

LAPORAN
PRAKTIK KERJA LAPANGAN



PENGUJIAN KERENTANAN PADA DEVELOPMENT
REST API DALAM CV. SOLUSI AUTOMASI INDONESIA

MUHAMMAD NUR IRSYAD

1807422020

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER

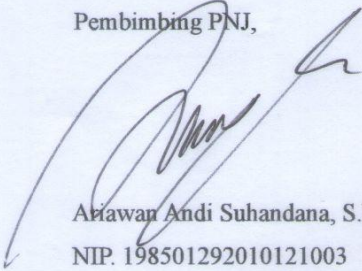
DEPOK

2022

HALAMAN PENGESAHAN
LAPORAN PRAKTIK KERJA LAPANGAN

- a. Judul : Pengujian Kerentanan pada *Development*
REST API dalam CV. Solusi Automasi Indonesia
- b. Penyusun
- 1) Nama : Muhammad Nur Irsyad
- 2) NIM : 1807422020
- c. Program Studi : Teknik Multimedia dan Jaringan
- d. Jurusan : Teknik Informatika dan Komputer
- e. Waktu Pelaksanaan : 2 September 2021 s.d. 2 Desember 2021
- f. Tempat Pelaksanaan : CV. Solusi Automasi Indonesia
Jl. Telekomunikasi Terusan Buah Batu, Bandung
Techno Park, kawasan Pendidikan Telkom, 40257,
Dayeuhkolot, Bandung, Jawa Barat

Pembimbing PNJ,

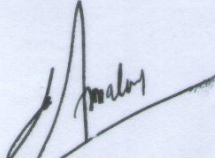

Anawan Andi Suhandana, S.Kom., M.T.I.
NIP. 198501292010121003

Depok, 10 Januari 2022
Pembimbing Perusahaan,


Automate All
CV. Solusi Automasi Indonesia

Irfan Nugraha, S.Kom.
NIP. 2010000120006

Mengesahkan,
KPS Teknik Multimedia dan Jaringan


Defiana Arnaldy, S.Tp., M.Si.
NIP. 198112012015041001

ABSTRAK

Penetration testing merupakan kegiatan yang komprehensif untuk melakukan pengujian yang lengkap, terintegrasi, serta berdasar, yang tidak lain ditujukan untuk mengevaluasi sisi keamanan infrastruktur aplikasi sehingga *user* dapat mengetahui potensi kerentanan lebih awal untuk melakukan mitigasi yang lebih cepat dan akurat. Dalam CV. Solusi Automasi Indonesia, adapun aplikasi REST API yang sedang dikembangkan oleh rekan magang divisi *Backend Developer* yang secara bersamaan diujikan kerentanannya oleh divisi *Software Security Developer*, untuk mengukur secara dini bagaimana potensi kerentanan yang dapat dimanfaatkan oleh penyerang. Bentuk paralel ini ditujukan agar memaksimalkan output dan evaluasi yang dikerjakan dengan waktu yang terbatas.

Tujuan dari laporan adalah untuk menjelaskan bagaimana pendekatan yang digunakan selama kegiatan berlangsung, tanpa adanya intensi dalam memberikan informasi internal industri kepada publik. Untuk memastikan keseluruhan kegiatan jalan secara struktural dan objektif, maka adapun penggunaan standar PTES yang diterapkan pada kegiatan *penetration testing* ini. Dengan adanya landasan yang jelas, maka *output* dari kegiatan pun tetap dapat bersifat informatif walaupun mengarah kepada sisi teknis.

Kerentanan yang didapatkan dilakukan pemodelan serangan terlebih dahulu agar *scope* nya tidak melenceng. Adapun kerentanan yang berhasil maupun tidak berhasil dilakukan eksploitasi, tetap mendapatkan rekomendasi mitigasi yang sesuai dengan teknologi yang digunakan, sehingga diharapkan dapat membantu secara spesifik untuk meningkatkan keamanan tersebut baik dari *scope* aplikasi maupun *server*.

Kata kunci: *Penetration Testing*, REST API, PTES

KATA PENGANTAR

Puji dan Syukur penulis panjatkan atas kehadiran Allah Subhanahu Wa Ta'ala. yang telah memberikan rahmat dan petunjuk-Nya sehingga penulis dapat menyelesaikan laporan Praktik Kerja Lapangan ini serta terlaksana dengan baik. Tidak lupa shalawat dan salam semoga tercurahkan kepada Rasulullah Shalallahu 'Alaihi Wassalam. Penulisan laporan Praktik Kerja Lapangan ini dilakukan dalam rangka memenuhi salah satu syarat untuk kelulusan dalam mencapai gelar Diploma Empat (D4) Politeknik Negeri Jakarta. Menyadari bahwa banyaknya bantuan serta bimbingan dalam proses penyusunan laporan ini, oleh karena itu, penulis mengucapkan terima kasih serta rasa kepada:

- Ariawan Andi Suhandana, S.Kom., M.T.I., selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan penulis dalam menyusun laporan Praktik Kerja Lapangan ini
- Pihak CV. Solusi Automasi Indonesia, yang telah memberikan kesempatan baik untuk waktu dan tempat untuk membuka peluang Praktik Kerja Lapangan dalam Batch 5 ini
- Irfan Nugraha, S.Kom., selaku pembimbing dari pihak instansi, yang telah mengarahkan penulis dalam melaksanakan tugas serta proses dokumentasi laporan ini
- Orang tua dan keluarga penulis yang memberikan dukungan moral serta material dalam keseluruhan rangkaian proses Praktik Kerja Lapangan
- Teman-teman kampus yang telah banyak membantu penulis dalam menyediakan medium serta fasilitas yang baik hingga memungkinkan penulis dapat memulai program serta menyelesaikan laporan ini

Menyadari akan kekurangan dalam proposal ini, penulis akan sangat menghargai untuk saran dan kritik yang sifatnya membangun. Akhir kata, semoga laporan ini dapat bermanfaat untuk pengembangan ilmu baik untuk penulis dan pembaca.

Jakarta, 14 Januari 2022

Penulis

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN	ii
ABSTRAK	iii
KATA PENGANTAR	iv
DAFTAR ISI	v
DAFTAR GAMBAR	vi
DAFTAR TABEL	vii
DAFTAR LAMPIRAN	viii
BAB I PENDAHULUAN	1
1.1 Latar Belakang Kegiatan	1
1.2 Ruang Lingkup Kegiatan	2
1.3 Waktu dan Tempat Pelaksanaan	3
1.4 Tujuan dan Kegunaan	3
1.4.1 Tujuan	3
1.4.2 Kegunaan	3
BAB II TINJAUAN PUSTAKA	4
2.1 Penetration Testing	4
2.1.1 Penetration Testing Execution Standard	4
2.1.2 Common Weakness Enumeration	5
2.1.3 Common Vulnerability Scoring System	6
2.1.4 Attack Tree	9
2.2 Application Programming Interface	11
2.3 Metodologi Scrum	12
BAB III HASIL PELAKSANAAN PKL	13
3.1 Unit Kerja Praktik Kerja Lapangan	13
3.1.1 Struktur Organisasi	14
3.1.2 Divisi Software Security Developer	14
3.2 Uraian Praktik Kerja Lapangan	15

3.3	Pembahasan Hasil Praktik Kerja Lapangan	18
3.3.1	Kondisi Sekarang	19
3.3.1	Kondisi yang Diharapkan	19
3.3.2	Tahapan Pengujian Kerentanan	20
3.3.2.1	Pre-Engagement	20
3.3.2.2	Intelligence Gathering	21
3.3.2.3	Threat Modelling	24
3.3.2.4	Vulnerability Analysis	26
3.3.2.5	Exploitation	32
3.3.2.6	Post Exploitation	41
3.3.3.7	Reporting	41
3.3.4	Tools Pendukung Pengujian Kerentanan	46
3.4	Identifikasi Kendala yang Dihadapi	48
3.4.1	Kendala Pelaksanaan Tugas	48
3.4.2	Cara Mengatasi Kendala	49
BAB IV PENUTUP		50
4.1	Kesimpulan	50
4.2	Saran	50
DAFTAR ISTILAH		51
DAFTAR PUSTAKA		52
LAMPIRAN		55

DAFTAR GAMBAR

Gambar 2.1 Relasi CWE dengan CVE	6
Gambar 2.2 Metric Kalkulasi CVSS 3.1 untuk Base Score	7
Gambar 2.3 Mekanisme REST API	11
Gambar 3.1 Logo CV. Solusi Automasi Indonesia	13
Gambar 3.2 Struktur Organisasi CV. Solusi Automasi Indonesia	14
Gambar 3.3 Diagram Attack Tree	25
Gambar 3.4 Command Usage pada port-sweeper	47
Gambar 3.5 Demonstrasi penggunaan port-sweeper	47
Gambar 3.6 Fitur auto-installer dalam port-sweeper	48

DAFTAR TABEL

Tabel 2.1 Deskripsi simbol dalam diagram Attack Tree	10
Tabel 3.1 Deskripsi Pre-Engagement	20
Tabel 3.2 Metode Intelligence Gathering	22
Tabel 3.3 Hasil Informasi Intelligence Gathering	23
Tabel 3.4 CVSS pada XSS Injection (Sensitive Data Exposure)	29
Tabel 3.5 CVSS pada System Misconfig (Sensitive Data Exposure)	30
Tabel 3.6 CVSS pada Unrestricted Resource (Broken Access Control & Sensitive Data Exposure)	31
Tabel 3.7 CVSS pada Improper Error Handling (Sensitive Data Exposure)	32
Tabel 3.8 Hasil eksploitasi dalam Unrestricted Public Repository	38
Tabel 3.9 Hasil eksploitasi dalam Improper Error Handling	39

DAFTAR LAMPIRAN

L-1 Sertifikat Keterangan Selesai PKL	55
L-2 Buku Penghubung Industri	56
L-3 User Requirement Industri	59
L-4 Lampiran Dokumentasi	60

BAB I

PENDAHULUAN

1.1 Latar Belakang Kegiatan

CV. Solusi Automasi Indonesia atau yang dikenal juga sebagai Automate All, merupakan sebuah *startup* yang berfokuskan pada bidang *Robotic Process Automation* atau RPA, yang sudah berdiri sejak tahun 17 Oktober 2020. Automate All sendiri berbasiskan di Techno Park, Bandung, Jawa Barat; yang juga dibangun oleh para 8 alumni dari Universitas Telkom di Bandung.

Dalam bisnisnya, produk dan layanan yang Automate All tawarkan dapat berjalan dalam berbagai macam sektor bisnis, seperti *Healthcare* (*inventory management, invoice settlement, update patient records*), *IT* (*ticket request handling, server monitoring, user-access management*), *Education* (*attendance management, class scheduling, auto-generate report and marksheet*), dan *Finance* (*invoice processing, auto-generate finance report, account closure*).

Dalam periode ini, Automate All telah bergerak untuk melakukan pengembangan terhadap salah satu aplikasi web utamanya yang dijadikan sebagai landing page industri. Web yang diisukan sebagai v2 ini kerap dikembangkan kembali oleh 2 divisi *intern*, yaitu *Frontend Developer* dan *Backend Developer*. Dalam kasusnya, aplikasi yang dibangun oleh divisi *Backend Developer*, yaitu REST API, membutuhkan evaluasi keamanan terhadap bagaimana informasi dan resource yang tersimpan tetap aman dan hanya dapat diakses dan dikelola dari pihak internal industri. Kondisi ini yang kemudian menugaskan divisi *Software Security Developer* untuk melakukan kegiatan pengujian kerentanan terhadap aplikasi tersebut, yang diharapkan dapat mengidentifikasi berbagai macam kerentanan yang memungkinkan digunakan sebagai *entry point* oleh *hacker* dalam melakukan serangan siber. *Output* yang ditujukan kemudian adalah seperti apa remediasi dan mitigasi yang sesuai dengan kerentanan, serta teknologi yang aplikasi gunakan. *Output* nanti dikemas dalam bentuk suatu *report* yang dikirimkan kepada kepala divisi, untuk nantinya didiskusikan dengan rekan divisi *Backend Developer*.

Hal ini ditujukan untuk adanya implementasi dan evaluasi terhadap hal-hal yang didapatkan dalam *report* tersebut. Keseluruhan rangkaian kegiatan ini sendiri didasarkan dengan kode etik dan protokol yang sudah ditetapkan oleh industri, dan disetujui oleh kedua belah pihak.

Disini, judul yang diangkat adalah “Pengujian Kerentanan pada Development REST API dalam CV. Solusi Automasi Indonesia”, yang merupakan satu dari dua kegiatan yang ditugaskan selama masa PKL ini berlangsung. REST API yang digunakan sebagai target pengujian merupakan milik internal Automate All yang berada dalam fase *development*. Hal dan ilmu yang dipelajari selama kegiatan berlangsung memotivasi untuk dapat memaparkan tahapan dan teknik yang digunakan saat *penetration testing* ke dalam laporan yang lebih terstruktur.

1.2 Ruang Lingkup Kegiatan

Program PKL yang disediakan oleh CV. Solusi Automasi Indonesia memiliki lebih dari 15 divisi dalam 3 departemen *internship*, yaitu IT, *Business*, dan *Marketing*. Dalam *batch* 5 ini, posisi yang dilamar adalah pada departemen IT dalam divisi *Software Security Developer Intern*. Seluruh tugas yang dipaparkan, dirancang menggunakan project management dengan metodologi Scrum. Selama menjalankan prosesi PKL, adapun didapatkan tugas yang berupa bersifat teknis dan non-teknis. Pada yang sifatnya teknis, tugas ditujukan untuk melakukan dan mengevaluasi pengujian kerentanan terhadap target yang sudah ditentukan, yaitu REST API internal dari pihak perusahaan. Pada tugas yang sifatnya non-teknis, tugas ditujukan untuk melakukan riset mengenai implementasi aspek keamanan dalam *workflow* RPA pada bagian robot dengan menggunakan *tools* UiPath. Dalam laporan ini, pembahasan yang lebih ditekankan adalah kepada tugas yang sifatnya teknis, yang nantinya dijelaskan pada bab 3 mengenai hasil pelaksanaan PKL.

1.3 Waktu dan Tempat Pelaksanaan

Waktu dan tempat pelaksanaan PKL di CV. Solusi Automasi Indonesia dijabarkan sebagai berikut:

- a. Waktu : 2 September 2021 s.d. 2 Desember 2021
- b. Perusahaan : CV. Solusi Automasi Indonesia
- c. Alamat : Jl. Telekomunikasi Terusan Buah Batu, Bandung
Techno Park, kawasan Pendidikan Telkom, 40257,
Dayeuhkolot, Bandung, Jawa Barat
- d. Kondisi Kerja : Dirumahkan / WFH
- e. Jam Kerja : 09.00 WIB s.d 17.00 WIB (5 hari kerja)

1.4 Tujuan dan Kegunaan

Tujuan serta manfaat implementasinya dari PKL di CV. Solusi Automasi Indonesia dalam divisi *Software Security Developer Intern* sebagai berikut:

1.4.1 Tujuan

Berikut tujuan dari pengujian kerentanan *development* REST API:

- a. Mendemonstrasikan pengujian kerentanan pada *development* REST API yang berbasiskan pada panduan PTES
- b. Menghasilkan laporan evaluasi pengujian yang nantinya digunakan untuk didiskusikan kepada divisi *Backend Developer Intern*

1.4.2 Kegunaan

Berikut manfaat dari pengujian kerentanan *development* REST API:

- a. Menambahkan referensi evaluasi mengenai tingkat serta jumlah kerentanan pada layanan *development* REST API

BAB II

TINJAUAN PUSTAKA

2.1 Penetration Testing

Dalam upaya untuk mengevaluasi sisi keamanan infrastruktur produk, pihak perusahaan dapat memanfaatkan jasa *white hat hacker* dalam melakukan *penetration testing* untuk mengetahui potensi kerentanan lebih awal daripada *cyber attacker* diluar sana. *Penetration testing* sendiri merupakan kegiatan yang komprehensif untuk melakukan pengujian yang lengkap, terintegrasi, serta berdasar. Kegiatan ini dapat terdiri dari layer *software*, *hardware*, hingga *human resource* nya. Hal-hal yang *discover* diantaranya adalah analisa adanya potensi kerentanan, konfigurasi yang salah / tidak baik, ataupun kegiatan operasional yang dinilai terlalu lemah untuk praktis keamanan-nya (Bacudio et al. 2011). Kegiatan *penetration testing* disarankan untuk dilakukan dalam periode tertentu, seperti saat pengeluaran suatu release, sehingga meningkatkan *awareness* kalau produk tersebut ter-cover dari kerentanan yang baru (Fachri, Fadlil & Riadi 2021).

2.1.1 Penetration Testing Execution Standard

Adapun suatu standard yang dijadikan sebagai pemandu untuk melakukan kegiatan pentest, yaitu PTES (*Penetration Testing Execution Standard*), yang ditujukan untuk memandu *pentester* dalam memenuhi tujuan secara prosedural. PTES sendiri terdiri dari tujuh tahapan sebagai berikut:

- a. *Pre-engagement*: menentukan tujuan kegiatan *pentest*, lingkup kegiatan, serta adanya kesepakatan ataupun kontrak pada dua pihak mengenai mekanisme dan *policy* selama kegiatan berlangsung
- b. *Intelligence Gathering*: mengumpulkan informasi yang berkaitan dengan target, baik secara aktif maupun pasif. Kualitas dan kuantitas informasi mempengaruhi output dari kegiatan *pentest*
- c. *Threat Modelling*: menggambarkan seberapa besar pengaruh potensi ancaman terhadap aset didalamnya. Ancaman tersebut dapat diprioritaskan dengan referensi *base score* CVSS

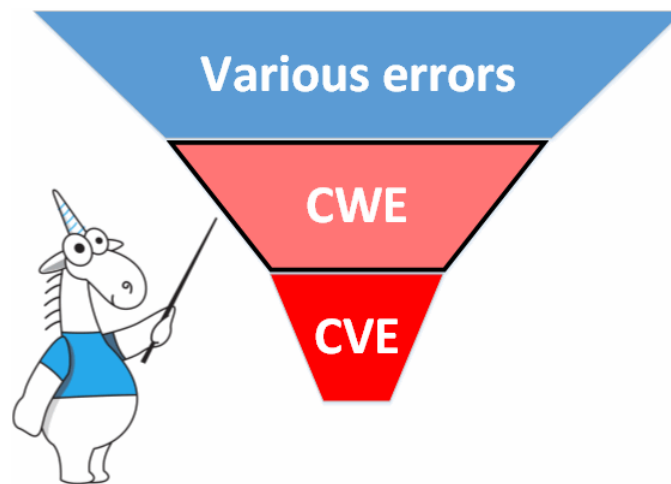
- d. *Vulnerability Analysis*: menganalisa potensi celah kerentanan pada target. *List* kerentanan yang didapatkan bisa didasarkan kepada referensi CVE & CWE untuk mendapatkan informasi rincinya
- e. *Exploitation*: melakukan pengujian target yang berdasarkan *modelling* dan kerentanan yang sudah didapat. Kedua hal tersebut menjadi *entry point* utama dalam melakukan pengujian
- f. *Post-Exploitation*: meningkatkan pengujian ketika berhasil masuk kedalam sistem target; mengukur seberapa jauh target dapat di-*exploit* hingga berpotensi mengganggu aset yang tidak diduga
- g. *Reporting*: mendokumentasi kegiatan *pentest* secara terstruktur, yang *mengcover* seluruh tahapan yang dilakukan, kerentanan yang ditemukan, serta mitigasi dari potensi serangan ke depan (Sunaringtyas & Prayoga 2021)

2.1.2 Common Weakness Enumeration

Dalam memahami konteks mengenai berbagai macam kerentanan yang tersedia beserta dengan deskripsinya, adapun referensi standar yang digunakan oleh para praktisi profesional, yaitu CVE (*Common Vulnerabilities and Exposure*). CVE menyediakan ratusan ribu *list* atau *record*, mengenai kerentanan umum yang ditemukan secara global (Anonym 2020). *Record* tersebut di-*publish* dan di-*maintain* oleh perusahaan di dunia yang telah menjadi *partner* dalam CVE *program*, dengan tujuan untuk memudahkan pertukaran informasi kerentanan secara global terhadap suatu spesifik *instance*, produk ataupun sistem (Anonym 2021).

Selain CVE, istilah yang umum didengar selanjutnya adalah CWE (*Common Weakness Enumeration*). Berbeda dengan CVE, CWE menyediakan *list* klaisifikasi dari seluruh macam kerentanan, yang tidak terikat terhadap suatu teknologi / sistem tertentu, yang dapat dilihat pada gambar 2.1. *List* tersebut menampung berbagai jenis *bug*, *flaws*,

kelemahan dalam implementasi *coding*, *design*, arsitektur, serta *networking* yang dapat memicu untuk rentan nya diserang oleh *cyber attacker*. Karena pendekatannya bersifat klasifikasi seperti struktur pohon, maka setiap *record* dapat memiliki *parent* ataupun *sub record* yang saling terikat satu sama lain (Anonym. 2021).



Gambar 2.1 Relasi CWE dengan CVE
(Sumber: <https://pvs-studio.com/en/blog/posts/0577/>)

2.1.3 Common Vulnerability Scoring System

Dalam melakukan prioritas antara satu kerentanan dengan yang lain, maka salah satu cara sistematis yang umum digunakan adalah menggunakan referensi *scoring* dari CVSS (*Common Vulnerability Scoring System*). CVSS dikembangkan oleh FIRST (*Forum of Incident Response and Security Teams*), yang merupakan organisasi *non-profit* Amerika dengan tujuan untuk membantu pembangunan karakteristik utama dari suatu kerentanan, dengan menggunakan format *scoring* untuk menggambarkan tingkat kerusakannya. Selain menyediakan algoritma kalkulasi, FIRST juga menyediakan *tools* kalkulasi CVSS secara online baik untuk versi 2.x maupun versi 3.x dalam situs resminya. Salah satu perbedaan mendasarnya adalah adanya penambahan beberapa matrik pada kategori *exploitability*, sehingga meningkatkan keakuratan *scoring* terhadap kerentanan yang diujikan (Anonym 2021). Dalam kategori *impact*, kedua versi memiliki *metric* yang sama, kurang lebih menggambarkan mengenai

bagaimana pengaruh kerentanan tersebut terhadap *resource internal* serta bagaimana layanan nya. Berikut merupakan contoh tampilan *metric* dari kalkulasi CVSS versi 3.1:

The image shows the CVSS v3.1 Base Score Calculator interface. It consists of several sections for selecting metrics, a central display for the score and vector, and a footer with the calculator's name and copyright.

ATTACK VECTOR	ATTACK COMPLEXITY	PRIVILEGES REQUIRED	USER INTERACTION
Network	Low	None	None
Adjacent	High	Low	Required
Local		High	
Physical			

CVSS v3.1

SCOPE	CONFIDENTIALITY	INTEGRITY	AVAILABILITY
Changed	High	High	High
Unchanged	Low	Low	Low
	None	None	None

SEVERITY · SCORE · VECTOR

Medium 5.4 CVSS:3.1/AV:A/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:H

CVSS v3.1 Base Score Calculator - Copyright 2019 © Chandan

Gambar 2.2 Metric Kalkulasi CVSS 3.1 untuk Base Score
(Sumber: <https://cvssjs.github.io/>)

CVSS versi 3.0 dengan versi 3.1 tidak mengalami perubahan *metric*, namun lebih mengidentifikasi setiap *attack vector* dapat memiliki *base score* dalam konteks yang berbeda. Hal ini memberikan gambaran yang lebih fleksibel terhadap bagaimana suatu kerentanan dalam satu sistem dapat memiliki *base score* yang lebih rendah dibandingkan dalam sistem yang lain (Sharma 2020). Berikut merupakan penjelasan singkat dari setiap *metric* yang digunakan sebagai kalkulasi CVSS pada versi 3.1 yang dapat dilihat pada gambar 2.2, yaitu:

1. *Attack Vector*: pemberian konteks mengenai bagaimana pendekatan proses pengujian yang dilakukan kepada suatu target yang rentan. Penilaian naik sebagaimana serangan dapat dilakukan *se-remote*

mungkin dari target, sehingga sifatnya menjadi scalable baik itu 1 hop lebih dari target (*physical* \rightarrow *local* \rightarrow *adjacent* \rightarrow *network*).

Metric dilambangkan dengan kode **AV**

2. *Attack Complexity*: penggambaran seperti apa kondisi maupun *requirement* yang dibutuhkan *attacker* sebelum dapat melakukan pengujian kepada target. Penilaian naik sebagaimana tidak dibutuhkannya suatu kondisi yang standar untuk *attacker* dapat mengurangi serangan terus menerus dengan hasil yang diharapkan (*high* \rightarrow *low*). *Metric* dilambangkan dengan kode **AC**
3. *Privileges Required*: penggambaran seberapa tinggi *user privilege* ataupun otorisasi yang dibutuhkan untuk *attacker* dapat melakukan pengujian secara utuh. Penilaian naik sebagaimana *attacker* tidak membutuhkan otorisasi sama sekali dalam melakukan penyerangan yang berhasil secara repetitif (*high* \rightarrow *low* \rightarrow *none*). *Metric* dilambangkan dengan kode **PR**
4. *User Interaction*: pemberian konteks mengenai seberapa ketergantungannya dengan interaksi aktif dari user / client dalam menjalankan serangan. Penilaian naik sebagaimana serangan dapat dilakukan dengan *unattended-user* (*required* \rightarrow *none*). *Metric* dilambangkan dengan kode **UI**
5. *Scope*: penggambaran seberapa besar dampak dan pengaruh serangan terhadap aset / kerentanan diluar dari yang diujikan. Penilaian naik sebagaimana serangan memberikan dampak yang signifikan terhadap aset yang bukan dalam lingkupnya, yang disebut juga sebagai *collateral damage* (*unchanged* \rightarrow *changed*). *Metric* dilambangkan dengan kode **S**
6. *Confidentiality*: penggambaran seberapa besar *exposure* yang didapatkan dari kerahasiaan informasi target dari hasil serangan *attacker*. Penilaian naik sebagaimana pengujian menyebabkan *total*

loss dari *confidentiality* pada *restricted resource* internal (*none* → *low* → *high*). *Metric* dilambangkan dengan kode C


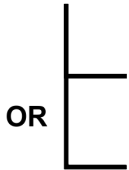

7. *Integrity*: penggambaran seberapa besar dampak dari pengujian kerentanan tersebut terhadap integritas *resource* baik yang sifatnya publik maupun internal terhadap *user* dan *client*. Penilaian naik sebagaimana pengujian menyebabkan *total loss integrity* ataupun proteksi dari *resource* tersebut (*none* → *low* → *high*). *Metric* dilambangkan dengan kode I
8. *Availability*: penggambaran seberapa besar dampak terganggunya ketersediaan *resource* dalam melayani *user* / *client* nya, yang dimana mencakup peran *bandwith*, CPU, GPU, serta *disk space*. Penilaian naik sebagaimana menyebabkan *total loss availability* mengenai akses terhadap *resource* tersebut, baik itu *sustain* ataupun *persistent* (*none* → *low* → *high*). *Metric* dilambangkan dengan kode A

2.1.4 Attack Tree

Seperti yang disebutkan dalam tahapan PTES, *threat modelling* ditunjukkan memberikan gambaran besar mengenai seperti apa potensi ancaman terhadap asset, lalu bagaimana memitigasinya. *Threat modelling* sendiri dapat digambarkan ke dalam 3 bentuk, yaitu *asset-centric* (menganalisa berbagai macam kerentanan dari setiap aset), *software-centric* (fokus terhadap konfigurasi dan keamanan transmisi data antara setiap *layer* nya), serta *attacker-centric* (menggambarkan rantai *possible attack* pada setiap kerentanan), yang salah satu contohnya adalah *Attack Tree*. *Attack tree* menggambarkan struktur serangan berdasarkan *bottom-up*, hingga menuju *goals* yang diinginkan, yang dimana juga dapat dikombinasikan dengan ukuran *metric cost* untuk setiap serangannya (Mohanakrishnan 2021).

Dalam pembuatannya, pertama diagram dapat ditentukan terlebih dahulu *root node* sebagai *overall goal* nya, sehingga memberikan *scope* yang jelas. Kemudian *goal* tersebut dipecah menjadi *subgoal*, yang dapat berbentuk sebagai *attack repository* dan berperan sebagai *branch node*. Penjabaran dapat terus dilakukan untuk membuat setiap task menjadi lebih kecil, membuatnya menjadi *leaf node* (Duan, Saini & Paruchuri 2008). Value antar *leaf node* dapat berupa OR (*independent*) ataupun AND (*dependent*) antara satu sama lain, menggambarkan kondisi yang harus terpenuhi untuk dapat mencapai *parent node* (Dzida & Wiklicky 2020). Setelah *attack tree* terbentuk, *user* dapat menganalisa bahwa setiap *node* bisa mendapatkan pendekatan keamanan nya masing-masing, sehingga mitigasi bisa dilakukan lebih terarah. Berikut merupakan deskripsi simbol-simbol dari komponen attack tree dalam gambar 2.1:

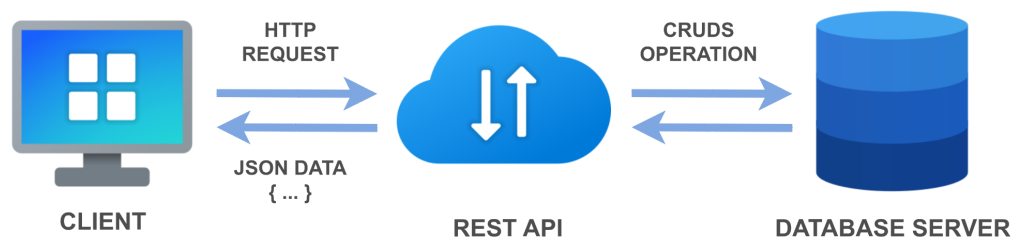
Tabel 2.1 Deskripsi simbol dalam diagram Attack Tree

Simbol	Deskripsi
Node / Objective 	Kotak yang menggambarkan sebuah node, yang berperan sebagai fungsi root, parent, serta leaf node (menggambarkan serangan & goals)
Disjunctive Refinement 	Relasi antara child nodes dengan parent node dengan keterangan OR. Hanya dibutuhkan satu child node yang sukses untuk menuju parent node
Conjunctive Refinement 	Relasi antara child nodes dengan parent node dengan keterangan AND. Dibutuhkan dua atau lebih child node yang sukses untuk menuju parent node

Met Condition —————	Suatu proses dalam node dapat dilaksanakan karena adanya kondisi yang terpenuhi yang memungkinkan
Unmet Condition - - - - -	Suatu proses dalam node belum dapat dilaksanakan karena adanya kondisi yang tidak terpenuhi yang tidak memungkinkan

2.2 Application Programming Interface

Dengan meningkatnya minat masyarakat dalam mendapatkan informasi dan layanan secara real-time, maka diharapkan teknologi dapat beradaptasi dalam melayani permintaan tersebut secara simultan. Salah satu solusi yang selalu dikembangkan untuk dipakai oleh perusahaan besar, terutama dalam bidang bisnis dan ekonomi, adalah dengan penggunaan API (*Application Programming Interface*) (Walkowski 2020). API memungkinkan untuk menyediakan *interface* antara *resource database* dengan aplikasi bisnis secara terintegrasi, sehingga mobilitas transfer data menjadi lebih efisien dan tersentralisasi (Priyatna & Hananto 2020).



Gambar 2.3 Mekanisme REST API

API sendiri hadir dalam beberapa bentuk implementasinya, salah satu yang umum digunakan adalah dalam arsitektur REST (*Representational State Transfer*). Dalam gambar 2.3, mekanisme REST menggunakan standar protokol HTTP untuk dapat berkomunikasi antara *resource database* dengan aplikasi ataupun program yang bersifat *third-party*. Fungsi dari REST API nantinya dapat diakses oleh aplikasi melalui *endpoint URL* (*Uniform Resource Locators*) yang telah

dikonfigurasi dalam server REST tersebut. Salah satu keuntungan menggunakan REST adalah transmisi datanya yang menggunakan *bandwith* yang kecil, sehingga *response* nya sangat ringan dan fleksibel untuk dilakukan *cache* (Manuaba & Rudiastini 2017).

2.3 Metodologi Scrum

Dalam dunia kerja, konsep manajemen proyek menjadi salah satu hal mendasar dalam mengerjakan tugas secara kolaboratif. Salah satu *framework* manajemen proyek yang umum digunakan adalah *Agile*, yang dimana memfokuskan dengan memberikan MVP (*Minimum Viable Product*) kepada *client* dalam periode interval yang rendah, sehingga menstabilkan performa kerja, kreativitas, serta produktivitas (Dingsøyr et al. 2012). *Agile* sendiri merupakan payung untuk beberapa metodologi yang berkembang di dalamnya, salah satunya adalah Scrum. Scrum mewarisi konsep *Agile* yang kemudian dikembangkan untuk meng-*handle* proyek yang kompleks secara *incremental* dan *iterative*, sehingga dapat terus beradaptasi dengan *business requirement* dari proyek tersebut. Karena itu, penggunaan Scrum relatif lebih menghemat biaya serta waktu, tanpa mengurangi gambaran besar proyek tersebut. Scrum sendiri dilakukan dalam beberapa tahap didalamnya yaitu:

- *Sprint Planning*: melakukan *self-assessment* terhadap tugas yang diberikan, seperti manajemen waktu pengerjaan, dan definisi goals nya, yang umumnya satu sprint dilaksanakan selama 1-2 minggu
- *Daily Scrum*: melakukan koordinasi dengan tim setiap harinya mengenai progress, perkembangan, serta ide yang dapat dikembangkan
- *Sprint Review*: melakukan review dengan tim serta seluruh *stakeholder*, sehingga dapat mendiskusikan *feedback* serta menganalisa output dan menjadikannya suatu input untuk sprint yang mendatang (Hema et al. 2020)

BAB III

HASIL PELAKSANAAN PKL

3.1 Unit Kerja Praktik Kerja Lapangan



Gambar 3.1 Logo CV. Solusi Automasi Indonesia
(Sumber: <https://entrepreneurship.telkomuniversity.ac.id/portfolio/>)

Program PKL dilakukan di CV. Solusi Automasi Indonesia pada departemen IT dalam divisi Software Security Developer Intern. Dalam menjalankan bisnisnya, Automate All memiliki visi dan misi yang melandaskan perusahaan tersebut untuk terus berkembang dan maju, detailnya sebagai berikut:

Visi: Menjadi nomor satu dalam menyediakan *tools* automasi di Indonesia dan membantu pelaku bisnis *go automate* pada setiap proses bisnis

Misi:

- Mengautomasi pekerjaan yang bersifat *repetitive task*
- Melatih sumber daya manusia mengerti RPA dan mampu bersaing
- Memudahkan pelaku bisnis di Indonesia dalam proses administrasi
- Memaksimalkan kinerja pekerja dengan menggunakan *bot*

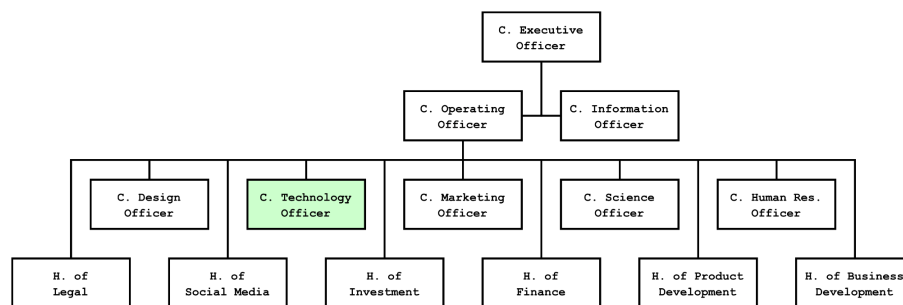
PKL (Praktik Kerja Lapangan) yang diselenggarakan oleh Automate All sudah masuk ke dalam *Batch 5*, mulai dari September hingga Desember 2021 ini. Proses ini dilakukan selama 3 bulan dengan metode *project-based learning*. Dikarenakan masa pandemi, seluruh kegiatan PKL dilakukan secara WFH (*Work From Home*) via *online*, sehingga memberikan kesempatan peserta diluar Bandung untuk dapat ikut serta dalam program PKL ini, seperti peserta yang berdomisili di Jakarta.

Adapun gambaran aktivitas secara umum yang dilakukan oleh peserta PKL selama program PKL berlangsung secara *online*, yaitu:

- Mengerjakan proyek riil yang ada di Automate All dan bertanggung jawab dalam pemecahan masalah tersebut
- Mendapat bimbingan dari mentor / pembimbing jika dibutuhkan
- Mendapatkan materi pengayaan dari perusahaan yang memperkuat *skill* dari proyek yang ditugaskan

3.1.1 Struktur Organisasi

CV. Solusi Automasi Indonesia merupakan *startup* baru yang berkembang ke arah digitalisasi dari kegiatan administrasi dan bisnis. Hal ini dapat digambarkan dari struktur organisasi yang berstruktur *horizontal*, sehingga memberikan ruang untuk inovasi dan tanggung jawab yang lebih kepada pegawainya. Adapun struktur organisasinya sebagai berikut:



Gambar 3.2 Struktur Organisasi CV. Solusi Automasi Indonesia
(Sumber: CV. Solusi Automasi Indonesia)

3.1.2 Divisi Software Security Developer

Divisi Software Security Developer merupakan salah satu divisi dari departemen IT yang dikelola oleh CTO (*Chief Technology Officer*), yang juga berperan sebagai pembimbing industri. Adapun *job description* yang menggambarkan seperti apa tugas maupun proyek yang nantinya akan diberikan kepada peserta PKL, list nya digambarkan sebagai berikut:

- Melakukan dan mengevaluasi *penetration testing*
- Membuat script keamanan *software*
- Mengamankan *software* dari serangan luar

3.2 Uraian Praktik Kerja Lapangan

Program PKL yang dijalankan merupakan kegiatan wajib untuk mahasiswa semester 7 (tujuh) aktif dalam jurusan Teknik Informatika dan Komputer, terkhusus untuk program studi Teknik Multimedia dan Jaringan. Pada laporan ini, kegiatan PKL dilaksanakan di CV. Solusi Automasi Indonesia pada departemen IT dalam divisi *Software Security Developer Intern*, yang dilaksanakan selama 3 (tiga) bulan terhitung sejak tanggal 2 September 2021 hingga 2 Desember 2021. Program PKL ini dilaksanakan sebagai salah satu persyaratan nilai dan pengajuan dalam mengikuti skripsi. Seluruh kegiatan PKL didokumentasikan dalam *logbook* terpisah yang disimpan dalam format *spreadsheet*. Berikut merupakan uraian dari *logbook* tersebut:

a. **Pekan Pertama** (02/09/2021 - 03/09/2021)

Melakukan *onboarding* bersama rekan peserta PKL *batch* 5 terhadap lingkungan kerja dan organisasi via *zoom meeting*. Adapun pengenalan dengan pembimbing divisi serta pemberian *overview* mengenai contoh *output* tugas dan bentuk timeline selama PKL berlangsung

b. **Pekan Kedua** (06/09/2021 - 10/09/2021)

Adanya pembekalan mengenai prosesi penilaian PKL, penggunaan manajemen proyek dengan metode Scrum, serta pembahasan materi mengenai API, struktur MVC, dan JWT. Kegiatan kemudian diikuti dengan melakukan *sprint planning* mengenai *backlog* pertama yang dipilih, yaitu pengujian kerentanan dari API internal perusahaan. Pada *daily standup* pertama, kegiatannya adalah mengumpulkan *tools* yang disesuaikan dengan kebutuhan *goals* dari *card* pertama *sprint* pertama ini, yaitu melakukan minimal 2 kegiatan *footprinting* kepada target

c. **Pekan Ketiga** (13/09/2021 - 17/09/2021)

Kegiatan selanjutnya yaitu mencoba untuk mencari *index directory* dari sistem target, baik menggunakan *google dork* maupun dalam *directory traversal*. Adapun percobaan untuk melakukan *port* serta *vulnerability scanning* yang dilakukan secara berkala, diharapkan mendapat hasil yang

tetap relevan. Adapun mengembangkan *tools port scanning* dalam *Bash* secara mandiri untuk mempelajari mekanisme tersebut. Dalam prosesnya, didapatkan juga *source code* dari target yang repositorinya adalah publik, sehingga dilakukan analisa *source code* untuk mengambil informasi sensitifnya dalam tahap *testing*. Hal yang juga mempelajari adalah cara kerja http request serta melakukan enumerasi dari DNS aplikasi

d. **Pekan Keempat** (20/09/2021 - 24/09/2021)

Setelah menyelesaikan card pertama pada *sprint* pertama, dilakukan *sprint review* bersama dengan rekan serta pembimbing divisi, mendiskusikan mengenai progress kemarin dan kendala yang dihadapi. Dalam *card* kedua ini, *goalsnya* adalah melakukan *code injection* serta mencari *sensitive data exposure* dan *broken access control*. Pada *code injection*, diujikan bentuk-bentuk XSS dalam beberapa kasus. Pencarian 2 *goals* lainnya didapatkan dengan pengujian *vulnerable* URI serta informasi dari *source code*, *request interception*, serta *error handling*

e. **Pekan Kelima** (27/09/2021 - 01/10/2021)

Pada hari terakhir *sprint* pertama, kegiatan lebih difokuskan untuk pengujian dalam bentuk *reflected* XSS. *Sprint* kedua kemudian diakhiri dengan melakukan *sprint review* bersama pembimbing dan rekan divisi, mendiskusikan hambatan serta *output* dari keseluruhan pengujian. Selain itu, dilakukan juga *sprint planning* terhadap *sprint* kedua pada *backlog* kedua mengenai finalisasi laporan yang terstruktur dan mudah dipahami oleh divisi lain

f. **Pekan Keenam** (04/10/2021 - 08/10/2021)

Pada pekan keenam, kegiatan dimulai dengan melakukan diskusi perihal penggabungan dokumentasi dengan referensi PTES bersama rekan tugas. Adapun penambahan aspek langkah mitigasi yang tepat dalam mengatasi setiap kerentanan yang diujikan. Digunakan pula kalkulasi CVSS dalam memprioritaskan setiap kerentanan berdasarkan *base score*. Berdasarkan *goals sprint* kedua ini, dilakukanlah penjelasan yang lebih detail terkait

istilah maupun kegunaan *tools* yang diisukan, sehingga dapat memberikan konteks yang lebih baik kepada pembaca, terutama pembimbing industri dan rekan divisi *Backend Developer*

g. **Pekan Ketujuh** (11/10/2021 - 15/10/2021)

Melakukan *sprint review* dengan rekan divisi dan pembimbing. Seluruh tim tugas melakukan presentasi secara utuh dan pemberian *feedback* dari pembimbing. Keseluruhan laporan kemudian di-*archive* ke dalam google drive internal agar dapat diakses untuk kebutuhan internal

h. **Pekan Kedelapan** (18/10/2021 - 22/10/2021)

Mulai melakukan *sprint planning* terhadap *sprint* ketiga ini dengan *backlog* yang baru, yaitu riset mengenai implementasi keamanan komponen UiPath Suite (Studio, Robot, dan Orchestra), yang salah satunya diambil adalah Robot. Karena hal baru, maka adanya upaya dalam mencoba untuk membuat program sederhana terlebih dahulu untuk memahami konteks dan cara kerja produk UiPath secara keseluruhan

i. **Pekan Kesembilan** (25/10/2021 - 29/10/2021)

Setelah berhasil membuat program, kegiatan selanjutnya yaitu mempelajari cara *deployment* program ke dalam Orchestrator, yang membutuhkan interaksi dari Robot untuk eksekusi. Hal ini juga termasuk dalam mempelajari mekanisme konfigurasi / *provisioning* Robot, autentikasi antara Robot dengan *host machine*, serta *role* dan *permission* Robot terhadap *environment* nya

j. **Pekan Kesepuluh** (01/11/2021 - 05/11/2021)

Mencoba konsep *storing credentials* serta melakukan *query* nya dengan memenuhi *security practice*, baik itu disimpan dalam Orchestrator, maupun *third-party* seperti WCM (*Windows Credential Manager*). Selain itu, adapun upaya dalam mempelajari konfigurasi VPN serta SFTP pada Robot untuk mengelola dan mendapatkan *resource* nya, serta interaksi Robot dengan dua macam *workspace* yang berbeda di Orchestrator

k. **Pekan Kesebelas** (08/11/2021 - 12/11/2021)

Mengimplementasikan keamanan dalam sisi *publishing package* dari UiPath studio menuju ke dalam Orchestrator, dengan membuat *certificate signing* dalam format PKCS#12 sebagai mekanisme *signature verification* nantinya. Hal ini menjadi salah satu keamanan yang krusial, dengan memastikan apakah *package* yang akan di-*download* secara lokal oleh Robot merupakan *package* yang sama dengan yang diisukan. Diadakan juga *sprint review* dalam *sprint* ketiga ini, dengan menjelaskan bagaimana hasil riset dan seperti apa bentuk implementasinya

l. **Pekan Kedua belas** (15/11/2021 - 19/11/2021)

Dikarenakan keseluruhan *sprint* sudah selesai dijalankan, maka setiap minggu akan dilanjutkan sebuah *spring* (presentasi) perihal keamanan dan IT secara *general*, yang dilakukan 1-2 kali dalam seminggu. Minggu ini adalah mengenai pengenalan terhadap CTF (*Capture The Flag*). Adapun perancangan *logbook* yang direferensikan pada *timeline* di *spreadsheet*

m. **Pekan Ketiga belas** (22/11/2021 - 26/11/2021)

Disini, 2 rekan mempresentasikan mekanisme *salted hashing* serta keamanan dasar dalam membangun aplikasi *mobile*. Adapun pengembangan *tools port scanning*-nya untuk dapat *support* dalam menerima argumen *port number* dalam bentuk *range* maupun *multiple*

n. **Pekan Keempat belas** (22/11/2021 - 26/11/2021)

Pada minggu terakhir, kegiatan ditutup dengan *spring* dari rekan mengenai SIEM (*Security Information & Event Management*) dan SOAR (*Security Orchestration Automation Response*). Adapun *closing PKL batch 5* secara seremonial yang dilakukan pada 14 desember 2021 via zoom meeting

3.3 Pembahasan Hasil Praktik Kerja Lapangan

Dalam pelaksanaan program PKL, secara besar terdapat dua tugas yang diberikan, yang pertama adalah pengujian kerentanan terhadap REST API beserta dengan dokumentasinya, dan yang kedua melakukan riset mengenai implementasi keamanan dalam UiPath Robot. Disini, laporan difokuskan untuk membahas

mengenai tugas pertama karena sifatnya yang lebih teknis. Adapun beberapa informasi maupun aset yang didapatkan selama pengujian yang tidak bisa disebarluaskan karena sifatnya *confidential* / rahasia untuk perusahaan secara internal. Berikut merupakan pembahasan hasil pelaksanaan PKL terhitung dari tanggal 2 September 2021 hingga 2 Desember 2021. Agar perusahaan dapat mengukur seberapa rentan suatu sistem dari sisi keamanan nya, maka sistem tersebut dapat dilakukan evaluasi secara sistematis dan terstruktur. Kegiatan ini termasuk dalam melakukan *system accreditation* serta *risk assessment* (Yaqoob et al. 2017).

3.3.1 Kondisi Sekarang

Sistem yang digunakan sebagai target merupakan REST API internal yang berada dalam fase *development*. Sistem tersebut juga dibangun secara berbarengan antara pembimbing divisi dengan rekan divisi *Backend Developer*. Karena program PKL dilakukan secara WFH, maka pengujian dilakukan secara *remote*.

3.3.1 Kondisi yang Diharapkan

Karena target masih dalam tahap pengembangan, maka diperlukannya *security assessment* yang baik sebelum produk masuk ke fase *production*. Dengan menggunakan standar PTES, maka *output* dari kegiatan diharapkan bisa memberikan suatu hasil yang bermanfaat dan berbobot oleh perusahaan, terkhusus untuk divisi *Backend Developer*, dalam meningkatkan aspek keamanan sistem tersebut. Karena itu, bentuk penjelasan laporan akan direferensikan dengan PTES tanpa mengeluarkan maupun membocorkan informasi internal perusahaan terkait hasil dari kegiatan pengujian tersebut.

3.3.2 Tahapan Pengujian Kerentanan

3.3.2.1 Pre-Engagement

Pada tahap ini, dilakukannya koordinasi dengan pembimbing serta rekan tugas terhadap seluruh kegiatan yang akan dilakukan dalam

pengujian tersebut. Tabel 3.1 berikut menerangkan hal mendasar seperti *scope*, *hourly time estimation*, *time duration*, dan *scope meeting* (Anonym 2021). Hal-hal dilakukan *mapping* dalam rangka untuk memperjelas bagaimana pelaksanaan pengujian kerentanan berikut berlangsung yang diikuti dengan instrumennya. Adapun penjelasan singkat terhadap atribut pada tabel 3.1 sebagai berikut:

- a. Target: produk / aset yang akan diujikan dalam penugasan
- b. Goal Time Estimation: estimasi lama waktu dan durasi yang dibutuhkan dalam penugasan untuk setiap *goals*-nya
- c. Scope Execution: lingkup serta cangkupan bidang ataupun instrumen yang digunakan dan dilaksanakan selama penugasan berlangsung
- d. PIC Information: pihak yang bertanggung jawab terhadap penugasan berlangsung, dalam konteks ini adalah pihak pembimbing industri

Tabel 3.1 Deskripsi Pre-Engagement

Target	REST API (development)		
Goal Time Estimation	Footprinting	30 hrs	10/09/21 - 17/09/21
	Testing	35 hrs	20/09/21 - 28/09/21
	Documentation	35 hrs	04/10/21 - 12/10/21
Scope Execution	Agreement	PKWT - PP No. 35 2021	
	Proj. Management	Scrum Framework	
	Online Platform	Discord	
		Google Meet / Zoom	
		Google Drive	
		Trello	
		WhatsApp	
	Location	Remote	

PIC Information	Name	Irfan Nugraha, S.Kom.
	Title	Chief Technology Officer
	Discord	Irfan Nugraha#3205

3.3.2.2 Intelligence Gathering

Pada tahap ini, dilakukannya *information gathering* baik secara pasif maupun aktif. Kegiatan ini ditujukan untuk mengumpulkan informasi sebanyak mungkin yang sifatnya relevan dan suportif dalam melakukan *vulnerability assessment* dan fase eksploitasi. Adapun berbagai macam tools dan pendekatan yang dilakukan untuk mengcover informasi seluas mungkin, yaitu *port scanning*, *vulnerability scanning*, *DNS enumeration*, *directory traversal & crawler*, serta *source code analysis*. Informasi yang dipaparkan telah mengalami proses *masking* untuk meminimalisir *exposure* lebih mengenai hasil dari fase ini yang sifatnya internal. Untuk memberikan konteks yang lebih jelas mengenai tahap ini, maka berikut akan diberikan informasi mengenai metode pendekatan terhadap *tools* yang digunakan serta hasil temuan yang didapatkan. Adapun penjelasan singkat terhadap atribut pada tabel 3.2 sebagai berikut :

- a. Pendekatan Internal: kegiatan *intelligence gathering* dilakukan di dalam lingkungan kerja, sehingga aktivitas mencakup diskusi serta pembekalan terhadap target
- b. Pendekatan Eksternal: kegiatan *intelligence gathering* dilakukan di luar lingkungan kerja, sehingga aktivitas lebih mencakup penggunaan tools terhadap informasi yang lebih dalam mengenai teknologi dari target

Tabel 3.2 Metode & Tools dalam Intelligence Gathering

Pendekatan Internal		
Pasif	Studi Literatur	
Aktif	FGD (Forum Group Discussion)	
Pendekatan External		
Pasif	DNS Enumeration	Netcraft
		Nslookup
	Indexing	Google Dork
Aktif	Port Scanning	Nmap 7.80
		Port-sweeper a.2.12
	Vulnerability Scanning	Nikto 2.1.5
	Directory Traversal & Listing	Dotdotpwn 3.0.2
		Screaming Frog 16.5
		DirBuster 1.0
	HTTP Req. Modifier	BurpSuite 2021.8.1
		Curl 7.68.0

Adapun penjelasan singkat terhadap atribut pada tabel 3.3 sebagai berikut :

- a. Target API: alamat URL API yang dijadikan uji target
- b. DNS Enumeration: informasi lingkup serta teknologi server melalui alokasi *DNS record*
- c. Open Port Service: informasi service yang terbuka beserta dengan port number, protokol, serta teknologinya
- d. SSL: informasi terkait dengan penggunaan protokol HTTPS

- e. Directory Traversal: informasi terhadap ukuran kapabilitas untuk mengakses resource diluar lingkup aplikasi yang diujikan menggunakan endpoint
- f. API Endpoints: URI aplikasi yang digunakan sebagai rute untuk menjalankan suatu fungsi beserta dengan method-nya
- g. Header Configuration: konfigurasi dalam request dan response header server dalam melayani aplikasi

Tabel 3.3 Hasil Informasi Intelligence Gathering

Target API	http://xxx.com		
DNS Enumeration	IP Address (A)		54.xxx.xxx.xxx
			54.xxx.xxx.xxx
			3.xxx.xxx.xxx
			3.xxx.xxx.xxx
	AS Number		ASxxx
	Domain		xxx.com
	Nameserver		dns1.xxx.xxx.net
	Hosting Country		US
Open Port Service	xxx / tcp	*	?
	xxx / tcp	*	?
	80 / tcp	http	Apache Cowboy httpd
	443 / tcp	ssl / http	Apache Cowboy httpd
	xxx / tcp	*	?
SSL	Cipher		ECDHE-RSA-AESxxx GCM-SHAxxx
	Certificate		X.509 v.xxx
	Root CA		*.xxx.com
	Protocol Version		TLS v.1xxx

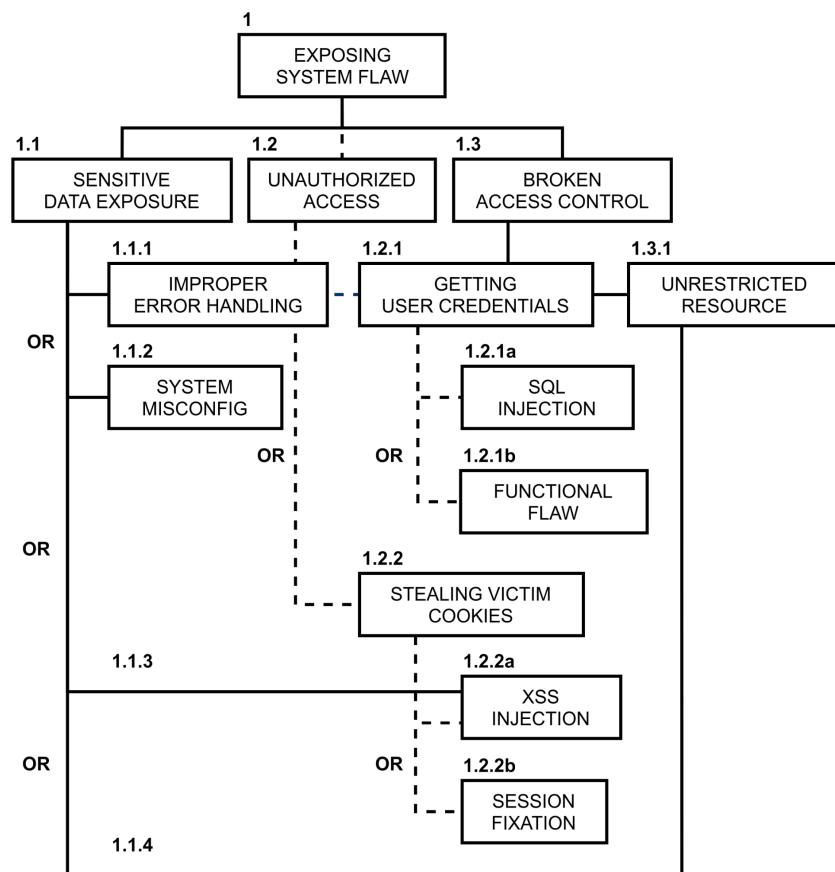
Dir. Traversal	1080 out of 5549 marked as vulnerable URI	
API Endpoints	GET	http://xxx.com/
	GET	http://xxx.com/blog
	GET	http://xxx.com/layanan
	GET	http://xxx.com/cobajwt
	POST	http://xxx.com/user/ubahpassword
	POST	http://xxx.com/auth/register
	POST	http://xxx.com/auth/login
	POST	http://xxx.com/sendmail
Header Conf.	Missing X-XSS-Protection header	

3.3.2.3 Threat Modelling

Pada tahap ini, dilakukan *threat modelling* menggunakan metode *attack tree*. Kegiatan ini memungkinkan penguji untuk memberikan gambaran yang lebih jelas mengenai beberapa macam *attack vector* yang dapat dilakukan untuk menuju *goals* yang sama, dari yang ditugaskan dalam card kedua. Dalam menjabarkan berbagai *attack vector*-nya, maka haruslah direfleksikan kembali apakah suatu *attack vector* tersebut cocok dengan target uji, yang merupakan aplikasi web dalam bentuk REST API. Hal ini tentunya meminimalisir beberapa *attack vector* karena tidak adanya interaksi secara langsung dengan elemen UI (User Interface) pada aplikasi.

Walaupun pemodelan *attack tree* dirancang secara *top-down approach*, pengujian *attack vector* dilakukan secara *bottom-up* karena *cost* nya relatif lebih rendah dan lebih cepat untuk dapat mencapai ke *node* di atasnya, dibandingkan dengan metode BDD (*Behaviour Driven Development*) apabila *tree attack* tersebut tidak

memiliki *shared sub-tree* (satu node memiliki 2 *parent*) dengan struktur kompleks (Kuipers 2020). Setelah *attack tree* nya terbentuk, maka setiap *attack vector* harus diverifikasi terlebih dahulu apakah target memenuhi kondisi untuk melakukan suatu *attack vector* tersebut. Berikut merupakan pemodelan *attack trees* dalam bentuk diagram:



Gambar 3.3 Diagram Attack Tree

Dikarenakan goals yang diberikan terbagi menjadi 3 (*code injection, sensitive data exposure & broken access control*) maka *root node* yang diambil adalah *exposing system flaw* sebagai generalisasi dari seluruh *goals* pada rangkaian kegiatan pengujian ini, yaitu mengekspos kerentanan sistem. Pada gambar 3.3, dapat dilihat bahwa *level 2* (1.1, 1.2, dan 1.3) merupakan judul dari *goals* itu sendiri, sedangkan pada *level 3* (1.1.1, 1.1.2, 1.1.3, 1.1.4, 1.2.1,

1.2.2 dan 1.3.1) merupakan *leaf node* yang digunakan sebagai *attack vector* saat tahapan *exploitation*. Disini, *goals code injection* sendiri dianggap sebagai *attack vector*, sehingga *parent node* 1.2 diasumsikan merupakan *goals* yang dapat diraih dalam melakukan *code injection*, yaitu untuk dapat melakukan *unauthorized access*. Adapun *level* 4 (1.2.1a, 1.2.1b, 1.2.2a dan 1.2.2b) yang merupakan penjabaran dari bagaimana *attack vector* dalam *unauthorized access* dapat dilakukan, namun masih dalam konteks *code injection* sesuai dengan *goals* awal yang diberikan.

Dalam kasusnya, sebelum masuk ke dalam periode *testing*, ditemukan terdapat kendala mengenai konektivitas antara *database* dengan REST API, menyebabkan seluruh fungsional ataupun *controller* yang membutuhkan koneksi ke dalam *database* tersebut tidak dapat dilayani oleh *server* API. Dikarenakan adanya ketergantungan tersebut, pada *parent node level* 2, yaitu 1.2, seluruh *attack vector*-nya tidak dapat dilakukan karena fungsinya yang membutuhkan konektivitas *database* tidak dapat dijalankan dan dimanfaatkan dalam pengujian. Maka dari itu, pada *leaf node* 1.2.2a pada gambar 3.3 diatas, *parent goals* nya masuk ke dalam kategori *sensitive data exposure* tanpa adanya tendensi untuk mendapatkan akses ke dalam sistem melalui serangan *cookie*, membuatnya tidak berlaku sebagai *shared-sub tree* kembali. Walaupun begitu, pada *node* 1.2.1, pengerjaan tetap dilakukan secara *bottom-up* karena *shared-sub tree* tidak tergolong kompleks dan tetap memiliki 2 referensi *parent node*, yaitu 1.1. dan 1.3.

3.3.2.4 Vulnerability Analysis

Setelah mendapatkan hasil dari tahapan *intelligence gathering*, dilakukannya analisa untuk mengambil beberapa kesimpulan dari informasi yang telah didapatkan yang kiranya relevan untuk

dilakukan *exploitasi*. Kemudian, kerentanan yang didapatkan akan direferensikan dengan CVE ataupun CWE beserta dengan *scoring* CVSS nya yang didasarkan dari *leaf node* sebagai *attack vector* untuk tahapan selanjutnya.

Dalam beberapa *service* yang terbuka, beberapa *port version name* nya bersimbolkan “?”. Hal ini menandakan kalau *port* tersebut terbuka namun tidak dapat sepenuhnya terbaca melalui *probing* dengan Nmap, sehingga digunakanlah nama *service* yang sesuai dengan *default port* tersebut. Pada tahapan enumerasi DNS, dapat juga terlihat bahwa server memiliki 4 *record* DNS type-A secara sekaligus, yang dikenal dengan istilah load balancing. Hal ini memastikan server untuk tetap *reliable* dan *scalable* dalam menghadapi *failover* kedepan (Naredo & Pardavila 2007). Teknologi ini umum digunakan pada *cloud service third-party*, sehingga ancaman seperti DDoS (*Distributed Denial of Service*) menjadi lebih tidak efektif dan efisien karena traffic yang di-bombardir akan terus terdistribusi kedalam berbagai *backend* yang tersedia.

Dalam mempermudah pembahasan, berikut dijelaskan mengenai 4 kerentanan yang akan dianalisa yang disesuaikan dengan *attack vector* pada gambar 3.3 mengenai pemodelan *attack tree*. Adapun penjelasan singkat terhadap atribut pada tabel-tabel yang digunakan sebagai referensi 4 objek kerentanan tersebut, yaitu pada tabel 3.4, 3.5, 3.6, dan 3.7 sebagai berikut:

- a. No.: nomor urut objek kerentanan yang dianalisa
- b. Title: penamaan objek kerentanan yang digunakan sebagai referensi pada tahap *exploitation*
- c. Target: *endpoint* / aset yang diisukan terhadap kerentanan

- d. Reference: referensi CWE yang digunakan sebagai dasar analisa kerentanan serta memberi gambaran bagaimana pembentukan *attack vector* yang sesuai
- e. Base Score: ukuran hipotesa terhadap tingkat kerentanan dari objek yang dianalisa. Nilai *base score* sendiri merupakan hasil dari kalkulasi *metric* yang dapat direferensikan pada gambar 2.2 serta penjelasannya

Merujuk pada tabel 3.4 untuk objek kerentanan pertama, dalam konfigurasi *header*, didapatkan bahwa proteksi X-XSS *header* tidak di-set dalam sisi *server*. Dalam konteks *browser* yang tidak modern atau *legacy*, *header* semacam ini dapat ditemukan dalam *Content-Security-Policy* dengan fitur *unsafe-inline script* yang ter-*enable*. Karena itu, info tersebut memberikan gambaran bahwa adanya potensi injeksi XSS dalam URL untuk semua *page* dalam *server*, yang dapat menyebabkan informasi internal aplikasi dapat diakses dari *unauthorized user* (Anonym 2021). Dikarenakan target berbentuk API, maka tipe serangan XSS yang akan digunakan adalah *reflected XSS (non-persistent)* karena tidak adanya elemen UI untuk di-*exploit*, layaknya *user input*. Hal ini berbeda dengan *stored XSS (persistent)*, karena *reflected XSS* hanya memanfaatkan injeksi *script* ke dalam *header request*, tanpa tersimpan maupun tertulis ke dalam server, sehingga membuatnya serangan yang lebih temporer dan *conditional* dibandingkan dengan yang bertipe *persistent* yang lebih stabil dan *reliable* (Marashdih & Zaaba 2016).

Dikarenakan bentuk XSS yang tidak *persistent* dapat membuat *attack vector*-nya menjadi lebih temporer, maka *metric* yang diberikan menghasilkan *base score* dengan tingkat kerentanan yang tergolong rendah, yaitu 3.1 dari 10.0.

Tabel 3.4 CVSS pada XSS Injection (Sensitive Data Exposure)

No.	1						
Title	X-XSS-Protection Header is not defined						
Target	http://xxx.com:80/						
Reference	CWE-644: Improper Neutralization of HTTP Headers for Scripting Syntax						
AV	N	AC	H	PR	N	UI	R
S	U	C	L	I	N	A	N
Base Score				3.1 / 10.0			

Merujuk pada tabel 3.5 untuk objek kerentanan kedua, adapun penggunaan *directory traversal* untuk mendeteksi apakah *file system* target dapat diakses melalui URI (Uniform Resource Identifier) nya, yang dimana memanfaatkan kelemahan dalam sisi *indexing* direktori aplikasi. Kelemahan dalam hal ini memperbolehkan user publik mengakses segala *file* dalam sistem *server*, yang dimana sudah diluar *scope* dari aplikasi *server*. Salah satu cara dalam mengganti direktori aplikasi menuju sistem, dengan menggunakan *dot-dot-slash* (*../*) dalam URI nya. Dalam 1080 dari 5549 URI yang dicoba, API berhasil me-*reject* penggunaan *dot-dot-slash* dengan memberikan status 500, 403, serta 400. Konfigurasi ini juga dapat ditemukan pada opsi *indexing* dalam konfigurasi *.htaccess* yaitu “Options All -Indexes”, yang membuat *server* tidak menyalakan fungsi *indexing* kepada publik, sehingga *root* sistem merupakan direktori aplikasi. Karena secara *default*, *indexing* dalam Apache akan bersifat publik. Sedangkan URI yang berstatuskan *vulnerable* adalah beberapa URI yang memiliki dan menerapkan kombinasi dari *percent-encoding*. *Percent-encoding* dalam kasus ini, khusus digunakan untuk

merepresentasikan komponen URI dalam bentuk digit *hexadecimal* yang merepresentasikan nilai *octet*, sehingga aplikasi yang tidak melakukan URI *decoding*, akan menjawab dan mentranslasikan *encoding* tersebut kembali sebagai *dot-dot-slash* yang valid (Berners-Lee 2005).

Dikarenakan adanya inkonsistensi terhadap status *vulnerable* walaupun URI sudah menggunakan pendekatan *percent-encoding*, maka *metric* yang diberikan menghasilkan *base score* dengan tingkat kerentanan yang tergolong rendah, yaitu 3.7 dari 10.0.

Tabel 3.5 CVSS pada System Misconfig (Sensitive Data Exposure)

No.	2						
Title	Vulnerable URI on /etc/passwd						
Target	http://xxx.com:80/						
Reference	CWE-548: Exposure of Information Through Directory Listing						
AV	N	AC	H	PR	N	UI	N
S	U	C	L	I	N	A	N
Base Score				3.7 / 10.0			

Merujuk pada tabel 3.6 untuk objek kerentanan ketiga, ditemukan juga adanya *source code* aplikasi tersebut yang ter-*hosting* dengan *visibility* publik. Namun, *source code* yang didapatkan berada dalam *release* yang berbeda dengan yang dikembangkan lebih lanjut oleh divisi *Backend Developer*. Meskipun begitu, apabila konsep *legacy code* diasumsikan masih diimplementasikan oleh aplikasi, sehingga beberapa informasi masih relevan untuk dipelajari, bahkan apabila memiliki *write access* terhadap *repository* tersebut (Shamay 2020). Terlebih dikarenakan proyek

bukan bersifat *open source*, sehingga *scope* aksesibilitas serta aktivitas tetap harus terjaga untuk penggunaan internal dengan *actor / user* yang sudah *credible* untuk identitasnya.

Dikarenakan terbukanya akses menuju *source code* secara publik walaupun dapat diasumsikan sebagai *legacy code*, maka *metric* yang diberikan menghasilkan *base score* dengan tingkat kerentanan yang tergolong sedang, yaitu 5.3 dari 10.0.

Tabel 3.6 CVSS pada Unrestricted Resource (Broken Access Control & Sensitive Data Exposure)

No.	3						
Title	Unrestricted Public Repository						
Target	public hosting cloud-service						
Reference	CWE-284: Improper Access Control						
	CWE-540: Inclusion of Sensitive Information in Source Code						
AV	N	AC	L	PR	N	UI	N
S	U	C	L	I	N	A	N
Base Score				5.3 / 10.0			

Merujuk pada tabel 3.7 untuk objek kerentanan keempat, terkait dengan isu konektivitas terhadap *database* dan API pada sub-bab sebelumnya, penggunaan *framework* besar dalam aplikasi, secara umum akan memberikan fungsi *traceback call* ataupun tracing terhadap bagian dimana isu tersebut muncul dalam *source code*. Hal ini dimanfaatkan oleh user maupun tim dalam melakukan *debugging* aplikasi, sehingga memudahkan untuk melakukan *overview* terhadap faktor apa yang menyebabkan aplikasi menjadi tidak berfungsi secara normal. Informasi yang diberikan dapat

berupa terkait mengenai *database*, *logs*, *routes*, serta *event*, yang dimana dapat dimunculkan dalam berbagai macam fase *development*, kecuali *production*. Di sisi lain, hal ini dapat menjadi ancaman untuk aplikasi karena informasi tersebut terbuka untuk dilihat bagi siapa saja yang mengalami *error* yang semacamnya, yang dimana dapat bersifat sensitif dan internal (Anonym 2021). Walaupun begitu, hal ini dapat pula tergantung dengan bagaimana aplikasi tersebut di *deploy* dan konfigurasi *server* tersebut dalam meng-*handle* adanya *error* saat service sudah berjalan.

Dikarenakan adanya potensi tereksposnya informasi aplikasi serta server yang bersifat internal dan lengkap walaupun pemanfaatan sangat bersifat *conditional*, maka *metric* yang diberikan tetap menghasilkan *base score* dengan tingkat kerentanan yang juga tergolong sedang, yaitu 5.9 dari 10.0.

Tabel 3.7 CVSS pada Improper Error Handling (Sensitive Data Exposure)

No.	4						
Title	Improper Error Handling						
Target	http://xxx.com:80/blog						
Reference	CWE-209: Generation of Error Message Containing Sensitive Information						
AV	N	AC	H	PR	N	UI	N
S	U	C	H	I	N	A	N
Base Score				5.9 / 10.0			

3.3.2.5 Exploitation

Setelah melakukan pemodelan terhadap beberapa kemungkinan ancaman, maka selanjutnya masuk kedalam fase eksploitasi kerentanan tersebut yang sesuai dengan 4 *leaf node* dalam diagram

attack tree. Disini, adapun pendemonstrasian contoh serangan yang diimplementasikan, serta menjelaskan bagaimana pendekatan yang digunakan serta memberikan gambaran terhadap *output* yang didapatkan. Penjabaran akan dilakukan secara berurutan sesuai dengan prioritas skor CVSS yang diberikan. Adapun hasil dari pengujian dapat disimpulkan dalam bentuk *confusion matrix*, yaitu:

- *True Positive*: pengujian berhasil dilakukan dengan kerentanan yang sesuai dan sudah tervalidasi. Diistilahkan sebagai *hits*
- *False Positive*: pengujian gagal dilakukan walaupun dengan kerentanan yang sesuai dan sudah tervalidasi. Diistilahkan juga sebagai *false alarm*
- *True Negative*: pengujian gagal dilakukan karena kerentanan memang tidak tervalidasi. Diistilahkan sebagai *correct rejection*
- *False Negative*: pengujian berhasil dilakukan walaupun dengan kerentanan yang tidak tervalidasi. Diistilahkan sebagai *miss* (Das 2021)

Pada kegiatan eksploitasi pertama, perihal X-XSS-Protection Header is not defined dalam objek kerentanan pada tabel 3.4, dilakukan beberapa bentuk *reflected* XSS dalam pengujiannya. Adapun penamaan “*reflected*” dikarenakan atribut *payload* dalam *request header* akan direfleksikan kembali pada response header untuk dapat dijalankan. Bentuk pertama yang diujikan adalah penggunaan *Web Cache Deception*, yang dimana memanfaatkan *cache* pada aplikasi terhadap *response header* untuk membawa *payload* XSS. Apabila request berhasil ter-*cache* dalam periode 60 detik, maka *request* yang *match* dapat menjalankan *payload* tersebut. Dalam seluruh bentuk XSS nanti akan digunakan *percent-encoding* dalam *request headernya*, untuk mem-*bypass*

blacklist ataupun filterisasi yang diimplementasikan dalam *business logic* API. Salah satu limitasi yang kemudian ditemukan adalah API tidak menerima *cache* dari *request*, yang diinfokan melalui *cache control* dengan parameter *max-age=0*, *no-store* dan *no-cache*, sehingga *payload* tidak dapat dijalankan. Berikut bentuk pengujian yang dilakukan, dengan adanya *snippet* pertama yang menggunakan *percent-encoding*, serta pada *snippet* kedua yang merupakan hasil translasinya:

```
http://xxx.com/\?payload\=%22%3E%3Cscript%3Ealert\(\document.domain\)%3C/script%3E\&cache\=60
```

```
http://xxx.com/?payload="><script>alert(document.domain)</script>&cache=60
```

Bentuk kedua yang diujikan adalah penggunaan *reflected* XSS dengan *On-Error Event*, yang dimana ditunjukkan untuk mengakses *resource* yang sebenarnya tidak ada dengan tujuan untuk *men-trigger error alert*. Hal ini dapat diraih dengan menggunakan tag HTML yang melakukan *reference* terhadap *resource remote* ataupun *local*, seperti *svg*, *img*, *audio*, serta *embed* tag. Konsep ini memiliki kemiripan dengan penggunaan *throw exception*, yang nantinya digunakan untuk menjalankan *payload* XSS. Salah satu limitasi yang dihadapi adalah tidak ditemukannya penggunaan parameter dalam method GET yang sesuai, sehingga *payload* tidak tereksekusi sama sekali. Berikut bentuk pengujian yang dilakukan, dengan adanya *snippet* pertama yang juga menggunakan *percent-encoding*, serta pada *snippet* kedua yang merupakan hasil translasinya:

```
http://xxx.com/\?payload\=%3Cimg%20src\=logo.png%20onerror\=alert\(\document.domain\)%3E
```

```
http://xxx.com/?payload=<img src=logo.png onerror=alert(document.domain)>
```

Bentuk terakhir yang diujikan adalah penggunaan *reflected XSS* dengan *Response Splitting*, yang dimana membuat refleksi *payload* dari *response header*, dipindahkan menuju *response body*. Hal ini ditunjukkan karena *payload* akan menjadi *executable* apabila berada dalam *response body*. Salah satu cara untuk dapat memindahkan *payload* tersebut adalah dengan menggunakan *escape character*, yaitu *carriage return* (\r) serta *newline* (\n). Dalam kasus ini, digunakan pembuatan *cookie dummy* yang parameternya disesuaikan dengan informasi yang ditemukan dalam *source code* mengenai pembuatan *cookie*, yaitu penggunaan *HttpOnly* (pembuatan *cookie* dan diakses hanya melalui *browser*) dan *SameSite* dengan value *Lax* (pengiriman *cookie* hanya melalui *top-level navigation* DOM, link tag ataupun form). Limitasi yang ditemukan adalah belum mendapatkannya atribut yang merefleksikan kedalam *response header*, terlebih *requirement* mengenai pembuatan *cookie* yang belum terpenuhi. Berikut bentuk pengujian yang dilakukan, dengan adanya *snippet* pertama yang menggunakan *percent-encoding* dalam sisi *request*, serta pada *snippet* kedua yang merupakan hasil dari sisi *response* yang diberikan aplikasi:

Request

```
GET /xxx.php?payload=temp%0D%0A%0D%0A%0D%0A%22%3E%3Cscript%3Ealert(document.domain)%3C/script%3E HTTP/1.1
Host: xxx.com
Cookie: payload=; HttpOnly; SameSite=Lax;
```

Response

```
HTTP/1.1 200 OK
Connection: close
Content-Type: application/json; charset=UTF-8
Content-Length: 56
```

Adapun hasil dari pengujian dalam X-XSS-Protection Header is not defined yang disimpulkan sebagai **False Positive**, karena

pengujian gagal walaupun terdapat potensi mengenai kerentanan mengenai XSS dalam aplikasi.

Pada kegiatan eksploitasi kedua, perihal Vulnerable URI on /etc/passwd dalam objek kerentanan pada tabel 3.5, dilakukan dengan mencoba URI yang ditemukan dengan berstatus *vulnerable* secara simultan. Dikarenakan *list* yang didapatkan relatif banyak, maka digunakannya suatu *Bash script* sederhana untuk menguji URI tersebut dengan memanfaatkan *background jobs* layaknya *threading*. Berikut *snippet source code*:

```
#!/bin/bash
file=$1; len_file=$(cat "$file" | wc -l)
worker=$2; inc=$((len_file/worker));
start=1; next=$inc

runCurl() {
    while read -r uri; do
        curl -X GET "$uri" -silent | grep -v 'xxx'
    done < "$1"; }

for i in `seq 1 $worker`; do
    sed -n $start,$nextp "$file" > "$file".part"$i"
    runCurl "$file".part"$i"
    start=$((next+1)); next=$((next+inc)); done
```

Pada *snippet* diatas, program memanfaatkan 2 parameter, yaitu file yang mengandung seluruh list URI serta jumlah worker threading. Adapun function *runCurl* yang ditujukan untuk menjalankan request berbentuk GET terhadap URI nya. Adapun penggunaan *grep* sebagai filterisasi untuk menampilkan konten selain dari *response endpoint* yang digunakan, sehingga diharapkan dapat menangkap 2 file yang diujikan, yaitu /etc/passwd (*list user* dan *group* yang teregistrasi dalam *server*) serta /etc/issue (informasi *server*). Setelah menggunakan seluruh URI, tidak ada hasil yang dikeluarkan yang sesuai dengan *output* yang diharapkan. Hal ini dapat dikarenakan tidak ditemukannya parameter yang sesuai

untuk menampung payload, sehingga seluruh *request* yang diujikan kembali kedalam *request* kepada *endpoint* yang sesungguhnya.

Dikarenakan hal tersebut, hasil dari pengujian dalam Vulnerable URI on /etc/passwd disimpulkan sebagai ***False Positive***, karena pengujian gagal walaupun terdapat potensi mengenai kerentanan mengenai *decoding* URL / *hexadecimal* dalam aplikasi.

Pada kegiatan eksploitasi ketiga, perihal Unrestricted Public Repository dalam objek kerentanan pada tabel 3.6, dilakukan dengan menganalisis *source code* lebih dalam untuk menemukan informasi yang kiranya sensitif terhadap aplikasi. Selain mendapatkan seluruh *route endpoint* yang dimanfaatkan dalam tahap sebelumnya, adapun informasi yang didapatkan seperti jenis *tech stack* yang digunakan, struktur MVC aplikasi, konfigurasi *override server*, mekanisme *hashing* yang digunakan dalam penyimpanan *password*, gambaran mengenai beberapa instance dalam *database* seperti *table* dan atributnya, serta digunakan untuk mempelajari *business logic* aplikasi tersebut.

Adapun penyimpanan *credential* yang tersimpan ke dalam *environment variable* aplikasi yang diasumsikan dalam bentuk file *dotenv* (.env), yang berpotensi untuk menyimpan seluruh *credential database* serta *API key*. Namun hal ini dilimitasi dengan adanya penggunaan *local scope*, sehingga *file* tersebut tidak disertakan ke dalam *repository*. Hal ini ditunjukkan untuk menjadi salah satu cara dalam menyimpan *credential* secara internal. Selain mendapatkan berbagai macam informasi terkait dengan aplikasi, *unauthorized actor* juga memiliki potensi untuk memberikan *malicious commit* dengan harapan admin dapat melakukan merging script tersebut kedalam *codebase* ataupun melakukan *zero day exploit* terhadap suatu *release*. Adapun penjelasan singkat terhadap

atribut pada tabel 3.8 serta hasil eksploitasi yang didapatkan sebagai berikut:

- a. Server Dependencies: aspek ketergantungan teknologi dalam berjalannya *server*
- b. Indexing Option: opsi fitur terhadap pembukaan indeks direktori terhadap *server*
- c. User's Model Attribute: kumpulan atribut dari model ataupun tabel user yang diisukan pada *database*
- d. Password Security: informasi mekanisme keamanan pada penyimpanan *password* dalam *database*

Tabel 3.8 Hasil eksploitasi dalam Unrestricted Public Repository

Server Dependencies	PHP x.x	
	Codeigniter x.x	
	Firebase/php-jwt release x.x	
	Apache HTTP Server x.x	
Indexing Opt.	Options All -Indexes (.htaccess)	
User's Model Attribute	xxx	Required
	xxx	Required 5 - 51 valid unique
	xxx	Required 7 - 255
	created_at	Required ** auto-generated
	updated_at	Required ** auto-generated
Password Security	Base64 Encoding Algorithm	
	Bcrypt Hashing Algorithm	

Dikarenakan hal tersebut, hasil dari pengujian dalam Unrestricted Public Repository disimpulkan sebagai **True Positive**, karena pengujian berhasil dilakukan sebagai *unauthorized actor* yang

direferensikan dengan kerentanan terhadap adanya potensi informasi sensitif terkait dengan aplikasi.

Pada kegiatan eksploitasi terakhir, perihal Improper Error Handling dalam objek kerentanan pada tabel 3.7, dilakukan dengan menganalisis *error message* yang dikeluarkan dari fitur *debugging* aplikasi. Dengan mengakses endpoint yang membutuhkan *controller* terhadap konektivitas *database*, maka dikeluarkanlah fitur *debugging* dengan memberikan kronologis *error* yang terjadi serta spesifik *snippet source code* yang memiliki relasi terhadap isu tersebut, keseluruhan struktur *file* dengan *file path* nya, serta *environment variable* dalam aplikasi. Secara default, *framework* seperti CodeIgniter akan memberikan informasi secara general yang dimana dapat memberikan informasi lebih dari yang dibutuhkan dalam *scope error* tersebut. Informasi sensitif tersebut yang memberikan potensi untuk penyerang dalam memberikan eksploitasi lanjutan kedalam aplikasi. Adapun penjelasan singkat terhadap atribut pada tabel 3.9 serta hasil eksploitasi yang didapatkan sebagai berikut:

- a. Server Memory: ukuran kapasitas RAM *server* dalam meng-*handle request client*
- b. Filepath Constant: variabel konstan yang digunakan sebagai referensi *path* dalam aplikasi
- c. Web Variable: variabel yang digunakan dalam menjalani fitur dalam lingkup *web server*
- d. Database Variable: variabel yang tersimpan secara *private* untuk konektivitas *database server*
- e. JWT Variable: variabel yang digunakan dalam menjalani fungsi autentikasi menggunakan JWT

Tabel 3.9 Hasil eksploitasi dalam Improper Error Handling

Server Memory	Peak Memory Usage	6 MB
	Memory Limit	xxx MB
Filepath Constant	FCPATH	
	SYSTEMPATH	
	ROOTPATH	
	VENDORPATH	
Web Variable	PWD	/xxx
	DOCUMENT_ROOT	/xxx/xxx
	PORT	xxx
	WEB_CONCURRENCY	xxx
	ENVIRONMENT	development
	SHOW_DEBUG_BACKTRACE	1
	REMOTE_ADDR	10.xx.xx.xx
	SERVER_ADDR	172.xx.xx.xx
Database Variable	app.baseURL	http://xxx.com
	database.default.hostname	xxx.com:xxx
	database.default.database	xxx
	database.default.username	xxx
	database.default.password	xxx
JWT Variable	JWT_SECRET_KEY	xxx
	JWT_TIME_TO_LIVE	3600

Dikarenakan hal tersebut, hasil dari pengujian dalam Improper Error Handling disimpulkan sebagai **True Positive**, karena pengujian berhasil mengambil informasi sensitif terkait dengan

aplikasi yang dapat digunakan untuk melakukan serangan yang lebih luas lagi.

3.3.2.6 Post Exploitation

Setelah didapatkan hasil dari eksploitasi, yang dimana merupakan informasi sensitif, hal yang dapat dilakukan selanjutnya adalah memanfaatkan informasi ataupun *credential* tersebut untuk digunakan dalam pengujian yang lebih dalam lagi. Dikarenakan fase *exploitation* diselesaikan sesuai dengan durasi dalam *timeline*, maka fase *post exploitation* belum dapat dilakukan secara bersamaan. Dikarenakan eksploitasi yang berhasil tidak merubah *scope* dari aplikasi, maka tidak dilakukan tahapan *clean-up* terhadap kondisi sebelum maupun sesudah fase eksploitasi. Adapun penyelesaian fase terakhir, yaitu dokumentasi dan *reporting* sebagai kegiatan terakhir dalam *sprint* kedua.

3.3.3.7 Reporting

Dalam fase *reporting* ini, hasil keseluruhan rangkaian kegiatan pentest akan disajikan dalam bentuk *executive summary*, yang dimana menjelaskan seperti apa rekomendasi serta mitigasi yang dapat dilakukan berdasarkan hasil analisa kerentanan serta fase eksploitasi. Mitigasi sendiri ditujukan dalam berbentuk saran untuk pembentukan dan pengembangan aplikasi yang lebih aman untuk dikemudian hari. Adapun *scope* penjelasan yang dilakukan berdasarkan 4 kegiatan eksploitasi yang dilakukan sebelumnya, sehingga saran dan rekomendasi bersifat spesifik untuk *use-case* tersebut.

Pada rekomendasi dalam kegiatan eksploitasi pertama, perihal X-XSS-Protection Header is not defined dalam objek kerentanan pada tabel 3.4, maka salah satu hal yang harus diperhatikan adalah bagaimana manajemen dan konfigurasi terhadap *header request*

dan apa saja yang perlu ditampilkan pada sisi *response*. Dengan begitu, maka solusi dan saran yang terkait dapat dijabarkan dalam beberapa sudut pandang, yang akan dijelaskan sebagai berikut:

- Penggunaan X-XSS Protection Header. Dalam konteks *browser* modern, header tersebut menjadi salah satu standar sebagai mitigasi serangan XSS dengan membantu aplikasi dalam mengaudit potensi serangan tersebut dengan menggunakan *XSS auditor*. Header tersebut dapat digunakan dalam semua *endpoint* pada aplikasi, sehingga proteksi bersifat menyeluruh
- Penggunaan CSP (*Content Security Policy*) Header. Hal ini dapat membantu pencegahan serangan *code injection* dengan melimitasi *resource* apa saja yang dapat di-load ke dalam aplikasi, yang mana termasuk *script* JavaScript, CSS, ataupun *payload* dalam *executable script* lainnya (*script execution*). CSP sendiri hadir dalam berbagai *level* untuk mendukung *browser* modern maupun *legacy*, sehingga diharapkan dapat meng-*cover* keamanan aplikasi secara luas
- Penggunaan WAF (*Web Application Firewall*). Peran WAF sendiri dalam infrastruktur server merupakan hal yang penting dan relevan dalam mitigasi serangan XSS, dengan salah satu fitur nya yaitu *signature based security rules*. Hal ini dimanfaatkan untuk memperbaiki sanitasi *input user* serta memblokir *request* yang kiranya abnormal dari *business rule* yang sudah ditentukan. Berdasarkan hasil *information gathering*, *cloud server* yang digunakan aplikasi untuk *hosting* sudah menggunakan WAF, yang juga digunakan sebagai *load balancer* nya

- Filterisasi request terhadap *metacharacter*. Hal ini dapat memanfaatkan fungsi *regex matching* (penggunaan `preg_match()` dalam aplikasi berbasis PHP), untuk membolehkan *request* yang tidak memiliki karakter seperti “`</>{}$\\^.*`”, yang kiranya digunakan dalam membuat *payload* XSS. Untuk mencegah *malicious request* yang menggunakan *percent-encoding*, maka URL haruslah *decode* terlebih dahulu dengan UTF-8 sebelum dapat diproses ke dalam *controller* API

Pada rekomendasi dalam kegiatan eksploitasi kedua, perihal Vulnerable URI on `/etc/passwd` dalam objek kerentanan pada tabel 3.5, salah satu prinsip utama dalam mengamankan aplikasi dari serangan *directory listing* maupun *directory traversal* adalah dengan melakukan *restriction access* terhadap direktori apa saja yang dapat diakses oleh publik. Solusi dapat dilakukan baik dari aspek *secure coding practice* ataupun manajemen konfigurasi server sebagai berikut:

- Mematikan fitur *indexing direktori*. Apabila aplikasi tidak membutuhkan fitur tersebut, maka hal ini dapat dilakukan dengan mengatur dalam *main config* dari *web server* tersebut, ataupun menggunakan fungsi `.htaccess` sebagai *config-based directory*, karena secara *default*, *indexing* bersifat *enable (allow from all)*. Berdasarkan hasil eksploitasi dalam kegiatan ketiga, aplikasi sudah menerapkan penggunaan `.htaccess` dalam mematikan *indexing* nya kepada publik
- Tidak menggunakan *absolute path* dalam meng-*import class* ataupun melakukan *routing*. Hal ini termasuk berbahaya karena *root directory* nya merupakan *root* dalam *file system* di *server*. Sedangkan dengan menggunakan

relative path / canonical, root directory nya adalah *root* dari *scope* aplikasi saja, sehingga tidak mengganggu *scope* di luarnya. Adapun dalam aplikasi PHP dengan menggunakan fungsi *realpath()*, yang berguna untuk mematikan *symbolic link* serta *metacharacter* terhadap direktori, layaknya penggunaan *dot-dot-slash*

- Memenjarakan user dalam direktori aplikasi saja. Hal ini juga dikenal dengan konsep *jailing*, yang digunakan untuk mencegah user dalam mengakses *parent directory* di atasnya, yang mana menjadi salah satu implementasi yang efektif untuk mencegah serangan tersebut. Adapun pemanfaatan saran tersebut dengan menggunakan *Chroot Jail* dalam server Apache, serta *open_basedir Protection* dalam aplikasi PHP

Pada rekomendasi dalam kegiatan eksploitasi ketiga, perihal Unrestricted Public Repository dalam objek kerentanan pada tabel 3.6, hal utama yang perlu diperhatikan adalah bagaimana manajemen *access control* yang meliputi *authentication*, *authorization*, serta *accountability* terhadap user dengan repositori aplikasi. Dengan begitu, solusi dapat berupa pendekatan yang dapat dilakukan dalam mengelola repositori agar lebih aman dan *scope* yang jelas sebagai berikut:

- Membuat *visibility* repositori menjadi *private*. Hal ini secara signifikan dapat meminimalisir ter-*expose* nya *source code* aplikasi kepada publik. Dalam konteks *cloud-hosting* repository seperti GitLab ataupun Github, mereka menyediakan fitur *collaborators* walaupun repositori dalam *scope private*, sehingga hanya *user* yang kredibel yang dapat masuk ke dalam repositori tersebut melalui izin dari pemilik repositori itu sendiri

- Pemberian *role* kepada setiap *collaborator* dalam repositori. Fitur ini membantu pemilik repositori dalam melakukan manajemen terhadap otoritas *collaborator*, sehingga terdapat protokol yang sesuai antara satu divisi dengan yang lain. *Role* tersebut berupa *Read*, *Triage*, *Write*, *Maintain*, dan *Admin*, yang masing-masing nya memiliki *permission* yang sesuai. Secara *default*, *role Admin* akan diberikan kepada pembuat repositori, dengan *permission* untuk mengelola keseluruhan objek di dalamnya, baik itu manajemen *source code* maupun *collaborator* nya

Pada rekomendasi dalam kegiatan eksploitasi keempat, perihal Improper Error Handling dalam objek kerentanan pada tabel 3.7, di satu sisi, fitur *error handling* dapat membantu *developer* dalam membantu proses *debugging* dan *maintenance* aplikasi. Meskipun begitu, dalam upaya untuk meminimalisir kebocoran informasi kedalam *scope* publik, maka solusi yang diberikan lebih terkait terhadap fase *deployment* dalam pembangunan aplikasi, yang dijelaskan sebagai berikut:

- Penggunaan *deployment environment* yang tepat. Agar aplikasi siap diakses oleh publik, maka penggunaan *environment* yang tepat adalah *production*, sehingga secara *default* dapat mematikan fungsi *debugging* tersebut. Dengan begitu, aplikasi akan berada dalam *environment* yang sama baik untuk penguji maupun *hacker* dalam menghadapi aplikasi dalam implementasi yang sesungguhnya. Dikarenakan API masih berada dalam tahap *development*, maka penggunaan tipe *production* tidaklah sesuai, sehingga saran berikutnya dibuat agar dapat diaplikasikan untuk keseluruhan penggunaan *environment*

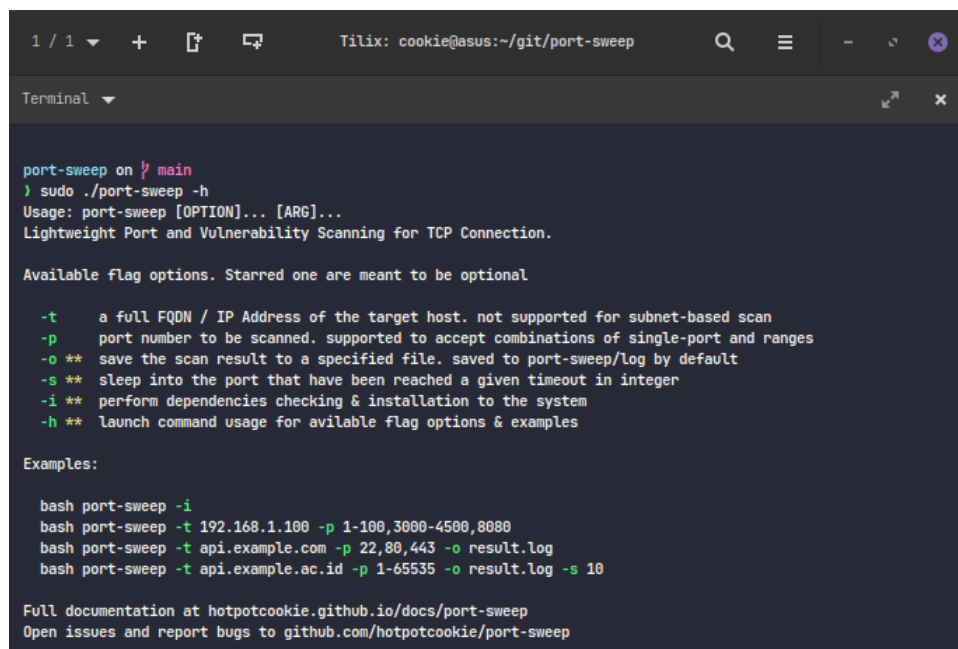
- Menghindari penggunaan *generic response* dalam *exception handler*. Dalam aspek *secure coding practices*, aplikasi diharapkan dapat menspesifikasikan *handler* tertentu untuk mengcover spesifik potensi yang dapat terjadi apabila diperlukan. Adapun penggunaan *logging* yang dapat membantu *audit* apabila adanya suatu *error* yang kiranya tidak dapat di *handle* secara instan, sehingga membutuhkan evaluasi lebih lanjut dari pengembang aplikasi
- Penggunaan *email-based tracing*. Dengan menggunakan fitur ini, maka aplikasi dalam fase *production* pun tetap bisa mendapatkan informasi error tersebut berdasarkan notifikasi *email*. Hal ini dapat meminimalisir adanya potensi kebocoran data kepada publik, dengan mem-*broadcast* notifikasi tersebut kepada user yang memiliki otoritas terhadap aplikasi tersebut. Salah satu cara penggunaannya adalah dengan memanfaatkan fungsi *mail()* dalam aplikasi PHP, yang juga diiringi dengan *service SMTP* maupun *third-party mail service* lainnya

3.3.4 Tools Pendukung Pengujian Kerentanan

Dalam membantu proses kegiatan *port scanning* dalam tahapan *information gathering*, dibuatlah suatu *tools* sederhana bernama *port-sweeper*, untuk mengiringi dan memverifikasi hasil dari *tools* port scanner lain secara komparatif. *Tools* dibuat dalam *Bash Script* dengan berbasiskan CLI (*Command-Line Interface*) yang memanfaatkan integrasi antara *tools* *curl*, *ping* serta *netcat*. *Curl* dan *ping* digunakan untuk mengecek apakah *host target* berstatus aktif, sedangkan *netcat* digunakan untuk melakukan *scanning* pada protokol TCP. Beberapa fitur dasar yang dapat dimanfaatkan dari *tools* ini sendiri adalah instalasi *package* otomatis, melakukan *port scanning* baik dalam bentuk IP maupun domain, support

dalam menggunakan *port range*, melakukan *logging* ke dalam file, serta melimitasi durasi *probing* setiap *port* dengan menggunakan *timeout*.

Berikut merupakan *command usage* yang digunakan sebagai panduan penggunaan *tools* yang dipaparkan pada gambar 3.4, contoh penggunaan scanning menggunakan *multiple port* dan mengeksport *output* ke dalam *log* pada gambar 3.5, serta salah satu fitur untuk instalasi *package* secara otomatis yang dapat dimanfaatkan sebagai penggunaan pertama kali pada gambar 3.6:



```

1 / 1 + [?] [?] Tilix: cookie@asus:~/git/port-sweep
Terminal
port-sweep on ? main
> sudo ./port-sweep -h
Usage: port-sweep [OPTION]... [ARG]...
Lightweight Port and Vulnerability Scanning for TCP Connection.

Available flag options. Starred one are meant to be optional

-t a full FQDN / IP Address of the target host. not supported for subnet-based scan
-p port number to be scanned. supported to accept combinations of single-port and ranges
-o ** save the scan result to a specified file. saved to port-sweep/log by default
-s ** sleep into the port that have been reached a given timeout in integer
-i ** perform dependencies checking & installation to the system
-h ** launch command usage for available flag options & examples

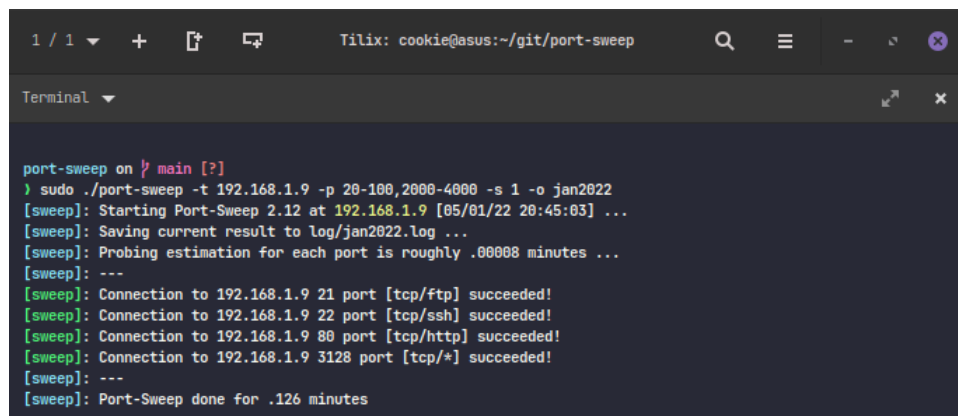
Examples:

bash port-sweep -i
bash port-sweep -t 192.168.1.100 -p 1-100,3000-4500,8080
bash port-sweep -t api.example.com -p 22,80,443 -o result.log
bash port-sweep -t api.example.ac.id -p 1-65535 -o result.log -s 10

Full documentation at hotpotcookie.github.io/docs/port-sweep
Open issues and report bugs to github.com/hotpotcookie/port-sweep

```

Gambar 3.4 Command Usage pada port-sweeper

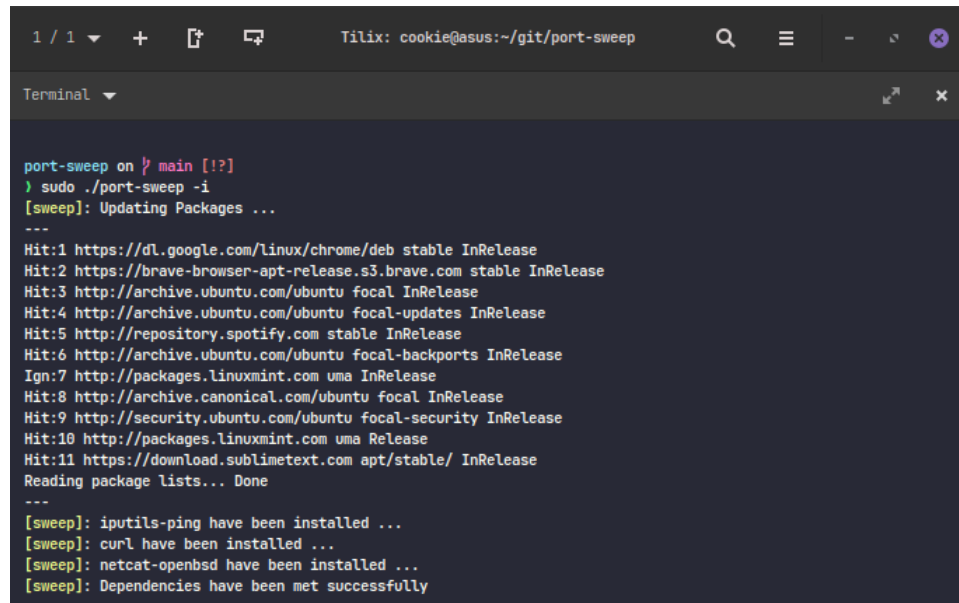


```

1 / 1 + [?] [?] Tilix: cookie@asus:~/git/port-sweep
Terminal
port-sweep on ? main [?]
> sudo ./port-sweep -t 192.168.1.9 -p 20-100,2000-4000 -s 1 -o jan2022
[sweep]: Starting Port-Sweep 2.12 at 192.168.1.9 [05/01/22 20:45:03] ...
[sweep]: Saving current result to log/jan2022.log ...
[sweep]: Probing estimation for each port is roughly .00008 minutes ...
[sweep]: ---
[sweep]: Connection to 192.168.1.9 21 port [tcp/ftp] succeeded!
[sweep]: Connection to 192.168.1.9 22 port [tcp/ssh] succeeded!
[sweep]: Connection to 192.168.1.9 80 port [tcp/http] succeeded!
[sweep]: Connection to 192.168.1.9 3128 port [tcp/*] succeeded!
[sweep]: ---
[sweep]: Port-Sweep done for .126 minutes

```

Gambar 3.5 Demonstrasi penggunaan port-sweeper



```

1 / 1 + [?] [?] Tilix: cookie@asus:~/git/port-sweep
Terminal
port-sweep on ? main [!?]
> sudo ./port-sweep -i
[sweep]: Updating Packages ...
---
Hit:1 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 https://brave-browser-apt-release.s3.brave.com stable InRelease
Hit:3 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:5 http://repository.spotify.com stable InRelease
Hit:6 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Ign:7 http://packages.linuxmint.com uma InRelease
Hit:8 http://archive.canonical.com/ubuntu focal InRelease
Hit:9 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:10 http://packages.linuxmint.com uma Release
Hit:11 https://download.sublimetext.com apt/stable/ InRelease
Reading package lists... Done
---
[sweep]: iputils-ping have been installed ...
[sweep]: curl have been installed ...
[sweep]: netcat-openbsd have been installed ...
[sweep]: Dependencies have been met successfully

```

Gambar 3.6 Fitur auto-installer dalam port-sweeper

Walaupun *output* dari *scanning* dari *tools* ini tidak lebih kompleks maupun informatif dari *tools* yang umum digunakan, seperti Nmap, perancangan *tools* ini memberikan gambaran seperti apa mekanisme *tools scanner* bekerja, dan bagaimana membuat *business logic* yang sesuai dengan fitur yang umumnya ditemukan dalam *tools* lainnya. *Tools port-sweeper* sendiri di *hosting* secara publik dalam repositori GitHub [port-sweep](https://github.com/0x00sec/port-sweep).

3.4 Identifikasi Kendala yang Dihadapi

Selama tiga bulan melaksanakan kegiatan PKL di CV. Solusi Automasi Indonesia dalam divisi *Software Security Developer*, selain mendapatkan ilmu dan pengalaman dalam menggali pengetahuan, adapun kendala-kendala yang dihadapi selama kegiatan tersebut berlangsung. Kendala sendiri sifatnya dapat berupa teknis maupun non teknis, yang akan dijabarkan dalam sub bab berikut.

3.4.1 Kendala Pelaksanaan Tugas

Kendala yang dihadapi selama melaksanakan kegiatan PKL baik teknis maupun non teknis adalah sebagai berikut:

1. Belum adanya pengalaman yang terfokus dalam melakukan *pentest* pada target yang berbentuk REST API

2. Belum ada pengalaman dalam melakukan pentest menggunakan standar PTES secara utuh
3. Terbatasnya durasi pengerjaan karena digunakan pula sebagai pembelajaran materi dari awal
4. Beberapa serangan tidak memberikan *output* yang diinginkan karena kurangnya pengumpulan informasi dan fase *modelling*

3.4.2 Cara Mengatasi Kendala

Adapun kendala-kendala yang disebutkan sebelumnya yang dapat diatasi baik saat pengerjaan, maupun untuk kedepannya sebagai berikut:

1. Mempelajari struktur aplikasi REST API, penggunaan HTTP *request*, serta serangan yang umum ditujukan kepada target baik dalam Jurnal, Artikel, maupun Video secara *online*
2. Membaca dokumentasi resmi dari PTES secara *online* dan mempelajari setiap fase PTES dengan menyesuaikan kegiatan dalam *timeline sprint*
3. Mengalokasikan pembelajaran terhadap materi maupun tugas yang dihadapi di luar dari jam kerja yang disepakati saat *onboarding*
4. Mempelajari aspek-aspek apa saja yang seharusnya dipahami dan dikuasai terlebih dahulu mengenai kondisi target yang akan diuji. Kondisi target yang pada akhirnya menentukan *attack vector* yang akan diberikan dalam *pengujian*, bukan sebaliknya

BAB IV

PENUTUP

4.1 Kesimpulan

Kegiatan *penetration testing* yang dilakukan menggunakan standar PTES selesai dilakukan dengan tepat waktu dari *timeline* dan *goal time estimation* yang sudah ditetapkan sebelumnya oleh para *stakeholder*. Dalam berlangsungnya kegiatan, 2 dari 4 percobaan *testing* yang dilakukan berstatus *true positive* dengan nilai kerentanan yang tergolong sebagai medium, yaitu mengenai kerentanan terhadap *unrestricted public repository* dan *improper error handling*, yang dikategorikan dalam parent node *sensitive data exposure* dan *broken access control*, yang direferensikan dari diagram attack tree pada gambar 3.3. Dengan adanya pendekatan evaluasi sistem yang bersifat struktural dan informatif, maka *report* diharapkan dapat memberikan kontribusi kepada kepala dan rekan divisi dalam memprioritaskan implementasi yang baik dalam rangka mengembangkan keamanan aplikasi yang bersifat responsif dan preventif.

4.2 Saran

Dari kajian ini, adapun saran yang diberikan terhadap pengembangan REST API setelah kegiatan *penetration testing* ini baik yang sudah maupun yang belum diterapkan, yaitu sebagai berikut:

1. Memberikan *proteksi header* yang aman serta melakukan sanitasi yang kuat dalam setiap request
2. Mematikan fitur *indexing* apabila tidak diperlukan dan memberikan *scope* aplikasi yang ketat terhadap akses menuju *server*
3. Melimitasi akses dan *permission* terhadap repositori aplikasi kepada *user*
4. Menggunakan *error handling* yang spesifik demi mencegah kebocoran data yang seharusnya tidak diperlukan untuk di-*expose*

DAFTAR ISTILAH

BurpSuite	: Aplikasi pemodifikasi dan intercept request header
Canonical	: Pengambilan path terpendek dari suatu direktori resource
Confusion Matrix	: Penyimpulan dengan mengklasifikasi hasil suatu kejadian
Curl	: Tool untuk mengambil dan mengirim data dari protokol
DirBuster	: Aplikasi pencari direktori dengan metode bruteforce
Dotdotpwn	: Aplikasi pencari direktori dengan percent-encoding
Google Dork	: Pencarian informasi secara eksklusif dengan Google
Legacy Source Code	: Kode aplikasi yang dibuat untuk inheritance aplikasi lain
Netcraft	: Aplikasi web untuk melakukan DNS enumeration
Nikto	: Aplikasi pencari kerentanan pada suatu target
Nmap	: Aplikasi pencari port yang terbuka pada suatu target
Nslookup	: Tool untuk melakukan query DNS record dari domain
Screaming Frog	: Aplikasi pencari direktori dengan metode crawling
Snippet Source Code	: Potongan sebagian kode dari aplikasi
Stakeholder	: Seluruh pemangku kepentingan dari suatu kegiatan
Trello	: Aplikasi web untuk manajemen proyek dengan scrum

DAFTAR PUSTAKA

- Anonym. (2020). “CVE-CWE-CAPEC Relationships”. MITRE Corporation. [CVE, CWE, and CAPEC Relationships | CVE Mitre](#). [16/12/2021]
- Anonym. (2021). “About the CVE Program”. MITRE Corporation. [CVE Program Overview | CVE Mitre](#). [16/12/2021]
- Anonym. (2021). “About CWE”. MITRE Corporation. [Overview - What is CWE? | CWE Mitre](#). [16/12/2021]
- Anonym. (2015-2021). “Common Vulnerability Scoring System SIG”. FIRST, Inc. [CVSS | FIRST](#). [16/12/2021]
- Anonym. (2021). “The Penetration Testing Execution Standard Documentation”. Rel. 1.1, Hal. 5-17. [The Penetration Testing Execution Standard Documentation](#). [26/12/2021]
- Anonym. (2021). “Missing ‘X-XSS-Protection’ Header”. Tenable, Inc. Sec. Plugin - Web Application Scanning, ID. 112526. [Missing 'X-XSS-Protection' Header | Tenable®](#). [29/12/2021]
- Anonym. (2021). “Debugging Your Application”. Codeigniter Foundation. The Debug Toolbar - Enabling the Toolbar. [Debugging Your Application — CodeIgniter 4.1.5 documentation](#). [29/12/2021]
- Bacudio, G, Yuan, X, Chu, B, Jones, M. (2011). “An Overview of Penetration Testing”. International Journal of Network Security & Its Applications. Vol. 3, No. 6, Hal. 19. [\(PDF\) An Overview of Penetration Testing](#). [14/12/2021]
- Berners-Lee, T. (2005). “Uniform Resource Identifier (URI): Generic Syntax”. IETF Datatracker. Hal. 11. [RFC3986](#). [28/12/2021]
- Das, D. (2021). “Confusion Matrix From a CyberSecurity Analyst Perspective”. Medium Corp. Terminologies and Derivation from Confusion Matrix. [Confusion Matrix From a CyberSecurity Analyst Perspective | by Dipaditya Das | Geek Culture | Medium](#). [01/01/2022]
- Dingsøyr, T, Nerur, S, Balijepally, V, Moe, N. (2012). “A Decade of Agile Methodologies: Towards Explaining Agile Software Development”. The Journal of Systems and Software. No. 9, Hal. 2. [\(PDF\) A decade of agile methodologies: Towards explaining agile software development](#). [17/12/2021]

- Duan, Q, Saini, V, Paruchuri, V. (2008). "Threat Modelling Using Attack Trees". Journal of Circuits, Systems and Computer. Vol. 23, No. 4, Hal. 127. [\(PDF\) Threat Modeling Using Attack Trees](#). [25/12/2021]
- Dzida, G, Wiklicky, R. (2020). "Attack Trees Presentation InfoSec 20/21 UIBK". University of Innsbruck - YouTube. [Attack Trees Presentation InfoSec 20/21 UIBK](#). [25/12/2021]
- Fachri, F, Fadlil, A, Riadi, I. (2021). "Analisis Keamanan Web Server Menggunakan Penetration Test". Jurnal Informatika. Vol. 8, No.2, Hal. 185. [Analisis Keamanan Webserver menggunakan Penetration Test | Fachri | Jurnal Informatika](#). [15/12/2021]
- Hema, V, Thota, S, Kumar, N, Padmaja, C, Krishna, R, Mahender, K . (2020). "Scrum: An Effective Software Development Agile Tool". IOP Publishing. Ser. 981 022060, Hal. 4,6. [Scrum: An Effective Software Development Agile Tool](#). [17/12/2021]
- Kuipers, L. (2020). "Analysis of Attack Trees: fast algorithms for subclasses". Bachelor Thesis Computing Science. Hal. 6. [Analysis of Attack Trees: fast algorithms for subclasses](#). [28/12/2021]
- Manuaba, I, Rudiastini, E. (2017). "API REST Web Service and Backend System of Lecturer's Assessment Information System on Politeknik Negeri Bali". IOP Publishing. Ser. 953 012069, Hal. 3. [\(PDF\) API REST Web service and backend system Of Lecturer's Assessment Information System on Politeknik Negeri Bali](#). [16/12/2021]
- Marashdih, A, Zaaba, F. (2016). "Cross Site Scripting: Detection Approaches in Web Application". International Journal of Advanced Computer Science and Applications . Vol. 7, No. 10, Hal. 157. [Cross Site Scripting: Detection Approaches in Web Application](#) . [28/12/2021]
- Mohanakrishnan, R. (2021). "What is Threat Modelling? Definition, Process, Examples, and Best Practices". Ziff Davis, LLC. [What Is Threat Modeling? Definition, Process, Examples, and Best Practices | Toolbox It-security](#). [25/12/2021]
- Naredo, I, Pardavila, L. (2007). "DNS load balancing in the CERN cloud". IOP Publishing. Ser. 898 062007, Hal. 1. [DNS load balancing in the CERN cloud](#). [29/12/2021]
- Priyatna, B, Hananto, A. (2020). "Implementation of Application Programming Interface (API) in Indonesian Dance and Song Application". Scientific Journal of Information Systems and Informatics. Vol. 2, No. 2, Hal. 47.

[Implementation of Application Programming Interface \(API\) in Indonesian Dance and Song Applications | SYSTEMATICS](#). [16/12/2021]

- Shamay, U. (2020). "5 Bad Coding Habits That Leave Your Source Code Exposed". Spectral Cyber Technologies, Ltd. API Security - Unrestricted Repository Access. [5 Bad Coding Habits That Leave Your Source Code Exposed - Spectral](#). [30/12/2021]
- Sharma, A. (2020). "What Does the New CVSS 3.1 Scoring Model Mean for Enterprise Security?". Sonatype, Inc. [What Does the New CVSS 3.1 Scoring Model Mean for Enterprise Security?](#). [24/12/2021]
- Sunaringtyas, S, Prayoga, D. (2021). "Implementasi Penetration Testing Execution Standard untuk Uji Penetrasi pada Layanan Single Sign-On". Edu Komputika Journal. Vol. 8, No.1, Hal. 49. [Implementasi Penetration Testing Execution Standard Untuk Uji Penetrasi Pada Layanan Single Sign-On | Edu Komputika Journal](#). [15/12/2021]
- Walkowski, D. (2020). "Still Mystified by API? What They Are (Really) and Why They Matter". F5 Labs. [What APIs Are and Why They Matter](#). [16/12/2021]
- Yaqoob, I, Hussain, S, Mamoon, S, Naseer, N, Akram, J, Rehman, A. (2017). "Penetration Testing and Vulnerability Assessment". Journal of Network Communications and Emerging Technologies. Vol. 7, Iss. 8, Hal. 11. [Penetration Testing and Vulnerability Assessment](#). [20/12/2021]

LAMPIRAN

L-1 Sertifikat Keterangan Selesai PKL

CERTIFICATE

Of Internship Completion

MUHAMMAD NUR IRSYAD

Has completed the internship at Automate All
as **Software Security Developer** during 3 month(s) with **excellent work**.

You Hard Work, Dedication, And Achievement Will Be Cherished

Bandung, 20 Desember 2021,



Irfan Nugraha
Chief Technology Officer



KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI JAKARTA

JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER

Jl. Prof. DR. G.A. Siwabessy, Kampus UI, Depok 16425
Telp: (021)91274097, Fax : (021) 7863531, (021)7270036 Hunting
Laman : <http://www.pnj.ac.id>, e-mail : tik.pnj@gmail.com

F8

**BUKU PENGHUBUNG
PEMBIMBING MAGANG INDUSTRI**

Nama Industri : CV. Solusi Automasi Indonesia
Alamat : Jl. Telekomunikasi Terusan Buah Batu, Bandung Techno Park, kawasan Pendidikan Telkom, 40257, Dayeuhkolot, Bandung, Jawa Barat
Judul Magang : Pengujian Kerentanan pada *Development* REST API dalam CV. Solusi Automasi Indonesia
Nama Pembimbing : Irfan Nugraha, S.Kom.

No	Hari / Tanggal	Aktivitas / Tugas	Paraf
1	Kamis – Jum'at 02/09 – 03/09 2021	<ul style="list-style-type: none"> Melakukan onboarding bersama peserta PKL batch 5 dan adanya pembekalan overview job desk 	<i>Irfan</i>
2	Senin – Jum'at 06/09 – 10/09 2021	<ul style="list-style-type: none"> Pemaparan teknis sprint dan backlog pertama dengan metode scrum Pembekalan materi, sprint planning, serta daily standup mengenai identifikasi tools di footprinting 	<i>Irfan</i>
3	Senin – Jum'at 13/09 – 17/09 2021	<ul style="list-style-type: none"> Melakukan tahapan scanning, directory traversal, analisa source code, serta studi terhadap http req 	<i>Irfan</i>
4	Senin – Jum'at 20/09 – 24/09 2021	<ul style="list-style-type: none"> Melakukan sprint review pada card pertama & pengujian code injection, vulnerable URI, serta data exposure pada error handling 	<i>Irfan</i>
5	Senin – Jum'at 27/09 – 01/10 2021	<ul style="list-style-type: none"> Melakukan pengujian reflected XSS, header request dan injection Melakukan sprint review pada card kedua & sprint planning mengenai finalisasi laporan 	<i>Irfan</i>

(lanjutan L-2)



KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI JAKARTA

JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER

Jl. Prof. DR. G.A. Siwabessy, Kampus UI, Depok 16425
Telp: (021)91274097, Fax : (021) 7863531, (021) 7270036 Hunting
Laman : <http://www.pnj.ac.id>, e-mail : tik.pnj@gmail.com

F8

6	Senin – Jum'at 04/10 – 08/10 2021	<ul style="list-style-type: none">Melakukan diskusi dengan rekan untuk format laporan dalam docs. penambahan aspek tingkat rentan serta cara mitigasi serangan nya	<i>I/a</i>
7	Senin – Jum'at 11/10 – 15/10 2021	<ul style="list-style-type: none">Finalisasi laporan untuk seluruh fase pengujian kerentanan hingga tahap reporting untuk divisi backendMelakukan sprint review pada back-log pertama secara keseluruhan	<i>I/a</i>
8	Senin – Jum'at 18/10 – 22/10 2021	<ul style="list-style-type: none">Melakukan sprint planning pada back log kedua mengenai implementasi keamanan pada UiPath Robot & studi untuk pembuatan program kecil	<i>I/a</i>
9	Senin – Jum'at 25/10 – 29/10 2021	<ul style="list-style-type: none">Melakukan percobaan untuk proses deployment robot, fase provisioning, serta manajemen autentikasi dan role & permission dengan instances	<i>I/a</i>
10	Senin – Jum'at 01/11 – 05/11 2021	<ul style="list-style-type: none">Melakukan tahap storing credentials, konfigurasi vpn dan sftp, serta studi mengenai environment workspaceMelakukan spring mengenai manipulasi data dengan Bash	<i>I/a</i>
11	Senin – Jum'at 08/11 – 12/11 2021	<ul style="list-style-type: none">Mengimplementasi package securing dan certificate signing & verificationMelakukan sprint review pada back-log kedua secara keseluruhan	<i>I/a</i>
12	Senin – Jum'at 15/11 – 19/11 2021	<ul style="list-style-type: none">Mendengarkan spring rekan mengenai pengenalan CTF & perancangan logbook dan format laporan	<i>I/a</i>

(lanjutan L-2)



KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI JAKARTA

JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER

Jl. Prof. DR. G.A. Siwabessy, Kampus UI, Depok 16425
Telp: (021)91274097, Fax : (021) 7863531, (021) 7270036 Hunting
Laman : <http://www.pnj.ac.id>, e-mail : tik.pnj@gmail.com

F8

13	Senin – Jum'at 22/11 – 26/11 2021	<ul style="list-style-type: none">Mendengarkan spring rekan mengenai salted hashing & pengembangan tools scanner yang bersifat mandiri	
14	Senin – Kamis 29/11 – 02/12 2021	<ul style="list-style-type: none">Mendengarkan spring rekan mengenai SIEM & SOAR & melakukan closing PKL untuk batch 5	

Bandung, 31 Desember 2021

Chief Technology Officer,

CV. Solusi Informatika
Irfan Nugraha, S.Kom.

NIP 2010000120006

L-3 User Requirement Industri



KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI JAKARTA

JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER

Jl. Prof. DR. G.A. Siwabessy, Kampus UI, Depok 16425
Telp: (021)91274097, Fax : (021) 7863531, (021)7270036 Hunting
Laman :<http://www.pnj.ac.id>, e-mail : tik.pnj@gmail.com

F10

USER REQUIREMENT

(Kepentingan Pengguna/Industri)

Nama Pembimbing : Irfan Nugraha, S.Kom.

Divisi/Departemen : Software Security Developer / IT

No	Modul / Unit	Spesifikasi	Paraf
1	REST API	Kegiatan Footprinting dan presentasi progress	
2	REST API	Kegiatan Code Injection, Sensitive Data Exposure, dan Broken Access Control	
3	Dokumentasi	Finalisasi laporan hasil kegiatan Penetration Testing	
4	Kajian Teknologi	Riset implementasi keamanan pada UiPath Robot	
5	Kajian Teknologi	Melakukan presentasi mengenai Security ataupun IT dalam general	

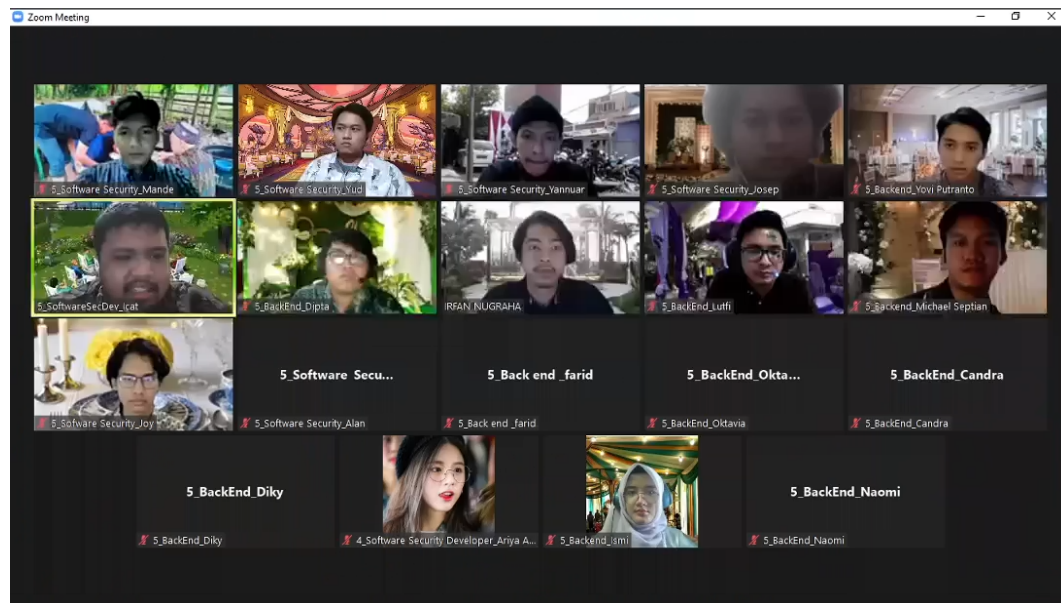
Jakarta, 31 Desember 2021

Pembimbing Industri,

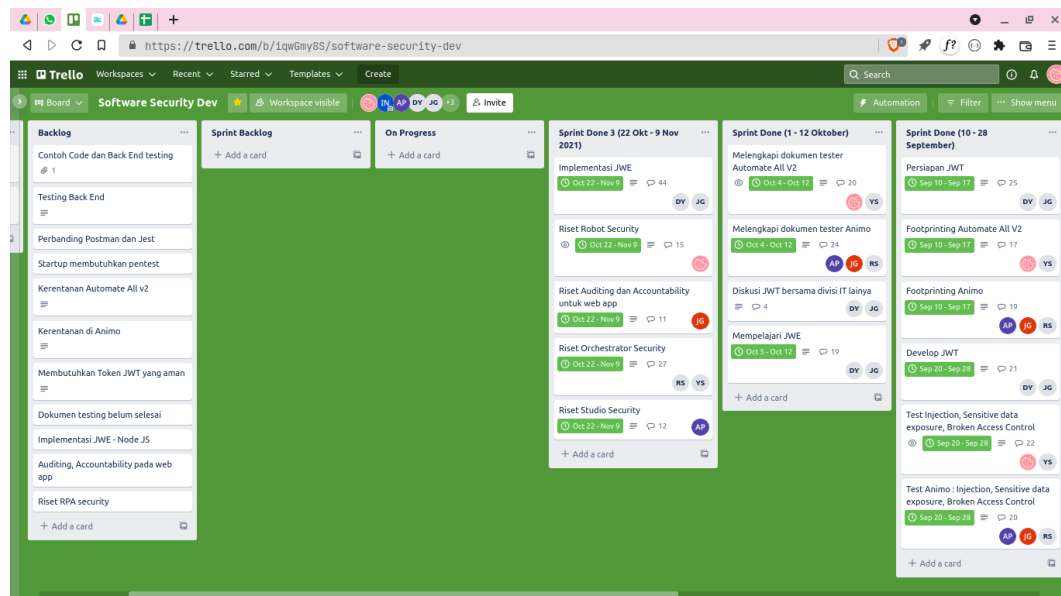
Automate All
CV. Solusi Teknologi Indonesia

NIP 2010000120006

L-4 Lampiran Dokumentasi

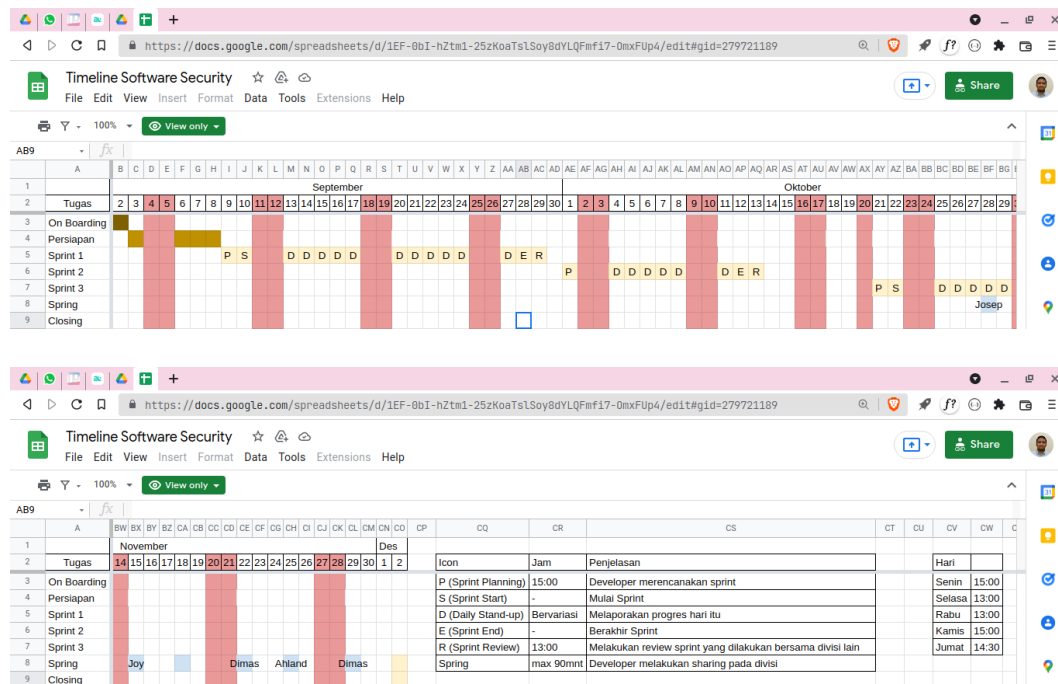


Gambar 1 Onboarding dengan rekan divisi Software Security Developer dan divisi Backend Developer bersama pembimbing industri

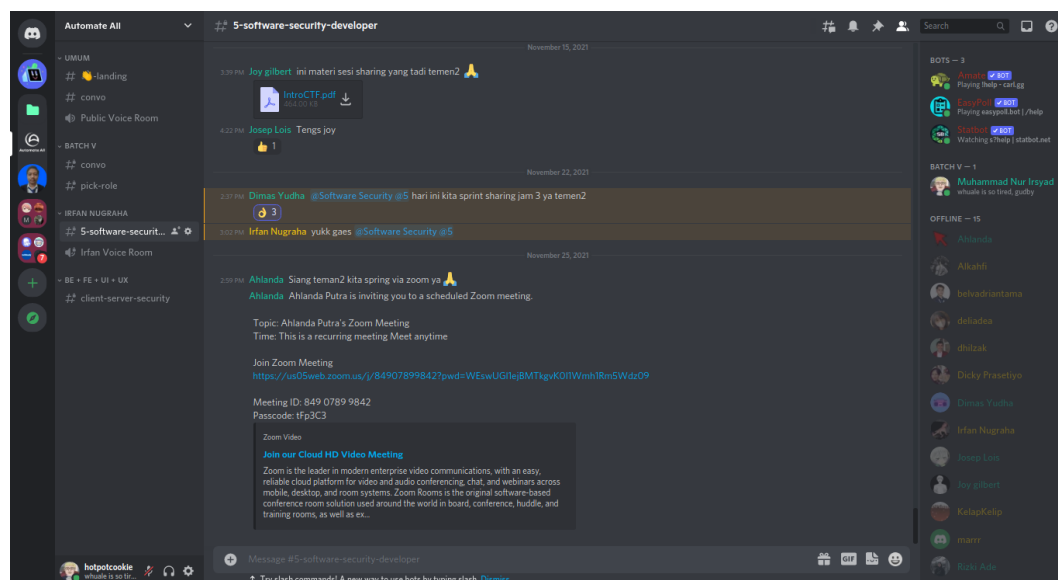


Gambar 2 Manajemen Tugas dalam metodologi Scrum menggunakan Trello

(lanjutan L-4)



Gambar 3 Keseluruhan timeline dan penjadwalan dalam kegiatan magang

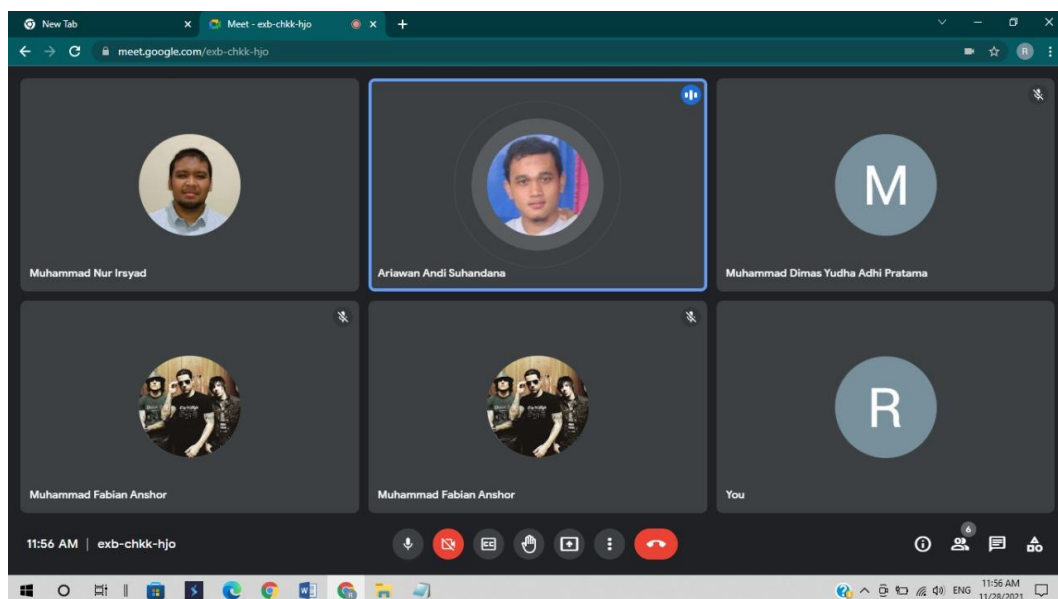


Gambar 4 Kegiatan virtual meeting dan diskusi menggunakan Discord

(lanjutan L-4)



Gambar 5 Closing dengan seluruh rekan magang Batch 5 bersama seluruh petinggi industri



Gambar 6 Bimbingan laporan magang pertama bersama dengan dosen pembimbing