Implementasi *Dynamic Application Security Testing* pada Aplikasi Berbasis Android

Fauzan Awanda Alviansyah[1]
Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta, Indonesia
17523147@students.uii.ac.id

Erika Ramadhani[2]
Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta, Indonesia
115230409@uii.ac.id

Abstract*— Perkembangan teknologi yang sangat pesat membuat proses komunikasi ikut berubah dengan signifikan. Peningkatan yang pesat ini diikuti dengan banyaknya pengguna perangkat mobile atau smartphone. Melalui Statcounter Global Stats salah satu website yang melakukan perhitungan berbagai jenis sistem operasi mobile yang digunakan di dunia, pada rentan waktu 2018 – 2020 tercatat sebanyak 74.95% pengguna smartphone menggunakan android sebagai sistem operasi yang digunakan. Banyaknya pengguna pengguna android menyebabkan aplikasi android menjadi target utama oleh Hacker dan Cracker dalam melakukan hacking. Hal tersebut terjadi karena pesatnya pertumbuhan aplikasi android saat ini. Dari sekian banyaknya aplikasi android yang beredar tidak semuanya sudah menerapkan pengujian keamanan dengan baik atau sesuai ISO/IEC 27001. Tingginya ancaman terhadap aplikasi android seiring dengan bertambahnya jumlah pengguna sistem operasi android membuat banyak pengembang membuat alat pengujian keamanan baik statis maupun dinamis. Diperlukan metode pengujian menggunakan metode DAST. Metode DAST dapat diterapkan menggunakan aplikasi MobSF. Tujuan akhir dari implementasi DAST pada aplikasi android adalah memberikan hasil jenis kerentanan keamanan pada aplikasi android.

Keywords*— android, Security, MobSF, DAST

I. PENDAHULUAN

Banyaknya pengguna sistem operasi android membuat jumlah aplikasi berbasis android meningkat, sejalan dengan pertumbuhan jumlah pengguna. Melalui Statcounter Global Stats salah satu website yang melakukan perhitungan berbagai jenis sistem operasi *mobile* yang digunakan di dunia, Pada rentan waktu 2018 – 2020 tercatat sebanyak 74.95% pengguna perangkat *mobile* menggunakan android sebagai sistem operasi yang digunakan [1]. Hal tersebut menjadikan aplikasi android menjadi target utama oleh *Hacker* dan *Cracker* dalam melakukan tidak kejahatan. Dari sekian banyaknya aplikasi android yang beredar tidak semuanya sudah menerapkan pengujian keamanan dengan baik sesuai ISO/27001 [2], salah satu standar keamanan manajemen informasi memberikan gambaran umum dalam mengimplementasikan keamanan pada sebuah aplikasi. Pesatnya perkembangan teknologi diiringi dengan banyaknya pengembangan aplikasi android selalu menghadirkan resiko keamanan baru.

Tingginya ancaman terhadap aplikasi android seiring dengan bertambahnya jumlah pengguna sistem operasi android, membuat banyak pengembang mengembangkan alat pengujian keamanan *Static Application Security Testing* (SAST) dan *Dynamic Application Security Testing* (DAST). Kedua metode pengujian tersebut mempunyai peran yang berbeda, SAST murni melakukan analisa pada sumber kode atau *source code* dari aplikasi yang diujikan, dalam hal ini menggunakan sudut pandang *whitebox*. Pengujian SAST murni berfokus pada analisa *source code* yang terdapat pada aplikasi. SAST tidak dapat menunjukan kerentanan aplikasi saat dijalankan [3]. Pengujian tidak hanya sebatas melakukan *source code review* melainkan harus dilakukan pengujian berdasarkan fungsi setiap *activity* yang terdapat pada aplikasi. Oleh karena itu, untuk mendapatkan hasil yang maksimal pada pengujian keamanan perlu dilakukan pengujian keamanan dengan menggunakan DAST. DAST dapat melakukan analisa saat aplikasi sedang dijalankan, dengan melakukan injeksi pada aplikasi, seperti memasukan *input* berbahaya untuk mengidentifikasi apakah aplikasi akan menampilkan kesalahan sesuai dengan *input* yang dilakukan. DAST mempunyai keunggulan dalam melakukan pengujian berdasarkan setiap *activity* yang terdapat pada aplikasi [4]. Dalam implementasi DAST memerlukan alat pengujian yang dapat menerapkan analisa dinamis. *Mobile Security Framework* (MobSF) sebagai salah satu alternatif pengujian aplikasi android secara dinamis. MobSF merupakan *framework* pengujian keamanan aplikasi *mobile* bersifat *open-source*. MobSF memiliki antar muka yang mudah dipahami dan mudah digunakan dalam melakukan analisa secara dinamis.

Karya ilmiah ini bertujuan untuk menjelaskan bagaimana implementasi DAST pada pengujian aplikasi android menggunakan MobSF. Adapun manfaat yang didapatkan adalah mendapatkan hasil laporan celah keamanan yang terdapat pada aplikasi android. Dengan dilakukan pengujian DAST diharapkan penguji keamanan atau *pentester* mendapatkan hasil celah keamanan yang lebih spesifik.

II. KAJIAN PUSTAKA

Dalam penulisan karya ilmiah ini, diperlukan informasi serupa terkait topik pengujian keamanan aplikasi android sebagai referensi penelitian. Dalam penelitian [4] memaparkan berbagai teknik pengujian keamanan aplikasi. Salah satu teknik pengujian yang digunakan adalah pengujian statis dan dinamis. Pengujian statis menggunakan sudut pandang *Whitebox*, sedangkan pengujian dinamis menggunakan sudut pandang *Blackbox*. Dalam implementasi pengujian keamanan secara dinamis disebut *Dynamic Application Security Testing* (DAST), sedangkan pengujian

keamanan secara statis disebut *Static Application Security Testing* (SAST).

Proses pengujian keamanan DAST merupakan proses pengujian aplikasi dimana perangkat lunak aplikasi dalam keadaan beroperasi. Adapun, tujuan utama dari DAST adalah melakukan analisa dan mencari kerentanan keamanan (*vulnerability*) dalam aplikasi android yang sedang berjalan. DAST dapat diterapkan saat pengembangan aplikasi sudah masuk pada fase produksi (*production*), atau setelah fase awal pengembangan (*development*) [5]. Pada penerapannya DAST menggunakan sudut pandang *blackbox*, yaitu merupakan wilayah pengujian yang dilakukan tanpa menggunakan akun ataupun tanpa memiliki akses terhadap aplikasi.

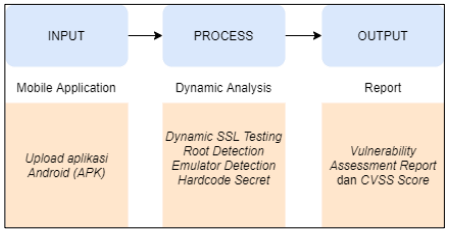
Implementasi pengujian keamanan aplikasi android dilakukan pada penelitian [6]. Penelitian tersebut menjelaskan tentang pengujian aplikasi android dengan secara statis atau SAST. Dalam penelitian tersebut dilakukan pengujian menggunakan *Mobile Security Framework* (MobSF) sebagai alat pengujian secara SAST. Dipilihnya MobSF sebagai alat pengujian karena MobSF mempunyai fitur dalam melakukan analisis secara statis maupun dinamis dengan menampilkan analisa izin, menampilkan *source code* dari aplikasi yang di uji, dan menampilkan file maupun malware yang terdapat dalam *source code* yang ada [6], akan tetapi pengujian secara statis hanya berfokus pada analisa *source code*, sehingga pengujian tidak dilakukan secara menyeluruh. Pengujian juga dilakukan secara manual, sehingga penguji keamanan masih harus melakukan validasi terlebih dahulu terhadap jenis celah keamanan yang diberikan.

Pada penelitian [7] memberikan usulan pengujian keamanan aplikasi berbasis website yang dilakukan secara dinamis atau DAST. DAST memiliki keunggulan dalam pengujian aplikasi yang uji daripada SAST. Pengujian keamanan aplikasi secara dinamis dapat mendeteksi kerentanan keamanan dengan mengirimkan *string* yang telah ditentukan sebelumnya ke aplikasi [8]. Dalam pengujian dengan DAST tidak diperlukan akses ke sumber kode, melainkan hanya membutuhkan APK yang akan diuji. Pengujian dapat dilakukan berdasarkan fungsi yang dimiliki oleh aplikasi, dapat berupa pengujian XSS, SQLinjection, dan IDOR.

Dari penelitian yang ada, telah dilakukan pengujian keamanan aplikasi menggunakan statis (SAST) dan dinamis (DAST) untuk berbagai macam aplikasi berbasis *mobile* android dan Website. Terdapat 2 penelitian terdahulu yang telah menerapkan pengujian secara dinamis (DAST) pada aplikasi berbasis website, dan salah satu penelitian menjelaskan penerapan pengujian statis (SAST) pada aplikasi berbasis android menggunakan MobSF. Namun, belum ada penelitian yang menggunakan MobSF sebagai DAST dalam pengujian aplikasi berbasis android. Oleh karena itu, pada penelitian ini akan melakukan implementasi DAST dalam pengujian kerentanan keamanan aplikasi android menggunakan MobSF sebagai alat pengujian.

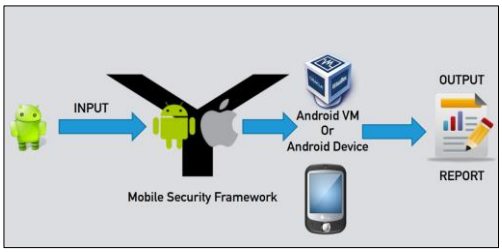
III. METODE ANALISIS

Dalam makalah ini metode yang digunakan dalam melakukan pengujian keamanan pada aplikasi android adalah metode pengujian dinamis atau DAST yang menggunakan *Blackbox* sebagai sudut pandang pengujian. *Blackbox* dilakukan tanpa menggunakan akun ataupun tanpa memiliki akses pada sistem. Dalam pengujian ini, penguji tidak akan memperoleh informasi apapun kecuali target berupa domain, IP *address*, atau aplikasi [9]. Aplikasi yang dapat digunakan dalam implementasi DAST pada makalah ini adalah MobSF yang merupakan aplikasi *open-source* yang dikembangkan oleh Ajin Abraham [10].



Gambar 1 Alur kerja DAST

Gambar 1 menjelaskan alur kerja tahapan yang dilakukan dalam menggunakan DAST terbagi menjadi 3 tahapan yaitu *Input*, *Process*, dan *Output*. Setiap tahapan memiliki peran dan fungsinya masing-masing. Pada tahap *Input*, dilakukan dengan mengunggah aplikasi android pada MobSF. Kemudian, pada tahap *Process* MobSF akan melakukan pengujian terhadap aspek-aspek keamanan yang terdapat dalam seluruh *activity* aplikasi. Dari hasil proses pengujian aspek-aspek pada secara dinamis akan menghasilkan temuan celah keamanan berdasarkan *activity* yang diuji.



Gambar 2 Gambaran Sistem MobSF

Pada Gambar 2 menjabarkan alur dari *Input* sampai dengan *Output* saat melakukan pengujian keamanan menggunakan MobSF. Adapun penjelasannya sebagai setiap tahapan yang dilakukan adalah sebagai berikut:

- *INPUT*: Pada tahapan ini dilakukan aplikasi MobSF dijalankan pada jaringan lokal Windows 10. Kemudian, dilakukan upload file aplikasi android yang akan diujikan pada *dashboard* MobSF.
- *PROCESS*: Pada tahap ini mulai dilakukan *dynamic analysis* menggunakan MobSF. Aplikasi android akan otomatis terinstall pada *virtual machine* android yang sudah disiapkan. Kemudian, MobSF akan secara otomatis melakukan pengujian secara dinamis bersamaan dengan berjalannya aplikasi atau aplikasi yang berjalan pada *runtime*.
- *OUTPUT*: Pada tahapan ini akan dihasilkan laporan kerentanan (*vulnerability assessment*) dari hasil *dynamic Analysis* pada pengujian aplikasi android yang dilakukan.

Dalam implementasi DAST diperlukan perangkat lunak pendukung, karena secara bentuk pengujian yang dilakukan yaitu dengan menjalankan aplikasi pada sistem operasi

android secara langsung, kemudian melakukan *monitoring* dan analisis melalui *runtime* pada android, maka diperlukan perangkat lunak pendukung jalannya pengujian menggunakan MobSF. Diperlukan perangkat lunak pendukung sebagai berikut:

- Python, sebagai bahasa pemrograman dalam menjalankan MobSF.
- *Virtual Machine*, sebagai mesin virtual yang digunakan dalam melakukan analisis secara dinamis.
- *Java Development Kit* (JDK) sebagai perangkat lunak yang digunakan dalam melakukan *compile* dan *decompile* APK.
- *Android Application Package* (APK) sebagai bahan pengujian keamanan dengan MobSF.
- Sistem operasi Windows 10 yang digunakan dalam menjalankan MobSF.

TABEL 1 SPESIFIKASI SOFTWARE

No	*Software*	Versi
1	Python	3.7
2	Frida & Objection	14.0.1
3	*Java Development Kit* (JDK)	8+
4	MobSF	3.1
5	Genymotion	3.1.2
7	Android *API*	5.0 API 21

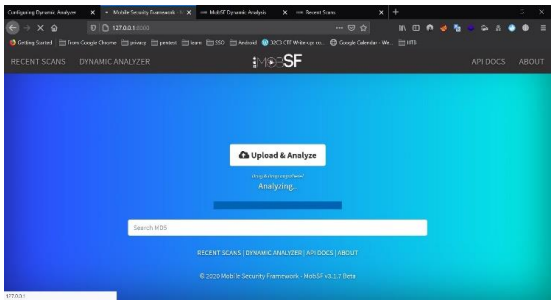
Pada Tabel 1 memaparkan versi dari perangkat lunak pendukung, yang digunakan untuk mendukung penelitian ini. Versi perangkat lunak pendukung yang digunakan dalam penelitian ini berdasarkan versi minimal dalam menjalankan aplikasi MobSF.

IV. IMPLEMENTASI DAN HASIL ANALISIS

Sesuai jenis *security testing* yang digunakan yaitu DAST menggunakan MobSF pada aplikasi android. Adapun, langkah yang harus dilakukan dalam implementasinya sebagai berikut:

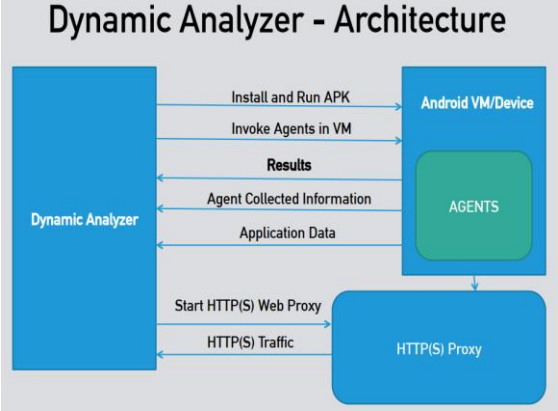
A. Implementasi Mobile Security Framework (MobSF)

MobSF terdiri dari *library python* yang digunakan dalam melakukan pengujian seperti Frida, Objection, pyOpenSSL, dan juga Java Development Kit (JDK). Setelah selesai melakukan instalasi MobSF, nantinya akan dijalankan pada jaringan lokal atau *localhost* pada Windows.



Gambar 3 Halaman Utama MobSF

Gambar 3 menampilkan halaman utama MobSF saat sudah berhasil dijalankan pada jaringan lokal Windows. Analisa secara dinamis menggunakan MobSF dilakukan dengan cara menjalankan sistem operasi android pada *virtual machine* android, pada makalah ini *virtual machine* yang digunakan adalah Genymotion dengan versi android 5.0 dengan API 25.



Gambar 4 Arsitektur DAST pada MobSF

Tahap pengujian secara DAST dilakukan dengan menjalankan aplikasi pada emulator android, kemudian dilakukan pengambilan data dari *runtime* emulator yang sudah terhubung dengan MobSF. Seluruh aktivitas yang dilakukan penguji akan terekam oleh MobSF melalui Web *Proxy* HTTP/HTTPS yang sudah otomatis terhubung dengan MobSF. Aktivitas tersebut dapat berupa hubungan antara *client-server* dalam bentuk POST dan GET *request* pada *endpoint* atau *Application Programing Interface* (API).

B. Hasil Analisis Pengujian Keamanan

Setelah melalui tahap instalasi MobSF, selanjutnya dilakukan pengujian kerentanan terhadap aplikasi android. Tabel 3 merupakan sampel pengujian keamanan aplikasi android dengan sampel aplikasi bernama Apotik K24 dan Diva.

TABEL 2 SAMPEL APLIKASI ANDROID

No	Sampel	Versi	Keterangan
1	K24	4.01.0	Aplikasi penyedia obat-obatan
2	Diva	-	DIVA (Damn insecure and vulnerable App) aplikasi yang sudah dikonfigurasi memiliki celah keamanan

Pengujian bertujuan untuk melakukan pengujian aplikasi guna memastikan pengujian dengan DAST menggunakan MobSF berjalan dengan lancar dan dapat menghasilkan berbagai macam celah keamanan yang terdapat pada aplikasi yang diuji. Pengujian menggunakan 2 aplikasi yang berbeda. Kedua aplikasi tersebut nantinya akan dilakukan perbandingan jenis keamanan antara K24 aplikasi penyedia obat-obatan dan Diva aplikasi yang sudah dikonfigurasi memiliki celah keamanan. Hasil pengujian nantinya akan berisikan temuan celah keamanan yang dilakukan secara dinamis pada kedua aplikasi.

1) Pengujian Fungsionalitas

Tahap pengujian fungsionalitas bertujuan untuk memastikan MobSF sudah berjalan sesuai dengan tahap perancangan sistem MobSF. Beberapa poin yang diujikan dalam pengujian fungsionalitas dari MobSF diuraikan seperti pada Tabel 3.

TABEL 3 ASPEK PENGUJIAN FUNGSIONALITAS

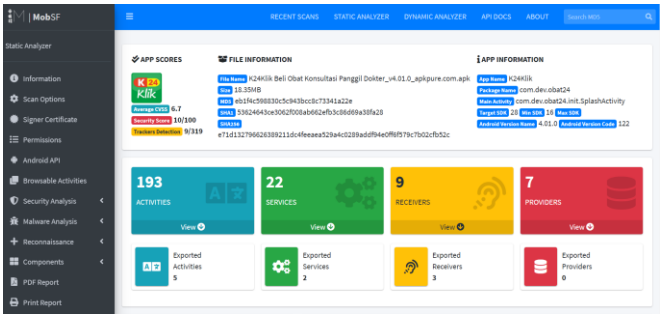
No	Nama Uji Kasus	Pengujian pada MobSF
1	Prosedur	Mengunggah file aplikasi android (APK)
2	Hasil	Sistem mampu mendeteksi aplikasi android (APK). Kemudian, dapat menampilkan informasi terkait aplikasi yang diunggah.
3	Status	Berhasil

2) Pengujian Dynamic Analysis

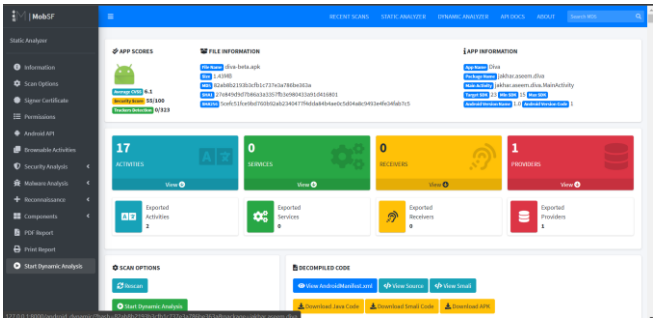
Pengujian secara dinamis dilakukan dengan menggunakan menjalankan aplikasi pada *virtual machine* yang berisikan sistem operasi android. Adapun, tahapan yang dilakukan dalam melakukan pengujian adalah sebagai berikut.

a) Unggah File APK

Unggah *file* APK pada *dashboard* yang sudah disediakan oleh MobSF. Setelah melakukan unggah *file*, maka secara otomatis MobSF akan melakukan statis.



Gambar 5 Informasi Aplikasi K24



Gambar 6 Informasi Aplikasi Diva

Pada Gambar 5 dan Gambar 6 menampilkan hasil dari *statis* dari aplikasi K24. Analisa tersebut dilakukan dengan menggunakan *scanning* otomatis pada aplikasi yang sudah dilakukan *decompile* oleh MobSF.

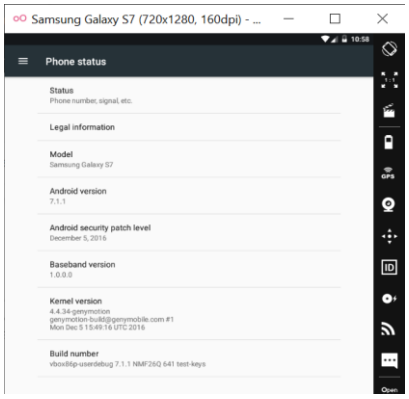
TABEL 4 HASIL STATIS

Possible Vulnerability	Diva	K24
Hard Coded Secret	No	Yes
Certificate	No	Yes
Anti-VM Code	No	Yes

Tabel 4 menampilan hasil rangkuman dari aspek keamanan yang sudah diterapkan oleh kedua sampel aplikasi. Hasil rangkuman tersebut dilakukan secara statis oleh MobSF. Hasil dari pengujian statis, aplikasi K24 sudah menerapkan teknik keamanan berupa *hard coded secret* dan *anti-VM Code*. Karena pada dasarnya aplikasi Diva diracancang memiliki celah keamanan, maka dari itu aplikasi Diva tidak menerapkan ketiga teknik tersebut. Hasil pengujian secara statis bersifat *false positive*. Hasil tersebut harus dilakukan validasi dengan dinamis untuk mengetahui apakah kedua sampel aplikasi sudah menerapkan aspek keamanan tersebut. Selain memvalidasi teknik keamanan yang sudah diterapkan, pengujian secara dinamis dapat dilakukan injeksi kode terhadap semua *activity* yang dimiliki oleh aplikasi android.

b) Menjalankan Virtual Machine

Dalam menjalankan pengujian secara dinamis perlu dilakukan instalasi aplikasi pada *virtual machine*. Pada makalah ini *virtual machine* yang digunakan adalah Genymotion.

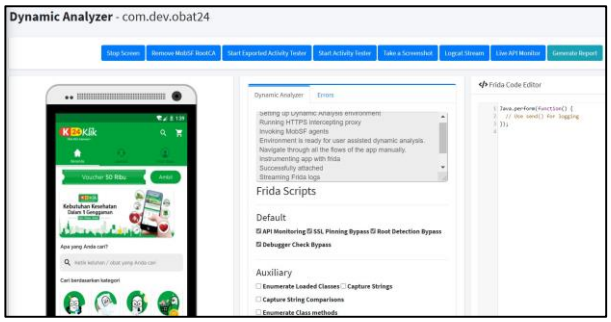


Gambar 7 *Virtual Machine* Genymotion

Pada Gambar 7 menampilkan versi android yang digunakan dalam pengujian. MobSF memberikan minimum versi android yang dapat digunakan dalam melakukan DAST yaitu di atas versi android 5.1. Apabila menggunakan versi android di bawah 5.1 maka diharuskan untuk melakukan konfigurasi pada HTTP Proxy yang ada pada MobSF

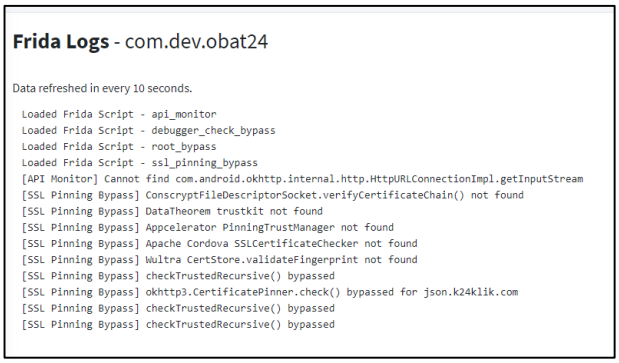
c) Melakukan Dynamic Analysis

Dalam melakukan pengujian secara dinamis harus dipastikan bahwa *virtual machine* sudah berjalan. Pada Gambar 8 menggambarkan sudah menjalankan *dynamic analysis* dengan ditandainya dibukanya aplikasi pada Genymotion. Gambar 8 menampilkan pilihan pengujian yang dapat dilakukan seperti *bypass SSL pinning* dan *root detection*.



Gambar 8 *Dynamic Analysis* pada MobSF

MobSF akan terhubung dengan *virtual machine* melalui *proxy* yang terdapat pada MobSF dan Genymotion. Saat aplikasi dijalankan nantinya akan dilakukan pengujian SSL yang terdapat pada kedua aplikasi tersebut. MobSF akan melakukan bypass SSL *pinning* menggunakan *tools* Frida.

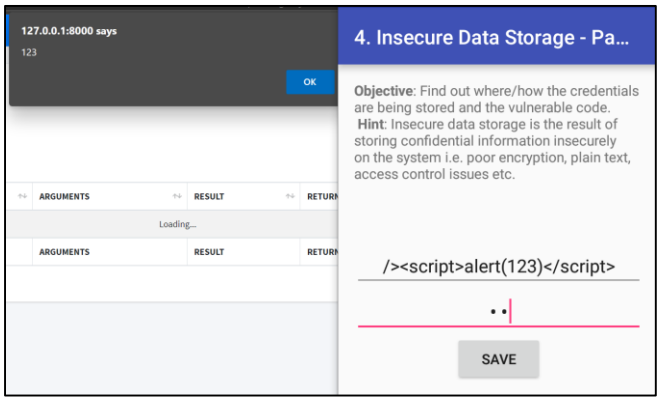


Gambar 9 Bypass SSL Pining pada K24

Gambar 10 Bypass SSL Pining pada Diva

Pada gambar 9 dan 10 menampilkan *log* dari teknik *Bypass* SSL Pining pada kedua aplikasi. Dalam implementasi DAST dalam pengujian keamanan aplikasi android yang dilakukan yaitu mencoba seluruh fitur atau *Activity* yang ada pada aplikasi tersebut.

Pengujian pertama dilakukan dengan memberikan injeksi pada *form* yang tersedia pada aplikasi. Injeksi berupa *script javascript* yang disisipkan pada *form* yang tersedia.



Gambar 11 Payload XSS

Gambar 11 menampilkan *script javascript* yang digunakan untuk mengetahui apakah aplikasi Diva mempunyai kerentanan terhadap serangan *Cross Site-Scripting* (XSS). Hal serupa juga dilakukan pada aplikasi K24. MobSF akan secara otomatis merekam seluruh aktivitas yang dilakukan. Seluruh aktivitas yang direkam tersebut nantinya akan dilakukan analisa oleh MobSF secara otomatis. Pada akhir akan memberikan daftar celah keamanan yang terdapat pada aplikasi.

3) Hasil

Pengujian diakhiri dengan melakukan setelah berhasil mengakses seluruh fitur atau menu yang terdapat pada aplikasi, kemudian pilih ***Generate Report*** agar seluruh aktivitas yang sudah terekam akan dianalisis oleh MobSF. Aktivitas yang terekam tersebut nantinya akan memperlihatkan hasil celah keamanan berdasarkan pengujian dinamis yang dilakukan oleh MobSF, didapatkan hasil seperti pada Tabel 5.

TABEL 5 HASIL PENGUJIAN DYNAMIC ANALYSIS

App	Bypass SSL Pining	No Hardcode Secret	No Root Detection	Stored XSS	Database Breach
K24	Yes	No	No	No	No
Diva	Yes	Yes	Yes	Yes	Yes

Tabel 5 memaparkan hasil pengujian secara DAST, dengan pengujian tersebut dapat dilakukan pengujian langsung terhadap fitur yang terdapat pada aplikasi. Pada Tabel 5 disebutkan bahwa kedua aplikasi memiliki hasil yang berbeda. Pada aplikasi K24 jika menggunakan analisa statis didapati bahwa K24 sudah menerapkan SSL Pining atau *Certificate*, tetapi setelah dilakukan analisa dinamis didapati bahwa SSL *Pinning* tersebut masih dapat dilakukan teknik *Bypass* SSL *Pinning*. Sebaliknya, aplikasi Diva yang dari awal memang didesain khusus agar memiliki celah keamanan tidak menerapkan aspek keamanan apapun sehingga menjadikannya aplikasi yang memiliki banyak kerenatanan. Lebih lanjut, berikut ini adalah penjelasan dari masing-masing dari aspek pengujian kerentanan.

a) SSL Pinning

SSL *Pinning* merupakan sebuah lapisan keamanan tambahan yang berfungsi untuk memberikan perlindungan terhadap serangan *man-in-the-middle*. Untuk itu, hanya Otoritas Sertifikat (CA) bersertifikat yang dapat menandatangani sertifikat untuk domain aplikasi. Penerapan SSL *Pinning* dalam pengembangan aplikasi bertujuan untuk menghindari adanya serangan *man-in-the-middle*. Pada Gambar 9 dan 10 menampilkan hasil *log* teknik Bypass SSL *Pining* yang dilakukan. Bypass SSL *Pinning* bertujuan untuk memonitoring setiap aktivitas yang dilakukan antara aplikasi dengan server.

b) Hardcode Secret

Harcode Secret merupakan hal yang berkaitan dengan enkripsi terhadap data yang disimpan oleh aplikasi. Data dapat berupa *password*, *key*, dan kredensial dari aplikasi yang tersimpan pada *local database*.

c) Stored XSS

Cross Side-Scripting (XSS) merupakan teknik dalam menyisipkan *script* HTML atau *Javascript*. Dengan memanfaatkan kerentanan tersebut, dapat dilakukan pencurian data seperti *cookie* yang berujung pada pengambilalihan akun. Celah keamanan XSS ditandai dengan dieksekusinya *script* yang disisipkan seperti pada Gambar 11.

d) Database Breach

Database Breach merupakan kerentanan terhadap penyimpanan data yang tidak tepat, sehingga menyebabkan data tersebut dapat diakses secara bebas.

e) Root Detection

Root Detection merupakan rangkaian teknik yang digunakan untuk mendeteksi apakah perangkat yang menjalankan aplikasi sudah dalam keadaan *root* atau tidak. *Root* sendiri mempunyai pengertian sebagai *root access*. Proses yang memungkinkan aplikasi berjalan sebagai *level user* tertinggi, dalam istilah sistem operasi Windows disebut *Administrator*.

V. KESIMPULAN DAN SARAN

Berdasarkan hasil yang didapat dari implementasi DAST pada aplikasi android K24 dan Dive yang dipaparkan pada Tabel 4 dan Tabel 5 dapat disimpulkan bahwa DAST dapat memberikan informasi celah keamanan atau *vulnerability* yang tidak terdapat pada SAST. DAST mampu melakukan pengujian secara otomatis pada setiap *activity* yang ada dibuktikan dengan hasil kerentanan pada *activity Insecure Data Storage* yang memiliki celah XSS dan *Database Breach*.

Adapun saran yang dapat dilakukan pada penelitian selanjutnya terkait DAST, yaitu diperlukan perbandingan aplikasi pengujian secara DAST, untuk mendapatkan efisiensi dan efektivitas dalam melakukan pengujian keamanan aplikasi android.

REFERENCES

[1] Statcounter, “Mobile Operating System Market Share Worldwide,” *Statcounter*, 2020. https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201909-202009 (accessed Nov. 10, 2020).

[2] I. P. W. A. A. Putra, O. D. Nurhayati, “Perencanaan dan Implementasi Information Security Management System Menggunakan Framework ISO/IEC 20071,” vol. 4, no. 1, pp. 60–66, 2016, doi: https://doi.org/10.14710/jtsiskom.4.1.2016.60-66.

[3] M. Imran, F. Eassa, and K. Jambi, “Dynamic Analysis for Security Testing of WEB Based Applications Using Agent Technology,” *24th Int. Conf. Softw. Eng. Data Eng. SEDE 2015*, pp. 3–9, 2015.

[4] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, “Security Testing: A Survey,” *Adv. Comput.*, vol. 101, pp. 1–51, 2016, doi: 10.1016/bs.adcom.2015.11.003.

[5] B. Filkins, “Testing Web Apps with Dynamic Scanning in Development and Operations,” 2019.

[6] C. Hanifurohman and D. Hutagalung, “Analisis Statis Menggunakan Mobile Security Framework Untuk Pengujian Keamanan Aplikasi Mobile E-Commerce Berbasis Android,” pp. 22–28, 2020.

[7] J. Im, J. Yoon, and M. Jin, “Interaction platform for improving detection capability of dynamic application security testing,” *ICETE 2017 - Proc. 14th Int. Jt. Conf. E-bus. Telecommun.*, vol. 4, no. Icete, pp. 474–479, 2017, doi: 10.5220/0006437104740479.

[8] P. Faruki *et al.*, “Android security: A survey of issues, malware penetration, and defenses,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 2, pp. 998–1022, 2015, doi: 10.1109/COMST.2014.2386139.

[9] Y. Zhauniarovich, A. Philippov, O. Gadyatskaya, B. Crispo, and F. Massacci, “Towards black box testing of android apps,” *Proc. - 10th Int. Conf. Availability, Reliab. Secur. ARES 2015*, pp. 501–510, 2015, doi: 10.1109/ARES.2015.70.

[10] A. Ajim, “Mobile Security Framework (MobSF),” 2015. https://github.com/MobSF/Mobile-Security-Framework-MobSF (accessed Nov. 18, 2020).