Manajemen User Berbasis Token Untuk Sistem Smart Agriculture

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

Abstract—**Nowadays, The Internet of Things (IoT) sensor which can be set to collect data every minute and second 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. However, 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 on smart agriculture system.. In every request that occurs, the client must include a registered token for the request can be served. In addition, user management will also be implemented based on user roles. Access from user to resource will be limited based on their role to improve the security of data. This research show thaty, the created system can improve the security aspects of IoT applications based on REST communication especially on smart agriculture system.**

Keyword: REST, web service, internet of things, authentication

Abstract—**Mudahnya sensor *Internet of Things* (IoT) 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, di beberapa penelitian aspek keamanan belum menjadi topik perhatian utama. Di dalam penelitian ini, akan dikembangkan otentikasi REST dengan menggunakan token pada sistem *smart agriculture*. Mekanismenya dalam setiap request yang terjadi, klien harus menyertakan token yang terdaftar agar request tersebut dilayani. Selain itu, akan diterapkan juga manajemen user berdasar peran pengguna. Akses dari pengguna ke *resource* akan dibatasi berdasar perannya untuk meningkatkan keamanan privacy data. Hasil dari penelitian ini, sistem yang dibuat mampu meningkatkan aspek keamanan pada aplikasi IoT yang menggunakan metode komunikasi REST khususnya pada sistem *smart agriculture*.**

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

I. 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 di lain 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 di atas, 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 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].

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, antara lain SQL *Injection*, *Man-in-the-Middle* (MITM*), Replay Attack, spoofing* dan *Cross-Site Scripting* (XSS) dan *Cross-Site Request Forgery* (CSRF).

Sistem *smart agriculture* yang menjadi bahan pada penelitian ini merupakan sistem IoT yang digunakan untuk melakukan monitoring pada perkebunan kelapa sawit. API sistem tersebut sudah dikembangkan oleh Anisa [5] pada penelitiannya.

Di dalam sistem tersebut, REST sudah dikembangkan secara baik namun masih memiliki beberapa kekurangan karena dalam pengembangan sistemnya aspek keamanan data belum menjadi prioritas. Sebagai contoh, belum adanya otentikasi *user* saat mengakses API tersebut, sehingga jika alamat URI dari API tersebut diketahui maka dapat di akses oleh semua orang. Masalah yang lain yaitu setiap *user* bisa melihat semua data dalam basis data. Baik itu data dari sensor maupun data yang bersifat rahasia seperti *username* dan *password user*. Untuk itu, penelitian ini bertujuan untuk membangun layanan otentikasi dan layanan manajemen *user* untuk mengatur dan membatasi hak akses dari *user* itu sendiri.

II. PENELITIAN TERKAIT

Penelitian pertama menggunakan metode HTTP *Basic Authentication* [6]. 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[7]. 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 di kirim 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 di klaim 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 [8] 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.

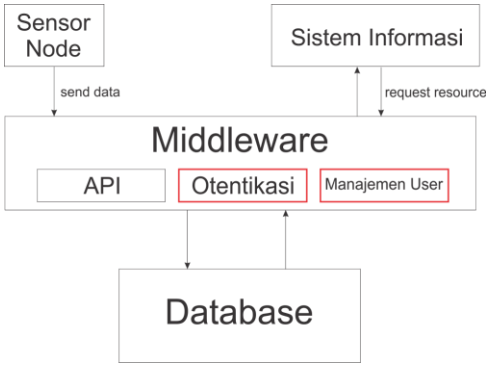
Metode lain dikembangkan oleh Lee et al [3]. Metode ini merupakan pengembangan token yang dipadukan dengan otentikasi berdasar *Identity-Based Encryption* yang dicetuskan oleh Boneh dan Franklin [9]. Pada metode ini terdapat 4 tahap yaitu *Setup, Extract, Encrypt* dan *Decrypt*. Semua otentikasi dan 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.

Pengembangan lain dari otentikasi klien dalam metode komunikasi REST yaitu otorisasi klien untuk melakukan request terhadap *resource* yang ada. Permasalahan keamanan dan privacy data menjadi fokus utama pada hal ini. Beberapa penelitian yang membahas tentang kontrol akses klien terhadap resource URI sudah diusulkan seperti WOT *Access Control* [10] dan SmartOrBAC [11]. WOT *Access Control* menerapkan sistem yang tersebar
(*Decentralized Access Control*) dimana setiap request yang terjadi akan memberikan response permission resource yang boleh atau tidak diakses. Sedangkan SmartOrBAC merupakan kebalikannya dengan menerapkan sistem kontrol yang terpusat.
SmartOrBAC membuat aturan berisi list-list klien yang memiliki permission untuk melakukan request terhadap resource di server.

III. METODE

Pada bab ini akan dijelaskan metode yang dikembangkan. Ada 2 buah layanan API yang dibuat pada sistem ini. API pertama digunakan untuk layanan otentikasi token. Sedangkan API kedua digunakan untuk melakukan validasi *user* saat melakukan *request* data

pada *resource*. Ilustrasi kedua layanan ini dapat dilihat pada Gambar 1.

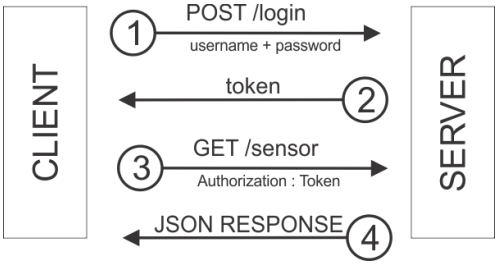


Gambar 1. Sistem API *smart agriculture*

Pada layanan API otentikasi, *user* diharuskan melakukan registrasi terlebih dahulu. Dalam proses ini *user* akan mengirim data berupa *name, username, email, password, role* dan *activation status.* Setelah data diterima dan tidak ada error maka server akan membuat token secara random. 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.

Untuk membuat token maka digunakan fungsi bin2hex. Bin2hex merupakan bagian dari *Cryptographically Secure Pseudo-random Number Generator* (CSPRNG) [12] 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.*

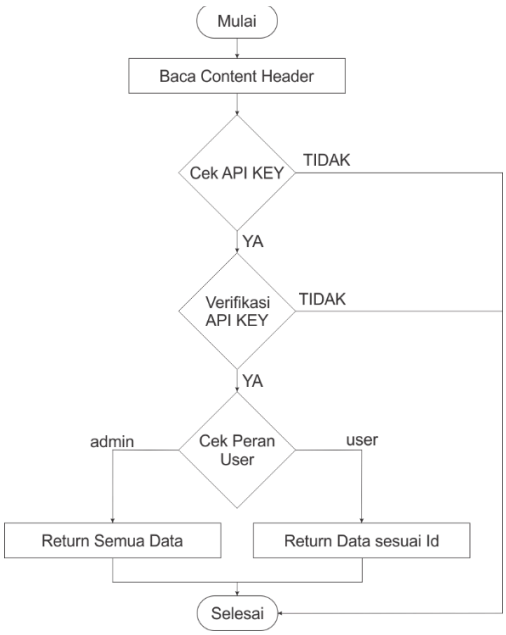
Untuk melihat token yang telah dibuat oleh server, maka *user* harus login kedalam sistem. Token ini nantinya digunakan oleh user untuk melakukan *request* terhadap *resource*. Jika tidak menggunakan token yang valid maka *request* tersebut akan ditolak oleh server. Ilustrasi dari proses ini dapat dilihat pada Gambar 2.



Gambar 2. Proses mendapatkan token dan *request resource*

API kedua berfungsi melakukan validasi token yang berfungsi untuk menentukan peran dari *user*. Untuk membantu pembuatan sistem ini maka digunakan Slim Framework. 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 kode yang sesuai dan mengembalikan *HTTP response*. Pada perancangan sistem ini Slim berperan sebagai *middleware* yang berfungsi untuk melakukan validasi peran dari user. Peran user dalam penelitian ini dibedakan menjadi 2 yaitu sebagai admin dan pengguna. Jika *user* berperan sebagai admin maka *user* berhak untuk melihat semua data. Sedangkan jika peran *user* sebagai pengguna, maka hanya data milik user tersebut yang dapat dilihat.

Contoh proses validasi 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 dan valid, maka proses akan berlanjut untuk menentukan peran dari *user*. Jika *user* tersebut admin maka server akan mengembalikan semua data. Jika user tersebut adalah pengguna biasa maka hanya data sesuai id user tersebut yang dikembalikan.



Gambar 3. Flowchart *request* data

IV. HASIL

Pada penelitian ini dilakukan pengujian dengan metode Black Box testing. Black Box Testing adalah pengujian yang dilakukan dengan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak tersebut. Dengan kata lain, input dengan output yang dihasilkan dari perangkat lunak diharapkan sudah sesuai dengan kebutuhan pengguna. Sebelum dilakukan pengujian maka akan dipetakan terlebih dahulu URI dan layanan
API dari sistem. URI yang dipetakan merepresentasikan dari layanan yang ada. Hasil dari pemetaan tersebut dapat dilihat pada Tabel 1.

Tabel 1. URI dan layanan API

No	URI	Layanan
1	/register	Pendaftaran user
2	/login	Login user
3	/apikey/:id	Request token baru
4	/shownodes	Melihat sensor node
5	-	Otentikasi Token

Setelah dilakukan pemetaan maka diketahui ada 5 layanan yang ada pada sistem API ini. Berdasarkan 5 layanan tersebut maka dikembangkan skenarioskenario yang akan dilakukan untuk melakukan pengujian terhadap fungsionalitas sistem.

Karena bagian *front end* pada sistem ini belum dikembangkan maka uintuk membantu melakukan pengujian digunakan aplikasi Postman, Postman merupakan aplikasi yang berguna untuk melakukan uji coba terhadap REST API. Hasil pengujian dari sistem ini dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian

No	Layanan	Jumlah Uji	Hasil
1	Pendaftaran	4 Skenario	Valid
2	Login	4 Skenario	Valid
3	Request Token	4 Skenario	Valid
4	Melihat Sensor node	3 Skenario	Valid
5	Otentikasi Token	3 Skenario	Valid

Berdasar hasil pengujian dari 5 layanan dengan menggunakan 18 skenario didapatkan kesimpulan bahwa layanan API yang dikembangkan telah memenuhi kebutuhan dari pengguna.

V. 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*. Manajemen user juga telah diterapkan pada sistem ini. *Request* pada *resource* akan divalidasi terlebih dahulu, data yang dikembalikan akan disesuaikan dengan peran pengguna, apakah sebagai admin atau sebagai pengguna biasa.

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] A. Azis, "Pengembangan Restful API Untuk Mendukung Sistem Pemantauan Perkebunan Kelapa Sawit,"Skripsi. Universitas Gadjah Mada 2017.

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

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

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

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

[10] S. W. Oh and H. S. Kim, "Study on access permission control for the Web of Things," *Int. Conf. Adv. Commun. Technol. ICACT*, vol. 2015–Augus, no. 1, pp. 574–580, 2015.

[11] A. Ouaddah, I. B. Anas, A. Elkalam, and A. A. I. T. Ouahman, "Security Analysis and proposal of new Access Control model in the Internet of Thing," 2015.

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