015.

[8] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[9] "PHP: CSPRNG - Manual." [Online]. Available: http://php.net/manual/en/book.csprng.php. [Accessed: 27-Apr-2017].