ISSN 2085-4552

Analisa Implementasi Protokol HTTPS pada Situs Web Perguruan Tinggi di Pulau Jawa

Willy Sudiarto Raharjo, Aloysius Airlangga Bajuadji
Program Studi Teknik Informatika Fakultas Teknologi Informasi UKDW Yogyakarta, Indonesia
willysr@staff.ukdw.ac.id; bayuaji@staff.ukdw.ac.id

Diterima 20 November 2016
Disetujui 21 Desember 2016

***Abstract*—**HTTPS protocol offers better data protection than regular HTTP protocol since it utilize cryptography, mainly encryption and authentication mechanism to provide confidentiality and authenticity to packets sent to and from servers. However, not all institutions have properly implemented HTTPS protocol for their web sites. This paper analyzed the implementation of HTTPS protocol for all higher education web sites in Java island. We found that only 28 out of 1505 (1.86%) of all higher education institution who have a domain name have been using HTTPS protocol for their main domain. Furthermore, not all of them have properly implemented HTTPS protocol. We analyzed all 28 domains and we found that 8 out of 28 (28.57%) institutions are still using SSLv3 protocol which is no longer recommended to be used since it's vulnerable to POODLE attack, 9 out of 28 (32.14%) institutions are still using an old algorithm RC4 which is proven to be insecure, 4 out of 28 (14.28%) institutions only support up to TLS 1.0, and 6 out of 28 (21.42%) institutions are still using SSLv2 or reusing same RSA keys thus vulnerable to DROWN attack. Many of the best practices of implementing HTTPS protocol were also neglected. HTTP Strict Transport Security (HSTS) is used by 5 out of 28 (17.8%) institutions and none of them have implemented HTTP Public Key Pinning (HPKP).

Index Terms—cryptography, HTTPS, SSL, TLS

I. Pendahuluan

Sejak diperkenalkan pada tahun 1991, *World Wide Web* telah mengalami evolusi yang sangat cepat sehingga kini telah menjadi salah satu teknologi yang digunakan oleh banyak orang diseluruh dunia dan tersedia di berbagai platform. Layanan web tidak hanya digunakan untuk berbagi informasi, namun sudah merambah ke hal-hal yang lebih bersifat personal seperti media sosial, *e-commerce*, dan lain sebagainya.

Meski demikian, layanan web pada awalnya tidak dirancang untuk menyelesaikan semua kebutuhan para pengguna di masa sekarang ini. Salah satu masalah penting yang belum terpikirkan pada waktu awal pembuatan adalah pengamanan pada data yang dikirimkan atau diterima melalui layanan web. Dengan semakin meningkatnya penggunaan layanan web di segala bidang, maka keamanan menjadi salah satu faktor penting yang perlu diperhatikan oleh pemilik informasi maupun penyedia layanan. Hal ini bisa diaplikasikan dalam bentuk penggunaan protokol HTTPS untuk pengamanan terhadap data yang lalu lalang di jalur komunikasi publik (*Internet*).

Perguruan tinggi sebagai institusi pendidikan memiliki banyak informasi yang perlu dirahasiakan dan bukan untuk konsumsi publik seperti misalnya informasi login, data mahasiswa, dosen, karyawan, dan lain sebagainya. Data-data ini banyak digunakan untuk layanan bagi civitas kampus melalui berbagai macam layanan sistem berbasis web. Pemanfaatan protokol HTTPS dapat digunakan untuk meminimalkan kebocoran data pada aplikasi yang berjalan pada *platform* web.

Dengan semakin banyaknya ancaman

keamanan di dunia maya, maka tidaklah cukup untuk hanya sekedar menggunakan protokol HTTPS, namun juga harus diikuti dengan pengaturan yang benar sesuai dengan rekomendasi yang sudah ditetapkan. Kesalahan dalam melakukan konfigurasi dapat berpotensi untuk mendapatkan kunci utama dari mesin *server* yang bersangkutan atau mampu membaca seluruh data yang dikirimkan atau diterima oleh *server* dan pengguna tanpa ada pengamanan sama sekali (*Man-in-the-Middle-Attack*).

Sejak beberapa tahun terakhir sudah banyak kesepakatan yang disetujui oleh berbagai konsorsium untuk mulai menggunakan protokol HTTPS sebagai syarat mutlak sebuah situs dianggap baik dan aman oleh *vendor browser*. Hal ini juga perlu dicermati oleh para pengelola situs web perguruan tinggi agar halaman situs web institusinya tetap dapat ditampilkan dan dinikmati oleh para penggunanya di masa yang akan datang.

II. LANDASAN TEORI

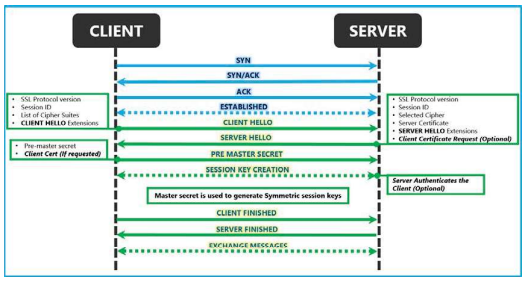
A. HTTP

HTTP (*Hypertext Transfer Protocol*) adalah sebuah protokol yang bersifat *stateless* yang dapat digunakan untuk sistem informasi terdistribusi, kolaboratif, dan berbasis pada *hypertext* yang diperkenalkan pada awal tahun 90an [1]. Protokol ini menggunakan model *request-response* dan diterapkan pada arsitektur *client-server* dimana *browser* sebagai *client* akan mengirimkan *HTTP Requests* melalui *port* 80 (*default*) dan *server* akan mengembalikan *HTTP Response*.

Masalah utama dari protokol HTTP adalah proses pengiriman *HTTP Request* dan *HTTP Response* dilakukan tanpa ada pengamanan sama sekali, sehingga seseorang yang memiliki akses di jaringan mampu menyadap informasi yang dikirimkan (*traffic sniffing*) dan bahkan bisa melakukan modifikasi (*data tampering*) tanpa diketahui oleh kedua belah pihak. Penyerang cukup memiliki akses ke salah satu infrastruktur jaringan yang dilalui dan memasang aplikasi yang mampu menyadap seperti halnya Wireshark atau Kismet. Setiap terjadi permintaan akses melalui protokol HTTP maka penyerang akan dapat melihat seluruh data yang dikirimkan, termasuk *username* dan *password* yang seharusnya menjadi informasi rahasia dari masing-masing pengguna

B. SSL/TLS

SSL merupakan protokol yang dikembangkan oleh Netscape dan bertujuan untuk menyediakan pengamanan data dengan menggunakan teknik kriptografi. SSL bisa digunakan untuk berbagai layanan, seperti halnya layanan web, email, *instant messaging*, dan lain sebagainya. Protokol ini mengijinkan *client* dan *server* untuk berkomunikasi melalui sebuah jalur yang dirancang untuk bebas dari serangan *eavesdropping*, *tampering*, dan *message forgery* [2]. SSL menggunakan algoritma enkripsi simetris untuk mengamankan data, sedangkan kunci yang digunakan untuk melakukan enkripsi/dekripsi akan dihasilkan pada saat melakukan negosiasi awal (*handshake*). Skema diagram proses *SSL Handshake* ditunjukkan pada Gambar 1.



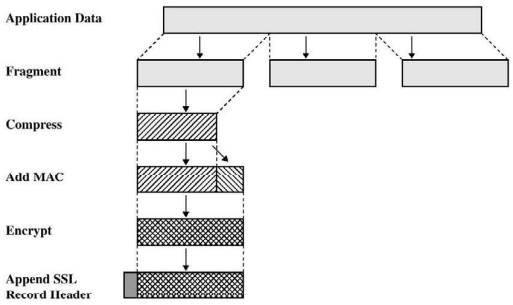
Gambar 1 SSL Handshake

C. TLS

TLS (*Transport Layer Security*) merupakan protokol yang bertujuan untuk memberikan layanan privasi dan integritas data antara dua aplikasi yang berkomunikasi. Protokol ini terdiri dari dua layer, yaitu *TLS Record Protocol* dan *TLS Handshake Protokol*. *TLS Record Protocol* menyediakan keamanan dengan mencakup dua hal yaitu koneksi yang bersifat rahasia (*confidential*) dengan menggunakan algoritma kriptografi simetris (AES) serta reliable dengan menyediakan jaminan integritas data menggunakan fungsi hash (SHA-256) [3].

TLS Record Protocol adalah protokol yang bertugas untuk menghasilkan blok data yang akan dikirimkan. Data asli yang akan dikirimkan dipecah menjadi fragmen-fragmen dimana setiap fragmennya akan dikompresi dan ditambahkan

hasil perhitungan dari MAC (*Message Authentication Code*) untuk mendeteksi terjadinya proses data tampering. Fragmen terkompresi kemudian dienkripsi menggunakan kunci yang sudah disepakati dan terakhir ditambahkan informasi *SSL Record Header* untuk menandai bahwa merupakan paket yang harus diproses menggunakan SSL/TLS. Skema *TLS Record Protocol* dapat dilihat pada Gambar 2.



Gambar 2 TLS Record Protocol

D. HTTPS

HTTPS adalah protokol HTTP yang diaplikasikan diatas protokol SSL/TLS sehingga semua fasilitas keamanan yang disediakan oleh protokol SSL/TLS juga akan dapat dinikmati oleh pengguna HTTPS. Protokol HTTPS secara default bekerja pada port 443 dan dapat dikonfigurasikan untuk menggunakan protokol SSL maupun TLS.

Meskipun SSL dan TLS sudah dikembangkan cukup lama, namun pada awalnya tingkat adopsinya masih sangat rendah sampai terjadi beberapa kasus yang kemudian membuat banyak orang sadar bahwa selama ini data yang mereka anggap aman ternyata dapat dibaca oleh orang lain dengan begitu mudahnya. Meski demikian, sampai dengan hari ini masih banyak serangan nyata yang ditujukan pada implementasi SSL/TLS atau pada protokol SSL/TLS itu sendiri.

FireSheep

Firesheep [4] adalah sebuah ekstension pada *browser* Firefox dan Chrome yang dibuat oleh Eric Butler pada tahun 2010 yang mampu mendemokan bagaimana melakukan HTTP *session hijacking* pada berbagai media sosial yang masih belum menggunakan protokol HTTPS. Seorang *attacker* cukup tersambung pada layanan WiFI yang sama dengan korban dan ketika korban mengakses sebuah situs tanpa menggunakan protokol HTTPS, maka Firesheep dapat membaca dan melakukan *session hijacking* terhadap akun korban. Pengguna tidak akan sadar bahwa akun miliknya diakses oleh orang lain dan attacker juga tidak perlu tahu *password* dari korban karena informasi *password* sudah tersimpan pada *session* yang berhasil dicuri.

HTTPS Stripping Attack

HTTPS Stripping merupakan serangan yang pertama kali dibuat oleh Moxie Marlinspike pada tahun 2009 dan sudah dipresentasikan pada acara konferensi BlackHat DC 2009 [5]. Serangan ini mampu memantau traffic koneksi HTTP dan mencoba mengarahkan link yang mengarah ke protokol HTTPS menjadi ke protokol HTTP.

BEAST

BEAST (*Browser Exploit Against SSL/TLS*) adalah serangan yang menggunakan metode chosen plaintext attack untuk memanfaatkan kelemahan pada mode CBC (*Cipher Block Chaining*) dimana seorang attacker yang mampu menebak nilai IV (*initialization vector*) dari blok data sebelumnya akan dapat menggunakannya untuk mendapatkan data *ciphertext* yang dikehendakinya [6]. Serangan ini diumumkan pada acara Ekoparty Security Conference pada tahun 2011 oleh Thai Duong dan Juliano Rizzo. Pada percobaan yang dilakukan pada situs PayPal, BEAST membutuhkan waktu sekitar 2 detik untuk melakukan dekripsi dari setiap byte dari cookie yang terenkripsi [7]. Solusi untuk mengatasi masalah ini adalah dengan menggunakan TLS versi terbaru.

CRIME

CRIME (*Compression Ratio Info-Leak Made Easy*) adalah sebuah serangan yang memanfaatkan *side-channel attack* untuk menyerang cookies pada koneksi yang menggunakan protokol HTTPS dan SPDY dan mengaktifkan fitur kompresi. Dengan memanfaatkan serangan ini, seorang *attacker* dapat mendapatkan sebagian

atau seluruh data dari *cookies* sehingga dapat digunakan untuk melakukan *session hijacking*. Solusi untuk mengatasi masalah ini adalah dengan menonaktifkan fasilitas kompresi pada protokol level SSL/TLS [8].

BREACH

Yoel Gluck, Neal Harris, dan Angelo Prado membuktikan bahwa solusi yang digunakan untuk melakukan mitigasi terhadap serangan CRIME tidaklah cukup dengan menghadirkan sebuah serangan baru yang disebut dengan BREACH (*Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext*). Mereka menemukan bahwa pertukaran data yang diamankan dengan menggunakan TLS tidaklah selalu aman terhadap serangan *side-channel attack*. Serangan dilakukan dengan cara menyerang *HTTP Response* yang masih menggunakan kompresi gzip. Karena banyak informasi seperti *CSRF token* dan *user input* juga disertakan pada *HTTP Response*, maka metode yang sama dapat diterapkan untuk serangan ini [9]. Terdapat beberapa solusi yang ditawarkan, seperti menyembunyikan informasi panjang *ciphertext*, memisahkan informasi rahasia dari input pengguna, menonaktifkan kompresi pada level HTTP, menyembunyikan informasi rahasia, dan membatasi *rate-limiting*.

POODLE

POODLE (*Padding Oracle on Downgraded Legacy Encryption*) merupakan serangan yang dicetuskan oleh tiga peneliti dari Google: Bodo Moller, Thai Duong, dan Krzysztof Kotowicz. Mereka menemukan bahwa apabila *client* dan *server* mendukung SSL dan TLS secara bersamaan, maka seorang *attacker* dapat memaksa *server* untuk menggunakan SSL versi 3.0 yang masih menggunakan algoritma yang tidak lagi dianggap aman, yaitu RC4 karena dianggap memiliki nilai bias yang bisa membocorkan informasi apabila informasi rahasia (bisa berupa kunci atau cookie) yang sama digunakan berkali-kali untuk mengirimkan data [10]. Solusi yang disarankan untuk mengantisipasi serangan POODLE adalah dengan menonaktifkan protokol SSL versi 3.0 pada HTTPS atau apabila dirasa SSL versi 3.0 masih diperlukan bisa menggunakan mekanisme TLS_FALLBACK_SCSV.

FREAK

FREAK merupakan sebuah jenis serangan yang diumumkan pada tanggal 3 Maret 2015 oleh peneliti dari INRIA di Paris, Microsoft Research, dan IMDEA. Serangan ini memungkinkan seorang *attacker* untuk meng-*intercept* koneksi HTTPS antara client dan *server* dan memaksa mereka untuk menggunakan algoritma enkripsi yang lemah sehingga *attacker* bisa mencuri data atau memanipulasi data. Masalah ini pada awalnya timbul karena adanya kebijakan dari pemerintah Amerika Serikat untuk melarang penggunaan algoritma enkripsi yang kuat diluar wilayah Amerika dan membatasi pada ukuran kunci sebesar 512 bit yang pada waktu itu dianggap cukup kuat dan aman. Sistem enkripsi yang ada lalu dirancang untuk mendukung penggunaan sistem yang kuat dan yang lemah, atau yang dikenal dengan istilah EXPORT GRADE [11].

Seiring dengan berjalannya waktu, ternyata ditemukan fakta bahwa TLS Client menerima kunci RSA yang lemah meskipun mereka tidak memintanya. Hal ini bisa dimanfaatkan oleh para *attacker* untuk mengubah permintaan ini dengan meminta kunci EXPORT GRADE RSA dan menggunakannya untuk mendapatkan kunci yang digunakan untuk melakukan dekripsi. Berdasarkan hasil penelusuran ke situs ALEXA, terdapat lebih dari 1 juta situs yang mengijinkan penggunaan RSA EXPORT.

LOGJAM

LogJam merupakan serangan yang hampir sama dengan FREAK, namun serangan ini dikarenakan adanya kelemahan pada protokol TLS itu sendiri dan fokus pada mekanisme pertukaran kunci yang menggunakan Diffie-Helman dan bukan RSA. Serangan ini memungkinkan seorang attacker untuk menjalankan serangan *man-in-the-middle-attack* dan memaksa koneksi untuk menggunakan TLS dengan algoritma kriptografi GRADE EXPORT 512 bit. Dari hasil penelitian ditemukan bahwa sekitar 8.4% dari

1 juta domain teratas dan semua *browser* web masih lemah terhadap masalah ini pada saat masalah ini diumumkan. Solusi untuk mengatasi masalah ini adalah dengan meningkatkan ukuran bit kunci yang digunakan menjadi minimal 2048 bit, menonaktifkan *export cipher suite*, dan mengupgrade versi OpenSSH yang menggunakan *Elliptic-Curve* Diffie-Hellman untuk pertukaran kunci [12].

DROWN

DROWN merupakan sebuah serangan antar protokol yang dapat mendekripsi session TLS yang telah berhasil dikumpulkan secara pasif dari *client* dengan menggunakan sebuah *server* yang mendukung protokol SSLv2 dan menggunakan *RSA key exchange*. Pada saat serangan ini ditemukan, 79% dari seluruh *server* HTTPS diseluruh dunia masih terkena dampak dari serangan ini. Solusi untuk menyelesaikan serangan ini adalah dengan menghentikan dukungan terhadap protokol SSLv2 yang sudah sangat *out-of-date* [13].

HEARTBLEED

Heartbleed adalah sebuah *vulnerability* yang ditemukan pada pustaka OpenSSL pada bulan April 2014 dan bisa digunakan untuk mengeksploitasi sebuah *server* untuk mendapatkan kunci rahasia dengan cepat [14]. Serangan ini mencoba untuk menyerang salah satu ekstensi pada protokol TLS/DTLS, yaitu heartbeat yang berfungsi untuk mempertahankan koneksi tanpa harus mengirimkan data terus menerus pada protokol TLS/DTLS [15]. Solusi untuk mengantisipasi serangan ini adalah dengan mengupgrade pustaka OpenSSL.

III. IMPLEMENTASI PENELITIAN

Pertama-tama ditentukan dahulu data perguruan tinggi yang akan digunakan. Sebagai acuan akan digunakan data awal dari halaman pangkalan data pendidikan tinggi kementrian riset, teknologi, dan pendidikan tinggi (PDPT KEMENRISTEKDIKTI). Data yang diambil adalah data dari wilayah DKI Jakarta, Jawa Barat, Jawa Tengah, DIY, dan Jawa Timur. Dari hasil pengumpulan data ini didapatkan 1942 data perguruan tinggi yang meliputi universitas, sekolah tinggi, politeknik, akademi, dan institut.

Langkah selanjutnya adalah melakukan validasi terhadap seluruh data mengingat banyak data yang bersifat duplikat dan tidak valid (misalnya alamat domain diisi alamat email atau alamat situs web yang tidak lagi bisa diakses). Dari hasil pembersihan tahap kedua, didapatkan 1928 data yang valid. Selanjutnya halaman web domain utama dari setiap institusi diakses dan dicari apakah sudah menggunakan protokol HTTPS atau belum. Hasilnya dapat terlihat pada Tabel 1.

Tabel 1 Rekap Perguruan Tinggi di Pulau Jawa

NO	PROPINSI	TIDAK DIKETAHUI		MEMILIKI WEBSITE						TOTAL PT
				HTTP		HTTPS		JUMLAH		
		Σ	%	Σ	%	Σ	%	Σ	%	
1	JAWA BARAT	71	13%	461	86%	4	1%	465	87%	536
2	JAWA TIMUR	127	25%	373	73%	13	3%	386	76%	513
3	DKI	103	25%	297	73%	5	2%	302	75%	405
4	JAWA TENGAH	91	27%	245	72%	2	1%	247	73%	338
5	DIY	31	23%	101	73%	4	4%	105	78%	136
TOTAL		423	21.94%	1477	76.61%	28	1.45%	1505	78.06%	1928

Beberapa pertimbangan lain yang digunakan selama proses validasi domain adalah domain yang dikelola sendiri dan bukan menumpang pada layanan yang disediakan pihak ketiga, misalnya Wordpress ataupun Blogspot meskipun mereka sudah menyediakan dukungan terhadap HTTPS. Dari hasil analisa tahap kedua, didapatkan 28 perguruan tinggi yang sudah menggunakan protokol HTTPS seperti terlihat pada Tabel 2.

Tabel 2 Rekap Perguruan Tinggi yang Menggunakan HTTPS

No	Nama institusi	Alamat Situs Web
1	Politeknik Negeri Media Kreatif	https://polimedia.ac.id
2	Universitas Prof Dr Moestopo	https://moestopo.ac.id
3	Universitas Yarsi	https://www.yarsi.ac.id
4	Universitas Nahdlatul Ulama Indonesia	https://www.unuindonesia.ac.id
5	Universitas Pertamina	https://universitaspertamina.ac.id
6	Institut Teknologi Bandung	https://www.itb.ac.id

7	Politeknik Negeri Jakarta	https://www.pnj.ac.id
8	Universitas Kristen Maranatha	https://www.maranatha.edu
9	Sekolah Tinggi Ilmu Dakwah Sirnarasa	https://stidsirnarasa.ac.id
10	Universitas Sanata Dharma	https://www.usd.ac.id
11	Universitas Gadjah Mada	https://www.ugm.ac.id
12	Akademi Keperawatan Notokusumo	https://www.akper-notokusumo.ac.id
13	Universitas Atma Jaya Yogyakarta	https://www.uajy.ac.id
14	Universitas Sebelas Maret	https://uns.ac.id
15	Akademi Statistika Muhammadiyah	https://www.aksibukartini.com
16	Universitas Jember	https://unej.ac.id
17	Universitas Negeri Malang	https://um.ac.id
18	Universitas Negeri Surabaya	https://www.unesa.ac.id
19	Institut Teknologi Sepuluh Nopember	https://www.its.ac.id
20	Politeknik Perkapalan Negeri Surabaya	https://ppns.ac.id
21	Universitas Surabaya	https://ubaya.ac.id
22	Universitas Narotama	https://www.narotama.ac.id
23	Universitas Gresik	https://www.unigres.ac.id
24	Institut Informatika Indonesia Surabaya	https://ikado.ac.id
25	Institut Bisnis dan Informatika STIKOM	https://www.stikom.edu
26	Sekolah Tinggi Teknik Atlas Nusantara	https://sttar.ac.id
27	STIKES Bahrul Ulum Jombang	https://www.stikes-bu.ac.id
28	Universitas Islam Negeri Sunan Ampel	https://www.uinsby.ac.id

IV. PENGUJIAN SISTEM

Pada fase pengujian, setiap domain akan diuji berdasarkan beberapa elemen, yaitu: panjang kunci, algoritma tanda tangan digital, mekanisme pembatalan, protokol yang didukung, ancaman terhadap berbagai ancaman serangan pada protokol SSL/TLS, mixed content, *secure renegotiation*, *forward Secrecy*, *Strict Transport Security* (HSTS), dan *HTTP Public Key Pinning* (HPKP). Untuk membantu pengujian, peneliti menggunakan layanan SSLTest dari SSLLabs dan juga script testssl.sh pada sistem operasi berbasis Linux.

A. Panjang Kunci dan Algoritma Tanda Tangan Digital

Panjang kunci dan algoritma tanda tangan digital merupakan dua parameter penting dalam menilai apakah sebuah sertifikat sudah diproteksi dengan aman. NIST (*National Institute of Standards and Technology*) pada tahun 2016 merekomendasikan panjang kunci minimal yang digunakan adalah 2048 bit untuk RSA atau 224 bit untuk ECC (*Elliptic-Curve Cryptography*) [16]. Untuk algoritma tanda tangan digital, NIST merekomendasikan penggunaan SHA-224, SHA-256, SHA-384, dan SHA-512.

Dari 28 perguruan tinggi yang dianalisa, semuanya sudah menggunakan panjang kunci yang dianggap aman, yaitu RSA 2048 bit atau EC 256 bit. Selain itu algoritma yang digunakan juga sudah menggunakan minimal SHA256 yang dikombinasikan dengan RSA atau ECDSA (*Elliptic-Curve Digital Signature Algorithm*).

B. Mekanisme Pembatalan, dan Extended Validation

Apabila sebuah sertifikat dianggap tidak lagi valid, maka sertifikat tersebut harus dibatalkan. Terdapat dua mekanisme pembatalan yang ada, yaitu CRL (*Certificate Revocation List*) dan OCSP (*Online Certificate Status Protocol*). CRL bekerja dengan cara mendata seluruh sertifikat yang sudah dibatalkan (*revoke*) oleh CA dan memungkinkan pihak client untuk melakukan pengujian terhadap keabsahan sebuah sertifikat sedangkan pada OCSP pihak *client* mengajukan request ke *server* OCSP. Kelemahan dari CRL adalah *overhead* yang besar, interval *cache* yang lama, dan perilaku default yang digunakan adalah menerima semua sertifikat yang tidak bisa divalidasi. Kelemahan OCSP adalah besarnya beban pada *server* karena semua *request* ditujukan ke *server* OCSP.

Untuk mengatasinya, digunakan OCSP *Stapling*, dimana *server* secara periodik akan mengirimkan OCSP *Request* dan hasilnya akan dikirimkan ke *client* pada saat melakukan SSL/TLS *handshake*. Dari pengujian diketahui 89.28% sudah menggunakan baik CRL dan OCSP sedangkan hanya 28.57% yang sudah menggunakan OCSP *Stapling*.

C. Protokol yang didukung

Pemilihan protokol yang didukung sangat dipengaruhi dari banyak faktor. Salah satunya adalah target pengguna yang ingin dicapai. Semakin tinggi protokol yang digunakan, maka semakin tinggi kebutuhan yang harus dipenuhi oleh pengguna.

Sangat penting bagi pengelola domain untuk menonaktifkan dukungan terhadap SSLv2 dan SSLv3 dan beralih ke protokol TLS yang memberikan jaminan keamanan yang lebih baik. Protokol TLS v1.2 adalah yang direkomendasikan karena penggunaan algoritma MD5 dan SHA-1 telah digantikan dengan SHA-256 dan algoritma 3DES telah digantikan dengan AES [3]. Selain itu TLS v1.2 juga memanfaatkan mode *authenticated encryption* seperti CCM [17] dan GCM [18].

Dari hasil pengujian didapatkan 4 dari 28 (14.28%) institusi yang masih belum menggunakan TLS v1.2, bahkan semua perguruan tinggi tersebut masih menggunakan TLSv1.0 yang rentan terhadap serangan *CBC Chaining Attack*.

D. Ancaman Keamanan

Layanan SSLTest dan script testSSL digunakan untuk menganalisa apakah *server* yang menjadi host dari domain tersebut rentan terhadap berbagai ancaman keamanan yang sudah dipublikasikan seperti DROWN, BEAST, POODLE, BREACH, FREAK, dan HeartBleed.

Dari semua perguruan tinggi yang ada, tidak ada satu pun yang terkena serangan BEAST, BREACH, FREAK, dan HeartBleed. Hal ini mengindikasikan bahwa administrator pengelola halaman web cukup rutin dalam mengupdate versi pustaka OpenSSL yang digunakan.

Namun demikian, sebanyak 8 dari 28 (28.57%) institusi masih rentan terhadap serangan POODLE yang menyerang pada penggunaan protokol SSLv3. Ancaman ini bisa dihindari dengan menonaktifkan modul SSLv3 sepenuhnya. Selain itu, masih terdapat 9 dari 28 (32.14%) institusi yang masih menggunakan algoritma RC4 yang sudah tidak lagi dianggap aman setelah berhasil ditemukan sebuah serangan praktis yang menyerang RC4 ketika diterapkan pada HTTPS atau WPA-TKIP dan mampu membobol hanya dalam waktu 52 jam [19].

E. Downgrade Prevention

Serangan *downgrade attack* merupakan sebuah bentuk serangan yang memaksa sebuah *server* untuk menggunakan versi protokol yang lebih rendah dan lebih lemah dibandingkan dengan daftar protokol yang didukung oleh sebuah *server*. Serangan ini pada umumnya dieksekusi dengan menggunakan model *man-in-the-middle-attack*. Salah satu serangan nyata adalah POODLE [20] yang menyerang skema enkripsi berbasis CBC yang dipakai pada SSLv3. Untuk mengatasinya adalah dengan menonaktifkan protokol SSLv3 dan beralih ke protokol TLS. Bagi yang masih belum bisa beralih ke protokol TLS, IETF telah mengeluarkan sebuah skema alternatif yang distandarisasi pada dokumen RFC 7507: *TLS Fallback Signaling Cipher Suite Value* [21] yang dapat digunakan untuk mencegah serangan *downgrade attack* dan sudah diimplementasikan pada pustaka OpenSSL sejak tahun 2014.

Dari hasil pengujian ditemukan bahwa sebanyak 7 dari 28 institusi yang masih mendukung SSLv3, namun hanya 1 perguruan tinggi yang belum menerapkan *downgrade prevention*. Perbedaan jumlah ini disebabkan karena satu perguruan tinggi tersebut masih menggunakan OpenSSL versi lama yang belum mendukung TLS_FALLBACK_SCSV, sedangkan 6 sisanya sudah menggunakan versi OpenSSL yang lebih baru.

F. Mixed Content

Istilah mixed content digunakan untuk menunjukkan bahwa dalam sebuah halaman web masih terdapat satu atau lebih *resource* yang diakses melalui protokol yang berbeda (HTTP dan HTTPS). Hal ini tidaklah dianjurkan karena informasi yang dikirimkan melalui protokol HTTP bisa disadap dengan mudah, rawan terhadap pencurian data (cookies), kebocoran data, dan penyisipan data pada situs web [22].

Berdasarkan hasil pengujian masih terdapat 11 dari 28 (39.2%) institusi yang masih memiliki masalah dengan *mixed content* dan belum sepenuhnya menggunakan protokol HTTPS.

G. Secure Renegotiation

Standar TLS v1.2 [3] mengijinkan salah satu pihak untuk melakukan negosiasi ulang, namun

masih menggunakan parameter kriptografi yang sama dengan yang digunakan pada proses *handshake* yang pertama. Hal ini memungkinkan seorang *attacker* untuk menyisipkan sebuah paket pada interaksi antara kedua belah pihak. Hal ini diatasi dengan menerapkan *secure renegotiation* yang distandarisasi oleh dokumen RFC 5746 [23]. OpenSSL sudah mendukung fitur ini sejak versi 0.9.8m. Hanya 1 perguruan tinggi yang tidak mendukung *secure renegotiation* dan setelah diteliti lebih lanjut, perguruan tinggi tersebut masih menggunakan versi library OpenSSL 0.9.8k yang belum mendukung *secure renegotiation*.

H. Forward Secrecy

Forward secrecy merupakan sebuah kondisi yang dimiliki oleh sebuah protokol apabila kunci jangka panjang yang sudah terbongkar (*compromised*) tidak akan mempengaruhi kunci sesi yang sudah lampau. Dengan kata lain, *forward secrecy* menjamin bahwa pertukaran data yang terjadi di masa lampau dengan menggunakan kunci sesi yang lama tidak akan dapat dibuka dengan menggunakan kunci sesi yang baru [24].

Hasil pengujian menunjukkan bahwa semua institusi yang menggunakan HTTPS sudah mendukung mekanisme *forward secrecy*, namun sebanyak 32.14% masih menggunakan kunci yang lemah dan terdapat 3.57% yang masih menggunakan kunci yang tidak aman dan berpotensi terkena serangan LOGJAM [12].

I. HTTP Strict Transport Security

HSTS merupakan sebuah fitur keamanan dimana pemilik domain memberitahukan kepada *browser* bahwa semua pertukaran data harus terjadi melalui protokol HTTPS dan bukan HTTP. Spesifikasi dan standarisasi tentang HSTS diatur oleh RFC 6797 [25]. Mekanisme ini bisa mengurangi resiko terjadinya pencurian data yang bisa terjadi pada protokol HTTP. HSTS juga memaksa pemilik situs untuk memastikan bahwa situs yang dikelolanya selalu menyediakan layanan HTTPS atau pengunjung situs akan mendapati pesan kesalahan. HSTS juga bisa mengatasi masalah seperti *mixed content* karena *request* yang menggunakan HTTP biasa akan ditolak dan pengembang situs akan memperbaiki kontennya dalam protokol yang aman. Penggunaan HSTS juga mampu menghindari serangan *man-in-the-middle-attack* dengan cara mencegah *user-agent* mengirimkan *request* pertama melalui protokol HTTP [26]. Satu-satunya kelemahan dari HSTS adalah apabila "IncludeSubDomains" diaktifkan, maka pemilik situs web harus memastikan bahwa semua sub domain dibawah domain utama juga harus mengaktifkan HSTS atau pengguna juga akan mendapati pesan kesalahan [27]. Sayangnya hanya 5 dari 28 (17.85%) institusi yang sudah mengimplementasikan HSTS pada situsnya.

J. HTTP Public Key Pinning

HPKP merupakan solusi untuk mengantisipasi serangan MITM (*man-in-the-middle-attack*) dengan cara memberitahukan *client* untuk menggunakan kunci publik yang sudah ditentukan oleh *web server*. Mekanisme ini menggunakan prinsip TOFU (*Trust on First Use*). Fitur ini sangat efektif untuk mencegah terjadinya penyalahgunaan sertifikat yang terjadi pada CA, namun belum ada satu perguruan tinggi pun yang sudah menggunakan fitur ini.

Protokol HPKP sendiri meskipun sudah distandarisasi pada dokumen RFC 7469 masih belum banyak diterapkan di berbagai situs besar. Salah satu kekhawatiran utamanya adalah adanya resiko yang cukup besar apabila terjadinya kehilangan identitas kriptografi atau kesalahan konfigurasi saat melakukan penggantian informasi sertifikat yang digunakan pada sebuah situs web, misalnya saat melakukan perpanjangan (*renewal*) sertifikat digital [28].

Penggunaan HSTS dan HPKP tidak serta-merta meningkatkan keamanan situs web secara langsung karena HSTS dan HPKP hanyalah mekanisme untuk mendukung penggunaan protokol HTTPS yang lebih baik. Tingkat keamanan lebih ditentukan dari berbagai aspek yang dibahas sebelumnya, misalnya versi protokol dan algoritma kriptografi yang digunakan, pengamanan terhadap berbagai ancaman *vulnerability*, dan lain sebagainya.

V. KESIMPULAN

Berdasarkan pembahasan dan analisis terhadap hasil pengujian sistem pada bab IV, maka dapat ditarik beberapa kesimpulan sebagai berikut :

1. Masih banyak perguruan tinggi yang

belum menggunakan protokol HTTPS. Dari total 1505 perguruan tinggi yang sudah memiliki situs web, hanya 28 dari 1505 (1.86%) institusi di pulau Jawa yang menggunakan protokol HTTPS. Hal ini mencerminkan fakta bahwa banyak pengelola situs web institusi masih belum sadar akan pentingnya protokol HTTPS untuk mengamankan salah satu aset sebuah institusi pendidikan, yaitu domain dan halaman web institusi.

2. Semua institusi sudah menggunakan panjang kunci dan algoritma untuk tanda tangan digital yang direkomendasikan dan aman untuk digunakan, namun ditemukan banyak permasalahan pada implementasi protokol HTTPS, baik dari sisi protokol yang digunakan maupun pada konfigurasi pada *web server*.
3. Dari 28 situs institusi yang diuji, sebanyak 28.57% masih menggunakan protokol SSLv3 dan masih rentan terhadap serangan POODLE, 32.14% masih menggunakan protokol RC4, 21.42% masih berpotensi terkena serangan DROWN karena masih menggunakan protokol SSLv2 atau menggunakan kunci RSA yang sama, dan 14.28% hanya mendukung protokol TLS 1.0 dan belum mendukung protokol TLS 1.2 yang direkomendasikan untuk digunakan. Hal ini menunjukkan bahwa tingkat security *awareness* pengelola situs web masih kurang.
4. Sebagian besar institusi pendidikan masih belum menerapkan fitur-fitur keamanan terbaru. Hanya sekitar 17.8% yang sudah mulai menggunakan *HTTP Strict Transport Security* (HSTS), namun tidak ada satu perguruan tinggi yang sudah menerapkan *HTTP Public Key Pinning* (HPKP). Hal ini mencerminkan rendahnya adopsi teknologi terbaru untuk peningkatan keamanan di situs web perguruan tinggi.

VI. SARAN

Beberapa saran yang dapat menjadi masukan untuk penelitian selanjutnya:

1. Memperluas area cakupan di luar pulau Jawa
2. Menguji aplikasi web yang digunakan apakah sudah aman terhadap berbagai ancaman serangan keamanan seperti *SQL Injection, Cross-site Scripting*, dan *Cross-site Request Forgery*.
3. Menganalisa infrastruktur lain seperti pemanfaatan DNSSEC dan DANE untuk pengamanan pada level DNS yang berbasiskan pada penggunaan *digital signature* dan *public-key cryptography*.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Mikhael Tjandrayana Setiawan yang telah membantu dalam penelitian ini dan juga Fakultas Teknologi Informasi Universitas Kristen Duta Wacana yang mendukung penelitian ini.

DAFTAR PUSTAKA

[1] Fielding, R., Reschke, J., "RFC 7230 - Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", IETF, 2014.

[2] Freier, A., Karlton, P., & Kocher, P., "RFC 6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0", IETF, 2011.

[3] Dierks, T., & Rescorla, E. "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2", IETF, 2008

[4] Butler, E. (2010). "Firesheep". Alamat situs: http://codebutler.com/firesheep.

[5] Marlinspike, M. "New Tricks for Defeating SSL in Practice", in BlackHat DC, 2009.

[6] Green, M., (2011). "A diversion: BEAST Attack on TLS/SSL Encryption", Alamat situs: https://blog.cryptographyengineering.com/2011/09/21/brief-diversion-beast-attack-on-tlsssl/.

[7] Goodin, D., (2011). "Hackers break SSL encryption used by millions of sites". Alamat situs: http://www.theregister.co.uk/2011/09/19/beast_exploits_paypal_ssl/.

[8] Ritter, T., (2012). "Details on the "Crime" Attack". Alamat Situs: https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2012/september/details-on-the-crime-attack/.

[9] Gluck, Y., Harris, N., Prado, A., "BREACH: Reviving The CRIME Attack", 2013.

[10] Moller, B., Duong, T., & Kotowicz, K. "This Poodle Bites Exploiting the SSL 3.0 Fallback", 2014.

[11] Green, M. (2015). "Attack of the week: FREAK (or 'factoring the NSA for fun and profit')". Alamat situs: http://blog.cryptographyengineering.com/2015/03/attack-of-week-freak-or-factoring-nsa.html.

[12] Adrian, D., Bhargavan, K., Durumeric, Z., dkk., "Imperfect Forward Secrecy". Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015.

[13] Aviram, N, Schinzel, S., Somorovsky J. dkk, "DROWN: Breaking TLS using SSLv2", 2016.

[14] Ristic, I. "*Bulletproof SSL and TLS*", London: Feisty Duck, 2015.

[15] Seggelman, R, Tuexen, M., Williams, M., " RFC 6520 - Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", IETF, 2012.

[16] Barker, E., "Recommendation for Key Management Part 1: General", NIST Special Publication 800-57 Part 1 Revision 4, 2016.

[17] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality" NIST Special Publication 800-38C, 2004.

[18] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", " NIST Special Publication 800-38D, 2007.

[19] Vanhoef, M., Piessens, F. "All Your Biases Belong To Us: Breaking RC4 in WPA-TKIP and TLS", Usenix Security, 2015

[20] Rizzo, J., Duong, T., "Practical Padding Oracle Attacks", 4th USENIX Workshop on Offensive Technologies, USA, Washington, DC, 2010

[21] Moeller, B., Langley, A. "RFC 7507 - TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", IETF, 2015.

[22] Chen, P., Nikoforakis, N., Huygens, C., Desmet, L. "A Dangerous Mix: Large-scale analysis of mixed-content websites", 2013

[23] Rescorla, E., Ray, M., Oskov, N., "RFC 5746 - Transport Layer Security (TLS) Renegotiation Indication Extension", IETF, 2010.

[24] Menezes, A., Oorschot, P., Vanstone, S., "Handbook of Applied Cryptography", Floria, CRC Press, 1997, bab 12, sub bab 12.15, hal 496.

[25] Hodges, J., Jackson, C., Barth, A. "RFC 6797 - HTTP Strict Transport Security (HSTS)", IETF, 2012.

[26] Helme, S. (2013, November) "HSTS - The missing link in Transport Layer Security". Alamat Situs: https://scotthelme.co.uk/hsts-the-missing-link-in-tls/

[27] Ricketts, R. (2016, April) "Header Strict Transport Security (HSTS), The Downfall of INCLUDESUBDOMAINS". Alamat Situs: http://rricketts.com/header-strict-transport-security-hsts-the-downfall-of-includesubdomains/

[28] Ristic, I. (2016, September) "Is HTTP Public Key Pinning Dead?". Alamat Situs: https://blog.qualys.com/ssllabs/2016/09/06/is-http-public-key-pinning-dead