Implementasi Dynamic Application Security Testing pada Aplikasi Berbasis Android

by Fauzan Awanda Alviansyah

Submission date: 24-Nov-2020 09:11PM (UTC+0700)
Submission ID: 1455180656
File name: Application_Security_Testing_pada_Aplikasi_Berbasis_Android.pdf (708.4K)
Word count: 2782
Character count: 18309

Implementasi *Dynamic Application Security Testing* pada Aplikasi Berbasis Android

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

Perkembangan teknologi pada saat ini telah mengalami perubahan yang sangat pesat, perkembangan yang cepat membuat proses komunikasi ikut berubah dengan signifikan. Di era teknologi saat ini *smartphone* sudah menjadi kebutuhan setiap manusia mulai dari berkomunikasi, transaksi, dan berselancar di internet. Banyak hal yang dapat dilakukan dengan *smartphone*, contohnya dalam melakukan transaksi melalui aplikasi yang sudah disediakan seperti layanan *mobile banking*, pesan makanan, dan pembelian barang secara *online* [1]. *Smartphone* dalam pengaplikasiannya mempunyai banyak fitur dan mudah dioperasikan, menjadikannya alat yang sangat bermanfaat bagi manusia.

Melalui Statcounter Global Stats salah satu website yang melakukan perhitungan berbagai jenis sistem operasi *mobile* atau *smartphone* yang digunakan di dunia, Pada rentan waktu 2018 – 2020 tercatat sebanyak 74.95% pengguna *smartphone* menggunakan Android sebagai sistem operasi yang digunakan. Di Indonesia sendiri pada rentan waktu tersebut tercatat sebanyak 92.03% menggunakan Android sebagai sistem operasi yang digunakan.

Smartphone dengan sistem operasi Android banyak digunakan saat ini dengan didukung dengan aplikasi-aplikasi pendukung kebutuhan manusia [2]. Namun dengan banyaknya pengguna pengguna Android menyebabkan aplikasi pada Android menjadi target utama oleh *Hacker* dan *Cracker* dalam melakukan hacking. Hal tersebut terjadi karena pesatnya pertumbuhan aplikasi Android saat ini. *Hacker* adalah seorang peretas yang mencoba masuk ke dalam sistem yang sudah dibuat dengan memanfaatkan celah keamanan yang ada. Sedangkan, *Cracker* adalah seorang peretas seperti *Hacker*, akan tetapi tidak hanya masuk ke dalam sistem melainkan juga melakukan berbagai tindakan seperti penghapusan data dan pencurian data, kemudian menggunakannya untuk kepentingan pribadi bahkan seringkali merugikan pihak terkait.

Dari sekian banyaknya aplikasi Android yang beredar tidak semuanya sudah menerapkan pengujian keamanan dengan baik sesuai ISO/27001, salah satu standar keamanan manajemen informasi memberikan gambaran umum dalam mengimplementasikan keamanan pada sebuah aplikasi. OWASP (*Open Source Foundation for Application Security*) salah satu organisasi yang berfokus pada keamanan aplikasi merilis setidaknya terdapat 10 jenis celah keamanan pada *mobile application* berupa *Insecure Data Storage*, *Improper Platform Usage*, *Insecure Authentication*, *Insufficient Cryptography,* dan lain sebagainya [3]. Beberapa dari celah keamanan tersebut berjalan di layer application pada aplikasi Android. Pesatnya perkembangan teknologi diiringi dengan banyaknya pengembangan aplikasi Android selalu menghadirkan resiko keamanan baru. Sistem operasi Android bisa dibilang lebih aman daripada sistem operasi desktop Windows, Linux, atau Mac OS, namun masalah masih dapat muncul jika tidak mempertimbangkan keamanan dengan cermat selama pengembangan aplikasi Android.

Tingginya ancaman terhadap aplikasi Android seiring dengan bertambahnya jumlah pengguna sistem operasi Android membuat banyak pengembang membuat alat pengujian keamanan baik statis maupun dinamis. *Mobile Security Framework (MobSF)* sebagai salah satu alternatif pengujian aplikasi Android secara dinamis. *Mobile Security Framework (MobSF)* merupakan *framework* pengujian keamanan *mobile application* bersifat *open-source*.

II. DASAR TEORI

A. *Keamanan Informasi*

Keamanan Informasi merupakan kegiatan dalam rangka untuk mengamankan informasi baik berupa data maupun infrastruktur. Fokus dari keamanan informasi berupa upaya dalam mengamankan semua data dari tindakan yang berbahaya atau tidak sah. Pada tingkat individu atau personal, keamanan informasi berupa melindungi data pribadi berupa identitas diri, aktivitas, dan perangkat yang digunakan. Pada tingkat yang lebih tinggi seperti perusahaan, keamanan informasi berupa tanggung jawab dalam menjaga informasi perusahaan, reputasi, dan pengguna [4]. Beberapa aspek keamanan yang berkaitan dengan keamanan informasi yaitu: *Confidentiality, Integrity, dan Availability (CIA)*.

B. *Pengujian Keamanan Aplikasi (Application Security Testing)*

Pengujian Keamanan Aplikasi (*Application Security Testing*) adalah teknik pengujian terhadap aspek keamanan informasi *Confidentiality, Integrity,* dan *Availability (CIA)*. Pengujian keamanan menentukan apakah aspek keamanan informasi sudah diterapkan dengan benar. Pengujian juga dapat dilihat dari proses mengevaluasi keamanan sistem atau jaringan komputer dengan memvalidasi dan memverifikasi keefektifan keamanan aplikasi yang diuji. Secara intuitif, pengujian keamanan mempertimbangkan kesesuaian antara masukan dan luaran yang didapatkan saat melakukan uji. Pengujian tersebut bertujuan untuk mengetahui apakah tidak terjadi kesalahan antara masukan dan luaran yang dihasilkan oleh sistem. Persyaratan keamanan menjadi positif dan fungsional saat fungsionalitas keamanan yang diharapkan dari mekanisme keamanan sudah diterapkan, sebaliknya apabila mekanisme keamanan belum diterapkan akan menjadikan persyaratan menjadi negatif dan *non* fungsional [5].

C. *Android*

Android merupakan sistem operasi berbasi Linux yang dirancang untuk perangkat mobile atau *smartphone*. Sistem operasi Android pertama kali diluncurkan pada tahun 2005, kemudian dibeli oleh Google, sampai saat ini Android sudah melakukan pembaharuan sampai pada versi 11 dengan codename Red Velvet Cake. Android termasuk perangkat lunak bersifat perangkat lunak sumber terbuka (*open source*), sehingga membuat banyak orang berkontribusi dalam mengembangkan aplikasi-aplikasi berbasis Android. Android menggunakan APK (Application Package File) sebagai berkas atau file format yang digunakan untuk mendistribusikan dan memasang perangkat lunak (software).

D. *Android Package Kit (APK)*

APK atau *Android Package Kit* adalah paket aplikasi pada sistem operasi Android. APK merupakan *format file* yang digunakan Android dalam untuk mendistribusikan dan menginstall aplikasi. Seperti format file zip atau rar, apabila APK dilakukan ekstraksi data maka akan terdapat beberapa file. Beberapa file merupakan struktur dari aplikasi Android. Dalam melakukan ekstraksi data pada APK disebut dengan decompile dan compile.

E. *Dynamic Application Security Testing (DAST)*

Proses pengujian *Dynamic Application Security Testing (DAST)* adalah proses pengujian aplikasi secara real-time atau dalam keadaan 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 memasuki runtime, setelah fase awal pengembangan (*development*). Pada penerapannya DAST menggunakan sudut pandang Black Box Testing yaitu merupakan wilayah pengujian yang dilakukan tanpa menggunakan akun ataupun tanpa memiliki akses. Dalam pengujian dengan DAST tidak diperlukan akses ke sumber kode, melainkan hanya membutuhkan APK yang akan diuji.

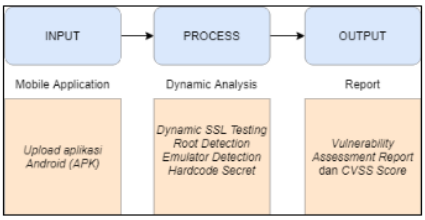
F. *Mobile Security Framework (MobSF)*

Mobile Security Framework (MobSF) adalah sebuah *framework* pengujian keamanan *mobile application* bersifat *open-source*. MobSF memberikan hasil berupa *vulnerability* assessment atau dokumen yang berisikan informasi celah keamanan berdasarkan metode pengujian yang dilakukan melalui MobSF. Metode yang dapat dilakukan *MobSF* diantaranya adalah *dynamic analysis* dan *static analysis*. MobSF menggunakan python sebagai bahasa pemrograman yang digunakan. Python mempunyai banyak *library* sehingga membuat python dapat mendukung pengembangan MobSF [6].

III. METODE ANALISIS

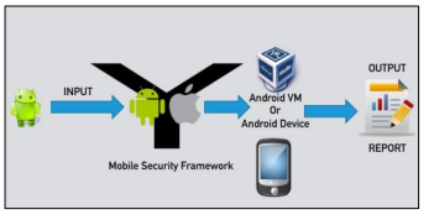
Dalam makalah ini metode yang digunakan dalam melakukan pengujian keamanan pada aplikasi Android adalah metode pengujian DAST yang menggunakan *Black Box Testing* sebagai sudut pandang pengujian. *Black Box Testing* dilakukan tanpa menggunakan akun ataupun tanpa memiliki akses pada sistem. Di dalam pengujian ini, penguji tidak akan memperoleh informasi apapun kecuali target berupa domain, IP *address*, atau aplikasi [7]. *Tools* yang dapat digunakan dalam implementasi DAST pada makalah

ini adalah MobSF yang merupakan aplikasi *open-source* yang dikembangkan oleh Ajin Abraham.



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 proses MobSF akan melakukan pengujian terhadap aspek-aspek yang terdapat dalam *dynamic analysis*. Dari hasil proses pengujian aspek-aspek pada *dynamic analysis* akan menghasilkan *vulnerability assessment report*.



Gambar 2 Gambaran Sistem MobSF

Pada Gambar 2 menjabarkan alur dari *Input* sampai dengan *Output* saat melakukan *security testing* menggunakan MobSF. Tahapan pengujian aplikasi Android. Adapun penjelasannya 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, *dynamic analysis* dilakukan 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 aplikasi Android (APK).
- Aplikasi android (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 minimum dalam menjalankan aplikasi MobSF.

IV. IMPLEMENTASI DAN HASIL ANALISIS

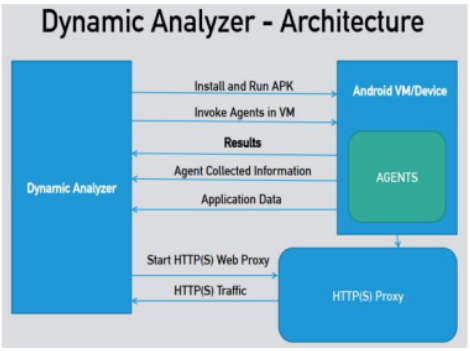
Sesuai jenis *security testing* yang digunakan yaitu *Dynamic Application Security Testing (DAST)* menggunakan *Mobile Framework Security Testing (MobSF)* pada aplikasi Android. Adapun, langkah yang harus dilakukan dalam implementasinya sebagai berikut:

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 API (*Application Programing Interface*).

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 AHM Mobile dan Apotik K24.

TABEL 2 SAMPEL APLIKASI ANDROID

No	Sampel	Versi	Keterangan
1	K24	4.01.0	Aplikasi penyedia obat-obatan
2	AHM Mobile	5.3.5	Aplikasi absensi karyawan

Pengujian bertujuan untuk melakukan *security testing* guna memastikan pengujian dengan DAST menggunakan MobSF berjalan dengan lancar dan dapat menghasilkan *Vulnerability Assessment*.

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 *dynamic analysis* dilakukan dengan menggunakan menjalankan aplikasi pada *virtual machine* yang berisikan sistem operasi Android. Adapun, tahapan yang dilakukan dalam melakukan pengujian secara *dynamic analysis* 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 *static analysis*.

Gambar 5 Informasi Aplikasi K24

Gambar 6 Informasi Aplikasi K24

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

TABEL 4 HASIL STATIC ANALYSIS

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

Tabel 4 menampilan hasil rangkuman dari aspek keamanan yang sudah diterapkan oleh kedua sampel aplikasi. Hasil rangkuman tersebut dilakukan secara *static analysis* oleh MobSF. Kedua aplikasi sudah menerapkan teknik keamanan berupa *hard coded secret* dan *anti-VM Code*. Sedangkan *certificate* berupa SSL hanya diterapkan oleh aplikasi K24. Hasil pengujian secara *static analysis* bersifat *false positive*. Hasil tersebut harus dilakukan validasi dengan *dynamic analysis* untuk mengetahui apakah kedua sampel aplikasi sudah menerapkan aspek keamanan tersebut.

b) Menjalankan Virtual Machine

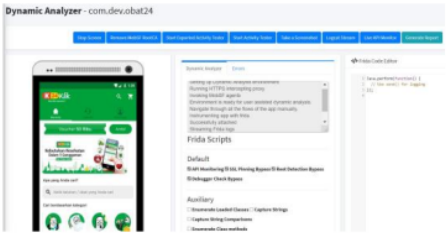
Dalam menjalankan pengujian secara *dynamic analysis* perlu dilakukan instalasi aplikasi pada *virtual machine*. Pada makalah ini *virtual machine* yang digunakan adalah Genymotion.

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

Pada salah satu menu terdapat pilihan untuk melakukan *dynamic analysis*. Pada Gambar 8 menggambarkan sudah menjalankan *dynamic analysis* dengan ditandainya dibukanya aplikasi pada Genymotion.



Gambar 8 *Dynamic Analysis* pada MobSF

Dalam implementasi DAST dalam pengujian keamanan aplikasi Android yang dilakukan yaitu mencoba seluruh fitur atau *Activity* yang ada pada aplikasi tersebut. MobSF akan secara otomatis merekam seluruh aktivitas yang dilakukan. Seluruh aktivitas yang direkam tersebut nantinya akan dilakukan analisa oleh MobSF yang pada akhirnya akan memberikan *vulnerability assessment*.

3) Hasil

Setelah berhasil mengakses seluruh fitur atau menu yang terdapat pada aplikasi, kemudian pilih ***Generate Report*** agar seluruh aktivitas yang sudah terekam tadi dianalisis oleh MobSF dan dilakukan *vulnerability assessment*. Berdasarkan *dynamic analysis* yang dilakukan oleh MobSF, didapatkan hasil seperti pada Tabel 4.

TABEL 5 HASIL DYNAMIC ANALYSIS

App	**SSL Pinning**	**Obfuscation**	**Root Detection**
K24	No	No	No
AHM Mobile	Yes	No	No

Pada tabel 4 disebutkan bahwa kedua aplikasi memiliki hasil dynamic analyst yang berbeda. Pada aplikasi K24 terdapat SSL Pinning namun tidak menerapkan obfuscation dan root detection. Sebaliknya, aplikasi AHM Mobile hanya menerapkan root detection. 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*.

b) Obfuscation

Obfuscation merupakan metode yang digunakan dalam melakukan kriptografi. *Obfuscation* membuat aturan untuk mengubah pesan agar tidak dapat dimengerti oleh manusia, akan tetapi dapat dimengerti oleh sistem. Hal tersebut dilakukan untuk meningkatkan aspek keamanan pada aplikasi.

c) 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

Berdasarkan pengujian kerentanan pada kedua aplikasi yakni K24 dan AHM Mobile memberikan hasil yang berbeda. Hal ini disebabkan oleh perbedaan waktu seiring dengan update yang secara rutin yang dilakukan oleh pengembang aplikasi. Pengujian kerentanan yang dilakukan pada aplikasi K24 dan AHM Mobile menunjukkan hasil dimana terdapat beberapa celah keamanan pada kedua aplikasi tersebut dengan tingkat keamanan yang relatif sama. Hal ini merupakan informasi penting bagi para pengguna Android, dapat menjadi *security awareness* untuk mengetahui celah-celah tersebut serta hasil analisis ini dapat dijadikan catatan keamanan yang harus diperbaiki.

REFERENCES

[1] D. Timbowo, "Manfaat Penggunaan Smartphone Sebagai Media Komunikasi (Studi pada Mahasiswa Jurusan Ilmu Komunikasi Fakultas Ilmu Sosial dan Politik Universitas Sam Ratulangi)," *e-journal "Acta Diurna,"* vol. V, no. 2, pp. 1–13, 2016.

[2] A. Kartono, A. Sularsa, and S. J. I. Ismail, "Membangun Sistem Pengujian Keamanan Aplikasi Android Menggunakan Mobsf," vol. 5, no. 1, pp. 146–151, 2019.

[3] OWASP, "OWASP Mobile Security Testing Guide - 1.1.3 Release," p. 535, 2019.

[4] Cisco, "Introduction to Cybersecurity," 2020. http://static-course-assets.s3.amazonaws.com/CyberSec2.1/id/index.html#1.0.1.1 (accessed Nov. 10, 2020).

[5] 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.

[6] C. Hanifurohman and D. Hutagalung, "Analisis Statis Menggunakan Mobile Security Framework Untuk Pengujian Keamanan Aplikasi Mobile E-Commerce Berbasis Android," pp. 22–28, 2014.

[7] 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.

Implementasi Dynamic Application Security Testing pada Aplikasi Berbasis Android

ORIGINALITY REPORT

4%	4%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.ar-raniry.ac.id Internet Source	1%
2	123dok.com Internet Source	<1%
3	repository.uksw.edu Internet Source	<1%
4	mobsf.github.io Internet Source	<1%
5	en.wikipedia.org Internet Source	<1%
6	hafidadhi.wordpress.com Internet Source	<1%
7	appkey.id Internet Source	<1%
8	jtsiskom.undip.ac.id Internet Source	<1%
9	nurmibatu.blogspot.com	

	Internet Source	<1%
10	moam.info Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		