HARDENING SISTEM INFORMASI SEKAWAN V2 MENGGUNAKAN *FRAMEWORK* OWASP

Disusun Oleh:

N a m a : Fadila Ahmad Sya'bani

NIM : 18523107

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

HARDENING SISTEM INFORMASI SEKAWAN V2 MENGGUNAKAN *FRAMEWORK* OWASP

TUGAS AKHIR

Disusun Oleh:

N a m a : Fadila Ahmad Sya'bani

NIM : 18523107

Yogyakarta, 25 Juli 2023

Pembimbing,

(Fayruz Rahma, S.T, M.Eng)

HALAMAN PENGESAHAN DOSEN PENGUJI

HARDENING SISTEM INFORMASI SEKAWAN V2 MENGGUNAKAN *FRAMEWORK* OWASP

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 25 Juli 2023

Tim Penguji

Fayruz Rahma, S.T., M.Eng. _____

Anggota 1

Aridhanyati Arifin, S.T.,M.Cs _____

Anggota 2

Rahadian Kurniawan, S.Kom., M.Kom. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Fadila Ahmad Sya'bani

NIM : 18523107

Tugas akhir dengan judul:

HARDENING SISTEM INFORMASI SEKAWAN V2 MENGGUNAKAN FRAMEWORK OWASP

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 7 Desember 2022

(Fadila Ahmad Sya'bani)

HALAMAN PERSEMBAHAN

Alhamdulillahi rabbil 'alamin, puji syukur ke hadirat Allah Swt. karena atas izin, rahmat, dan karunia-Nya proses pengerjaan tugas akhir dapat selesai dengan baik. Tak lupa saya ucapkan rasa terima kasih kepada kedua orang tua Bapak Hariyadi dan Ibu Dwi Wahyuni untuk sebuah kesempatan menuntut ilmu pada jenjang perguruan tinggi, kepada ibu dosen pembimbing yang telah memberikan arahan dan masukan pada setiap permasalahan yang dihadapi selama menjalani masa penelitian, dan terima kasih juga kepada teman-teman semua dan Tim Pengembang Sekawan v2 atas bantuannya dalam proses penelitian. Semoga Allah Swt. membalas setiap kebaikan yang telah diberikan. Amin.

HALAMAN MOTO

لَا يُكَلِّفُ ٱللَّهُ نَفْسًا إِلَّا وُسْعَهَا

"Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya"

QS. Al Baqarah: 285

"*A bug is never just a mistake. It represents something bigger. An error of thinking. That makes you who you are*"

Elliot Alderson

"*We can't change the past, but we can plan the future*"

Fadila Ahmad Sya'bani

KATA PENGANTAR

Puji dan syukur kepada Allah Subhanahu Wata'ala tuhan yang Maha Esa, Maha Pengasih, dan Maha Penyayang. Shalawat dan salam semoga senantiasa tercurah kepada Nabi Muhammad saw. Karena atas karunia dan nikmat Allah Swt., proses pengerjaan tugas akhir dengan judul "Hardening Sistem Informasi Sekawan V2 Menggunakan *Framework* OWASP" dapat dilaksanakan dan diselesaikan dengan baik.

Penulis menyadari bahwa proses pengerjaan penelitian tidak dapat terlaksana dengan baik apabila tidak ada dukungan dan bantuan dari berbagai pihak. Maka dari itu, dalam kesempatan ini perkenankan penulis dengan rendah hati mengucapkan rasa terima kasih kepada:

- Kedua orang tua dan seluruh keluarga besar yang senantiasa memberikan dukungan.
- Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika - Program Sarjana Universitas Islam Indonesia
- Ibu Fayruz Rahma, S.T, M.Eng. selaku Dosen Pembimbing tugas akhir yang selalu memberikan arahan dan masukan dalam penelitian.
- Tim Pengembang Sekawan V2 (Raihan Rafid, Rayhan Mahardhika Wijaya, Anwaruddin Ridho Novianto, Yuanda Hanif Hisyam, dan Muhammad Ihsan Syafiul Umam) atas bantuan dan bimbingannya pada sistem Sekawan V2.
- Seluruh teman-teman seperjuangan, seangkatan, dan satu almamater yang telah memberikan dukungan moral, penulis ucapkan terima kasih.

Dengan terlaksananya tugas akhir ini semoga dapat memberikan barokah, ilmu pengetahuan, dan manfaat kepada penulis dan semua pihak yang berkenan mengkaji karya ini. Semoga Allah Swt. melimpahkan rahmat, karunia, dan balasan kebaikan bagi semua pihak yang telah membantu. Atas kekurangan dan kekeliruan yang terjadi selama pengerjaan tugas akhir ini, penulis ucapkan permohonan maaf sebesar-besarnya.

Yogyakarta, 25 Juli 2023

(Fadila Ahmad Sya'bani)

SARI

Teknologi sistem informasi telah digunakan dalam berbagai bidang. Salah satu penerapan hal tersebut adalah sistem informasi perguruan tinggi berupa aplikasi berbasis web yang telah digunakan di berbagai universitas dan sekolah di Indonesia. Besarnya jumlah pengguna teknologi tersebut, tidak luput dari pantauan oknum-oknum yang tidak bertanggung jawab. Data-data yang bersifat sensitif berisiko tersebar ke dunia maya, dan dapat digunakan oleh individu ataupun organisasi dengan tujuan yang negatif. Dengan berbagai risiko dan ancaman tersebut, diperlukan tindakan lanjut untuk mencegah hal tersebut terjadi.

Hardening atau pengerasan keamanan adalah salah satu cara untuk mengurangi celah keamanan dan ancaman yang dapat menyerang sistem informasi. Secara garis besar tahap-tahap yang dilakukan adalah dengan mencari celah keamanan pada sistem informasi tersebut baik dengan pengujian ataupun inspeksi kode sumber aplikasi. Lalu dilanjutkan dengan melakukan perbaikan pada sistem tersebut dengan cara mengubah kode atau konfigurasi yang sebelumnya telah diterapkan menjadi yang lebih aman ataupun dengan menambahkan fungsi keamanan lainnya. Langkah terakhir yang ditempuh adalah melakukan pengujian kembali dengan tujuan untuk mengetahui hasil dari perbaikan keamanan yang telah dilakukan.

Proses *hardening* dilakukan dengan kerangka kerja *Open Web Application Security Project* (OWASP). Dua sumber yang digunakan adalah: OWASP Top 10 2021 dan OWASP *Web Security Testing Guide* (WSTG) *Stable Version*: *During Development*. Pemilihan OWASP Top 10 2021 didasarkan pada kelengkapan informasi (deskripsi, pencegahan, contoh skenario serangan, referensi, *mapping* CWE) dan keterbaruan informasi celah keamanan yang terdapat dalam publikasi tersebut. OWASP WSTG-Stable dipilih karena sesuai dengan tahap pengembangan perangkat lunak yang sedang dilakukan pada saat penelitian dilaksanakan, yaitu *during development.*

Pada penelitian ini terdapat empat kategori celah keamanan yang termasuk dalam OWASP Top 10 2021 yang telah ditemukan, diuji, dan diperbaiki pada Sistem Sekawan v2. Pada aplikasi *backend* terdapat satu celah keamanan, aplikasi *frontend* dua celah keamanan, dan server sebanyak dua kerentanan.

Kata kunci: *security*, *hardening*, OWASP TOP 10, OWASP WSTG, *vulnerability*.

GLOSARIUM

Attack surface	jumlah titik atau tempat yang berbeda di mana pengguna yang tidak sah dapat mencoba memasukkan data ke atau mengekstrak data dari lingkungan.
Checksum	blok data berukuran kecil yang berasal dari blok data digital lain untuk tujuan mendeteksi kesalahan yang mungkin telah terjadi selama transmisi atau penyimpanannya.
CWE	*Common Weakness Enumeration* adalah sistem kategori untuk kelemahan dan kerentanan perangkat keras dan perangkat lunak
Fail2ban	kerangka kerja perangkat lunak pencegahan intrusi. Ditulis dalam bahasa pemrograman Python, ini dirancang untuk mencegah serangan *brute force.*
Firewall	suatu sistem yang dirancang untuk mencegah akses yang tidak diinginkan dari atau ke dalam suatu jaringan internal. Perangkat lunak yang digunakan untuk menjaga keamanan jaringan pribadi dengan cara memblokir akses tidak sah ke atau dari jaringan pribadi.
Hardening	mengamankan sistem dengan mengurangi permukaan serangan yang ada dalam desainnya. Sistem Hardening adalah proses mengurangi permukaan serangan dalam sistem sehingga membuatnya lebih kuat dan aman. Ini adalah bagian integral dari praktik keamanan sistem.
Hash	Fungsi *hash* adalah sebuah algoritma matematika satu arah yang memetakan data arbitrer menjadi nilai bit tetap.
IPv4	versi keempat dari Protokol Internet. Protokol ini adalah salah satu protokol inti dari metode internetworking berbasis standar di internet dan jaringan *packet-switched* lainnya.
IPv6	versi terbaru dari Protokol Internet, protokol komunikasi yang menyediakan sistem identifikasi dan lokasi untuk komputer di jaringan dan merutekan lalu lintas di Internet. IPv6 dikembangkan oleh Internet Engineering Task Force untuk menangani masalah kelelahan alamat IPv4 yang telah lama diantisipasi.
Kali Linux	sebuah distribusi sistem operasi Linux yang ditujukan untuk pengujian penetrasi dan audit keamanan tingkat lanjut. Hal tersebut dilakukan

	dengan menyediakan alat umum, konfigurasi, dan otomatisasi yang memungkinkan pengguna untuk fokus pada tugas yang perlu diselesaikan, bukan aktivitas di sekitarnya.
Metavulnerability	sebuah dependensi yang rentan berdasarkan dependensi pada versi rentan dari paket yang rentan.
MIME *type*	*Multipurpose Internet Mail Extensions* atau tipe media berfungsi untuk identifikasi format *file* dan format konten yang ditransmisikan di internet.
Salt	Dalam kriptografi, *salt* adalah sebuah data acak yang ditambahkan dalam *input* fungsi *hash*. Penambahan ditujukan untuk memastikan *output* unik dari fungsi *hash*.
Socket	titik komunikasi dari lalu lintas komunikasi antar proses di dalam sebuah jaringan komputer. Alamat *socket* terdiri atas kombinasi sebuah alamat IP dan sebuah nomor *port*.
SSH	sebuah protokol jaringan kriptografi untuk komunikasi data yang aman dan diberikan akses untuk melakukan perintah dari jarak jauh (*remote*) antara dua jaringan komputer.
Systemd	Sebuah sistem init Linux yang bertugas untuk melakukan manajemen sistem dan layanan.
Terminal	terminal *emulator* Linux adalah antarmuka berbasis teks yang digunakan untuk mengontrol komputer Linux dengan cara menuliskan perintah ke dalam terminal. Juga sering disebut sebagai *shell*, terminal, *console*, *prompt* atau berbagai nama lainnya.
Ubuntu	sistem operasi berbasis Linux Debian yang dapat digunakan pada desktop, server, dan IoT.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi yang semakin pesat telah memengaruhi aktivitas masyarakat. Berbagai macam kegiatan yang semula dilakukan dengan *offline* kini berubah menjadi *online*. Hal tersebut membuat semua orang lebih bergantung kepada internet, mulai dari sektor pendidikan, kesehatan, bisnis, dan hiburan kini beralih menggunakan platform digital. Penggunaan internet yang meningkat juga dimanfaatkan oleh *cybercriminal* untuk melaksanakan aksi mereka. Terhitung sejak awal virus covid-19 muncul ke publik, telah banyak kasus kejahatan siber yang terjadi di berbagai negara (Lallie, et al., 2021). Pelaku tersebut mencari celah-celah kerentanan yang terdapat di dalam teknologi yang sedang digunakan ataupun yang terdapat pada para pengguna untuk dieksploitasi. Beberapa contoh serangan yang dilakukan *cybercriminal* seperti *Ransomware, Phising, Spam, Social Engineering, Vishing,* dan *Smishing* (Fontanilla, 2020).

Dampak yang ditimbulkan akibat serangan siber dapat menjadi sangat merugikan. Di bidang pendidikan, maraknya pembelajaran secara daring menyebabkan kenaikan penggunaan platform *e-learning* dan *video conference* dan hal tersebut memicu munculnya ancaman yang menyamar sebagai platform pendidikan. Tidak sedikit perusahaan di sektor lain juga mengalami hal serupa, dan diperkirakan mengalami kerugian mencapai jutaan dolar. Hal ini dapat terjadi karena perusahaan tersebut membayar uang tebusan agar sistem mereka dapat digunakan kembali (Pawlicka, Michał , Pawlicki, & Kozik , 2021). Dampak dari serangan siber tidak hanya dirasakan oleh organisasi besar, melainkan masyarakat umum juga menjadi sasaran. Kerugian dari serangan tersebut dapat berimbas kepada korban pada sisi: (1) finansial, salah satunya melalui pemerasan, (2) sosial yaitu turunnya rasa kepercayaan akibat maraknya pesan *spam* dan penipuan *online*, dan (3) mental bertambahnya beban pekerjaan atau kehilangan pendapatan di kala pandemi dapat memengaruhi kondisi kesehatan kejiwaan yang mana akan mengakibatkan depresi, penyesalan, dan rasa malu yang dalam (Monteith, et al., 2021). Dengan begitu kemungkinan keberhasilan kejahatan siber yang dilakukan pelaku terhadap korban akan semakin besar.

Kerentanan merupakan suatu kelemahan atau cacat sistem yang dapat menyebabkan terjadinya serangan siber (Blankenship & M, 2020). Kecacatan tersebut dapat berupa kesalahan

pada saat pembuatan desain maupun dalam tahap implementasi. *Vulnerability* dapat menjadi permasalahan keamanan yang dapat mengancam pengguna pada sisi: (1) *Confidentiality* atau kerahasiaan yaitu data hanya dapat diakses oleh pengguna yang memiliki hak akses legal, (2) *Integrity* atau integritas artinya data hanya diubah oleh *user* yang memiliki otoritas, dan (3) *Availability* atau ketersediaan yakni data dapat diakses kapan pun dan di manapun. Sesuai dengan OWASP *Top* 10 *Vulnerabilities* 2021 yang dikelola oleh OWASP *Foundation*, beberapa *bug/vulnerability* yang paling sering terjadi di sebuah web adalah *Broken Access Control*, Kegagalan kriptografi, Injeksi, Desain yang tidak aman, dan kesalahan konfigurasi keamanan (van der Stock, et al., 2021). Kelemahan keamanan seperti *Cross Site Scripting* (XSS) telah ada sejak tahun 90-an dan masih tetap eksis hingga masa kini (Shinde & Ardhapurkar, 2016).

Dengan begitu banyaknya celah keamanan serta ancaman dari para peretas yang tidak bertanggung jawab, diperlukan suatu aksi untuk agar terhindar dari situasi tersebut. Pengerasan sistem merupakan suatu cara untuk menghilangkan hal-hal yang dapat digunakan para peretas untuk mengeksploitasi sistem (Barker, Smid, & Branstad, 2015). Penerapan pengerasan keamanan sistem dapat dilakukan dari sisi *framework* aplikasi yang digunakan dalam pembangunan web (Nur, Na'am, Nurcahyo, & Arlis, 2019), ataupun *hardening* di sisi infrastruktur server dan jaringan yang digunakan sebagai tempat *hosting* aplikasi web tersebut (Olenčin & Perháč, 2021). Memperbaiki keamanan juga dapat dilakukan dengan metode lain yang berfokus pada *data security* dan manajemen risiko (Budiarto, 2017), sehingga potensi kegagalan dapat dihilangkan sejak dini.

Salah satu layanan akademik yang sedang dikembangkan versi terbarunya oleh Prodi Informatika – Program Sarjana Universitas Islam Indonesia adalah Sekawan. Sebagai aplikasi web penunjang administrasi jalur kelulusan, Sekawan memberikan fasilitas berupa layanan pencatatan *logbook*, melihat nilai, serta mengunggah dokumen yang dibutuhkan. Web ini juga membantu dosen pembimbing dan sekretaris prodi dalam melakukan kegiatan akademik yang berkaitan dengan tugas akhir (Program Studi Informatika Program Sarjana, 2021). Dengan adanya layanan ini, Sekawan V2 dapat menjadi salah satu sarana penting yang membantu dalam kegiatan administratif tahun terakhir mahasiswa.

Di dalam penelitian ini, akan dijawab pertanyaan mengenai situasi keamanan Sistem Informasi Sekawan V2 saat ini, cara memperbaiki kelemahan tersebut, dan implementasinya. Sistem Sekawan saat ini sedang dalam masa pengembangan versi terbarunya. Maka dari itu,

penelitian ini diharapkan dapat memberikan kontribusi untuk meningkatkan keamanan Sistem Sekawan V2.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, dapat dirumuskan beberapa rumusan masalah, yaitu:

a. Bagaimana kondisi keamanan pada Sekawan V2 dalam masa pengembangan?

b. Bagaimana cara mengatasi kerentanan yang telah ditemukan?

c. Bagaimana kondisi keamanan Sekawan V2 setelah proses pengerasan keamanan?

1.3 Batasan Masalah

Penelitian dilakukan dengan menerapkan batasan masalah. Ruang lingkup permasalahan dibatasi agar tidak terlalu luas atau lebar. Dengan begitu penelitian yang dilakukan akan lebih terfokus.

a. Penelitian dilakukan pada tahap pengembangan perangkat lunak dalam WSTG-*Stable*.
b. Pencarian kerentanan berdasarkan kategori OWASP Top 10 2021.
c. Penelitian dilakukan pada sisi aplikasi dan *server*.

1.4 Tujuan Penelitian

Tujuan dari penelitian Hardening Sistem Sekawan v2 Menggunakan Framework OWASP adalah mengamankan sistem Sekawan v2 dari *bugs* atau kesalahan dalam kode dan konfigurasi yang dapat mengakibatkan terjadinya celah keamanan serta serangan dari pihak luar.

1.5 Manfaat Penelitian

Proses pengerasan keamanan yang dilakukan diharapkan memiliki pengaruh positif terhadap keamanan sistem Sekawan V2. Manfaat utama dari *hardening* adalah menjamin keamanan data-data yang diakses melalui web Sekawan V2 sesuai dengan *triad* CIA yaitu:

a. *Confidentiality* (Kerahasiaan).
b. *Integrity* (Integritas).
c. *Availability* (Ketersediaan)

1.6 Sistematika Penulisan

Pada penelitian ini, sistematika penulisan digunakan agar laporan menjadi lebih sistematis dan terurut sehingga lebih mudah dipahami. Secara garis besar, penelitian ini terdiri atas lima bab dengan rincian:

1.6.1 BAB 1 PENDAHULUAN

Bab 1 Pendahuluan membahas mengenai latar belakang penelitian, rumusan masalah, batasan-batasan yang dihadapi, tujuan, manfaat, dan sistematika penulisan yang digunakan selama kegiatan tersebut berlangsung.

1.6.2 BAB 2 LANDASAN TEORI

Bab 2 Landasan Teori memaparkan tentang teori ilmiah pada penelitian. Hal tersebut berisi mengenai definisi dan referensi yang digunakan selama penelitian berlangsung. Penggunaan landasan teori bertujuan untuk memperkuat landasan atau dasar dari penelitian itu sendiri.

1.6.3 BAB 3 METODOLOGI PENELITIAN

Bab 3 Metodologi Penelitian membahas mengenai metode yang digunakan dalam penelitian. Pembahasan tersebut akan mengkaji secara umum tentang tahap-tahap yang dilaksanakan selama penelitian.

1.6.4 BAB 4 HASIL DAN PEMBAHASAN

Bab 4 Hasil dan Pembahasan mengulas tentang hasil yang ditemukan dan pembahasannya. Hasil diperoleh melalui tahap sebelumnya dipaparkan pada bab ini. Pembahasan juga dilakukan pada hasil dengan menjabarkan keadaan yang terjadi pada hasil tersebut.

1.6.5 BAB 5 KESIMPULAN

Tahapan terakhir dalam penelitian Hardening Sistem Informasi Sekawan V2 Menggunakan OWASP adalah pembuatan kesimpulan. Penarikan kesimpulan didasarkan pada hasil dan pembahasan yang telah dilakukan.

BAB II
LANDASAN TEORI

Bab 2 Landasan Teori mengulas mengenai teori dan gagasan yang digunakan pada penelitian. Pada bab tersebut membahas teori aplikasi web, sistem informasi akademik, pengerasan keamanan, OWASP *Top* 10 2021, OWAS WSTG-*Stable,* perangkat keras, perangkat lunak, dan penelitian serupa dari berbagai sumber.

2.1 Dasar Teori

Pengujian dan pengerasan keamanan aplikasi web menggunakan framework keamanan adalah pendekatan terstruktur untuk melindungi aplikasi dari serangan siber. Pengujian mengidentifikasi kerentanan, sementara pengerasan menerapkan langkah-langkah pencegahan yang disediakan oleh *framework* keamanan siber. Hal ini bertujuan untuk memastikan aplikasi web aman dan dapat mengatasi risiko serangan dengan efektif.

2.1.1 Aplikasi Web

Aplikasi web merupakan sebuah perangkat lunak yang berjalan dan diakses oleh pengguna menggunakan peramban yang terhubung dengan internet. *Software* tersebut umumnya digunakan sebagai *e-commerce*, web mail, sosial media, dan hiburan. Komponen yang digunakan untuk menjalankan aplikasi web berupa: web server yang berfungsi untuk mengelola *request*, aplikasi server untuk menyelesaikan *request*, dan basis data sebagai penyimpanan (TechTarget Contributor, 2019).

Menurut (Martin, 2022) perangkat lunak berbasis web memiliki kelebihan:

- Dapat dirilis kapanpun. Tidak perlu untuk mengingatkan pengguna untuk melakukan *update* aplikasi.
- Dapat digunakan pada semua platform, seperti Windows, Mac, dan Linux.
- Dapat diakses menggunakan perangkat komputer maupun perangkat bergerak.

Namun, pada sisi lain juga memiliki kekurangan, antara lain:

1. Keamanan yang tidak terjamin, sehingga berisiko terjadi pelanggaran akses.
2. Aplikasi belum tentu dapat diakses dan didukung oleh semua peramban.
3. Memiliki cakupan terbatas untuk mengakses fitur pada perangkat.

2.1.2 Sistem Informasi Akademik

Di masa kini, penggunaan platform digital berbasis web dapat membantu masyarakat dalam mengakses informasi mengenai sekolah. Hal ini tentu memudahkan siswa, guru, dan orang tua dalam melakukan kegiatan akademis karena informasi dapat diakses melalui internet, sehingga tidak perlu repot-repot untuk datang ke sekolah hanya untuk memperoleh informasi. Sebelumnya pelajar dan orang tua harus datang ke sekolah, hanya untuk memperoleh informasi, yang tentu hal tersebut memerlukan waktu serta ongkos yang tidak sedikit (Susanti, 2016).

Di dalam dunia pendidikan sendiri, kebutuhan akan informasi dapat diartikan sebagai kemampuan, syarat atau kriteria yang harus ada dan dipenuhi oleh sistem informasi. Sistem informasi akademik sendiri sudah diterapkan oleh perguruan tinggi di Indonesia. Tujuan utama dari implementasi hal tersebut adalah untuk mempermudah penyampaian informasi kepada peserta didik dari tenaga pengajar maupun tenaga administrasi (Kurniawan & Riadi, 2018).

2.1.3 Pengerasan Keamanan

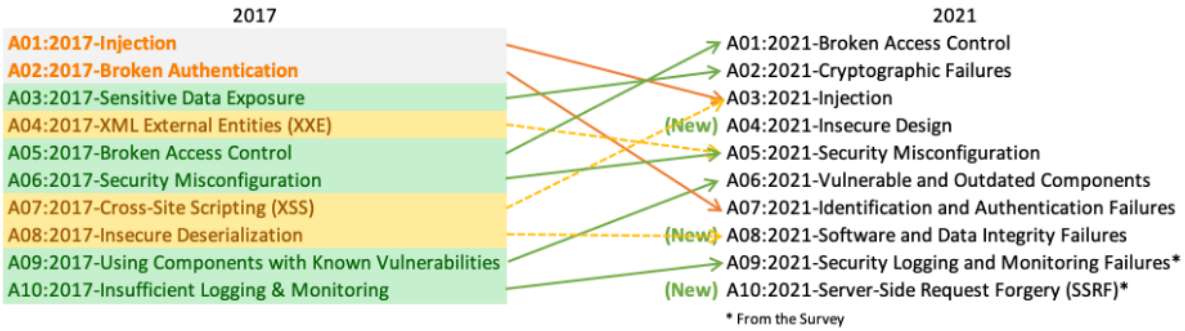
Keamanan merupakan salah satu aspek penting dalam sebuah sistem informasi. Dengan banyaknya ancaman dari luar maupun dalam, keamanan seringkali luput dari daftar hal-hal yang dianggap penting pada sistem. Sebagai contoh, apabila sistem mengalami serangan *Denial of Service* (DOS), pelayanan yang diberikan oleh server akan terganggu. Hal tersebut merupakan salah satu gangguan yang mengancam *availability* atau ketersediaan akses dari sistem tersebut (Rahardjo, 2005).

Dalam hal keamanan, *hardening* atau pengerasan merupakan suatu metode yang dilakukan dengan tujuan untuk memperkuat keamanan suatu sistem, baik web maupun server. *Hardening* diperlukan agar dapat menghindari kerugian-kerugian yang dapat terjadi pada pihak pengelola sistem dan *client* yang menggunakan layanan tersebut (Prabowo, Darusalam, & Ningsih, 2020).

2.1.4 OWASP *Top* 10 *Web Security Vulnerability* 2021

Open Web Application Security Project merupakan salah satu komunitas *open-source* yang bekerja di area keamanan aplikasi web. Salah satu publikasi yang dibuat dan sering dipakai oleh pengembang aplikasi dan pemerhati keamanan adalah OWASP Top 10. Hal tersebut adalah laporan celah keamanan web yang diperbaharui secara berkala setiap empat tahun sekali dengan fokus sepuluh celah keamanan paling kritis yang sering terjadi. Publikasi tahun 2021 menjadi versi terbaru yang dikeluarkan oleh OWASP, dengan beberapa perbedaan dari terbitan pada tahun-tahun sebelumnya. Pada OWASP Top 10 terdahulu data-data yang dikumpulkan hanya berfokus pada 30 CWE (*Common Weakness Enumeration*), sedangkan pada OWASP

Top 10 2021 telah terdapat hampir 400 CWE yang telah dianalisis. Data-data tersebut diperoleh melalui sumbangan, *bug bounty*, dan jual beli. (Torsten, et al., 2021). Pada Gambar 2.1 terlihat bahwa OWASP Top 10 telah mengalami perubahan seiring dengan perkembangan teknologi.



Gambar 2.1 Perubahan OWASP Top 10 2017 ke 2021

Daftar pemeriksaan keamanan dibuat berdasarkan informasi pada OWASP *Top* 10 *Web Security Vulnerability* 2021. Dalam publikasi tersebut terdapat sepuluh celah keamanan yang dibahas, dengan rincian pada Tabel 2.1:

Tabel 2.1 OWASP Top 10 2021

Rank	*Vulnerability*
1	*Broken Access Control*
2	*Cryptographic Failures*
3	*Injection*
4	*Insecure Design*
5	*Security Misconfiguration*
6	*Vulnerable and Outdated Components*
7	*Identification and Authentication Failures*
8	*Software and Data Integrity Failures*
9	*Security Logging and Monitoring Failures*
10	*Server-Side Request Forgery (SSRF)*

Pada masing-masing celah keamanan, terdapat tujuh subkonten yang berisi tentang:

a. *Factors*
b. *Overview*
c. *Description*
d. *How to Prevent*
e. *Example Attack Scenarios*
f. *References*

g. *List of Mapped CWEs*

Berikut rincian celah keamanan pada OWASP *Top* 10 2021:

Broken Access Control

Kontrol akses memberlakukan kebijakan sehingga pengguna tidak dapat bertindak di luar izin yang diberikan. Kegagalan biasanya menyebabkan terjadinya kebocoran informasi yang tidak sah, modifikasi, atau penghancuran semua data atau melakukan fungsi bisnis di luar batas pengguna.

Cryptographic Failures

Kesalahan dalam kriptografi dalam sistem. Enkripsi umumnya diterapkan pada data *at transit* (data yang sedang dikirimkan) dan data *in rest* (data yang sedang disimpan). Kelalaian yang umumnya terjadi yakni penggunaan protokol yang belum menerapkan enkripsi (HTTP, SMTP, Telnet, dan FTP) dan penggunaan algoritma yang sudah usang atau lemah pada operasi enkripsi dan *hash*.

Injections

Kerentanan injeksi, terjadi ketika masukan yang diberikan oleh pengguna tidak divalidasi dan disanitasi dengan benar. Beberapa serangan yang berkaitan dengan injeksi: SQL *Injection* (SQLi), *Cross Site Scripting* (XSS), dan *Buffer Overflow*.

Insecure Design

Desain yang tidak aman merujuk pada kurangnya keefektifan atau tidak adanya desain pada akses kontrol. Salah satu penyebab utama dalam terjadinya kerentanan ini adalah kurangnya informasi mengenai profil risiko bisnis pada perangkat yang sedang dikembangkan. Hal itu menyebabkan kegagalan dalam menentukan tingkat pengamanan yang diperlukan di perangkat tersebut.

Security Misconfiguration

Kesalahan dalam melakukan konfigurasi keamanan mengakibatkan tingkat kerentanan aplikasi dan server meningkat terhadap ancaman. Kelalaian yang umumnya terjadi: fitur atau layanan yang tidak diperlukan belum dinonaktifkan, sistem masih menggunakan kredensial *default*, pesan *error* yang terlalu informatif kepada *end-user*.

Vulnerable and Outdated Components

Menggunakan komponen-komponen yang telah kadaluarsa atau usang dapat meningkatkan risiko sistem terkena serangan. Hal tersebut dikarenakan informasi mengenai kelemahan perangkat lunak tersebut sudah beredar luas di internet. Sistem operasi, kerangka

kerja, dan *library*/dependensi yang digunakan merupakan beberapa komponen yang rentan di sistem.

Identification and Authentication Failures

Penyerang dapat mengeksploitasi proses identifikasi dan autentikasi sistem. Serangan dan kerentanan yang umum terjadi: serangan *bruteforce* dengan daftar kredensial yang telah diketahui, mengizinkan penggunakan kata sandi lemah, dan pemulihan kredensial yang tidak efektif.

Software and Data Integrity Failures

Integritas data dan perangkat lunak berfungsi untuk memastikan bahwa komponen yang digunakan adalah orisinil dan tidak mengalami perubahan. Kegagalan dalam penanganan hal tersebut dapat mengekspos sistem terhadap serangan. Contoh serangan yang dapat terjadi adalah penyerang dapat mengunggah pembaruan mereka sendiri untuk didistribusikan dan dijalankan.

Security Logging and Monitoring Failures

Proses *logging* dan *monitoring* berfungsi untuk mendeteksi, meningkatkan, dan menanggapi apabila terjadi pelanggaran atau kesalahan di dalam sistem. Kegagalan dalam proses ini dapat mengakibatkan tidak terdeteksinya pelanggaran keamanan. Kesalahan yang umum terjadi: kurang atau tidak adanya pencatatan *login* dan *login* gagal, peringatan dan eror tidak memberikan *log response* yang jelas, dan arsip *log* hanya disimpan dalam sistem lokal.

***Server-Side Request Forgery* (SSRF)**

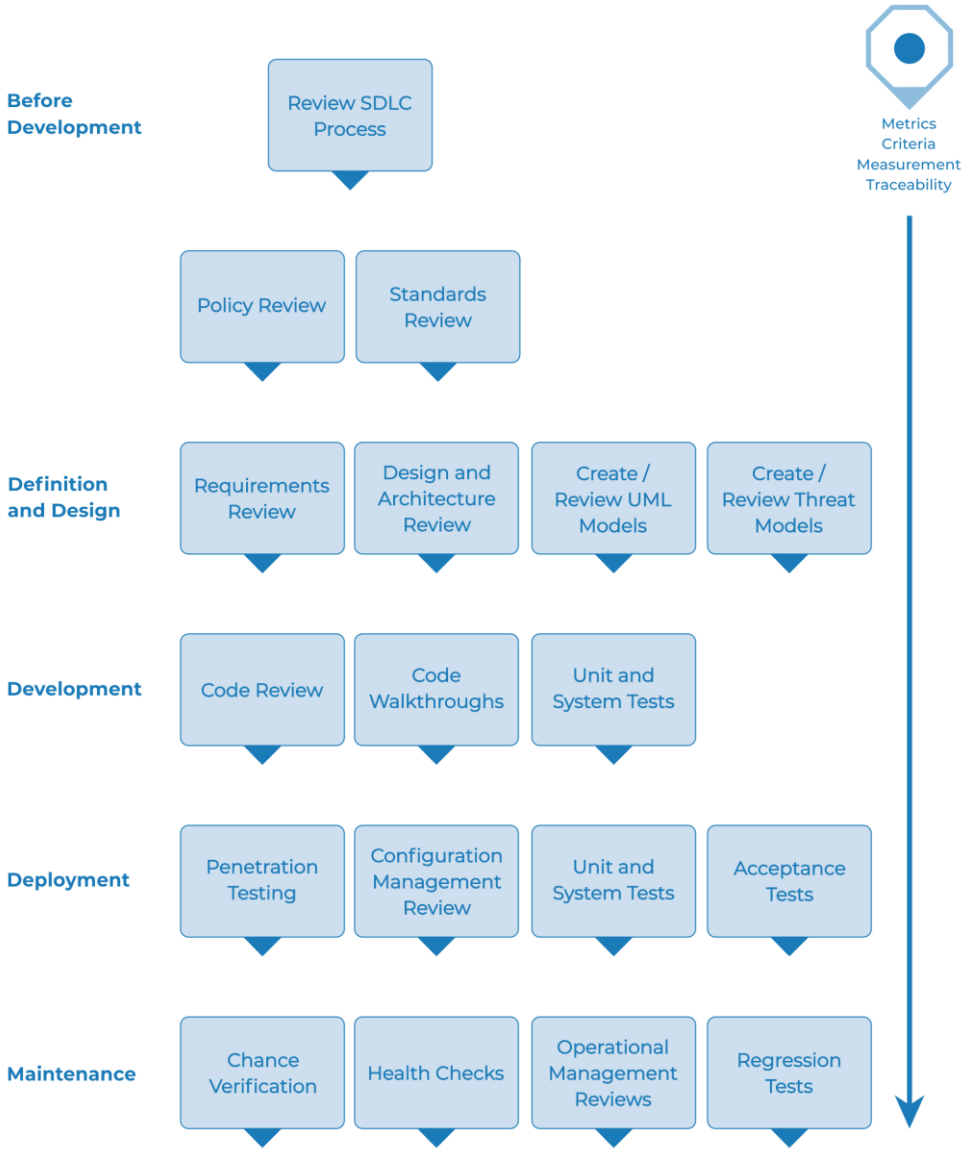
Kerentanan SSRF atau *Server-Side Request Forgery* dapat dimanfaatkan oleh penyerang untuk memanipulasi aplikasi pada sisi server untuk melakukan *request* ke lokasi yang tidak diinginkan. Kerentanan tersebut dapat terjadi ketika akses kontrol tidak memvalidasi koneksi dibuat kembali ke server itu sendiri.

Penggunaan kerangka kerja keamanan mempunyai beberapa manfaat yang dapat diperoleh, antara lain: struktur terarah, kepatuhan regulasi, reputasi, dan resiko lebih rendah. Beberapa *framework* keamanan lain yang dapat digunakan, antara lain adalah CWE (*Common Weakness Enumeration*), NIST *Cybersecurity Framework*, ISO 27001, SANS Top 20 *Critical Security Control*, dan PCI DSS. Kerangka kerja OWASP Top 10 2021 secara spesifik dipilih karena materi yang dimiliki komprehensif dan terkini, pendekatan yang lebih praktis, informasi yang detail, dan telah diuji oleh berbagai pihak jika dibandingkan dengan *framework-framework* lain yang telah disebutkan.

2.1.5 OWASP WSTG-*Stable*: *Phase 3 During Development*

The Web Security Testing Guide adalah salah satu proyek dari OWASP yang mengeluarkan panduan pengujian keamanan siber untuk pengembang aplikasi web dan para professional. WSTG dibuat dari upaya kolaborasi dari professional keamanan siber dan sukarelawan. Kerangka kerja ini menyediakan prosedur kerja praktik yang digunakan oleh para penguji penetrasi dan organisasi di seluruh dunia (wstgbot, 2020).

Framework WSTG telah tersedia dengan berbagai versi, salah satunya adalah versi *stable*. Kerangka kerja pengujian ini dibuat berdasarkan atas fase-fase yang sesuai pada proses *Software Development Life Cylces* (SDLC):



Gambar 2.2 Alur *Software Development Life Cycle* (SDLC) dalam OWASP WSTG-*Stable*

Berdasarkan Gambar 2.2, terdapat lima bagian dalam pengembangan perangkat lunak, yaitu:

- *Before Development Begins*
- *During Definition and Design*
- *During Development*
- *During Deployment*
- *During Maintenance and Operations*

Dengan Sekawan V2 yang sedang dalam proses pengembangan, tahap yang sesuai dan digunakan pada *The Web Security Testing Guide* versi stabil adalah *During development* atau tahap pengembangan perangkat lunak. Dalam tahap tersebut terdapat dua fase yang dilakukan:

Code Walkthrough

Aktivitas *Code Walkthrough* adalah pemeriksaan kode sumber secara umum dengan tujuan untuk mendapatkan informasi mengenai aplikasi.

Code Review

Tahap *Code Reviews* merupakan fase di mana kode sumber diperiksa berdasarkan kriteria yang telah ditetapkan, dalam hal ini adalah *security checklist*.

2.1.6 *Security Checklist*

Security Checklist adalah sebuah daftar pemeriksaan keamanan yang berisi mengenai informasi celah keamanan yang telah diketahui. Informasi tersebut berupa deskripsi kerentanan, contoh kasus, dan upaya pencegahan.

2.2 Perangkat Keras

Menurut Sri Rahayu, perangkat keras adalah sekumpulan piranti fisik yang dapat dilihat, disentuh, dan diraba. Fungsi dari *hardware* adalah untuk menjalankan perintah dalam bentuk program (Rahayu, 2016).

Definisi *hardware* atau perangkat keras menurut Badan Pengembangan dan Pembinaan Bahasa Kemdikbud adalah:

a. Barang-barang yang terbuat dari logam (pesawat televisi, proyektor, dan peralatan lain) yang berkaitan dengan suatu sistem.
b. Peralatan fisik (komputer dan sebagainya).
c. Perangkat dan peranti yang mendukung sistem komputer

(Badan Pengembangan dan Pembinaan Bahasa, Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia, n.d.).

Perangkat keras pada penelitian digunakan sebagai *platform* untuk melakukan pengujian, simulasi, dan menjalankan aplikasi untuk penelitian.

2.3 Perangkat Lunak

Terdapat beberapa definisi *software* atau perangkat lunak. Menurut ISO/IEC *software* adalah keseluruhan bagian dari program, prosedur, aturan, dan dokumentasi terkait dari sistem pemrosesan informasi (the International Electrotechnical Commission & the International Organization for Standardization, 2015). Sementara itu menurut Badan Pengembangan dan Pembinaan Bahasa Kemdikbud, perangkat lunak adalah:

a. Perangkat program, prosedur, dan dokumen yang berkaitan dengan suatu sistem (misalnya sistem komputer).
b. Bagian dari alat (komputer dan sebagainya) yang berfungsi sebagai penunjang alat utama.
c. Program atau aplikasi yang diisikan ke dalam memori internal komputer (Badan Pengembangan dan Pembinaan Bahasa, Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia, n.d.).

Penggunaan perangkat lunak tidak spesifik terpaku dengan satu vendor atau spesifikasi tertentu saja. Perangkat lunak yang memiliki fungsionalitas serupa juga dapat digunakan. Beberapa perangkat lunak yang digunakan selama penelitian:

2.3.1 Kali Linux

Kali Linux adalah sebuah sistem operasi GNU/Linux yang dibangun dengan basis dari Debian Linux dan ditujukan untuk *penetration testing* dan audit keamanan. Sistem Operasi Kali telah disesuaikan dengan kebutuhan *penetration tester* profesional untuk membantu aktivitas yang dilakukan dengan menyediakan alat, konfigurasi, dan otomatisasi yang dibutuhkan, sehingga memungkinkan pengguna untuk lebih fokus ke tugas yang dilakukan (g0tmi1k, 2022). Fitur-fitur yang disediakan oleh Kali Linux:

A. Lebih dari 600 alat uji penetrasi.
B. Gratis, pengunduhan tidak dipungut biaya.
C. Lisensi sumber terbuka (*Open-Source*).
D. Kepatuhan *File Hierarchy Standard* (FHS).
E. *Custom kernel*, telah mendapatkan *patch* dari injeksi.

Sebagai sebuah distro yang berfokus pada pengujian keamanan dan audit keamanan, Kali Linux telah memiliki berbagai perangkat lunak yang siap digunakan untuk hal tersebut. Beberapa perangkat lunak terpasang dalam Kali Linux yang digunakan dalam penelitian: Nmap dan Hydra.

2.3.2 Ubuntu Server

Sistem Operasi Ubuntu adalah OS yang berbasis Linux Debian dan dikembangkan oleh perusahaan Cannonical. Salah satu varian dari sistem operasi tersebut adalah Ubuntu Server yang mana digunakan untuk perangkat server. Perbedaan mendasar antara versi yang digunakan pada desktop umum dan server adalah tidak tersedianya *Graphical User Interface* (GUI) pada *server*, sehingga dalam penggunaannya seluruh operasi dilakukan dengan menggunakan *Command Line Interface* (CLI) dengan *shell* seperti *bash*, *zsh*, *fish*, dll (Canonical Ltd., 2022).

2.3.3 Google Chrome

Google Chrome adalah sebuah peramban yang dikembangkan oleh Google (Google LLC, 2022). Fitur-fitur yang ditawarkan oleh peramban tersebut antara lain:

a. *Google Address Bar*. Pada fitur bilah alamat tersebut dapat digunakan untuk melakukan pencarian, perhitungan, konversi mata uang, penerjemahan bahasa, dll.
b. *Password Check*. Peramban Google Chrome tidak hanya dapat membuat kata sandi, namun juga dapat melakukan pemeriksaannya.
c. *Sync*. Chrome juga dapat melakukan sinkronisasi *bookmarks,* kata sandi yang disimpan, dan info pembayaran pada berbagai perangkat.
d. *Dark Mode*. Berbagai tema dan warna dapat diaplikasikan pada peramban dan juga telah tersedia mode gelap.
e. *Tabs*. Fitur tersebut dapat membantu pengguna untuk mengatur halaman dan tugas.
f. *Extension*. Fitur tersebut tersedia pada peramban Google Chrome berfungsi untuk menambah fungsionalitas pada peramban.

2.3.4 DBeaver

DBeaver adalah perangkat lunak *open-source* yang berfungsi untuk manajemen basis data bagi pengembang dan administrator *database*. *Software* ini dapat digunakan untuk memanipulasi data dalam bentuk *spreadsheet,* membuat laporan analisis berdasarkan catatan dari berbagai penyimpanan data, dan melakukan ekspor informasi dalam berbagai format. Selain penggunaan tersebut, DBeaver juga memiliki fitur: dukungan untuk sumber data *cloud*, dukungan untuk standar keamanan perusahaan, integrasi dengan Microsoft Excel dan Git, dan tersedia dalam berbagai platform (DBeaver, 2022).

Penggunaan perangkat lunak DBeaver terdapat pada proses pengujian. Proses pengujian yang dilakukan adalah dengan melakukan injeksi kode ke dalam penyimpanan basis data. Maka

dari itu, untuk melihat hasil dari pengujian tersebut, diperlukan alat untuk melakukan manajemen basis data.

2.3.5 Postman

Perangkat lunak Postman adalah *platform* API (*Application Programming Interface*) untuk membangun dan menggunakan API. Penggunaan perangkat lunak ini memudahkan pengembang dengan menyederhanakan siklus hidup API. Beberapa fitur dalam Postman antara lain: kemudahan dalam penyimpanan, kolaborasi, dan manajemen artefak API dalam satu tempat. Postman juga menyertakan kumpulan peralatan yang dapat memudahkan mempercepat pengembangan API, mulai dari proses desain, pengujian, dokumentasi, dan pembuatan *prototype* (Postman, Inc, 2022).

Software Postman digunakan pada saat proses pengujian berlangsung. Kode dari API yang telah melalui proses *hardening* akan diuji menggunakan Postman. Skenario yang diuji adalah respon API dalam menghadapi *input* data dan *request* yang berbahya oleh pengguna.

2.3.6 Visual Studio Code

Dalam melakukan aktivitas pemrograman diperlukan perangkat lunak yang digunakan sebagai alat untuk menulis kode, salah satu *software* yang sering digunakan adalah Visual Studio Code. Program tersebut adalah sebuah kode yang dapat digunakan oleh pengembang perangkat lunak untuk melakukan *edit*, *build,* dan *debug* kode (Microsoft, 2022). Vscode dapat digunakan dalam sistem operasi Windows, MacOS, dan Linux. Keunggulan dari Visual Studio Code adalah terdapat dukungan bawaan untuk bahasa pemrograman Javascript, TypeScript, dan Node.js. Kode editor tersebut juga memiliki dukungan ekstensi untuk bahasa dan *runtime* lain seperti C++, C#, Java, Python, Go, .NET (Microsoft, 2022).

Kode editor Visual Studio Code digunakan dalam proses *hardening*. Perangkat lunak tersebut digunakan pada proses *code review*, peneliti melakukan pemeriksaan, perbaikan, dan penambahan fitur keamanan dalam kode sumber aplikasi.

2.4 Penelitian Terkait

Peneliti melakukan studi literatur terhadap 14 jurnal ilmiah yang diterbitkan antara tahun 2017 hingga 2022. Dari artikel-artikel tersebut, dipilih sebanyak enam artikel dikarenakan terdapat kesamaan dalam hal metode penelitian atau perangkat lunak yang digunakan.

Tabel 2.2 Penelitian terkait

Tahun	Metode	Hasil	Celah Keamanan	Penulis
2017	*Vulnerability Assesment* dan *Penetration Testing*	Pengujian manual memberikan hasil yang lebih banyak dan mendetail mengenai kerentanan dibandingkan dengan pengujian manual.	Arachni, Burpsuite, OWASP Zap	Prof. Sangeeta Nagpure; Sonal Kurkure
2020	*Hardening* dan *Testing*	Pengerasan keamanan terbukti dapat meminimalisir dan mencegah serangan, sehingga penyerang tidak dapat mendapatkan akses ke sistem.	Akses ilegal	Reza Rivaldo Fakhry
2020	*Penetration Testing*	Proses pengujian penetrasi dapat digunakan untuk mencari dan memvalidasi adanya celah kerentanan yang ada pada sistem.	SQL Injection, XSS, dan port TCP terbuka	I Gede Ary Suta Sanjaya; Gusti Made Arya Sasmita; Dewa Made Sri Arsa
2020	*Hardening*	Hardening sistem lebih baik dilakukan dalam berbagai komponen atau lapisan-lapisan dalam sistem, agar keseluruhan bagian sistem terlindungi.	*Privileged Escalation*, *port exploitation*, dan *Denial of Service*.	Maraghi Agil Prabowo; Ucuk Darusalam; Sari Ningsih
2020	JSON Web Token dan Enkripsi	Penambahan fitur enkripsi pada token JWT (JSON Web Token) berhasil menambah keamanan walaupun waktu komputasi lebih lama.	Kesalahan kriptografi	Raymond Cahyadi Saputra; YB Dwi Setianto
2022	OWASP	Pengujian dengan panduan kerangka kerja lebih terarah dan terstruktur	OWASP Top 10 2017	Buket Erşahin; Mustafa Erşahin

Dengan merujuk pada Tabel 2.2, dapat ditarik kesimpulan:

Menurut Nagpure, dkk (2017) proses untuk menemukan celah keamanan dalam web aplikasi dengan menggunakan metode pengujian manual dapat memperoleh hasil lebih baik. Jenis celah keamanan ditemukan: XSS, SQL *Injection*, Clickjacking, *File Upload*, *Browser*

Cache, *Directory Traversal*, *Authentication Bypass*, dan CSRF. Sementara itu, pengujian dengan menggunakan *software* pengujian otomatis tidak dapat memperoleh hasil pengujian sebanyak pengujian manual.

Fakhry (2020) melakukan penelitian seputar pengerasan keamanan server berbasis Linux Ubuntu. Pada penelitian tersebut dilakukan pengamanan jaringan dengan menerapkan perangkat lunak *Intrusion Detection System* (IDS) *Portsentry, port knocking* dan *Honeypot* Cowrie pada server. Pada penelitian tersebut hasil pemindaian port tidak dapat menemukan port yang telah disembunyikan, sehingga informasi mengenai sistem dapat tersembunyi dengan baik. Kesimpulan yang dapat ditarik adalah proses *hardening* pada jaringan server dapat meminimalisir atau mengurangi tingkat kerawanan terhadap penyerang.

Berdasarkan penelitian oleh Sanjaya, dkk (2020) pengujian keamanan terhadap web Lembaga X Menggunakan *Framework* ISSAF, tahap-tahap *hacker* dalam penyerangan web dapat dibagi menjadi sembilan fase: *information gathering, Network Mapping, Vulnerability Identification, Penetration Tesing, Gaining Acccess & Privilege Escalation, Enumerating Further, Compromise Remote User/Sites, Maintaining Access, Covering Tracks.* Hasil yang diperoleh berupa celah keamanan berupa SQL *Injection*, XSS, dan *port* TCP terbuka.

Menurut Prabowo, dkk (2020) server komputer memiliki peran vital dalam mengatur keamanan jaringan, mengatur pengguna, membatasi akses, dan menyediakan layanan. Studi ini mencoba menerapkan konfigurasi keamanan pada platform server Linux dengan pendekatan manual dan audit. Meskipun langkah-langkah keras ini berhasil meningkatkan perlindungan di lapisan permukaan dan paling dalam, tetapi lapisan tengah juga harus diperkuat untuk meraih keamanan menyeluruh.

Penelitian oleh Erşahin, dkk (2022) panduan dan standar dari OWASP bisa membantu pengembang dan organisasi menuju aplikasi web yang aman. *Security Knowledge Framework* (SKF) dapat mendisiplinkan pengembang dan meningkatkan kesadaran keamanan. Melacak kerentanan dependensi aplikasi juga penting dan alat pelacakan *Common Vulnerability Exposure* (CVE) harus digunakan. Dukungan dari SKF diharapkan dapat menciptakan aplikasi web yang lebih aman dan mengurangi usaha mengikuti standar keamanan yang berkembang.

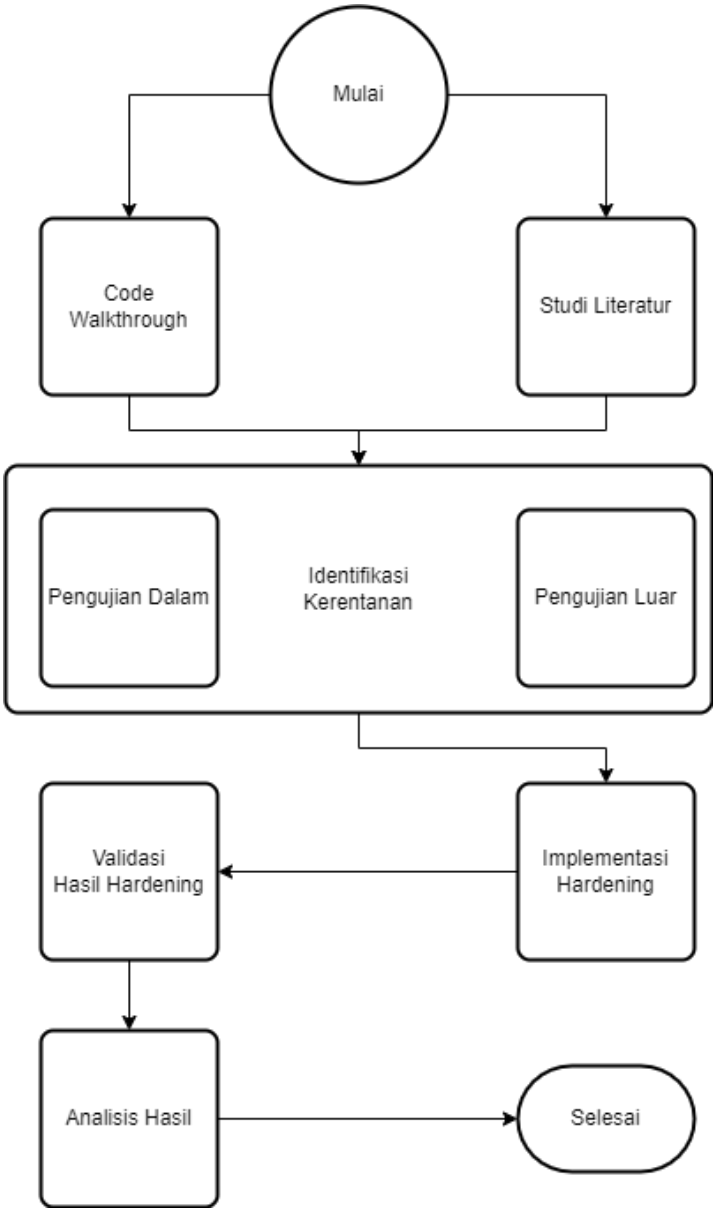
Menurut Saputra, dkk (2020) dalam rangka meningkatkan keamanan layanan, penggunaan token dalam konteks JSON Web Token (JWT) telah terbukti menjadi solusi yang efektif. Melalui mekanisme otentikasi yang diimplementasikan, setiap layanan yang disediakan menjadi lebih terlindungi dari akses yang tidak sah. Penggunaan token ini juga dikombinasikan dengan enkripsi tambahan menggunakan metode AES, yang mampu memberikan lapisan

perlindungan ekstra terhadap data yang dikirim melalui layanan. Meskipun ada peningkatan sedikit dalam waktu yang dibutuhkan untuk mengenkripsi pesan, langkah ini masih memberikan nilai positif dalam hal keamanan dan melindungi integritas data. Dengan demikian, penggunaan token dan enkripsi dalam konteks ini memberikan hasil yang menguntungkan dan mampu mengatasi ancaman keamanan yang dapat timbul.

Artikel-artikel ilmiah tersebut mempunyai persamaan dan perbedaan dengan penelitian *Hardening* Sistem Informasi Sekawan V2 Dengan Menggunakan *Framework* OWASP. Persamaan pada artikel-artikel tersebut, selain dari metode dan alat yang digunakan, aplikasi yang diuji telah melalui fase *deployment.* Perbedaan pada penelitian ini dan penelitian lainnya terdapat pada fase pengembangan perangkat lunak dan pada masing-masing penelitian tersebut tidak mencakup keseluruhan sistem, hanya salah satu komponen, server saja contohnya.

BAB III
METODOLOGI PENELITIAN

Penelitian dilakukan dengan melewati delapan tahap yang secara garis besar dapat dilihat pada Gambar 3.1. Penelitian Hardening Sistem Informasi Sekawan V2 menggunakan panduan *The Web Security Testing Framework* oleh OWASP (wstgbot, 2020). Hal tersebut juga didukung dengan pembuatan daftar pemeriksaan keamanan berdasarkan OWASP Top 10 2021. Berikut ini adalah gambaran alur penelitian



Gambar 3.1 Diagram alur penelitian

Penjelasan secara mendetail mengenai tahapan-tahapan yang dilakukan, sebagai berikut:

3.1 Studi Literatur

Pada tahapan ini, studi literatur dilakukan dengan tujuan untuk mencari dan mengumpulkan makalah-makalah ilmiah yang akan digunakan untuk mendukung dan memperjelas penelitian *Hardening* Sekawan V2. Selanjutnya literatur ilmiah yang dikumpulkan dan akan dilakukan penyeleksian informasi berdasarkan dua kategori yang telah ditentukan. Kategori tersebut antara lain:

a. *Offensive*
b. *Defensive*

Literatur ilmiah akan dimasukkan ke dalam kategori *offensive* apabila di dalam artikel tersebut memuat informasi seputar serangan dan eksploitasi sistem. Sementara itu, kategori *defensive* akan berisikan artikel yang mengandung informasi seputar pengamanan atau perbaikan sistem dari serangan. Literatur akan dikategorikan dalam *other* jika di dalam literatur tersebut tidak membahas mengenai eksploitasi maupun perbaikan. Tujuan dari pengelompokan literatur ilmiah adalah sebagai referensi dalam penelitian. Kategori *offensive* digunakan sebagai acuan dalam tahap pengujian dan kategori *defensive* sebagai acuan dalam proses pengerasan keamanan. Output dari tahapan ini digunakan untuk referensi pada tahap identifikasi kerentanan. Salah satu hasil dari studi literatur yang dilakukan adalah mendapatkan beberapa contoh code program yang rentan terhadap celah keamanan yang terdapat pada *security checklist*.

3.2 Code Walkthrough

Peneliti bersama dengan tim pengembang melakukan *code walkthrough* dengan melihat kode secara umum di mana pengembang dapat menjelaskan logika dan aliran kode yang diimplementasikan. Tujuan dari kegiatan ini adalah untuk mendapatkan pemahaman umum tentang kode, aliran, *layout*, dan struktur kode yang membentuk aplikasi. Pada fase *Code Walkthrough* juga diterapkan untuk memeriksa konfigurasi pada server, untuk mendapatkan informasi mengenai server dan keamanannya yang sudah diterapkan. Output dari tahapan ini adalah pemahaman tentang cara kerja dan spesifikasi sistem yang digunakan.

3.3 Identifikasi Kerentanan

Tahap ini dilakukan setelah peneliti memahami cara kerja sistem melalui tahap code walkthrough sebelumnnya. Identifikasi celah keamanan pada aplikasi adalah proses penting

dalam mengamankan sistem digital. Celah keamanan mengacu pada titik lemah dalam aplikasi yang dapat dieksploitasi oleh penyerang untuk mendapatkan akses yang tidak sah atau merusak integritas data. Melalui analisis mendalam, para ahli keamanan mencari berbagai jenis celah, seperti SQL *injection*, *cross-site scripting* (XSS), dan kerentanan terkait autentikasi yang dapat memberi celah bagi serangan. Dengan mengidentifikasi celah-celah ini, pengembang dapat mengambil langkah-langkah pencegahan yang diperlukan untuk meminimalkan risiko dan melindungi aplikasi serta data pengguna dari ancaman *cyber*. Dalam penelitian ini metode untuk mendeteksi celah kerentanan yang ada, dapat dilakukan dengan dua metode, yakni pengujian dalam dan pengujian luar. Pada tahap ini, contoh kode yang rentan pada security checklist yang diperoleh pada tahap studi literatur akan diperiksa. Output pada tahap ini adalah kerentanan sistem yang ditemukan. Berikut ini adalah rincian dari dua metode teresbut:

3.3.1 Pengujian Dalam

Pengujian dalam merupakan pengujian yang dilakukan dengan memeriksa kode sumber atau konfigurasi dari sistem. Dengan landasan informasi yang diperoleh melalui *code walkthrough* mengenai struktur kode atau konfigurasi dan alasan di balik penggunaan kode tersebut, dapat dilakukan pengujian dalam atau dapat disebut *code review* terhadap kode sumber secara langsung untuk mencari celah keamanan. Pemeriksaan *source code* dilakukan dengan menerapkan daftar pemeriksaan keamanan yang telah dibuat terhadap kode sumber dan konfigurasi *server*.

3.3.2 Pengujian Luar

Pengujian luar merupakan pengujian yang dilakukan tanpa mengetahui kode sumber. Pada penelitian ini, proses pengujian dilakukan sebanyak dua kali. Pengujian luar dilakukan sebelum proses pengerasan keamanan dilakukan. Hal tersebut bertujuan untuk memperoleh data dari sudut pandang penyerang eksternal. Pengujian dilaksanakan dengan melakukan *penetration testing* dengan referensi pada *security checklist* yang telah dibuat ataupun dengan referensi yang berasal dari sumber eksternal.

3.4 Implementasi *Hardening*

Kerentanan sistem yang didapatkan pada tahap sebelumnya, akan dilakukan perbaikan pada tahap ini. Pada tahap implementasi pengerasan keamanan, terdapat dua hal yang menjadi fokus utama, yaitu *hardening* pada sisi aplikasi web dan server. Pada aplikasi web Sekawan dilakukan tinjauan kode sumber untuk memastikan keamanan aplikasi pada bagian *frontend* maupun *backend*. Apabila terdapat kode yang rawan menimbulkan celah keamanan maka akan

dilakukan perbaikan kode. Fokus lain implementasi *hardening* terdapat pada sisi server. Proses yang akan dilakukan adalah pemeriksaan keamanan sistem operasi dan jaringan. Apabila terdapat konfigurasi yang dijalankan pada sistem operasi dan jaringan yang digunakan dapat menimbulkan kerentanan keamanan, dilakukan perbaikan dengan mengganti konfigurasi tersebut dengan keamanan yang lebih baik.

Hal-hal teknis yang dilakukan selama proses implementasi dapat menggunakan panduan yang berasal dari berbagai sumber. Salah satu sumber tersebut adalah bagian *how to prevent* yang telah tersedia pada *security checklist* ataupun informasi yang tersedia pada sumber resmi kerangka kerja, bahasa pemrograman, ataupun perangkat lunak yang digunakan untuk membangun aplikasi. Output dari tahap ini adalah sistem yang telah dilakukan proses hardening, sehingga sistem dianggap kuat terhadap permasalahan *security*.

3.5 Validasi Hasil Hardening

Proses yang dilakukan setelah proses implementasi *hardening* selesai adalah dengan melakukan pengujian. Tujuan dari dilakukannya pengujian keamanan adalah untuk mengetahui hasil dari proses implementasi yang telah dilakukan.

Dalam *cybersecurity* pengujian keamanan dapat berupa kegiatan *penetration testing*, yaitu menguji kerentanan yang ada dengan mengeksploitasi celah keamanan tersebut. Penguji dapat mengeksploitasi kerentanan dengan melakukan serangan terhadap aplikasi dengan metode *scripting* ataupun menggunakan *tools* yang telah tersedia luas di internet. Hasil validasi yang diperoleh kemudian dianalisis untuk menentukan keberhasilan proses pengerasan keamanan Sekawan. Luaran yang diperoleh pada proses ini dapat digunakan untuk menentukan status keamanan aplikasi web Sekawan v2 setelah proses *hardening*.

3.6 Alat yang Digunakan

Alat atau perangkat yang digunakan selama penelitian mencakup perangkat keras (*hardware*) dan perangkat lunak (*software*). Berikut ini adalah rincian perangkat yang digunakan:

3.6.1 Laptop

Spesifikasi *hardware* yang digunakan selama penelitian berlangsung tercantum pada Tabel 3.1:

Tabel 3.1 Spesifikasi perangkat keras yang digunakan

Komponen	Spesifikasi

Jenis perangkat	Laptop
Central Processing Unit (CPU)	AMD Ryzen 5 4600H with Radeon Graphics 6 Cores (12 CPUs), ~3.0GHz
Random Access Memory (RAM)	16GB DDR4 3200MHz
Graphics Processing Unit (GPU)	Nvidia GeForce GTX 1650
Penyimpanan	SSD 512GB
Sistem Operasi	Windows 10

Berdasarkan Tabel 3.1, perangkat keras yang digunakan berjenis laptop. Perangkat tersebut menggunakan prosesor AMD Ryzen 5 4600H dengan GPU bawaan Radeon Graphics. Prosesor tersebut mempunyai enam inti fisik dan 12 prosesor logis serta berjalan dengan kecepatan ~3.0GHz. RAM yang digunakan berukuran 16GB berjenis DDR4 dan mempunyai kecepatan 3200MHz. Pada sisi pemrosesan gambar, laptop tersebut menggunakan GPU eksternal Nvidia GeForce GTX 1650. Penyimpanan yang dipakai berukuran 512GB berjenis *Solid State Drive* (SSD) dan menggunakan Sistem Operasi Windows 10. Perangkat keras tersebut digunakan untuk menjalankan aplikasi Sekawan V2 yang sedang dikembangkan dan sebagai lab pengujian virtual.

Spesifikasi perangkat keras yang telah disebutkan merupakan spesifikasi rekomendasi dan bukan spesifikasi tetap. Maka dari itu sistem Sekawan V2 juga dapat berjalan di perangkat yang memiliki spesifikasi di bawahnya.

3.6.2 Perangkat Lunak

Penelitian dilakukan dengan menggunakan berbagai macam perangkat lunak Tabel 3.2. *Software* yang digunakan terpasang pada server, Sistem Operasi Kali Linux, dan Sistem Operasi Windows 10. Penggunaan masing-masing perangkat lunak terdapat pada tahap *code review*, pengujian pertama, implementasi *hardening*, dan Validasi Hasil Hardening. Terdapat enam perintah Linux pada Tabel 3.2, karena perintah tersebut juga merupakan *executable* atau sebuah perangkat lunak.

Tabel 3.2 Perangkat lunak dan penggunaannya dalam penelitian

Perangkat Lunak	Penggunaan
DBeaver	Digunakan pada proses pengujian pertama dan kedua untuk mengetahui hasil uji dalam *database.* Terpasang pada Windows 10

Google Chrome	Peramban digunakan pada *code review* untuk memeriksa pesan kesalahan kustom dan pengujian untuk memeriksa *http response header*. Terpasang pada Sistem Operasi Windows 10.
NPM	Manajer paket dari NodeJS yang dipasang dalam server untuk melakukan audit dependensi yang digunakan pada tahap *code review.* Terpasang pada Ubuntu server.
Apt	Manajer paket Ubuntu. Digunakan pada tahap code review untuk mencari *unused package* dan implementasi hardening untuk *uninstall* paket. Terpasang pada Ubuntu server.
Fail2ban	Perangkat keras yang digunakan pada tahap implementasi *hardening.* Berfungsi untuk melakukan pemblokiran apabila terjadi kesalahan *login* berulang-ulang. Terpasang pada Ubuntu server.
Journalctl	Digunakan untuk memeriksa *log* sistem pada tahap *code review.* Terpasang pada Ubuntu server.
hydra	*Software* pada Sistem Operasi Kali Linux untuk melakukan *bruteforce password* ssh pada pengujian pertama.
ls	Pada tahap *code review*, perintah ls berfungsi untuk memeriksa perizinan suatu *file* atau direktori.
passwd	Perintah passwd pada Linux berfungsi untuk mengganti atau menerapkan kata sandi untuk pengguna. Digunakan pada implementasi *hardening.*
Pm2	Manajer proses untuk NextJS, diperiksa pada tahap *code review*.
systemctl	systemctl dapat digunakan untuk memeriksa dan mengontrol status sistem "systemd" dan manajer layanan. Digunakan pada proses *code review*
ufw	*Uncomplicated Firewall* digunakan untuk memanipulasi atau memeriksa pengaturan *firewall* di Ubuntu pada tahap *code review*.
Postman	Perangkat lunak yang digunakan pada tahap pengujian pertama dan kedua untuk melakukan *request* terhadap Sekawan-API.
Visual Studio Code	Digunakan pada saat proses *code review* untuk melihat kode sumber dari aplikasi.

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini akan dikaji mengenai hasil dari tahapan-tahapan yang dilakukan dan pembahasannya. Proses penelitian dimulai dengan mencari informasi tentang sistem Sekawan V2 pada tahap Code Walkthrough, pembuatan *security checklist*, lalu pemeriksaan kerentanan melalui *code review* dan pengujian pertama, implementasi *hardening* dan tahap terakhir adalah Validasi Hasil Hardening untuk memastikan bahwa proses *hardening* telah berhasil.

4.1 Code Walkthrough

Aplikasi web Sekawan v2 saat ini dalam masa pengembangan dan akan memasuki fase *deployment* dalam waktu dekat. Kegiatan yang dilakukan oleh Tim Pengembang Sekawan v2 berupa penambahan fitur, pengujian fitur, perbaikan kode, dan melakukan konfigurasi server. Berdasarkan aktivitas *code walkthrough* dapat diperoleh informasi mengenai web aplikasi yang dibuat dan server yang digunakan. Aplikasi sistem informasi Sekawan v2 terdiri atas dua bagian, yaitu aplikasi *front-end* atau aplikasi yang dapat diakses oleh pengguna, aplikasi *back-end* yakni aplikasi yang mengelola proses bisnis, dan *server* yang digunakan sebagai platform *hosting* untuk menjalankan kedua aplikasi tersebut. Detail informasi Sekawan v2:

Aplikasi *Back-end* Sekawan v2

a. Aplikasi bernama Sekawan-API.
b. Bahasa pemrograman utama Go.
c. Dibangun secara *native* (Tidak memakai kerangka kerja seperti GORM, Laravel, Codeigniter, dll).
d. Menggunakan paradigma *Model, View,* dan *Controller* (MVC).
e. Proses autentikasi menggunakan JSON Web Token (JWT).
f. DBMS : Postgresql

Alur kerja Sekawan-API dapat dibagi menjadi tiga bagian, yakni: *controller*, *service*, dan *repository*. Bagian *controller* bertugas untuk menerima *request* dan *response* dari *endpoint* API, *service* berfungsi untuk melakukan proses bisnis sesuai dengan fitur yang diimplementasikan, dan *repository* bertugas untuk melakukan transaksi dengan basis data yang digunakan dalam server Sekawan v2.

Aplikasi *Front-end* Sekawan v2

a. Aplikasi bernama Sekawan-FE.
b. Bahasa pemrograman utama adalah Typescript.
c. Menggunakan Javascript Runtime Node.js.
d. Menggunakan manajer paket NPM (*Node Package Manager*) dan YARN.
e. Menggunakan kerangka kerja NextJS.
f. Proses autentikasi menggunakan Next-Auth.

Server

a. Sistem Operasi : Ubuntu 18.04.6 LTS (*Bionic Beaver*)
b. RAM : 1.8 GB
c. *Processor* : Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz
d. Inti CPU : 1
e. Web *Server* : Nginx
f. Menggunakan pm2 sebagai manajemen proses untuk Node.js.

4.2 *Security Checklist*

Berdasarkan tujuh subkonten yang tersedia pada Tabel 2.1, terdapat tiga subkonten yang akan dijadikan sebagai daftar pemeriksaan keamanan yang akan digunakan pada tahap *code review*. Subkonten *Description* dipilih karena bagian tersebut menyediakan informasi tentang deskripsi celah keamanan dan penyebab umum terjadinya kerentanan. Kedua, pemilihan *How to Prevent* didasarkan pada panduan teknis yang dapat membantu dalam tahap implementasi *hardening*. *Example Attack Scenarios* dipilih dikarenakan subkonten tersebut memberikan informasi mengenai metode yang dapat digunakan oleh penyerang untuk mengeksploitasi celah keamanan yang bersangkutan. Hasil *security checklist* yang telah dibuat terdapat pada lampiran.

Tabel 4.1 Tabel pemeriksaan keamanan

Rank	OWASP Top 10 2021	Sekawan-API	Sekawan-FE	Server
1	*Broken Access Control*	✓	✓	✓
2	*Cryptographic Failures*	✓	✓	✓
3	*Injection*	✓	✓	
4	*Insecure Design*			✓
5	*Security Misconfiguration*	✓	✓	✓

6	*Vulnerable and Outdated Components*	✓	✓	✓
7	*Identification and Authentication Failures*			✓
8	*Software and Data Integrity Failures*	✓	✓	✓
9	*Security Logging and Monitoring Failures*	✓	✓	✓
10	*Server-Side Request Forgery* (SSRF)	✓		

Pada Tabel 4.1 terlihat komponen-komponen Sistem Informasi Sekawan V2 yang diperiksa berdasarkan OWASP Top 10 2021. Tanda centang (✓) menandakan komponen tersebut melalui proses pemeriksaan dan sebaliknya jika tidak ada maka komponen tersebut tidak diperiksa karena alasan tertentu. Celah keamanan *Injection* tidak diperiksa pada server karena hanya dapat terjadi apabila aplikasi atau layanan yang berjalan melakukan *input* pada *system shell*. Pada kerentanan peringkat empat *Insecure Design*, Sekawan-API dan Sekawan-FE tidak dilakukan *code review* maupun pengujian karena penelitian dilakukan hanya pada tahap pengembangan perangkat lunak dan tidak ikut serta pada proses desain. Kerentanan peringkat tujuh *Identification and Authenticatioin Failures* pada Sekawan-API dan Sekawan-FE tidak melalui proses *code review* dan pengujian karena proses autentikasi pada aplikasi telah menggunakan mekanisme *Single Sign On* (SSO). Kerentanan *Server-Side Request Forgery* hanya diperiksa pada bagian Sekawan-API karena pada komponen tersebut yang menjalankan proses bisnis di sistem Sekawan V2.

4.3 Identifikasi Kerentanan

Dalam upaya mengamankan sistem aplikasi web yang kompleks, identifikasi kerentanan keamanan pada berbagai komponen menjadi tahap penting. Komponen-komponen ini mencakup aspek seperti basis data, server web, lapisan antarmuka pengguna, serta berbagai modul fungsional yang mendukung fungsionalitas aplikasi. Proses identifikasi kerentanan pada komponen-komponen ini melibatkan analisis mendalam terhadap cara komponen berinteraksi dan memproses data. Pada tahap ini menggunakan security checklist yang dibuat berdasarkan *framework* OWASP Top 10 sebagai panduan. Pembahasan akan dilakukan berdasarkan komponen. celah kerentanan, dan tahapan identifikasi kerentanan. Terdapat tiga komponen yang akan diidentifikasi kerentanannya, yaitu: Sekawan-API, Sekawan-Fe, dan Server.

4.3.1 Sekawan-API

Proses code review pada komponen Sekawan-API akan memeriksa kode sumber dengan celah kerentanan: *Broken Access Control, Cryptographic Failures, Injection, Security Misconfiguration, Vulnerable and Outdated Components, Software and Data Integrity Failures, Security Logging and Monitoring Failures, dan Server-Side Request Forgery* (SSRF).

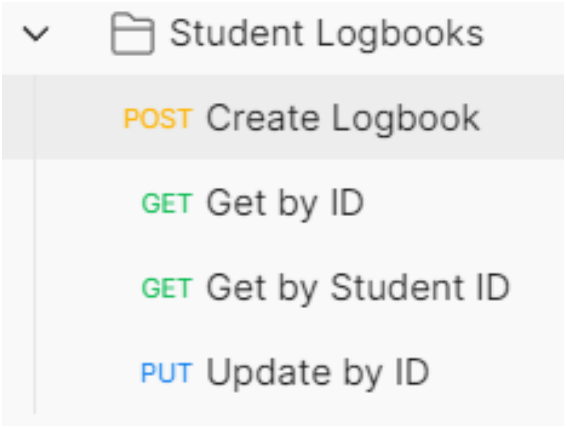
4.3.1.1 Broken Access Control

Memeriksa kerentanan akses yang rusak pada API sangat penting untuk mencegah akses tidak sah, melindungi data sensitif, menjaga integritas data, mencegah penyalahgunaan logika bisnis, memenuhi persyaratan kepatuhan, meningkatkan kepercayaan, dan meminimalkan permukaan serangan. Kontrol akses yang tepat memastikan hanya pengguna atau aplikasi yang berwenang yang dapat mengakses dan berinteraksi dengan data dan fungsionalitas yang diekspos melalui Sekawan-API.

A. Pengujian Luar

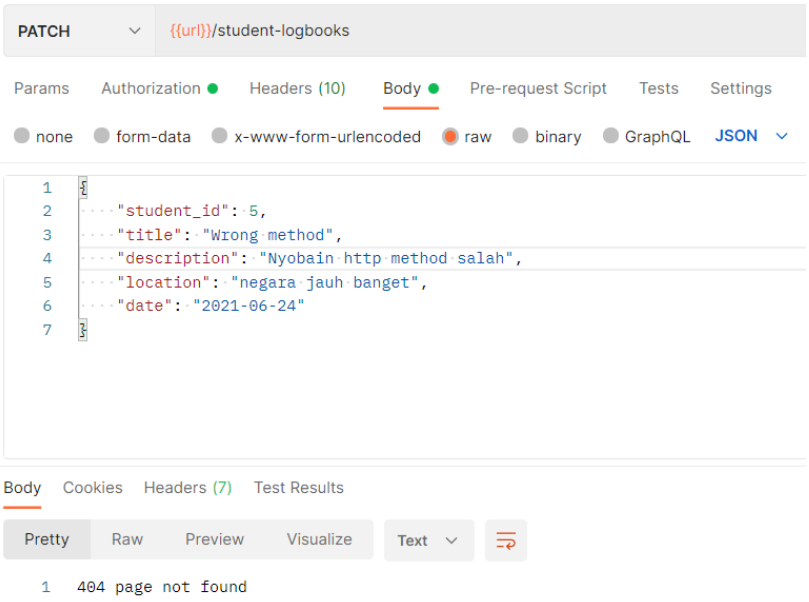
Wrong HTTP Method

Salah satu celah keamanan yang dijelaskan dalam bagian deskripsi OWASP Top 10 2021 *Broken Access Control* adalah penyerang dapat mengakses API dengan atau tanpa metode http yang salah. Ancaman yang dapat terjadi adalah penyerang dapat mengunggah atau menghapus berkas/*script* berbahaya ke dalam server, penyerang juga dapat melakukan *bypass* proses autentikasi dengan membuat *request* dengan metode http yang berbeda (wstgbot, M, & ThunderSon, WSTG - Latest | Test HTTP Methods, 2022). Pengujian pada API dalam kategori kerentanan *Broken Access Control* adalah dengan melakukan tes mengenai cara API dalam menangani *request* dengan metode http yang salah. *Endpoint* API yang akan digunakan dalam pengujian mempunyai fungsi untuk mencatat *logbook* mahasiswa. Pengujian dilakukan dengan bantuan perangkat lunak Postman, yang digunakan untuk melakukan *request* ke *endpoint* API dengan berbagai dukungan metode http yang berbeda.



Gambar 4.1 Koleksi *request* untuk *Student Logbooks*

Pada Gambar 4.1 terdapat empat koleksi *request* untuk pencatatan logbook yang dapat dilakukan yaitu *Create Logbook*, *Get by ID*, *Get by Student ID*, dan *Update by ID*. Masing-masing *request* memiliki metode http yang sesuai yaitu POST untuk mengirim data, GET digunakan dalam pengambilan data, dan penggunaan PUT pada saat pembaruan data (Perrier, Smith, & Baska, HTTP request Method - HTTP | MDN, 2022).



Gambar 4.2 Hasil pengujian dengan metode http yang salah

Hasil pengujian dapat dilihat pada Gambar 4.2. Pada saat Postman melakukan *request* dengan *http method* yang tidak tepat, API akan memberikan tanggapan "*404 page not found*" yang mengindikasikan bahwa apabila metode http yang digunakan tidak sesuai, aplikasi memberikan pesan kesalahan permintaan tidak diketahui. Maka dari itu API telah berhasil melakukan mitigasi kerentanan kategori *Broken Access*

Control tentang kesalahan penggunaan *http method* dalam *request* dan tidak memerlukan tindakan lanjut berupa *hardening*.

B. Pengujian Dalam

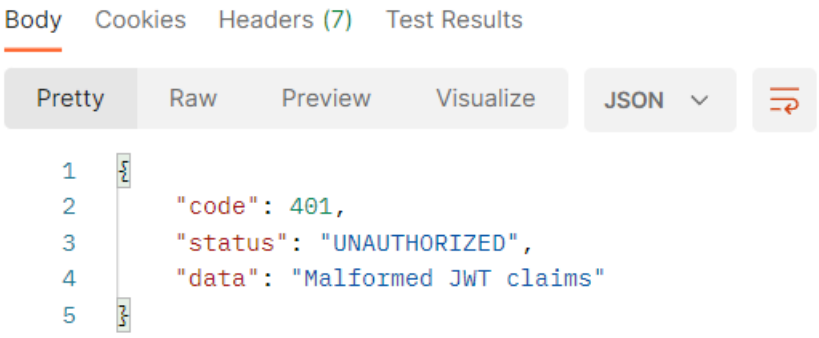
JSON Web Token (JWT) digunakan oleh Sistem Sekawan V2 untuk autentikasi dan session. Pada Sekawan-API terdapat fungsi yang men-generate token JWT Gambar 4.3 yang akan diberikan pada pengguna apabila melakukan proses login pada Aplikasi Sekawan V2. Bagian deskripsi dari OWASP Top 10 2021 kerentanan Broken Access Control menjelaskan salah satu serangan yang dapat terjadi pada JWT adalah tampering dengan memanipulasi nilai dari token tersebut. Risiko yang dihadapi adalah peningkatan hak akses menjadi lebih tinggi yang mana hal itu merupakan sesuatu yang berbahaya apabila terjadi pada sistem. Dengan ancaman tersebut, tim pengembang Sekawan V2 telah menerapkan sistem validasi token JWT untuk melakukan pengecekan token Gambar 4.4, apabila token tidak valid maka request tidak dapat dilakukan Gambar 4.5.

```
// Generate JWT
    jwtToken, err := NewJWTService().GenerateToken(user.Id, accessToken)
    utils.PanicIfError(err)

    return &web.JWTToken{
        Token: jwtToken,
        User:  utils.ToUserResponse(user),
    }
```

Gambar 4.3 Fungsi untuk men-*generate* token JWT

```
func (service *JWTServiceImpl) ValidateToken(encodedToken string)
(*jwt.Token, error) {
    token, err := jwt.Parse(encodedToken, func(t *jwt.Token) (interface{},
error) {
        _, ok := t.Method.(*jwt.SigningMethodHMAC)
        if !ok {
            return nil, errors.New("Invalid Token")
        }

        return []byte(os.Getenv("JWT_SECRET_KEY")), nil
    })
    return token, nil
}
```

Gambar 4.4 Fungsi untuk validasi token JWT



Gambar 4.5 *Request* dengan JWT salah

4.3.1.2 Cryptographic Failures

Tujuan dari memeriksa kegagalan kriptografis dalam OWASP Top 10 2021 pada Sekawan-API adalah untuk mengidentifikasi dan mengatasi kerentanan dalam enkripsi dan implementasi kriptografi. Ini membantu memastikan kerahasiaan, integritas, dan keaslian data yang pertukarkan melalui API, mencegah akses tidak sah dan pelanggaran data.

1. Pengujian Luar

   Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Sekawan-API pada celah keamanan *Cryptographic Failures.* Sekawan-API tidak meng-*handle* enkripsi pada password karena proses autentikasi tidak menggunakan *password*, enkripsi pada *data in transit* telah di tangani oleh implementasi https pada server, dan Sekawan-API tidak menggunakan protokol lawas yang menerapkan algoritma enkripsi lemah.

2. Pengujian Dalam

   Untuk memastikan bahwa token JWT yang digunakan tidak diubah, Sekawan-API menerapkan *signature* pada token. Algoritma yang digunakan pada proses tersebut adalah HS256 yakni kombinasi dari HMAC (*Hash-based Message Authentication Code*) dan SHA256. HS256 adalah algoritme simetris yang berbagi satu kunci rahasia antara penyedia identitas dan aplikasi. Kunci yang sama digunakan untuk menandatangani JWT dan mengizinkan verifikasi tanda tangan digital. Pada Gambar 4.6 terlihat implementasi algoritma tersebut pada Aplikasi Sekawan-API. Proses pembuatan JWT juga telah digunakan sebuah data rahasia tambahan yang membuat

nilai dari *hash* token tersebut semakin unik. Dengan begitu token tidak akan mudah untuk dimanipulasi nilainya.

```
token := jwt.NewWithClaims(jwt.SigningMethodHS256, claim)
signedToken, err := token.SignedString([]byte(os.Getenv("SECRET")))
```

Gambar 4.6 Algoritma tanda tangan digital JWT

4.3.1.3 Injection

Memeriksa kerentanan injeksi dalam konteks OWASP Top 10 2021 pada Sekawan-API memiliki tujuan untuk mencegah dan mengurangi risiko terkait serangan injeksi kode. Mendeteksi kerentanan ini membantu melindungi Sekawan-API dari akses tidak sah, manipulasi data, dan potensi pelanggaran keamanan. Hal ini memastikan integritas data dan kerahasiaan informasi sensitif tetap terjaga dalam lingkungan API.

A. Pengujian Luar

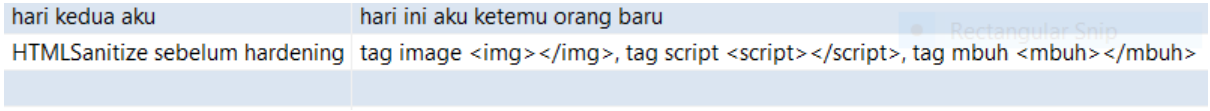
Stored XSS

Salah satu celah keamanan yang dapat terjadi pada aplikasi berbasis web adalah Cross Site Scripting (XSS). Dengan adanya vulnerability tersebut, maka penyerang dapat melakukan serangan berupa injeksi kode ke dalam aplikasi dimana kode tersebut memerlukan tag html tertentu agar dapat dijalankan. Perangkat lunak Postman digunakan untuk membuat permintaan berupa JavaScript Object Notation (JSON) yang akan dikirimkan ke endpoint /student-logbooks. Payload atau muatan yang dikirim mengandung tag html yang dapat dieksploitasi seperti <img>, <script>, dan tag lain <mbuh>.

Gambar 4.7 Pembuatan *request* dengan payload *tag* html pada Postman

Hasil dari permintaan Postman pada Gambar 4.7, dapat dilihat dalam *database* dengan menggunakan perangkat lunak DBeaver. Pada Gambar 4.8 terlihat bahwa *tag* HTML yang diberikan dapat disimpan dalam basis data. Salah satu kategori dalam

celah keamanan Cross-Site Scripting (XSS) adalah *stored* XSS atau XSS yang disimpan. Celah *Cross Site Scripting* dengan kategori tersebut dapat terjadi ketika *script* berbahaya yang dimasukkan ke dalam server milik target secara permanen, seperti di dalam database, pesan forum, log pengunjung, kolom komentar, dll. Skenario yang dapat terjadi adalah korban mendapatkan *script* berbahaya tersebut ketika sedang melakukan *request* ke *server* bersamaan dengan informasi yang diminta (Baars, et al., 2022).



Gambar 4.8 Hasil dari *request* yang akan disimpan dalam basis data

B. Pengujian Dalam

Pada bagian API, setiap eksekusi sql *statement* sudah menggunakan parameter sebagai pencegahan kerentanan injeksi sql. Pada saat sebuah *statement* sql dijalankan, sql *package* mengubah sql *statement* menjadi *prepared statement* atau pernyataan sql yang berparameter lalu dikirimkan bersamaan dengan parameter secara terpisah. Pada *Database Management System* Postgresql menggunakan tanda *dollar* ($) sebagai parameter seperti terlihat pada Gambar 4.9 (Google LLC, 2022). Dengan implementasi *prepared statement*, kemungkinan celah injeksi sql akan semakin kecil.

```
SQL := `INSERT INTO
    research_students(
        student_id, title, category_id
    )
    VALUES ($1, $2, $3)
    RETURNING id`
```

Gambar 4.9 *Parameterized query* pada pernyataan SQL

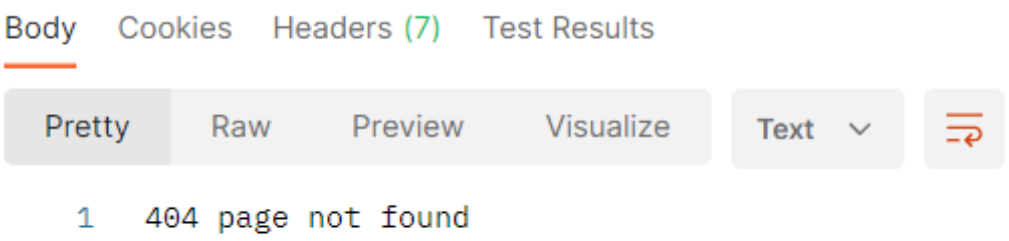
4.3.1.4 Security Misconfiguration

Memeriksa konfigurasi keamanan dalam OWASP Top 10 2021 pada Sekawan-API memiliki tujuan untuk mengidentifikasi dan memperbaiki pengaturan yang tidak terkonfigurasi dengan benar yang dapat menyebabkan akses tidak sah, ekspos data, atau kerentanan keamanan lainnya. Hal ini membantu memastikan bahwa Sekawan-API beroperasi secara aman dan sesuai dengan yang dimaksudkan.

A. Pengujian Luar

Salah satu poin yang terdapat pada kerentanan *security misconfiguration* adalah sistem menampilkan pesan kesalahan yang terlalu informatif kepada *end user*.

Sekawan-API telah menerapkan *custom error handling* yang dapat dilihat pada Gambar 4.10.



Gambar 4.10 Pesan *error* Sekawan-API yang tidak terlalu informatif

B. Pengujian Dalam

Sekawan-API telah menerapkan beberapa konfigurasi keamanan. Hal tersebut terlihat pada penerapan http security header, salah satu header yang diterapkan adalah X-CSRF-Token. Header tersebut befungsi untuk memitigasi serangan Cross Site Request Forgery (CSRF). Sekawan-API akan membuat dan mengirim token yang berfungsi sebagai "identitas" sebuah request dan *session* pengguna, sehingga penyerang tidak dapat melakukan "*forgery*" atau perubahan terhadap *request* yang dikirim Gambar 4.11. Hal lain yang menunjukan keamanan konfigurasi adalah pembatasan metode http yang dapat digunakan, sehingga pengguna tidak dapat menggunakan sembarang metode http Gambar 4.12.

```
c.Header("Access-Control-Allow-Headers", "Content-Type, Content-Length, Accept-Encoding, X-CSRF-Token, Authorization, accept, origin, Cache-Control")
```

Gambar 4.11 Penerapan Security Header dalam Sekawan-API

```
c.Header("Access-Control-Allow-Methods", "POST,HEAD,PATCH, OPTIONS, GET, PUT")
```

Gambar 4.12 Http method yang diperbolehkan dalam Sekawan-API

4.3.1.5 Vulnerable and Outdated Components

Tujuan mengatasi komponen yang rentan dan usang dalam OWASP Top 10 2021 pada API adalah untuk mengurangi risiko yang terkait dengan penggunaan perangkat lunak yang memiliki cacat keamanan yang sudah dikenal. Dengan memperbarui atau mengganti komponen-komponen ini, Sekawan-API dapat menghindari potensi eksploitasi dan kerentanan yang mungkin ditargetkan oleh penyerang. Hal ini membantu menjaga keamanan dan performa dari komponen Sekawan-API.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Sekawan-API pada celah keamanan *Vulnerable and Outdated Components*. Pemeriksaan versi dan kerentanan masing-masing komponen ataupun library yang digunakan pada Sekawan-API hanya dimungkinkan dengan pemeriksaan dari dalam.

B. Pengujian Dalam

Untuk mengecek ketersediaan *update* pada dependensi yang digunakan pada *Application Programming Interface* (API), dapat digunakan perintah terminal:

```
go list -u -f '{{if (and (not (or .Main .Indirect)) .Update)}}{{.Path}}: {{.Version}} -> {{.Update.Version}}{{end}}' -m all 2> /dev/null
```

Gambar 4.13 Perintah untuk memeriksa ketersediaan pembaruan pada go

Pada Gambar 4.13 perintah `go list` akan menampilkan daftar dependensi yang digunakan. Flag -u digunakan untuk menampilkan dependensi yang telah tersedia pembaruan. Flag -f digunakan untuk melakukan format pada hasil yang dikeluarkan. Flag -m menyebabkan go melakukan *list* terhadap modul dibandingkan dengan *package* (Google LLC, n.d.). Hasil pada Gambar 4.14 memperlihatkan terdapat empat *packages* yang dapat diperbarui, yaitu, gin, validator, pq, dan bluemonday. Salah satu pokok pembahasan dalam deskripsi *security list vulnerable and outdated components* adalah perangkat lunak yang telah usang, maka dari itu diperlukan tindakan lanjut *hardening* dengan cara melakukan pembaruan.

```
Lenovo LEGION@Legion MINGW64 /c/laragon/www/sekawan-api (hardening)
$ go list -u -f '{{if (and (not (or .Main .Indirect)) .Update)}}{{.Path}}: {{.Version}} -> {{.Update.Version}}{{end}}' -m all 2> /dev/null
github.com/gin-gonic/gin: v1.7.4 -> v1.8.1
github.com/go-playground/validator/v10: v10.9.0 -> v10.11.1
github.com/lib/pq: v1.10.3 -> v1.10.7
github.com/microcosm-cc/bluemonday: v1.0.18 -> v1.0.21
```

Gambar 4.14 Hasil pemeriksaan pembaruan *packages* pada go

4.3.1.6 Software and Data Integrity Failures

Memeriksa kegagalan integritas perangkat lunak dan data dalam OWASP Top 10 2021 pada Sekawan-API memiliki tujuan untuk mengidentifikasi dan mencegah perubahan atau manipulasi tidak sah terhadap perangkat lunak dan data. Dengan menjaga integritas komponen, Sekawan-API dapat mempertahankan akurasi, kehandalan, keamanan informasi, dan operasionalnya.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Sekawan-API pada celah keamanan *Software and Data*

Integrity Failures. Pengujian integritas kode sumber dan library yang digunakan oleh Sekawan-API hanya dimungkinkan oleh pengujian dalam (*code review*) karena pemeriksaan tersebut juga dapat memeriksa checksum atau hash yang dimiliki oleh masing-masing komponen.

B. Pengujian Dalam

Setiap proses *import* modul, `go` akan melakukan proses *hash* terhadap dependensi yang bersangkutan. Gambar 4.15 menampilkan hasil dari *hash* dependensi akan disimpan dalam berkas go.sum dan berfungsi untuk validasi *checksum* dari setiap dependensi. Hal tersebut dilakukan untuk memastikan bahwa dependensi yang digunakan tidak dimodifikasi sehingga keaslian dependensi dapat terjaga. Dapat dilihat pada gambar, berkas go.sum berisi tentang nama, versi, dan *hash* dari dependensi yang digunakan oleh aplikasi Sekawan-API.

```
github.com/aymerick/douceur v0.2.0 h1:Mv+mAeH1Q+n9Fr+oyamOlAkUNPWPlA8PPGR0QAaYuPk=
github.com/aymerick/douceur v0.2.0/go.mod h1:wlT5vV2O3h55X9m7iVYN0TBM0NH/MmbLnd30/FjWUq4=
github.com/creack/pty v1.1.9/go.mod h1:oKZEueFk5CKHvIhNR5MUki03XCEU+Q6VDXinZuGJ33E=
github.com/davecgh/go-spew v1.1.0/go.mod h1:J7Y8YcW2NihsgmVo/mv3lAwl/skON4iLHjSsI+c5H38=
github.com/davecgh/go-spew v1.1.1 h1:vj9j/u1bqnvCEfJOwUhtlOARqs3+rkHYY13jYWTU97c=
github.com/davecgh/go-spew v1.1.1/go.mod h1:J7Y8YcW2NihsgmVo/mv3lAwl/skON4iLHjSsI+c5H38=
github.com/gin-contrib/sse v0.1.0 h1:Y/yl/+YNO8GZSjAhjMsSuLt29uWRFHdHYUb5lYOV9qE=
github.com/gin-contrib/sse v0.1.0/go.mod h1:RHrZQHXnP2xjPF+u1gW/2HnVO7nvIa9PG3Gm+fLHvGI=
github.com/gin-gonic/gin v1.7.4 h1:QmUZXrvJ9qZ3GfWvQ+2wnW/1ePrTEJqPKMYEU3lD/DM=
github.com/gin-gonic/gin v1.7.4/go.mod h1:jD2toBW3GZUr5UMcdrwQA10I7RuaFOl/SGeDjXkfUtY=
```

Gambar 4.15 Isi konten dari berkas go.sum

4.3.1.7 Security Logging and Monitoring Failures

Menginspeksi kegagalan logging keamanan dan pemantauan dalam OWASP Top 10 2021 pada Sekawan-API memiliki tujuan untuk memastikan pelacakan yang komprehensif dan deteksi tepat waktu terhadap masalah keamanan. Dengan memperbaiki kekurangan ini, organisasi dapat dengan cepat mengidentifikasi tindakan yang tidak sah, pelanggaran, atau aktivitas yang mencurigakan dalam Sekawan-API, memungkinkan respons yang cepat dan mitigasi risiko.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Sekawan-API pada celah keamanan *Security Logging and Monitoring Failures.* Hal tersebut dikarenakan data-data mengenai log dan fungsi *monitoring* hanya dapat diakses melalui pengujian dalam dan tidak dapat dilihat dari luar, sehingga pemeriksaan tidak dimungkinkan.

B. Pengujian Dalam

Tim pengembang Sekawan telah menambahkan fitur *logging* pada sisi API. Fitur tersebut akan melakukan pencatatan aktivitas yang terjadi pada aplikasi *backend*. Kode dari fungsi log yang telah diimplementasikan terdapat pada Gambar 4.16. Terdapat tiga tipe log yang ditulis dalam berkas, yaitu: *info, warning,* dan *error*. Pada log bertipe infolog akan menampilkan status apabila aplikasi berjalan dengan sukses, sedangkan log dengan tipe error akan digunakan apabila aplikasi mengalami permasalahan. Hasil dari fungsi log tersebut akan dituliskan dalam berkas logs.log. Dalam Gambar 4.17 terlihat hasil dari fungsi log yang telah diterapkan dengan detail:

a. Jenis *log*.
b. Waktu, dengan format: tahun/bulan/tanggal jam:menit:detik
c. Berkas dan nomor baris fungsi yang dijalankan.

```
func NewLogger() {
    file, err := os.OpenFile("logs.log",
    os.O_APPEND|os.O_CREATE|os.O_WRONLY, 0666)
    PanicIfError(err)
    InfoLogger = log.New(file, "INFO: ",
    log.Ldate|log.Ltime|log.Lshortfile)
    WarningLogger = log.New(file, "WARNING: ",
    log.Ldate|log.Ltime|log.Lshortfile)
    ErrorLogger = log.New(file, "ERROR: ",
    log.Ldate|log.Ltime|log.Lshortfile)
}
```

Gambar 4.16 Fungsi logging pada Sekawan-API

```
INFO: 2022/05/13 21:44:00 app.go:37: Env file successfully loaded
INFO: 2022/05/13 21:44:00 app.go:45: Database successfully initiated
INFO: 2022/05/13 21:44:00 app.go:47: Validator successfully initiated
INFO: 2022/05/16 21:44:52 app.go:37: Env file successfully loaded
INFO: 2022/05/16 21:44:52 app.go:45: Database successfully initiated
INFO: 2022/05/16 21:44:52 app.go:47: Validator successfully initiated
INFO: 2022/05/16 22:02:00 app.go:37: Env file successfully loaded
INFO: 2022/05/16 22:02:00 app.go:45: Database successfully initiated
INFO: 2022/05/16 22:02:00 app.go:47: Validator successfully initiated
INFO: 2022/05/18 21:18:22 app.go:37: Env file successfully loaded
INFO: 2022/05/18 21:18:22 app.go:45: Database successfully initiated
INFO: 2022/05/18 21:18:22 app.go:47: Validator successfully initiated
```

Gambar 4.17 Hasil fungsi logging pada Sekawan-API

4.3.1.8 Server-Side Request Forgery

Memeriksa kerentanan server-side request forgery (SSRF) dalam OWASP Top 10 2021 pada Sekawan-API memiliki tujuan untuk mengidentifikasi dan mencegah penyerang memanfaatkan kerentanan yang memungkinkan mereka membuat permintaan

tidak sah ke sumber daya internal atau sistem pihak ketiga. Mendeteksi kerentanan SSRF membantu melindungi data sensitif, mencegah akses tidak sah, dan mengurangi potensi pelanggaran keamanan.

A. Pengujian Luar

Sekawan-API tidak menampilkan URL yang terekspos dengan parameter yang dapat dieksploitasi oleh penyerang. Sehingga percobaan pengujian tidak dimungkinkan.

B. Pengujian Dalam

Kerentanan SSRF terjadi ketika aplikasi menggunakan sumber daya *remote* tanpa melakukan validasi terhadap nilai yang diberikan pengguna. Target utama dari kerentanan tersebut adalah sistem *backend*, yang biasanya koneksi dari komponen tersebut akan selalu dipercaya. Contoh serangan dapat terlihat pada Gambar 4.19 dan Gambar 4.19. Sistem Sekawan V2 tidak memiliki URL yang memungkinkan pengguna mencantumkan *resource* dari lokal ataupun sumber lain dan mengeksekusinya.

```
GET https://example.com/page?page=about.php
```

Gambar 4.18 Contoh URL yang yang dapat dieksploitasi

```
GET https://example.com/page?page=file:///etc/passwd
```

Gambar 4.19 Contoh serangan *Server-Side Request Forgery* (SSRF)

4.3.2 Sekawan-FE

Proses *code review* pada aplikasi Sekawan-FE dilakukan dengan memeriksa kode sumber. Tahap ini dilakukan dengan menggunakan perangkat lunak Visual Studio Code dan manajer paket NPM. Kerentanan yang diperiksa adalah *Broken Access Control, Cryptographic Failures, Injection, Security Misconfiguration, Vulnerable and Outdated Components, Software and Data Integrity Failures*, dan *Security Logging and Monitoring Failures*.

4.3.2.1 Broken Access Control

Tujuan dari pemeriksaan terhadap broken access control dalam OWASP Top 10 2021 pada aplikasi Sekawan-FE adalah untuk mengidentifikasi dan memperbaiki kerentanan yang dapat memungkinkan pengguna yang tidak sah untuk mengakses fungsionalitas terbatas atau data sensitif. Mendeteksi dan mengatasi masalah-masalah ini membantu memastikan otorisasi pengguna yang tepat, perlindungan data, dan kepatuhan terhadap kebijakan keamanan.

A. Pengujian Luar

Akses kontrol pada Aplikasi Sekawan-FE dapat terlihat pada pesan kesalahan yang diberikan. Contoh skenario yang dapat terjadi adalah ketika seorang pengguna dengan *role* mahasiswa mencoba mengakses halaman milik *role* dosen ataupun sekprodi dengan mengubah URL. Hasil yang ditampilkan berupa halaman *error* yang menunjukan informasi bahwa halaman tidak dapat dimuat karena tidak memiliki izin atau *role* yang sesuai Gambar 4.20. Dengan akses yang terbatas, pengguna tidak dapat mengakses informasi secara bebas.

Gambar 4.20 Halaman tidak dapat dimuat karena tidak memiliki izin akses

B. Pengujian Dalam

Setiap halaman yang diakses oleh pengguna terikat dengan *session* yang telah ditetapkan oleh Sekawan-FE. Pada *session* tersebut juga terdapat data mengenai *role* dari setiap pengguna. Pada masing-masing halaman juga terdapat validasi *role* yang dapat mengakses halaman tersebut Gambar 4.21.

```
Acara.auth = { role: [Role.LECTURER, Role.ADMIN] };
```

Gambar 4.21 Validasi *role* pengguna pada halaman acara

4.3.2.2 Cryptographic Failures

Alasan memeriksa kegagalan kriptografis dalam OWASP Top 10 2021 pada komponen Sekawan-FE adalah untuk mendeteksi dan mengatasi kelemahan dalam praktik enkripsi dan kriptografi. Hal ini berfungsi untuk melindungi kerahasiaan dan integritas data yang pertukarkan dalam aplikasi, mencegah potensi kompromi keamanan

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Sekawan-FE pada celah keamanan *Cryptographic*

Failures. Sekawan-FE tidak meng-*handle* proses enkripsi *data in transit* maupun enkripsi pada *password*. Implementasi https dilakukan pada server.

B. Pengujian Dalam

Mekanisme kriptografi telah digunakan oleh Sekawan-FE pada kerangka kerja Nextjs. Penerapanya dapat dilihat pada Gambar 4.22, yaitu *hash* yang digunakan oleh nextjs pada dependensi yang digunakan. Jenis *hash* yang dipakai adalah SHA512 dan tidak menggunakan algoritma lemah dan lawas seperti MD5 dan SHA1 karena kedua algoritma tersebut telah memiliki banyak celah keamanan yang diketahui.

Gambar 4.22 *Hash* SHA512 pada dependensi Nextjs

4.3.2.3 Injection

Tujuan dari pemeriksaan kerentanan injeksi dalam OWASP Top 10 2021 pada aplikasi Sekawan-FE adalah untuk mengidentifikasi dan mencegah serangan injeksi kode berbahaya. Dengan mengatasi kerentanan ini, aplikasi dapat menghindari manipulasi data yang tidak sah, pelanggaran keamanan, dan potensi gangguan, serta memastikan integritas dan kehandalan interaksi pengguna dan pemrosesan data.

A. Pengujian Luar

Reflected XSS

Pengujian kerentanan *Reflected Cross Site Scripting* dilakukan dengan memasukan kode berbahaya pada form input di Sekawan-FE. Salah satu fitur di Sekawan-FE adalah pembuatan *logbook*, yang mana fitur tersebut akan menjadi target uji kali ini. Kode

Kerangka kerja web modern saat ini telah menerapkan lapisan keamanan tambahan secara *default*. Pada Sekawan-FE menggunakan Nextjs, yakni sebuah kerangka kerja berbasis bahasa pemrograman Javascript. *Framework* tersebut telah menerapkan keamanan tambahan, termasuk dengan perlindungan terhadap serangan *Reflected Cross Site Scripting* (XSS) (Security Lit Limited, 2022).

B. Pengujian Dalam

Pemeriksaan kode sumber Sekawan-FE telah menerapkan sanitasi pada *input*. Hal tersebut terlihat pada fitur unggah berkas. Pada fitur tersebut hanya menerima berkas dengan ekstensi pdf atau svg saja Gambar 4.23.

```
accept: ".pdf, .svg",
```

Gambar 4.23 Sanitasi ekstensi berkas

4.3.2.4 Security Misconfiguration

Memeriksa kesalahan konfigurasi keamanan dalam OWASP Top 10 2021 pada aplikasi Sekawan-FE bertujuan untuk mendeteksi dan mengurangi risiko yang timbul akibat pengaturan yang tidak terkonfigurasi dengan benar. Dengan memperbaiki kesalahan konfigurasi tersebut, aplikasi dapat menghindari akses tidak sah, kebocoran data, dan potensi insiden keamanan, sehingga meningkatkan keamanan secara keseluruhan dan mempertahankan kepercayaan pengguna.

A. Pengujian Luar

Clickjacking

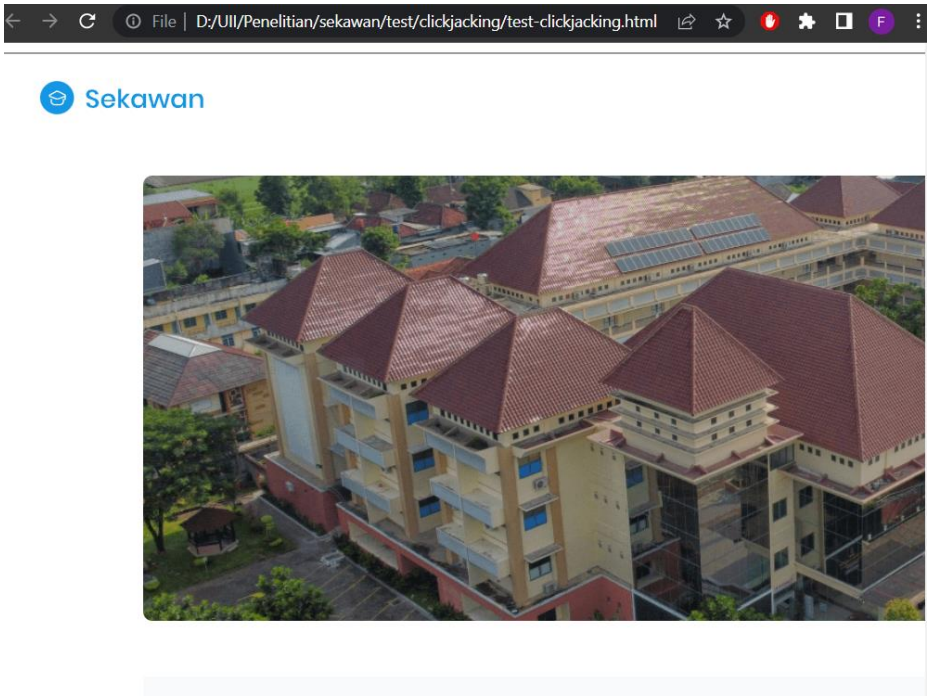
Pengujian berikutnya adalah melakukan percobaan serangan dengan Clickjacking. Penyerang melakukan penipuan terhadap korban dengan cara menggunakan lapisan transparan pada UI (*User Interface*) yang menutupi tampilan asli dari web yang diakses. Korban tidak menyadari bahwa tombol yang diklik adalah *button* palsu pada lapisan transparan milik penyerang (M, Alon, Mavrakis, Blankenship, & Buckley, 2022). Salah satu hal yang menyebabkan terjadinya celah keamanan Clickjacking adalah penggunaan HTML *tag* iframe atau *inline frame* yang berfungsi untuk menyematkan dokumen lain ke dalam halaman web.

Pengujian sederhana dapat dilakukan dengan membuat halaman web lain yang akan menyematkan web Sekawan v2 didalamnya. Tujuan dari pengujian ini adalah untuk memastikan bahwa *tag* HTML iframe yang rentan tidak dapat dimuat oleh browser, sehingga dapat meminimalisir terjadinya serangan Clickjacking. Pada Gambar 4.24 adalah kode yang digunakan untuk membangun halaman web palsu oleh penyerang (wstgbot, 2020).

```
<html>
    <head>

    </head>
    <body>
        <iframe src="http://localhost:3000" width="1500"
height="800"></iframe>
    </body>
</html>
```

Gambar 4.24 Kode sumber berkas untuk penyerangan *Clickjacking*

Pada Gambar 4.25 terlihat bahwa sebelum proses *hardening* dilakukan. Tag HTML iframe dapat dimuat oleh peramban sehingga halaman web lain dalam kasus ini adalah Sekawan V2 dapat disematkan di dalamnya. Dengan hasil tersebut, maka hal ini memerlukan tindakan pengamanan.



Gambar 4.25 Hasil pengujian serangan *Clickjacking*

B. Pengujian Dalam

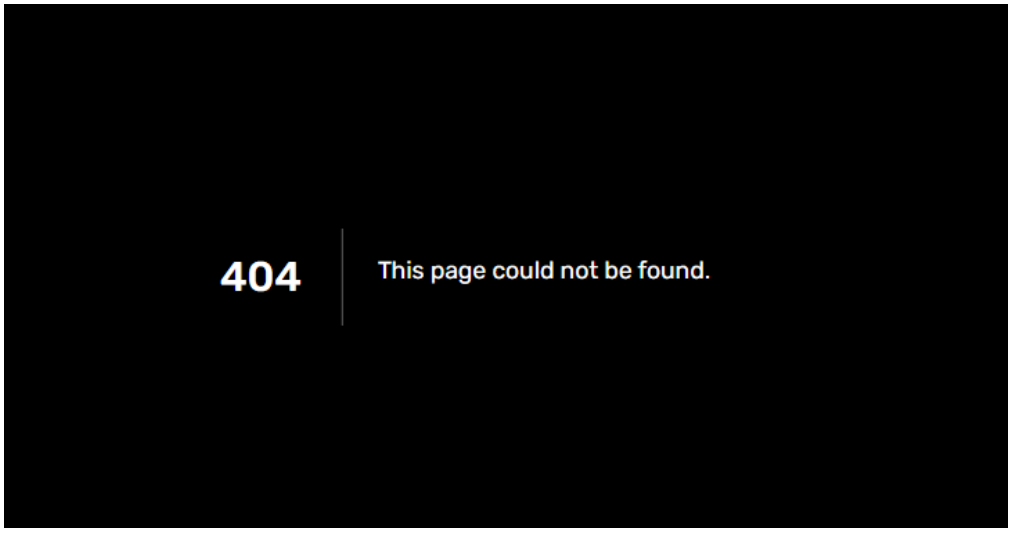
Pesan Kesalahan

Sekawan-FE telah menerapkan *custom error handling*. Fungsi tersebut bertujuan untuk tidak memberikan pesan kesalahan yang terlalu informatif kepada *end-user*, sehingga tidak ada informasi sensitif mengenai sistem yang diketahui publik. Pesan *error* yang terlalu informatif dapat digunakan oleh penyerang untuk mendapatkan data-data mengenai sistem yang digunakan seperti web server dan port yang digunakan Gambar 4.29. Dengan informasi tersebut, penyerang dapat mengetahui celah keamanan yang terdapat dalam sistem target. Skenario yang dapat terjadi untuk pesan *custom error* muncul antara lain:

a. Halaman yang di-*request* tidak tersedia Gambar 4.26

Mengakses URL dengan role yang berbeda (tidak memiliki akses). Contoh tampilan pada

b. Gambar 4.28 dan kode sumber pada Gambar 4.27.

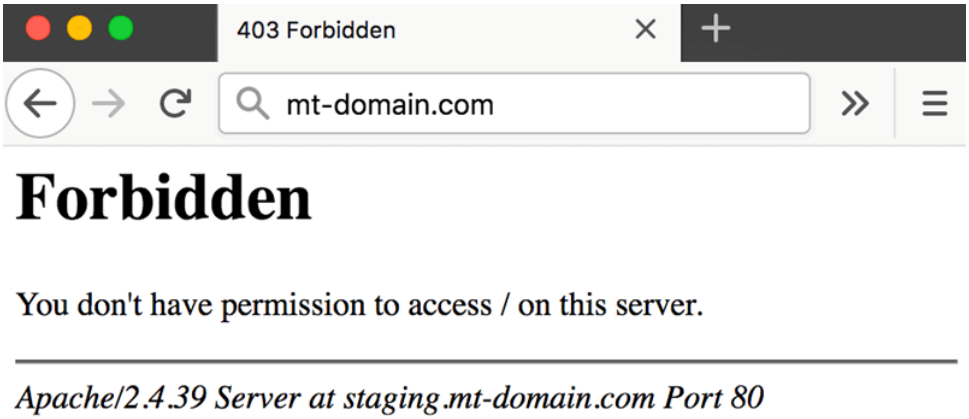


Gambar 4.26 Tampilan *custom error* untuk halaman yang tidak tersedia

```
return (
      <div className="flex flex-col gap-y-1 justify-center items-center h-
screen text-slate-600">
        <h1 className="text-5xl font-light text-slate-800">OOPS!</h1>
        <span className="text-base">
          You don&apos;t have permission to view this page
        </span>
      </div>
    );
```

Gambar 4.27 Kode *Custom error* untuk izin akses yang salah

Gambar 4.28 Tampilan *custom error* untuk izin akses yang salah



Gambar 4.29 Tampilan *error* yang terlalu informatif

Security Response Header

Aplikasi *front-end* Sekawan v2 belum menerapkan *http response header* keamanan tambahan. Pada saat pengguna melakukan *request* ke *server*, hanya terdapat *http response header* standar yang digunakan dan belum terdapat tambahan yang berkaitan dengan keamanan. Konfigurasi *header* terdapat pada berkas `next.config.js` seperti dalam Gambar 4.30. Maka dari itu, diperlukan tindakan lanjut berupa penambahan *header* keamanan.

```
/** @type {import('next').NextConfig} */
module.exports = {
  reactStrictMode: true,
}
```

Gambar 4.30 Konten dari berkas next.config.js

4.3.2.5 Vulnerable and Outdated Components

Memeriksa komponen yang rentan dan usang dalam OWASP Top 10 2021 pada aplikasi Sekawan-FE memiliki tujuan untuk mendeteksi dan mengurangi potensi ancaman keamanan yang timbul akibat penggunaan komponen perangkat lunak yang usang atau rentan. Dengan mengatasi kerentanan ini, aplikasi dapat mencegah pelanggaran potensial, akses tidak sah, dan kelemahan keamanan, sehingga meningkatkan posisi keamanan secara keseluruhan dan melindungi data pengguna.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Sekawan-FE pada celah keamanan *Vulnerable and Outdated Components*. Pemeriksaan versi dan kerentanan masing-masing komponen

ataupun library yang digunakan pada Sekawan-FE hanya dimungkinkan dengan pemeriksaan dari dalam.

B. Pengujian Dalam

Pada Nextjs proses pencarian komponen kedaluwarsa dan rentan dapat menggunakan *package manager* bawaan yakni npm. Tahap audit keamanan dapat dilakukan dengan mengetikan perintah pada terminal: `npm audit`. Perintah dari tersebut akan mengirimkan deksripsi dependensi yang dikonfigurasi ke *registry default* dan akan meminta laporan kerentanan yang telah diketahui. Laporan hasil audit memuat informasi mengenai celah keamanan yang terdapat pada dependensi proyek yang dapat digunakan untuk melakukan perbaikan (Thomson, 2020). Pada perintah `npm audit`, terdapat beberapa bagian informasi:

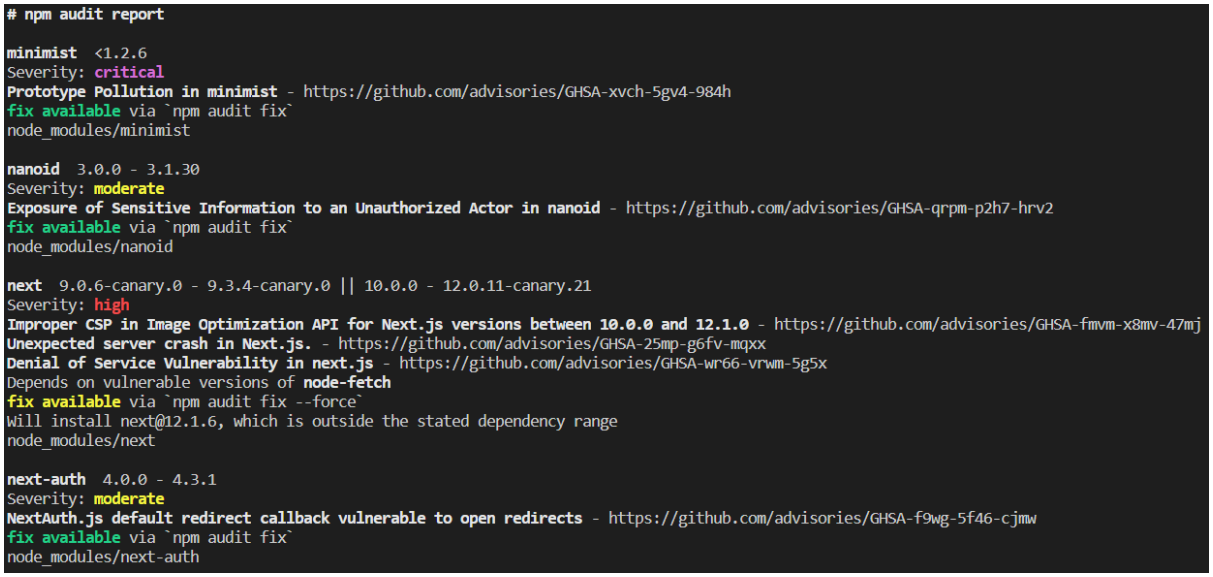
A. Tingkat Kerentanan
B. Nama Paket
C. Dependensi dari
D. Path

Apabila terdapat *package* yang mengandung celah keamanan, `npm audit` juga telah menyertakan tingkat kerentanan dan rekomendasi aksi yang dapat dilakukan. Detail mengenai hal tersebut tercantum pada Tabel 4.2.

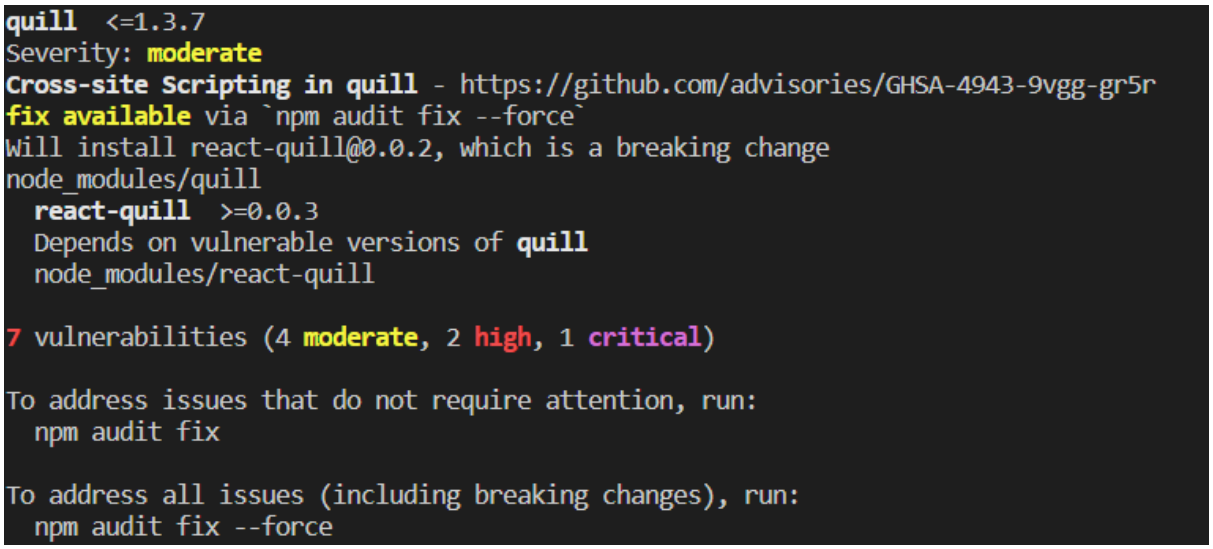
Tabel 4.2 Tabel hubungan antara tingkat kerentanan dan rekomendasi aksi

Tingkat Kerentanan	**Rekomendasi Aksi**
Critical	Langsung segera atasi
High	Atasi secepat mungkin
Moderate	Atasi jika waktu memungkinkan
Low	Atasi sesuai dengan kebijakan

Setelah perintah `npm audit` dijalankan, hasil dari proses audit keamanan akan dikeluarkan dengan rincian pada Gambar 4.31 dan Gambar 4.32:



Gambar 4.31 Hasil perintah `npm audit` pada aplikasi *front-end*



Gambar 4.32 Hasil perintah `npm audit` pada aplikasi *front-end*

Pada setelah mengeksekusi perintah npm audit, terlihat pada Gambar 4.31 dan Gambar 4.32 hasil yang didapatkan berupa tujuh celah keamanan dengan tiga tingkat bahaya yang berbeda. Terdapat satu *vulnerability* berstatus *critical*, dua *vulnerability* dengan tingkat *high*, dan empat kerentanan mempunyai tingkat *moderate*. Dengan hasil tersebut, maka untuk meminimalkan *attack surface* pada aplikasi diperlukan langkah lanjutan berupa *hardening*.

4.3.2.6 Software and Data Integrity Failures

Memeriksa kegagalan integritas perangkat lunak dan data dalam OWASP Top 10 2021 pada Sekawan-FE memiliki tujuan untuk mengidentifikasi dan mencegah perubahan atau manipulasi tidak sah terhadap perangkat lunak dan data. Dengan menjaga

integritas komponen, Sekawan-FE dapat mempertahankan akurasi, kehandalan, keamanan informasi, dan operasionalnya.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Sekawan-FE pada celah keamanan *Software and Data Integrity Failures*. Pengujian integritas kode sumber dan library yang digunakan oleh Sekawan-FE hanya dimungkinkan oleh pengujian dalam (*code review*) karena pemeriksaan tersebut juga dapat memeriksa checksum atau hash yang dimiliki oleh masing-masing komponen.

B. Pengujian Dalam

Setiap package yang dipasang dan digunakan pada NextJS dengan manajer paket, npm akan mencatat nama, versi, *engine*, *resolved*, *dependensi*, dan *integrity* dari setiap *package* ke dalam berkas package-lock.json. Algoritma yang dipakai oleh npm untuk melakukan proses *hash* adalah SHA512. Contoh dari konten berkas package-lock.json terdapat pada Gambar 4.33.

Gambar 4.33 Isi dari berkas package-lock.json

4.3.2.7 Security Logging and Monitoring Failures

Menginspeksi kegagalan logging keamanan dan pemantauan dalam OWASP Top 10 2021 pada Sekawan-FE memiliki tujuan untuk memastikan pelacakan yang komprehensif dan deteksi tepat waktu terhadap masalah keamanan. Dengan memperbaiki kekurangan ini, organisasi dapat dengan cepat mengidentifikasi tindakan yang tidak sah, pelanggaran, atau aktivitas yang mencurigakan dalam Sekawan-FE, memungkinkan respons yang cepat dan mitigasi risiko.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Sekawan-API pada celah keamanan Security Logging

and Monitoring Failures. Hal tersebut dikarenakan data-data mengenai log dan fungsi monitoring hanya dapat diakses melalui pengujian dalam dan tidak dapat dilihat dari luar, sehingga pemeriksaan tidak dimungkinkan.

B. Pengujian Dalam

Aplikasi Sekawan-FE dijalankan pada server dengan menggunakan pm2 sebagai manajer proses. Log dari perangkat lunak yang dijalankan dengan pm2 dapat dilihat secara *real time* dengan digunakan perintah Gambar 4.34:

```
pm2 log
```

Gambar 4.34 Perintah `pm2 log`

Hasil dari perintah Gambar 4.34 dapat dilihat pada Gambar 4.35. Pada gambar tersebut terlihat bahwa pm2 telah menjalankan Aplikasi Sekawan-FE dengan sukses, server nextjs dijalankan pada *port* 3000, berkas *environment* telah di-*load*, dan tidak ada *error* yang ditampilkan.

Gambar 4.35 Log dari Sekawan-FE pada pm2

4.3.3 Server

Pada server yang digunakan oleh sekawan, terdapat enam kategori OWASP Top 10 2021 yang diperiksa yakni: *Broken Access Control, Cryptographic Failures, Insecure Design, Security Misconfiguration, Vulnerable and Outdated Components,* dan *Security Logging and Monitoring Failures*.

4.3.3.1 Broken Access Control

Tujuan dari pemeriksaan *broken access control* dalam OWASP Top 10 2021 pada server Linux adalah untuk mengidentifikasi dan memperbaiki kerentanan yang dapat menyebabkan akses tidak sah ke sumber daya atau fungsionalitas sensitif. Mendeteksi dan mengatasi masalah-masalah ini membantu memastikan otorisasi pengguna yang tepat, mencegah pelanggaran data, dan menjaga keamanan lingkungan server.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Server pada celah keamanan *Broken Access Control.* Hal tersebut dikarenakan kontrol akses hanya tersedia setelah pengguna terautentikasi dan terautorisasi ke dalam sistem. Maka dari itu pemeriksaan yang lebih sesuai adalah dengan pengujian dalam dengan memeriksa konfigurasi pada server.

B. Pengujian Dalam

Sistem Operasi Linux Ubuntu secara *default* mempunyai akses kontrol yang mengatur izin akses pengguna terhadap berkas atau direktori. Perizinan tersebut diatur berdasarkan pemilik, grup, dan pengguna lain. *File permission* sebuah *file* dapat dilihat dengan perintah `ls -l` dengan hasil pada Gambar 4.36. Dari sebelah kiri, satu karakter pertama mengindikasikan berkas atau direktori, tiga karakter setelahnya menandakan akses yang dimiliki oleh pemilik dari berkas, tiga karakter selanjutnya menunjukan akses yang dimiliki oleh pengguna berada masuk grup yang sama, dan tiga karakter terakhir adalah akses yang dimiliki oleh bukan pemilik berkas maupun berada dalam grup tertentu. Terdapat tiga jenis akses:

1. *Read* (`r`): izin untuk membaca berkas. Contoh aktivitas adalah dengan menampilkan isi dari *file* .txt dengan menggunakan perintah `cat`.
2. *Write* (`w`): izin untuk menulis ke dalam berkas. Contoh aktivitas adalah mengubah teks dalam *file* dengan *text editor*.
3. *Execute* (`x`): izin untuk mengeksekusi berkas/program. Contoh aktivitas adalah mengeksekusi *script* python3.

Apabila sebuah *file* memiliki *permission* "`-rwx------`" maka berkas tersebut hanya dapat di baca, tulis, dan eksekusi oleh pemilik berkas. Direktori dengan *permission* "`d---rwx---`" mengindikasikan bahwa *file* dalam direktori tersebut hanya dapat di baca, tulis, dan eksekusi oleh pengguna masuk dengan grup yang sama. *File* dengan izin akses "`-------rwx`" menunjukan bahwa berkas tersebut hanya dapat diakses apabila pengguna bukan pemilik dan bukan dalam grup. Jika tidak terdapat *permission* maka karakter yang merepresentasikan jenis akses (`r, w, x`) maka karakter tersebut akan ditampilkan sebagai tanda minus (-).

```
drwxr-xr-x 6 penelitian-18523107 penelitian-18523107
drwxr-xr-x 7 root                root
-rw-rw---- 1 penelitian-18523107 penelitian-18523107
-rw-r--r-- 1 penelitian-18523107 penelitian-18523107
-rw-r--r-- 1 penelitian-18523107 penelitian-18523107
drwx------ 2 penelitian-18523107 penelitian-18523107
drwx------ 3 penelitian-18523107 penelitian-18523107
drwx------ 3 penelitian-18523107 penelitian-18523107
-rw-r--r-- 1 penelitian-18523107 penelitian-18523107
drwx------ 2 penelitian-18523107 penelitian-18523107
-rw-r----- 1 penelitian-18523107 penelitian-18523107
```

Gambar 4.36 Hak akses pada berkas atau direktori

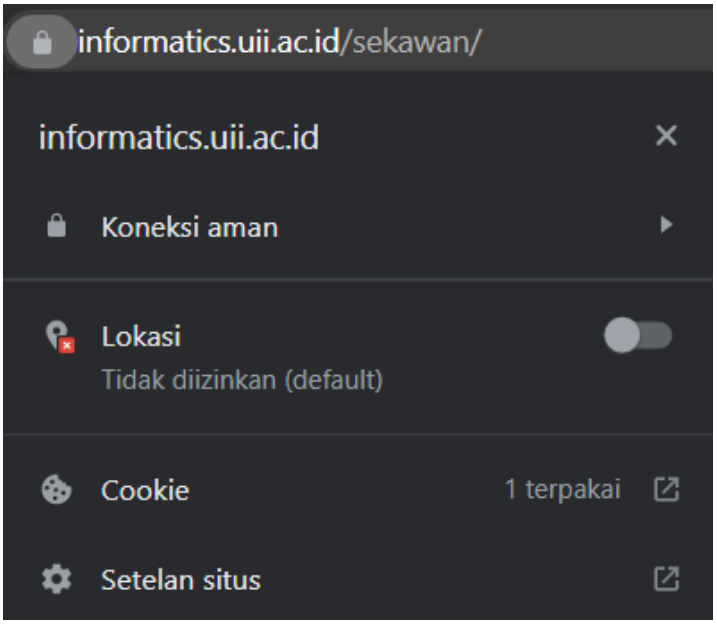
4.3.3.2 Cryptographic Failures

Memeriksa kegagalan kriptografis dalam OWASP Top 10 2021 pada server memiliki tujuan untuk memastikan penerapan yang benar dari enkripsi dan teknik kriptografi. Hal ini penting untuk melindungi data sensitif dari akses tidak sah, manipulasi data, dan potensi pelanggaran, sehingga meningkatkan keamanan pada lingkungan server.

A. Pengujian Luar

HTTPS

Pada server Sekawan v2 juga sudah terpasang sertifikat SSL (*Secure Socket Layer*). Hal tersebut memungkinkan layanan Sekawan v2 untuk menggunakan protokol HTTPS (*Hypertext Transfer Protocol Secure*). Tujuan dari proses tersebut adalah untuk mengamankan data pengguna pada saat data berpindah tempat (*data at transit*) dengan melalui operasi enkripsi. Salah satu indikator yang menunjukan bahwa web Sekawan V2 telah menggunakan protokol HTTPS adalah terdapatnya logo “gembok” di samping kiri dari URL (*Unified Resource Locator*) Gambar 4.37.



Gambar 4.37 Penggunaan HTTPS pada peramban Chrome

B. Pengujian Dalam

Password Hash

Sistem operasi yang digunakan oleh server Sekawan v2 adalah Linux Ubuntu. Pada OS tersebut, setiap kata sandi yang digunakan oleh masing-masing akun pengguna akan melalui proses *hashing*. Hasil dari operasi tersebut akan disimpan pada berkas `shadow` dan akan digunakan kembali pada saat pengguna *login* ke dalam sistem. Dalam berkas tersebut terdapat beberapa bagian yang digunakan untuk menyimpan informasi mengenai pengguna yang disimpan dan tiap bagian tersebut dipisahkan oleh tanda titik dua ":" seperti yang terlihat pada Gambar 4.38.

Gambar 4.38 Contoh *hash* pada akun server

4.3.3.3 Insecure Design

Memeriksa desain yang tidak aman dalam OWASP Top 10 2021 pada server bertujuan untuk mendeteksi dan mengurangi kerentanan yang berasal dari keputusan arsitektural yang tidak memadai dan kelemahan dalam desain. Dengan memperbaiki masalah-masalah ini, organisasi dapat mencegah potensi kerentanan keamanan, akses tidak sah, dan pelanggaran data, sehingga meningkatkan posisi keamanan dan ketahanan lingkungan server.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar Server pada celah keamanan *Insecure Design.*

B. Pengujian Dalam

Ketersediaan Layanan

Salah satu komponen dari CIA Triad adalah *Availability*. Hal tersebut telah diimplementasikan dalam server sekawan melalui manajemen layanan. Sekawan-API dijalankan pada *server* dengan membuat systemd *custom service* yang akan meng-*handle* aktivitas layanan. Di sisi lain, Sekawan-FE menggunakan *process manager* untuk NodeJS yaitu pm2. Kedua hal tersebut diperlukan untuk mengelola dan menjaga aplikasi agar tetap *online* dan dapat selalu diakses, meskipun terjadi gangguan seperti server *restart*.

Pada Linux Ubuntu, pengguna dapat membuat layanan *custom* yang berjalan di *server*. Pada Gambar 4.39 terlihat kode yang digunakan untuk membangun *custom service* (Docile, 2018), terdapat tiga bagian:

1. `[Unit]`: Bagian ini mengandung deskripsi dari layanan dan informasi yang berkaitan tentang *behaviour* dan dependensi.
2. `[Service]`: Pada bagian *service* terdapat informasi yang lebih spesifik mengenai suatu layanan, seperti perintah yang digunakan untuk mengeksekusi pada saat layanan dijalankan, atau tipe dari layanan tersebut.
3. `[Install]`: Bagian install mengandung informasi yang berkaitan dengan proses instalasi layanan.

```
[Unit]
Description=sekawan-api
After=multi-user.target

[Service]
User=root
Group=root
ExecStart=/root/sekawan-api/sekawan-api

[Install]
WantedBy=multi-user.target
```

Gambar 4.39 Kode pembuatan layanan *custom* pada Linux Ubuntu

Untuk memeriksa status dari *custom service* yang telah diimplementasikan pada Sekawan-API, perintah yang digunakan terdapat pada Gambar 4.40 dan hasil yang didapatkan pada Gambar 4.41.

```
sudo systemctl status sekawan-api.service
```

Gambar 4.40 Perintah terminal Linux untuk memeriksa status layanan Sekawan-API

```
informaticssekawan@informatics-sekawan:/etc/systemd/system$ sudo systemctl status se
kawan-api.service
● sekawan-api.service - sekawan-api
   Loaded: loaded (/etc/systemd/system/sekawan-api.service; disabled; vendor preset:
 enabled)
   Active: active (running) since Sun 2022-05-29 07:41:12 UTC; 2 weeks 0 days ago
 Main PID: 213813 (sekawan-api)
    Tasks: 5 (limit: 2178)
   CGroup: /system.slice/sekawan-api.service
```

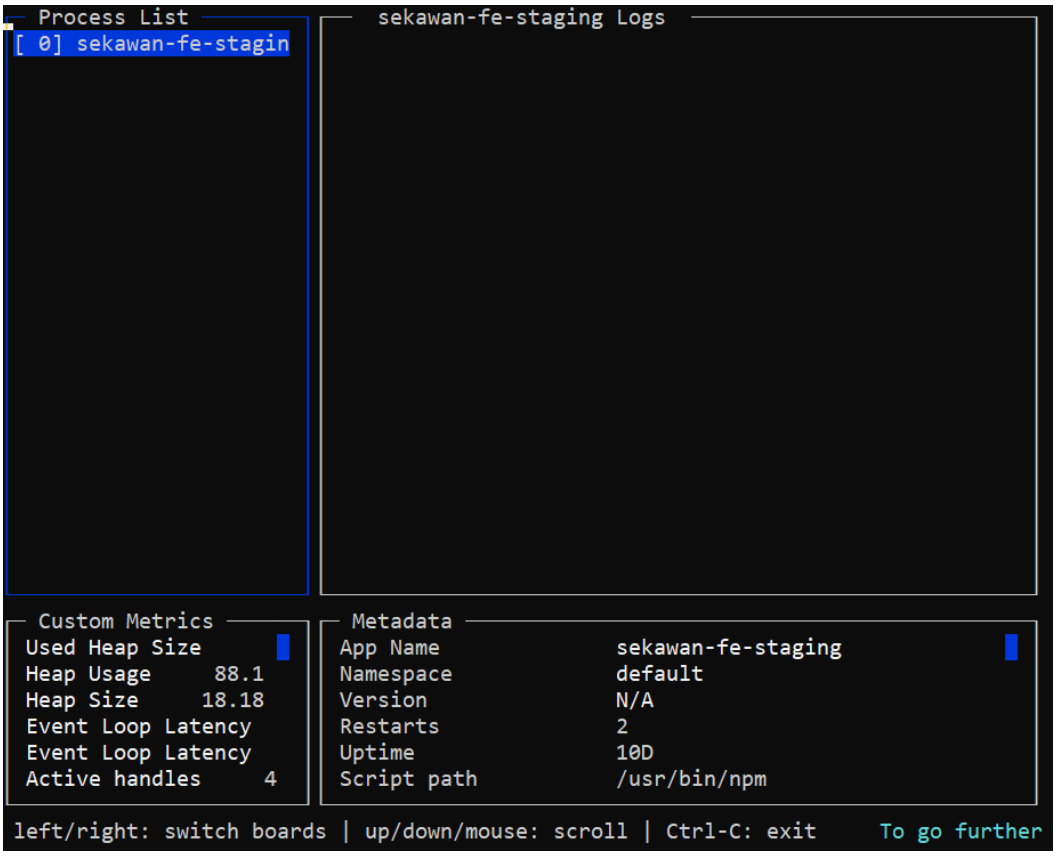
Gambar 4.41 Status Sekawan-API

Setelah proses pembuatan layanan dan memastikan bahwa layanan dapat berjalan dengan baik, selanjutnya dapat dilakukan konfigurasi yang bertujuan agar layanan dapat otomatis berjalan kembali apabila terjadi kendala. Untuk menerapkan konfigurasi tersebut pada Linux, dapat digunakan perintah pada Gambar 4.42:

```
sudo systemctl enable sekawan-api.service
```

Gambar 4.42 Konfigurasi layanan untuk *startup* otomatis

Untuk aplikasi Sekawan-FE pada server menggunakan pm2 sebagai manajer proses pada latar belakang. Pada pm2, untuk memeriksa status dari layanan yang dijalankan dengan perangkat lunak tersebut, dapat menggunakan perintah `pm2 monit` Gambar 4.43:



Gambar 4.43 Hasil dari perintah `pm2 monit`

4.3.3.4 Security Misconfiguration

Alasan memeriksa kesalahan konfigurasi keamanan dalam OWASP Top 10 2021 pada server adalah untuk mengidentifikasi dan memperbaiki pengaturan yang tidak terkonfigurasi dengan benar yang dapat menyebabkan kerentanan. Dengan mengatasi kesalahan konfigurasi ini, organisasi dapat mencegah akses tidak sah, eksposur data sensitif, dan potensi pelanggaran keamanan, sehingga memastikan lingkungan server yang lebih aman.

A. Pengujian Luar

Port Scanning

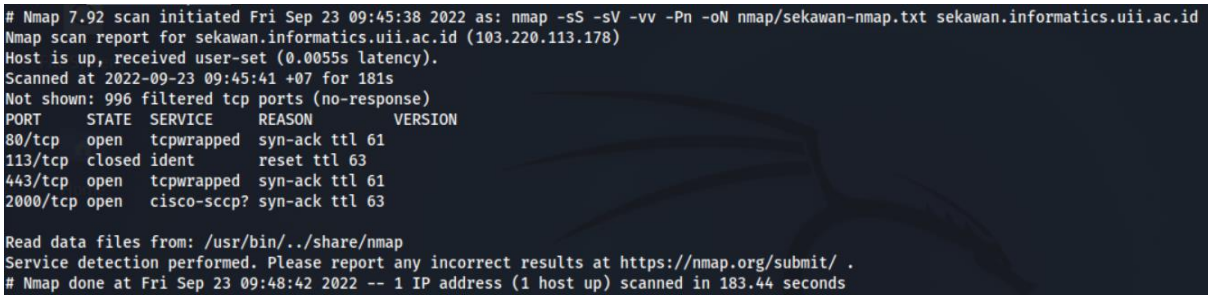
Pengujian yang akan dilakukan adalah dengan melakukan serangan pemindaian *ports* pada server dan mengidentifikasi layanan yang berjalan. Penyerangan tersebut dilakukan menggunakan bantuan perangkat lunak Nmap (*Network Mapper*). Pengujian tersebut bertujuan untuk mengetahui keadaan *port* yang terdapat pada server dan layanan yang berjalan pada port tersebut. Perintah yang digunakan untuk melakukan pengujian dengan nmap pada:

```
Nmap -sS -sV -vv -Pn -oN [nama file] [ip]
```

Gambar 4.44 Perintah Nmap

Pada Gambar 4.44, terdapat perintah yang digunakan untuk melakukan pemindaian dengan nmap. Pada perintah tersebut terdapat empat *flag* atau pilihan pemindaian yang digunakan, pilihan tersebut adalah:

- sS: menggunakan protokol TCP.
- sV: memindai port jaringan untuk mengetahui informasi dan versi layanan.
- Vv: meningkatkan level verbositas. Semakin tinggi, maka informasi yang ditampilkan pada layar akan semakin banyak dan sebaliknya. Informasi yang disimpan tetap sama.
- -Pn: menganggap semua *host online*. Tidak melakukan tahap pencarian *host* lain dalam jaringan.
- -oN: menyimpan hasil pemindaian dengan format normal.



Gambar 4.45 Hasil pemindaian server menggunakan nmap

Hasil pada pemindaian *port* dan layanan dapat dilihat pada Gambar 4.45. Pada hasil tersebut terdapat empat *port* yang telah berhasil dipindai dan layanan yang dijalankan dengan rincian:

A. 80: tcpwrapped
B. 113: ident
C. 443: tcpwrapped
D. 2000: cisco-sccp

Setelah melakukan pengujian menggunakan nmap, layanan Sekawan V2 tidak dapat diakses baik melalui peramban maupun dengan ssh. Hal tersebut dikarenakan akses telah diblokir oleh firewall yang telah diterapkan oleh BSI UII. Penyebab pemblokiran akses tersebut adalah pengujian yang dilakukan dapat dikategorikan sebagai serangan terhadap sistem jaringan milik UII.

Hasil yang diperoleh oleh perangkat lunak Nmap tidak sesuai dengan hasil *code review*. Terlihat pada hasil Nmap Gambar 4.45 dan perintah `secure socket` Gambar 4.50 tidak ada *port* dan layanan yang sama. Terdapat empat *port* dan layanan yang berjalan pada hasil nmap, sedangkan pada perintah secure socket terdapat 14 *port* dan layanan. Pada kedua hal tersebut layanan yang berjalan juga tidak sama, tidak ada layanan "tcpwrapped", "ident", dan "cisco-sccp" yang terlihat pada hasil `secure socket` begitu pun sebaliknya. Dengan begitu dapat dikatakan bahwa pemindaian port dan layanan oleh Nmap gagal dilakukan dan tidak perlu melakukan tindakan lanjut pengerasan keamanan.

B. Pengujian Dalam

User Management

Salah satu poin mengenai keamanan akun pada *security checklist* adalah penggunaan kredensial *default* yang belum atau tidak diubah. Pada akun *root server,* mempunyai akses tertinggi pada sistem operasi berbasis *unix* atau setingkat dengan

administrator pada Sistem Operasi Windows. Dengan hak akses tertinggi, pengguna tersebut dapat melakukan banyak hal pada sistem yang digunakan termasuk dengan mengubah pengaturan sistem. Maka dari itu, salah satu tujuan penyerang adalah mendapatkan akses ke dalam akun *root* (Sanjaya, Sasmita, & Arsa, 2020).

Pada Linux Ubuntu, untuk melakukan manajemen pengguna terdapat dua berkas yang digunakan untuk menyimpan informasi mengenai *user* yang terdaftar di dalam server. Berkas tersebut adalah `passwd` dan `shadow` yang berada di dalam direktori `/etc/`. Berkas `passwd` mengandung informasi penting yang diperlukan oleh sistem operasi pada saat proses login dilakukan. Sedangkan berkas `shadow` berisi tentang informasi *password* pengguna. Isi konten dari berkas `passwd` dan `shadow` pada *user root* dapat dilihat pada Gambar 4.46.

```
penelitian-18523107@informatics-sekawan:~$ sudo cat /etc/shadow | grep root
[sudo] password for penelitian-18523107:
root:*:18295:0:99999:7:::
penelitian-18523107@informatics-sekawan:~$ cat /etc/passwd | grep root
root:x:0:0:root:/root:/bin/bash
penelitian-18523107@informatics-sekawan:~$
```

Gambar 4.46 Akun *root* dalam berkas `shadow` dan `passwd`

Pada gambar terlihat bahwa akun root belum mempunyai password. Apabila sebuah akun pengguna telah memiliki kata sandi, maka simbol *asterisk* (*) pada berkas `shadow` dan akan berisi mengenai *hash* dan *salt* yang dipakai (Gite, 2022). Ketiadaan password tersebut dapat menjadi ancaman mengingat hak akses yang dimiliki. Maka dari itu diperlukan tindakan pengamanan lanjutan berupa penambahan password pada akun root.

Firewall

Pada kategori kerentanan kesalahan konfigurasi keamanan, salah satu celah keamanan yang disebutkan adalah *port* terbuka yang tidak digunakan. Pada server Sekawan v2 telah menggunakan perangkat lunak *Uncomplicated Firewall* (UFW) untuk melakukan manajemen *firewall*. Untuk mengetahui status dari ufw, dapat digunakan perintah pada Gambar 4.47:

```
Sudo ufw status
```

Gambar 4.47 Perintah untuk memeriksa status *firewall*

Gambar 4.48 menampilkan hasil perintah untuk memeriksa keadaan *firewall* yang sedang berjalan. Terdapat total delapan *rules* yang diimplementasikan, yang mengatur tujuan trafik jaringan ke *port* tujuan dengan tipe jaringan IPv4 dan IPv6.

```
penelitian-18523107@informatics-sekawan:~$ sudo ufw status
[sudo] password for penelitian-18523107:
Status: active

To                         Action      From
--                         ------      ----
2421                       ALLOW       Anywhere
443                        ALLOW       Anywhere
80                         ALLOW       Anywhere
Nginx Full                 ALLOW       Anywhere
2421 (v6)                  ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
Nginx Full (v6)            ALLOW       Anywhere (v6)
```

Gambar 4.48 Hasil pemeriksaan status *firewall*

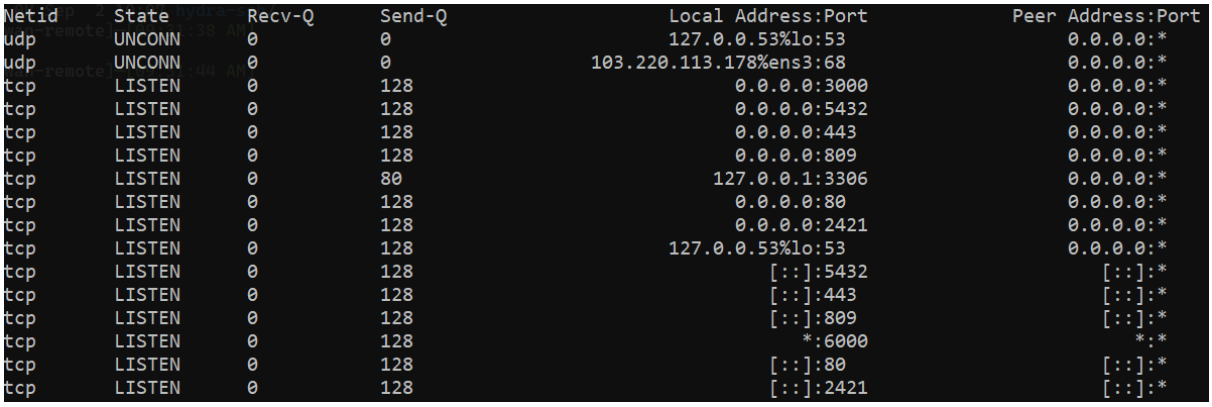
Status dari *port* yang terdapat pada server Sekawan V2 juga dapat diperiksa menggunakan perintah `ss` (*Secure Socket*). Perintah yang digunakan terdapat pada Gambar 4.49:

```
ss -tulpn
```

Gambar 4.49 Perintah untuk memeriksa status *socket*

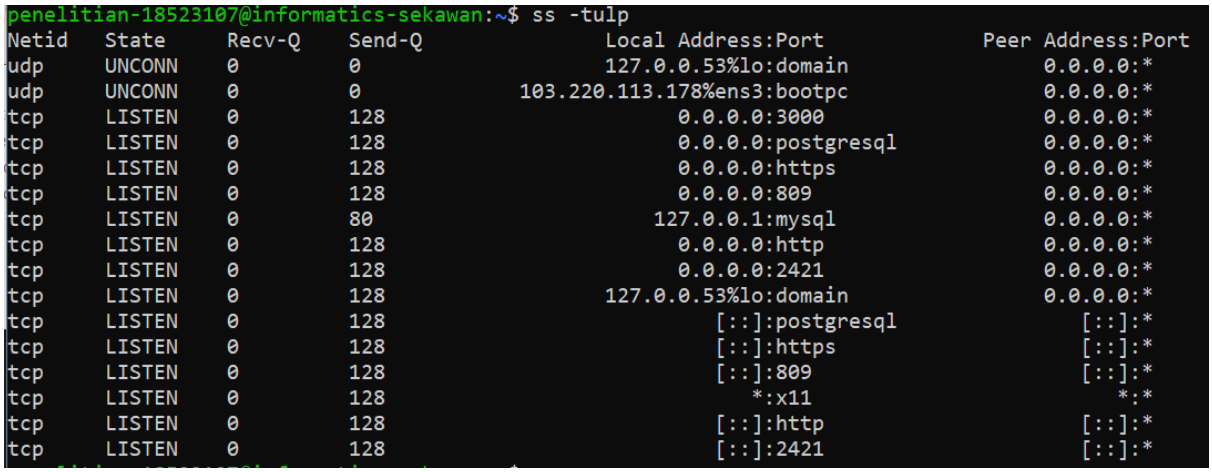
Pada gambar, terlihat perintah ss mempunyai *flag* atau *options* yang dapat digunakan untuk menambahkan utilitas pada perintah `ss` untuk menampilkan detail lainnya. Apabila perintah ss dieksekusi tanpa menambahkan *flag* atau pilihan tersebut, maka command tersebut akan menampilkan daftar *socket* yang berkategori *non-listening* dan telah membentuk koneksi. Berikut detail mengenai *flag* yang digunakan pada gambar:

a. t : menampilkan *socket* dengan jenis koneksi TCP
b. u : menampilan *socket* dengan jenis koneksi UDP
c. l : menampilkan *socket* dengan status *listening*
d. p : menampilkan proses yang sedang menggunakan *socket*
e. n : menampilkan angka *port* dan tidak menampilkan nama layanan. Apabila pilihan ini tidak dipakai maka akan ditampilkan nama layanan pada hasil, seperti pada Gambar 4.51.



Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
udp	UNCONN	0	0	127.0.0.53%lo:53	0.0.0.0:*
udp	UNCONN	0	0	103.220.113.178%ens3:68	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:3000	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:5432	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:443	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:809	0.0.0.0:*
tcp	LISTEN	0	80	127.0.0.1:3306	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:80	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:2421	0.0.0.0:*
tcp	LISTEN	0	128	127.0.0.53%lo:53	0.0.0.0:*
tcp	LISTEN	0	128	[::]:5432	[::]:*
tcp	LISTEN	0	128	[::]:443	[::]:*
tcp	LISTEN	0	128	[::]:809	[::]:*
tcp	LISTEN	0	128	*:6000	*:*
tcp	LISTEN	0	128	[::]:80	[::]:*
tcp	LISTEN	0	128	[::]:2421	[::]:*

Gambar 4.50 Perintah ss untuk memeriksa port yang terbuka tanpa nama layanan



```
penelitian-18523107@informatics-sekawan:~$ ss -tulp
```

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
udp	UNCONN	0	0	127.0.0.53%lo:domain	0.0.0.0:*
udp	UNCONN	0	0	103.220.113.178%ens3:bootpc	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:3000	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:postgresql	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:https	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:809	0.0.0.0:*
tcp	LISTEN	0	80	127.0.0.1:mysql	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:http	0.0.0.0:*
tcp	LISTEN	0	128	0.0.0.0:2421	0.0.0.0:*
tcp	LISTEN	0	128	127.0.0.53%lo:domain	0.0.0.0:*
tcp	LISTEN	0	128	[::]:postgresql	[::]:*
tcp	LISTEN	0	128	[::]:https	[::]:*
tcp	LISTEN	0	128	[::]:809	[::]:*
tcp	LISTEN	0	128	*:x11	*:*
tcp	LISTEN	0	128	[::]:http	[::]:*
tcp	LISTEN	0	128	[::]:2421	[::]:*

Gambar 4.51 Perintah ss untuk memeriksa port yang terbuka dengan nama layanan

Hasil dari perintah ss dapat dilihat pada Gambar 4.51. Pada gambar tersebut terdapat 14 *port* dengan status *listening,* yang mana berarti *port-port* tersebut sedang mendengarkan permintaan yang akan masuk ke server. Layanan yang berjalan pada masing-masing *port* terlihat pada Gambar 4.51, yaitu: postgresql, https, mysql, http, dan x11. Pemeriksaan dilakukan sebelum proses *hardening*, ditunjukkan pada layanan mysql yang berjalan pada port 3306. Hal tersebut akan diperbaiki dengan penghapusan *software* dalam proses *hardening*.

4.3.3.5 Vulnerable and Outdated Components

Menganalisis kesalahan konfigurasi keamanan dalam OWASP Top 10 2021 pada server sangat penting karena membantu mengidentifikasi dan menyelesaikan pengaturan yang tidak terkonfigurasi dengan benar yang dapat mengakibatkan kerentanan. Dengan mengatasi kesalahan konfigurasi ini, risiko akses tidak sah, kebocoran data, dan potensi pelanggaran keamanan dapat diminimalkan. Pendekatan proaktif ini memperkuat

keamanan keseluruhan lingkungan server, memastikan keamanan informasi sensitif dan mempertahankan kepercayaan pengguna dan *stakeholders*.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar server pada celah keamanan *Vulnerable and Outdated Components*. Pemeriksaan versi dan kerentanan masing-masing komponen ataupun library yang digunakan pada server hanya dimungkinkan dengan pemeriksaan dari dalam.

B. Pengujian Dalam

***Software* yang Tidak Digunakan**

Server menggunakan sistem operasi Ubuntu, yang mana menggunakan perangkat lunak APT (*Advanced Package Tool*) untuk melakukan manajemen paket/perangkat lunak yang dipasang dalam OS. Tidak semua dari *software* yang dipasang tersebut digunakan oleh server. Untuk meminimalisir *attack surface,* diperlukan audit perangkat lunak yang telah dipasang. Perangkat lunak yang terpasang dalam server, diperiksa berdasarkan kebutuhan dari Aplikasi Sekawan-API dan Sekawan-FE. Gambar 4.52 berisi tentang perintah yang digunakan untuk menampilkan paket/*software* yang dipasang pada Sistem Operasi Ubuntu:

```
apt list --installed
```

Gambar 4.52 Perintah terminal Linux untuk memeriksa perangkat lunak yang terpasang

Untuk melakukan pencarian perangkat lunak yang dengan kata kunci spesifik, perintah pada Gambar 4.52 dapat dimodifikasi menjadi seperti pada Gambar 4.53:

```
Apt list -installed | grep ["keyword"]
```

Gambar 4.53 Perintah terminal Linux untuk memeriksa perangkat lunak yang terpasang dan pencarian *package*

Pada perintah dalam Gambar 4.53, terdapat beberapa tambahan yang diberikan yakni simbol "|" dan perintah `grep`. Simbol "|" pada perintah terminal Linux digunakan untuk melakukan *pipeline* atau menyalurkan *output* dari perintah sebelumnya untuk digunakan sebagai input dalam perintah berikutnya. *Command* `grep` digunakan untuk melakukan pencarian berdasarkan pola atau kata kunci yang diberikan. Apabila perintah pada Gambar 4.53 dieksekusi, terminal akan melakukan pencarian *software* yang terpasang lalu hasil tersebut akan difilter oleh perintah `grep`. Hasil pencarian terdapat pada Gambar 4.54, Gambar 4.55, Gambar 4.56, dan Gambar 4.57.

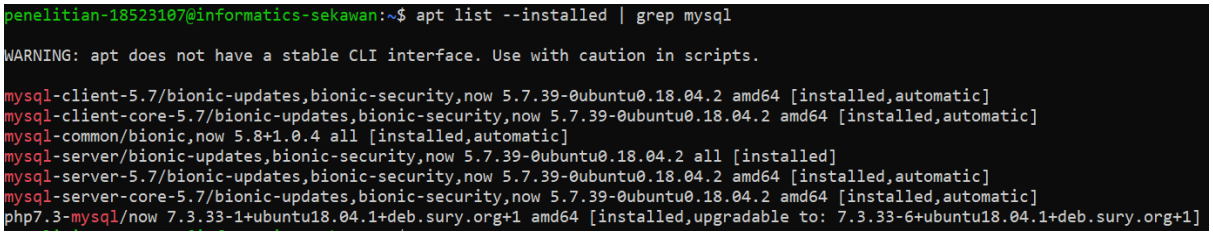
Berdasarkan aktivitas *code review* Sekawan V2 menggunakan web server nginx, DBMS postgresql, dan bahasa pemrograman go. Sementara itu, server juga telah terpasang web server apache2, DBMS mysql, dan bahasa pemrograman php. Perangkat lunak telnet juga ditemukan terpasang dan tidak diperlukan karena akses *remote* ke server dilakukan dengan menggunakan protokol *Secure Shell* (SSH). Pencarian dilakukan menggunakan perintah pada Gambar 4.53 dan keyword "apache2", "mysql", "php", dan "telnet" dan berhasil menemukan 33 *package* yang tidak dibutuhkan dan terpasang pada server. Rincian *packages*: apache2, apache2-bin, apache2-data, apache2-utils, mysql-client, mysql-client-core-5.7, mysql-common, mysql-server, mysql-server-5.7, mysql-server-core-5.7, php-common, php7.3-fpm, php7.3-common, php7.3-zip, php7.3-curl, php7.3-xml, php7.3-xmlrpc, php7.3-json, php7.3-mysql, php7.3-pdo, php7.3-gd, php7.3-imagick, php7.3-ldap, php7.3-imap, php7.3-mbstring, php7.3-intl, php7.3-cli, php7.3-recode, php7.3-tidy, php7.3-bcmath, php7.3-opcache, dan telnet. Dengan adanya perangkat lunak tidak terpakai, maka dari itu diperlukan tindakan lanjutan berupa *hardening*.

```
penelitian-18523107@informatics-sekawan:~$ apt list --installed | grep apache2

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

apache2/bionic-updates,bionic-security,now 2.4.29-1ubuntu4.25 amd64 [installed,auto-removable]
apache2-bin/bionic-updates,bionic-security,now 2.4.29-1ubuntu4.25 amd64 [installed,auto-removable]
apache2-data/bionic-updates,bionic-security,now 2.4.29-1ubuntu4.25 all [installed,auto-removable]
apache2-utils/bionic-updates,bionic-security,now 2.4.29-1ubuntu4.25 amd64 [installed,auto-removable]
```

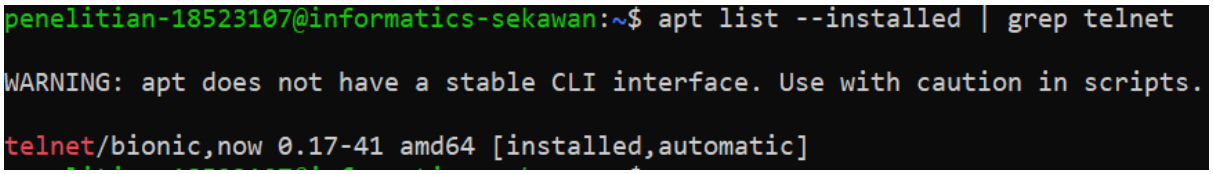
Gambar 4.54 Pencarian *software* Apache2

```
penelitian-18523107@informatics-sekawan:~$ apt list --installed | grep php

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

php-common/bionic,now 2:92+ubuntu18.04.1+deb.sury.org+2 all [installed,automatic]
php7.3-bcmath/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-cli/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-common/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-curl/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-fpm/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-gd/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-imagick/now 3.6.0-4+ubuntu18.04.1+deb.sury.org+10 amd64 [installed,upgradable to: 3.7.0-2+ubuntu18.04.1+deb.sury.org+2]
php7.3-imap/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-intl/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-json/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-ldap/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-mbstring/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-mysql/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-opcache/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-readline/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-recode/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-tidy/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-xml/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-xmlrpc/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
php7.3-zip/now 7.3.33-1+ubuntu18.04.1+deb.sury.org+1 amd64 [installed,upgradable to: 7.3.33-6+ubuntu18.04.1+deb.sury.org+1]
```

Gambar 4.55 Pencarian *software* php



Gambar 4.56 Pencarian *software* mysql



Gambar 4.57 Pencarian *software* telnet

4.3.3.6 Identification and Authentication Failures

Menganalisis kegagalan identifikasi dan otentikasi dalam OWASP Top 10 2021 pada server sangat penting karena memungkinkan identifikasi dan penyelesaian kerentanan yang dapat mengancam identitas pengguna dan akses sistem. Dengan mengatasi kegagalan-kegagalan ini, risiko akses tidak sah, kebocoran data, dan potensi insiden keamanan dapat diminimalkan. Pendekatan proaktif ini memperkuat keamanan keseluruhan lingkungan server, memastikan identifikasi pengguna yang akurat, mekanisme otentikasi yang kuat, dan menjaga integritas informasi sensitif.

A. Pengujian Luar

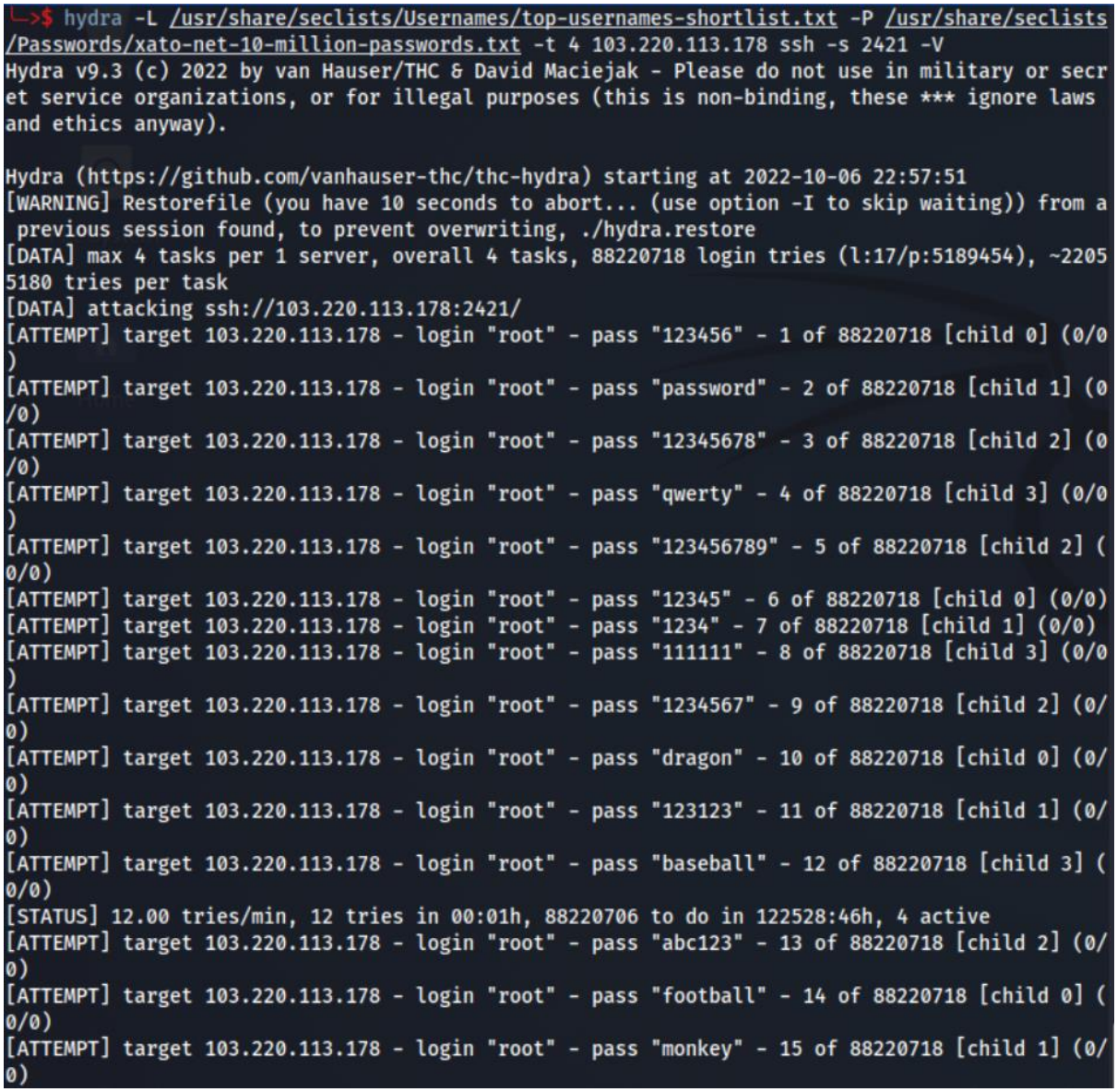
SSH *Bruteforce*

Salah satu protokol yang digunakan untuk mengakses server dari jarak jauh adalah ssh, salah satu serangan yang dapat menyerang protokol tersebut adalah *bruteforce.* Merujuk pada Proses yang dilakukan adalah dengan melakukan percobaan *login* dengan menggunakan kombinasi dari *username* dan *password*.

Pengujian dilakukan dengan menggunakan perangkat lunak Hydra. *Software* tersebut digunakan untuk melakukan otomatisasi serangan *bruteforce*, sehingga penyerang tidak perlu melakukan *input* data nama pengguna dan kata sandi secara manual. Target dari serangan adalah layanan *Secure Shell* (SSH), yang digunakan untuk melakukan akses server dari jarak jauh. Perintah Hydra yang digunakan pada Gambar 4.58.

```
Hydra -L [file username] -P [file password] -t 4 [ip] -s [port] -V
```

Gambar 4.58 Perintah hydra untuk serangan *bruteforce* protokol ssh



Gambar 4.59 Hasil pengujian serangan *bruteforce* dengan Hydra

Pada Gambar 4.59 terlihat hasil yang diperoleh dari proses pengujian *bruteforce* ssh dengan Hydra. Pada pengujian tersebut Hydra menggunakan *wordlist* atau kumpulan dari *username* dan *password* yang populer sebagai kombinasi. Hasil pengujian terlihat bahwa Hydra belum bisa mendapatkan kredensial yang digunakan untuk proses *login* pada server Sekawan V2. Hal tersebut dapat terjadi karena kombinasi dari *username* dan *password* yang digunakan tidak terdapat kombinasi yang sesuai dengan kredensial yang digunakan pada Server Sekawan V2. Setelah melakukan pengujian menggunakan hydra, layanan Sekawan V2 tidak dapat diakses baik melalui peramban maupun dengan login ssh. Hal tersebut dikarenakan akses telah diblokir oleh firewall yang telah diterapkan oleh BSI UII karena pengujian tersebut dapat dikategorikan sebagai serangan. Namun pengujian yang

dilakukan tidak sepenuhnya memitigasi serangan, penyerang dapat mencoba untuk melakukan serangan lain dengan menggunakan *wordlist* yang berbeda maupun jenis serangan lainnya. Server Sekawan V2 tidak melakukan pemblokiran pada percobaan login yang berulang-ulang. Dengan begitu diperlukan tindakan lanjut untuk pengerasan keamanan.

B. Pengujian Dalam

Salah satu poin yang ditekankan oleh OWASP Top 10 2021 pada kerentanan *Identification and Authentication Failures* adalah penggunaan *password* yang lemah. Pada sistem berbasis linux, pemeriksaan kekuatan dan kompleksitas kata sandi dapat menggunakan bantuan alat `cracklib-check`. Sintaks perintah yang digunakan terlihat pada Gambar 4.60.

```
echo "[password]" | cracklib-check
```

Gambar 4.60 Perintah pengujian kekuatan kata sandi

Hasil yang diperoleh dari perintah cracklib-check dapat dilihat pada Gambar 4.61. Pada gambar tersebut kata sandi yang dimasukan mendapat hasil "OK" yang berarti kata sandi tersebut sudah cukup panjang dan kompleks. Kata sandi yang ditampilkan pada Gambar 4.61 bukan merupakan kata sandi asli, namun kata sandi yang asli mendapatkan hasil yang serupa dalam pengujian.

Gambar 4.61 Hasil dari pemeriksaan kata sandi

4.3.3.7 Software and Data Integrity Failures

Menganalisis kegagalan integritas perangkat lunak dan data dalam OWASP Top 10 2021 pada server sangat penting untuk mengidentifikasi dan mengatasi kerentanan yang dapat mengancam integritas komponen perangkat lunak dan data yang disimpan. Dengan mengatasi kegagalan-kegagalan ini, organisasi dapat mencegah perubahan yang tidak sah, manipulasi data, dan potensi pelanggaran keamanan. Proses ini memperkuat keamanan keseluruhan lingkungan server, memastikan akurasi dan kepercayaan dalam perangkat lunak dan data.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar server pada celah keamanan *Software and Data Integrity Failures*. Pengujian integritas kode sumber dan library yang digunakan oleh

server hanya dimungkinkan oleh pengujian dalam (*code review*) karena pemeriksaan tersebut juga dapat memeriksa checksum atau hash yang dimiliki oleh masing-masing komponen.

B. Pengujian Dalam

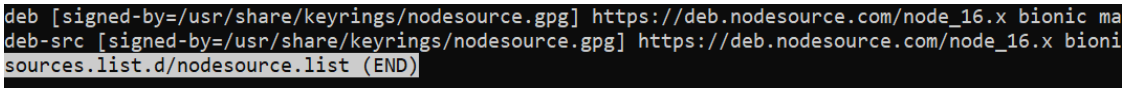
Penggunaan Repositori

Linux menggunakan repositori *online* yang digunakan untuk menyimpan perangkat lunak penting atau tambahan yang dapat dipasang di dalam server. Pada Sistem Operasi Ubuntu, untuk melihat alamat repositori yang digunakan dapat membuka berkas `/etc/apt/sources.list`. Dalam *file* tersebut, terdapat URL yang digunakan oleh manajer paket `apt` Ubuntu untuk melakukan instalasi, *update* repositori, dan *upgrade software*. Isi konten dari berkas `sources.list` default yang digunakan oleh Ubuntu terdapat pada Gambar 4.62:

```
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://id.archive.ubuntu.com/ubuntu bionic main restricted
# deb-src http://id.archive.ubuntu.com/ubuntu bionic main restricted

## Major bug fix updates produced after the final release of the
## distribution.
deb http://id.archive.ubuntu.com/ubuntu bionic-updates main restricted
# deb-src http://id.archive.ubuntu.com/ubuntu bionic-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://id.archive.ubuntu.com/ubuntu bionic universe
# deb-src http://id.archive.ubuntu.com/ubuntu bionic universe
deb http://id.archive.ubuntu.com/ubuntu bionic-updates universe
# deb-src http://id.archive.ubuntu.com/ubuntu bionic-updates universe

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://id.archive.ubuntu.com/ubuntu bionic multiverse
# deb-src http://id.archive.ubuntu.com/ubuntu bionic multiverse
deb http://id.archive.ubuntu.com/ubuntu bionic-updates multiverse
# deb-src http://id.archive.ubuntu.com/ubuntu bionic-updates multiverse

## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes
## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.
deb http://id.archive.ubuntu.com/ubuntu bionic-backports main restricted universe multiverse
# deb-src http://id.archive.ubuntu.com/ubuntu bionic-backports main restricted universe mult

## Uncomment the following two lines to add software from Canonical's
## 'partner' repository.
## This software is not part of Ubuntu, but is offered by Canonical and the
## respective vendors as a service to Ubuntu users.
# deb http://archive.canonical.com/ubuntu bionic partner
# deb-src http://archive.canonical.com/ubuntu bionic partner

deb http://id.archive.ubuntu.com/ubuntu bionic-security main restricted
# deb-src http://id.archive.ubuntu.com/ubuntu bionic-security main restricted
```

Gambar 4.62 Isi berkas `sources.list`

Repositori tambahan juga digunakan oleh server Sekawan V2. Hal tersebut bertujuan untuk menambahkan perangkat lunak yang tidak terdapat pada repositori *default* dari Ubuntu. Repositori yang ditambahkan berasal dari nodesource, yang mana digunakan untuk melakukan instalasi *software* nodejs dengan vesi 16 secara spesifik terlihat pada Gambar 4.63.



Gambar 4.63 Isi berkas `nodesource.list`

Repositori yang digunakan oleh Server Sekawan V2 merupakan repositori resmi dari Ubuntu dan Nodejs, sehingga tidak memerlukan tindakan lanjut berupa *hardening*.

4.3.3.8 Security Logging and Monitoring Failures

Menganalisis kegagalan pencatatan dan pemantauan keamanan dalam OWASP Top 10 2021 pada server sangat penting karena memungkinkan identifikasi dan penyelesaian kerentanan yang berasal dari pelacakan aktivitas sistem dan insiden keamanan yang tidak memadai. Dengan memperbaiki kegagalan-kegagalan ini, organisasi dapat meningkatkan kemampuan mereka dalam mendeteksi dan merespons tindakan tidak sah, potensi pelanggaran, dan peristiwa keamanan. Pendekatan proaktif ini memperkuat keamanan keseluruhan lingkungan server, memastikan identifikasi ancaman yang efektif, penanganan insiden yang tepat waktu, dan menjaga integritas data sensitif.

A. Pengujian Luar

Tidak terdapat skenario ataupun panduan yang dapat digunakan pada security checklist untuk pengujian luar server pada celah keamanan *Security Logging and Monitoring Failures.* Hal tersebut dikarenakan data-data mengenai log dan fungsi *monitoring* hanya dapat diakses melalui pengujian dalam dan tidak dapat dilihat dari luar, sehingga pemeriksaan tidak dimungkinkan.

B. Pengujian Dalam

Linux Journal Log

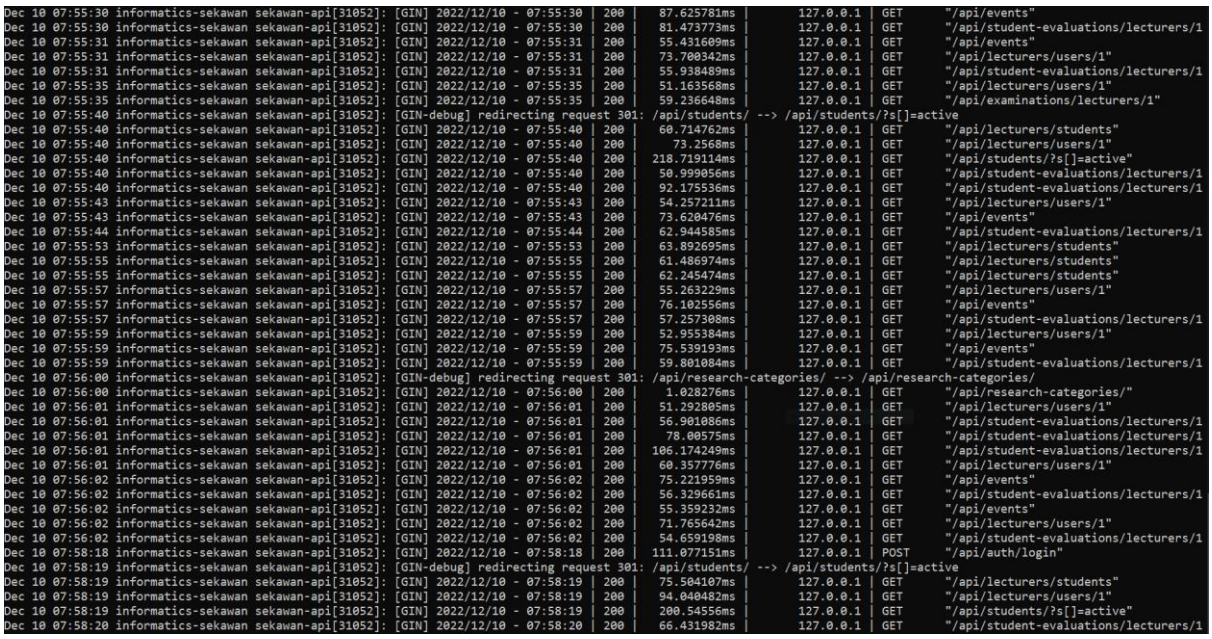
Sistem Operasi Linux Ubuntu telah menggunakan sistem init Systemd yang bertugas untuk mengelola sistem dan layanan dalam Linux. Pada sistem operasi tersebut berjalan berbagai layanan termasuk dengan aplikasi Sekawan V2 yang nantinya akan dijalankan oleh server pada fase *deployment*. Untuk mengetahui status dari layanan yang berjalan, diperlukan suatu fungsi yang mencatat keadaan sistem baik dalam keadaan normal maupun jika terjadi *error*. Pada Linux dengan Systemd

telah mempunyai perintah yang dapat digunakan untuk melakukan *query* log pada sistem yakni `journalctl`. Pada server Sekawan V2 untuk memeriksa status *log* server dapat digunakan perintah Gambar 4.64:

```
sudo journalctl -u [nama layanan]
```

Gambar 4.64 Perintah `journalctl`

Hasil dari perintah journalctl dapat dilihat pada Gambar 4.65 dan Layanan yang diperiksa adalah sekawan-api. Pada gambar tersebut hanyalah sebagian kecil dari log yang telah ditulis dan tidak dapat ditampilkan seluruhnya karena banyaknya konten yang telah di-*generate* oleh aplikasi. Hasil yang ditampilkan menunjukan bahwa sistem berjalan dengan normal dengan log yang ditampilkan berupa *timestamp* permintaan web, nama layanan, kode http, metode http, dan URL yang diakses. Kode http yang ditampilkan menunjukan status 200 yang mana mengindikasikan bahwa *request* berhasil. Fungsi logging pada sistem berjalan dengan baik dan tidak membutuhkan tindakan lanjut berupa *hardening*.



Gambar 4.65 Hasil perintah `journalctl`

4.4 Hasil Identifikasi Kerentanan

Hasil dari proses pengujian dalam atau code review dapat dilihat pada Tabel 4.3. Pada tabel tersebut hasil dari pengujian dalam ditampilkan dalam beberapa kategori, yaitu: komponen, peringkat kerentanan, hasil, dan keputusan untuk proses *hardening*. Pada komponen Sekawan-API, terdapat delapan celah keamanan yang diperiksa berdasarkan *security checklist* dan hanya

terdapat satu celah kerentanan yang akan dilanjutkan ke proses *hardening* sedangkan tujuh lainnya tidak perlu proses *hardening*. Komponen Sekawan-FE hasil yang diperoleh dapat dikelompokan dalam enam peringkat kerentanan *security checklist*. Dari enam kerentanan tersebut, terdapat dua hasil yang akan masuk ke proses *hardening*. Pada komponen server terdapat tujuh peringkat celah keamanan yang diperiksa dan dua diantaranya memerlukan perbaikan dalam proses *hardening*.

Tabel 4.3 Tabel hasil Pengujian Dalam (*code review*)

Komponen	Peringkat Kerentanan	Hasil	*Hardening*	
			Ya	Tidak
Sekawan-API	1	Sekawan-API telah menerapkan validasi JWT.		✓
	2	JSON Web Token menggunakan data tambahan pada pembuatannya.		✓
	3	Sekawan-API telah menerapkan *prepared sql statements.*		✓
	5	Sekawan-API tidak memberikan respon kesalahan yang terlalu informatif.		✓
	6	Sekawan-API terdapat dependensi yang dapat diperbaharui.	✓	
	8	Proses *import modules* di go telah menerapkan proses *hash.*		✓
	9	Pengembang telah menerapkan fitur *logging.*		✓
	10	Tidak memproses *resource remote* dan lokal pada URL.		✓
Sekawan-FE	1	Pengguna tidak dapat mengakses halaman diluar *role* yang telah ditentukan.		✓
	2	Telah menggunakan algoritma *hash* yang kuat.		✓
	5	Aplikasi telah menerapkan pesan kesalahan *custom* yang tidak informatif ke *end user*.		✓
		Belum menerapkan *http response header* tambahan untuk keamanan.	✓	
	6	Terdapat celah keamanan yang ditemukan pada dependensi dengan berbagai tingkat keamanan.	✓	
	8	Dependensi yang dipasang telah melalui proses *hash*		✓
	9	Log Sekawan-FE dapat ditampilkan, dan tidak *error.*		✓
Server	1	Linux memiliki izin akses dengan *role* yang berbeda.		✓
	2	Sistem Operasi Ubuntu telah menerapkan fungsi *hash* dan *salt* untuk kata sandi.		✓

		Server telah menggunakan koneksi https.		✓
	4	Telah menerapkan manajemen layanan untuk aplikasi Sekawan V2 untuk memastikan ketersediaan layanan.		✓
	5	Akun *root* server belum mempunyai *password.*	✓	
		Status firewall aktif dan mengawasi port dengan aturan yang telah diterapkan.		✓
	6	Terdapat 33 *software* yang tidak digunakan.	✓	
	8	Server menggunakan repositori resmi.		✓
	9	Fungsi *logging* server berfungsi dengan normal dan tidak ada *error.*		✓

Keterangan:

a. Komponen: komponen-komponen yang menyusun sistem Sekawan V2, yakni: Sekawan-API, Sekawan-FE, dan server.
b. Peringkat Kerentanan: Nomor peringkat berdasarkan OWASP Top 10 2021 pada Tabel 2.1.
c. Hasil: hasil dari proses *code review.*
d. *Hardening*: Tanda centang (✓) mengindikasikan tindakan lanjut yang akan diambil. Apabila "Ya" akan dilanjutkan dengan *hardening* dan sebaliknya.

Hasil dari proses pengujian luar dapat dilihat pada Tabel 4.4. Hasil yang ditemukan pada komponen Sekawan-API terdapat satu kerentanan berdasarkan security checklist yang akan masuk pada proses *hardening*. Pada komponen Sekawan-FE satu celah keamanan yang ditemukan akan diperbaiki pada hardening. Pada komponen Server terdapat satu kerentanan yang memerlukan tindakan lanjut berupa *hardening*.

Tabel 4.4 Tabel hasil pengujian Luar

Komponen	Peringkat Kerentanan	Hasil	Hardening	
			Ya	Tidak
Sekawan-API	1	API tidak memberikan respon apabila *request* menggunakan metode http yang salah.		✓
	3	tidak terdapat filter *input* yang diterapkan, dapat menggunakan tag html secara bebas ke dalam masukan.	✓	

Sekawan-FE	5	Halaman Sekawan V2 dapat dimuat oleh halaman lain menggunakan html *tag* iframe.	✓	
Server	7	Sistem tidak melakukan pemblokiran pada serangan *bruteforce.*	✓	
	5	hasil pemindaian *port* dan layanan berbeda dengan hasil pemeriksaan pada code review.		✓

Keterangan:

e. Komponen: komponen-komponen yang menyusun sistem Sekawan V2, yakni: Sekawan-API, Sekawan-FE, dan server.
f. Peringkat Kerentanan: Nomor peringkat berdasarkan OWASP Top 10 2021 pada Tabel 2.1.
g. Hasil: hasil dari proses *code review.*
h. *Hardening*: Tanda centang (✓) mengindikasikan tindakan lanjut yang akan diambil. Apabila "Ya" akan dilanjutkan dengan *hardening* dan sebaliknya.

Hasil dari tahapan identifikasi kerentanan menghasilkan luaran bagian-bagian komponen dari Sekawan V2 yang harus diperbaiki Tabel 4.5. Perbaikan tersebut didasarkan atas penemuan kerentanan yang termasuk di dalam *security checklist*. Dua tahap (pengujian luar dan dalam) yang dilakukan pada proses identifikasi kerentanan menemukan celah keamanan dengan rincian lima celah pada tahap pengujian dalam dan tiga celah pada pengujian luar. Masing-masing komponen Sekawan V2 mempunyai kerentanan dengan dua kerentanan pada Sekawan-API, tiga kerentanan pada Sekawan-FE, dan tiga kerentanan pada server. Celah keamanan pada OWASP Top 10 2021 yang terdapat pada komponen Sekawan V2 adalah: *Injection*, *Security Misconfiguration*, *Vulnerable and Outdated Components*, dan *Identifitacion and Authentication Failures*. Kerentanan-kerentanan tersebut akan dimitigasi pada tahap Implementasi Hardening.

Tabel 4.5 Kerentanan yang perlu *hardening* pada komponen Sistem Sekawan V2

Rank	OWASP Top 10 2021	Sekawan-API	Sekawan-FE	Server
1	*Broken Access Control*			
2	*Cryptographic Failures*			
3	*Injection*	✓		
4	*Insecure Design*			

5	*Security Misconfiguration*		✓	✓
6	*Vulnerable and Outdated Components*	✓	✓	
7	*Identification and Authentication Failures*			✓
8	*Software and Data Integrity Failures*			
9	*Security Logging and Monitoring Failures*			
10	*Server-Side Request Forgery (SSRF)*			

4.5 Implementasi *Hardening*

Sistem Informasi Sekawan v2 dibangun dengan arsitektur *microservice*. Maka dari itu implementasi *hardening* keamanan dilakukan pada masing-masing komponen yang membangun sistem tersebut berdasarkan hasil dari pengujian luar dan pengujian dalam. Tahap ini akan mengimplementasi pada tiga komponen sistem Sekawan V2, yaitu: Sekawan-API, Sekawan-Fe, dan Server.

4.5.1 Sekawan-API

Pengerasan keamanan pada Sekawan-API sangat penting untuk melindungi data sensitif, mencegah serangan siber, dan menjaga integritas sistem. Dengan menerapkan langkah-langkah pengamanan seperti validasi input, autentikasi yang kuat, enkripsi data, dan pembatasan akses, SekawanAPI dapat menjadi lebih tahan terhadap serangan seperti SQL injection, serangan terhadap kerentanan akses, dan upaya peretasan. Dengan mengamankan Sekawan-API, tim pengembang dapat memastikan bahwa layanan yang mereka tawarkan tetap aman, dapat diandalkan, dan sesuai dengan standar keamanan yang diperlukan. Celah keamanan yang akan diperbaiki adalah: *Injection* dan *Vulnerable and Outdated Components*

A. Injections

***Stored Cross Site Scripting* (XSS)**

Salah satu celah keamanan injeksi yang ditermukan pada proses identifikasi kerentanan adalah *Cross Site Scripting* (XSS) yang tersimpan. Celah tersebut memanfaatkan *form input* yang tidak divalidasi sehingga dapat digunakan untuk mengeksekusi *script* yang dijalankan (Sanjaya, Sasmita, & Arsa, 2020).

Hardening yang dilakukan adalah dengan melakukan sanitasi terhadap *input*. Proses yang dilakukan adalah dengan melakukan sanitasi dan memodifikasi input yang diberikan dengan mengubah atau menghapus simbol atau karakter tertentu (Alemi, 2021). Apabila pengguna memasukan *malicious tag HTML*, maka proses sanitasi tersebut akan menghapus *tag HTML* pada input yang tidak sesuai dengan aturan yang telah ditetapkan pada kode sumber.

Sanitasi input diimplementasikan ke dalam sebuah fungsi. Tujuan dari hal tersebut adalah untuk dapat digunakan berulang-ulang dalam kode yang berbeda. Fungsi sanitasi dibuat dengan bahasa pemrograman Go dan menggunakan *package* bluemonday yang dibuat oleh Microcosm (Microcosm, 2022). Pada Gambar 4.66, fungsi `HTMLSanitize` mempunyai satu parameter yang akan menerima masukan bertipe `string` dan akan mengembalikan satu nilai bertipe `string`.

Langkah selanjutnya adalah dengan membuat aturan yang akan digunakan untuk sanitasi dan dimasukkan dalam variabel `p`. Kode `p.AllowedElements` digunakan untuk mendeskripsikan elemen HTML yang diperbolehkan untuk digunakan. Proses sanitasi dilakukan pada kode `p.Sanitize(input)` dan hasil dari operasi tersebut akan disimpan pada variabel `sanitized`. Tahap terakhir yang dilakukan pada fungsi HTMLSanitize adalah mengembalikan hasil operasi sanitasi dengan kode `return` diikuti dengan nilai atau variabel yang akan dikembalikan.

```
package utils

import (
    "github.com/microcosm-cc/bluemonday"
)

// This function will strip all html elements from user's input
func HTMLSanitize(input string) string {
    p := bluemonday.NewPolicy()

    // Allowed elements
    p.AllowElements("p")
    p.AllowElements("ol")
    p.AllowElements("li")

    sanitized := p.Sanitize(input)
    return sanitized
}
```

Gambar 4.66 Kode fungsi HTMLSanitize

Fungsi HTMLSanitize diterapkan pada proses penyimpanan data ke basis data pada eksekusi SQL. Sebelum penerapan fungsi sanitasi, parameter dimasukkan secara langsung ke dalam *query*. Contoh *query* SQL sebelum menggunakan fungsi HTMLSanitize dapat dilihat pada Gambar 4.67.

```
err := tx.QueryRow(SQL, studentLogbook.StudentId, studentLogbook.Title,
studentLogbook.Description, studentLogbook.Location,
studentLogbook.Date).Scan(&createdId)
```

Gambar 4.67 Query SQL sebelum penerapan fungsi HTMLSanitize

Setelah penerapan fungsi sanitasi, kode akan terlihat pada Gambar 4.68. Kode `utils.HTMLSanitize` dipanggil ke dalam *query* SQL dan parameter *query* sebelumnya

menjadi masukan untuk fungsi sanitasi. Fungsi tersebut hanya dapat menerima *input* bertipe *string*, maka dari itu parameter yang akan menjadi masukan juga harus memiliki tipe data yang sama. Pada *query* SQL, terdapat tiga parameter bertipe *string*, yaitu `studentLogbook.Title`, `studentLogbook.Description`, dan `studentLogbook.Description`.

```
err := tx.QueryRow(SQL, studentLogbook.StudentId,
utils.HTMLSanitize(studentLogbook.Title),
utils.HTMLSanitize(studentLogbook.Description),
utils.HTMLSanitize(studentLogbook.Description),
studentLogbook.Date).Scan(&createdId)
```

Gambar 4.68 Query SQL setelah penerapan fungsi HTMLSanitize

B. Vulnerable and Outdated Components

Go *Outdated Packages*

Pada tahap *code review*, telah ditemukan sebanyak empat *packages* yang telah usang yaitu: gin-gonic, validator, pq, dan bluemonday. Maka dari itu sesuai dengan *security checklist* dengan kerentanan *vulnerable* and *outdated components* bagian *How to Prevent* dependensi-dependensi tersebut akan diperbarui. Gambar 4.69 memperlihatkan pembaruan `gin-gonic` bersamaan dengan dependensi validator. Gambar 4.70 menampilkan pembaruan `pq` dan Gambar 4.71 adalah proses pembaruan dependensi bluemonday. Setelah proses selesai, seluruh *packages* yang bermasalah telah berhasil diperbarui.

```
$ go get github.com/gin-gonic/gin@v1.8.1
go: downloading github.com/golang/protobuf v1.5.0
go: downloading github.com/stretchr/testify v1.7.1
go: upgraded github.com/gin-gonic/gin v1.7.4 => v1.8.1
go: upgraded github.com/go-playground/validator/v10 v10.9.0 => v10.10.0
```

Gambar 4.69 Pembaruan *package* gin gonic

```
$ go get github.com/lib/pq@v1.10.7
go: downloading github.com/lib/pq v1.10.7
go: upgraded github.com/lib/pq v1.10.3 => v1.10.7
```

Gambar 4.70 Pembaruan *package* pq

```
$ go get github.com/microcosm-cc/bluemonday@v1.0.21
go: downloading github.com/microcosm-cc/bluemonday v1.0.21
go: downloading golang.org/x/net v0.0.0-20221002022538-bcab6841153b
go: upgraded github.com/microcosm-cc/bluemonday v1.0.18 => v1.0.21
```

Gambar 4.71 Pembaruan *package* bluemonday

4.5.2 Sekawan-FE

Pengerasan keamanan pada web front-end adalah langkah penting untuk mengurangi risiko serangan siber dan melindungi pengguna serta data. Dengan menerapkan langkah-langkah

seperti sanitasi input dan prinsip keamanan seperti enkripsi, aplikasi web front-end dapat mencegah serangan. Hal ini membantu menjaga privasi pengguna dan mencegah kerentanan yang dapat dimanfaatkan oleh pihak tidak sah. Celah kerentanan yang akan diperbaiki pada Sekawan-FE adalah: *Security Misconfiguration* dan *Vulnerable and Outdated Components.*

a. Security Misconfiguration

HTTP *Security Response Header*

Pada Sekawan-FE, pengerasan keamanan dilakukan dengan cara menambahkan *custom http response header* tambahan, terutama *response header* yang berkaitan dengan keamanan (Steven, et al., 2022). Penambahan tersebut dilakukan dengan pembuatan *key headers* di dalam berkas `next.config.js`. Dalam fungsi *headers*, terdapat *array* yang akan menampung dua objek yaitu `source` yang berisi *request path* yang akan masuk dan `headers` yang mana berupa *array* objek *response header*. Objek tersebut berisi mengenai properti *key* dan *value* (Kasper, et al., 2022). Implementasi *response header* dapat dilihat pada Gambar 4.72.

```
...
module.exports = {
  reactStrictMode: true,
  async headers() {
    return [
      {
        // Apply these headers to all routes in your application.
        source: "/:path*",
        headers: securityHeaders,
      },
    ];
  },
};
...
```

Gambar 4.72 Implementasi *custom http response header*

Objek *headers* memiliki sebuah nilai berupa securityHeaders dan akan menampung daftar http *response header* yang akan digunakan. Pada proses ini menggunakan rekomendasi dari situs resmi NextJS yang dikelola oleh Vercel untuk menentukan *header* keamanan yang digunakan (Steven, et al., 2022). *List* response header yang diimplementasikan dapat dilihat pada Gambar 4.73. Terdapat enam *http response header* tambahan dengan detail sebagai berikut:

a. `Strict-Transport-Security`

Header ini akan memberi tahu peramban untuk bahwa situs web hanya diperbolehkan untuk diakses dengan protokol HTTPS. Apabila pengguna melakukan akses dengan protokol HTTP akan dialihkan menjadi HTTPS secara otomatis. Penggunaan *directives*

`includeSubDomains` akan menerapkan pengaturan tersebut pada subdomain yang terdapat pada web (Perrier, et al., 2022).

b. `X-XSS-Protection`

HTTP *response header* `X-XSS-Protection` merupakan sebuah fitur dari Peramban Internet Explorer, Google Chrome, dan Safari. Penggunaan *response header* tersebut menghentikan halaman web untuk ditampilkan ketika mendeteksi serangan *reflected Cross-Site Scripting* (XSS) (Perrier, Ruikar, queengooborg, & Pardal, X-XSS-Protection - HTTP | MDN, 2022).

c. `X-Frame-Options`

HTTP response header `X-Frame-Options` dapat digunakan untuk menunjukkan tentang perizinan peramban untuk me-*render tag* HTML <iframe>. Situs web dapat menggunakan hal tersebut untuk menghindari serangan *clickjacking* (Perrier, et al., X-Frame-Options - HTTP | MDN, 2022).

d. `X-Content-Type-Options`

Response header `X-Content-Type-Options` digunakan oleh server untuk menggunakan *Multipurpose Internet Mail Extensions* (*MIME Types*) yang terdapat pada *response header* `Content-Types` dan tidak diubah. *Directive* atau arahan yang diberikan untuk *header* ini adalah `nosniff`, yang berfungsi untuk melakukan pemblokiran *request* apabila tujuan dari permintaan tersebut adalah *style* dan MIME *Type* yang diberikan bukan `text/css`, `script`, dan bukan tipe MIME JavaScript (Perrier, Ruikar, queengooborg, Willee, & Sam, X-Content-Types-Options, 2022).

e. `Referrer-Policy`

Referrer-Policy berfungsi untuk mengatur jumlah informasi yang dikirim melalui *http response header* `Referrer-Header` yang akan diikutsertakan dalam *request*. *Directive* yang digunakan adalah `origin-when-cross-origin`. Hal tersebut akan mengirimkan *origin* (protokol, domain, dan *port* pada url yang digunakan untuk akses), *path*, dan *query string* ketika melakukan permintaan pada protokol http yang sama. Sebagai contoh adalah: HTTPS ke HTTPS dan HTTP ke HTTP (Perrier & Sam, Referrer-Policy - HTTP | MDN, 2021).

f. `Content-Security-Policy`

Penggunaan *http response header* `Content-Security-Policy` adalah untuk mendeteksi dan melakukan mitigasi serangan tertentu, termasuk *Cross-Site Scripting* (XSS) dan injeksi data. Serangan tersebut dapat digunakan untuk melakukan pencurian

data, mengubah tampilan web, dan distribusi *malware.* Pada penggunaan *Content-Security-Policy*, terdapat aturan yang ditetapkan pada variabel `ContentSecurityPolicy`. Isi dari variabel tersebut mengandung *directives* yang berfungsi untuk mendeskripsikan aturan-aturan yang berlaku pada *response header* `Content-Security-Policy` (Perrier, et al., Content-Security-Policy - HTTP | MDN, 2022). Detail mengenai directives yang diterapkan dapat dilihat pada Gambar 4.74.

```
const securityHeaders = [
  {
    key: "Strict-Transport-Security",
    value: "max-age=63072000; includeSubDomains; preload",
  },
  {
    key: "X-XSS-Protection",
    value: "1; mode=block",
  },
  {
    key: "X-Frame-Options",
    value: "SAMEORIGIN",
  },
  {
    key: "X-Content-Type-Options",
    value: "nosniff",
  },
  {
    key: "Referrer-Policy",
    value: "origin-when-cross-origin",
  },
  {
    key: "Content-Security-Policy",
    value: ContentSecurityPolicy.replace(/\s{2,}/g, " ").trim(),
  },
];
```

Gambar 4.73 Variabel `securityHeaders` yang berisi *response header* yang akan digunakan

```
const ContentSecurityPolicy = `
  default-src 'self';
  font-src 'self' fonts.googleapis.com fonts.gstatic.com;
  img-src 'self' data: w3.org/svg/2000;
  object-src 'none';
  script-src 'self' 'unsafe-eval';
  style-src 'self' 'unsafe-inline' fonts.googleapis.com;
`;
```

Gambar 4.74 *Directives* untuk *response header* Content-Security-Policy

C. Vulnerable and Outdated Components

Vulnerable packages

Pada tahap *code review*, terdapat tujuh *known vulnerability* yang telah ditemukan melalui perintah `npm audit`. Hasil dari perintah tersebut juga menyertakan perintah npm yang dapat digunakan untuk melakukan perbaikan pada masing-masing celah keamanan. Perintah tersebut adalah `npm fix` dan `npm fix --force`. Perintah `npm fix` akan memperbaiki celah keamanan yang terdapat pada *package* minimist, nanoid, dan next-auth Gambar 4.75.

```
$ npm audit fix
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was created with an old version of npm,
npm WARN old lockfile so supplemental metadata must be fetched from the registry.
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please be patient...
npm WARN old lockfile
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'react-lottie@1.2.3',
npm WARN EBADENGINE   required: { npm: '^3.0.0' },
npm WARN EBADENGINE   current: { node: 'v16.15.0', npm: '8.5.5' }
npm WARN EBADENGINE }
npm WARN deprecated core-js@2.6.12: core-js@<3.4 is no longer maintained and not recommended for usage
ions could cause a slowdown up to 100x even if nothing is polyfilled. Please, upgrade your dependencies

added 513 packages, and audited 514 packages in 17s

99 packages are looking for funding
  run `npm fund` for details
```

Gambar 4.75 Perintah `npm audit fix`

Sementara itu perintah `npm fix -force` digunakan untuk melakukan perbaikan pada *package* next, node-fetch, quill, dan react-quill Gambar 4.76. *Flag* `-force` pada perintah `npm audit` digunakan apabila cakupan *metavulnerabilities* meluas hingga ke akar projek dan tidak dapat diperbaiki jika tidak melakukan perubahan dalam dependensi yang digunakan.

```
$ npm audit fix --force
npm WARN using --force Recommended protections disabled.
npm WARN audit Updating next to 12.1.6,which is outside your stated dependency range.
npm WARN audit Updating react-quill to 0.0.2,which is a SemVer major change.
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'react-lottie@1.2.3',
npm WARN EBADENGINE   required: { npm: '^3.0.0' },
npm WARN EBADENGINE   current: { node: 'v16.15.0', npm: '8.5.5' }
npm WARN EBADENGINE }

added 3 packages, removed 178 packages, changed 8 packages, and audited 339 packages in 10s

78 packages are looking for funding
  run `npm fund` for details
```

Gambar 4.76 Perintah `npm audit fix --force`

4.5.3 *Server*

Pengerasan keamanan pada server penting untuk melindungi data, sistem, dan mencegah serangan siber. Dengan pengaturan yang benar, pembaruan perangkat lunak, dan manajemen hak akses, risiko kerentanan dan serangan dapat berkurang. Pengerasan ini menjaga operasi

yang handal, kerahasiaan data, dan tingkat keamanan yang sesuai. Kerentanan yang akan ditutup pada server adalah: *Security Misconfiguration*, *Vulnerable and Outdated Components*, dan *Identification and Auhtorization Failures*.

A. Security Misconfiguration

User Management

Pada proses *code review* ditemukan bahwa akun *root* pada server belum menggunakan *password*. Mengingat ancaman dan dampak yang ditimbulkan apabila akun dengan hak akses tertinggi di sistem di akses secara ilegal oleh penyerang, maka langkah pengamanan yang dilakukan adalah dengan menambahkan kata sandi pada akun tersebut. Pada Linux, untuk mengubah kata sandi pengguna dapat digunakan perintah:

```
sudo passwd [user]
```

Gambar 4.77 Perintah `passwd`

Ketika perintah pada Gambar 4.77 dijalankan untuk akun *root*, maka pengguna tersebut akan mengubah sandi lama menjadi sandi baru. Pada Gambar 4.78 dapat dilihat terdapat *prompt* yang muncul untuk memasukkan kata sandi baru dan menuliskan *password* tersebut kembali untuk langkah verifikasi. Apabila *password* yang dimasukkan kembali sama dengan kata sandi yang diinput pertama kali, pengguna telah berhasil melakukan pergantian kata sandi. Apabila berbeda, pengguna gagal untuk mengubah kata sandi.

```
penelitian-18523107@informatics-sekawan:~$ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Gambar 4.78 Perintah `passwd` mengganti *password* pada akun *root*

B. Vulnerable and Outdated Components

Perangkat lunak yang terpasang dalam server, diperiksa berdasarkan kebutuhan dari Aplikasi Sekawan-API dan Sekawan-FE. Apabila terdapat *software* yang tidak dibutuhkan, tindakan yang diambil adalah dengan menghapus paket tersebut. Pada *software* manajer paket *Advanced Pacakage Tools* (APT), untuk menghapus perangkat lunak yang terpasang dapat menggunakan dua perintah yang dapat dipakai, yaitu Gambar 4.79 dan Gambar 4.80:

```
sudo apt remove [nama paket]
```

Gambar 4.79 Perintah apt *remove*

```
sudo apt purge [nama paket]
```

Gambar 4.80 Perintah apt *purge*

Walaupun perintah `apt remove` dan `apt purge` berfungsi untuk menghapus perangkat lunak/*package* pada OS Ubuntu, kedua perintah tersebut memiliki perbedaan. `Apt remove` akan menghapus perangkat lunak yang ditentukan, namun berkas konfigurasi yang telah dibuat tidak akan ikut dihapus, sedangkan pada perintah `apt purge`, *software* akan dihapus beserta dengan konfigurasi yang telah dibuat. Karena perangkat lunak yang akan dihapus sudah dipastikan tidak dibutuhkan, perintah yang sesuai adalah `apt purge`.

Pada perintah dalam Gambar 4.79 dan Gambar 4.80, kata kunci `sudo` digunakan untuk menjalankan perintah dengan hak akses administrator atau dalam sistem operasi berbasis unix adalah `root`. Hal tersebut dikarenakan hanya pengguna yang memiliki hak akses *root* yang berhak mengubah pengaturan sistem.

Dengan menggunakan referensi perintah pada Gambar 4.80, perintah yang digunakan untuk menghapus paket-paket yang tidak dibutuhkan pada tahap *code review* adalah:

```
sudo apt purge php7.3* apache2* mysql* nc
```

Gambar 4.81 Perintah apt untuk menghapus perangkat lunak dan konfigurasi yang terpasang

Hasil dari perintah `apt purge` ditampilkan pada Gambar 4.82.

```
Purging configuration files for php7.3-json (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-imap (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for apache2 (2.4.29-1ubuntu4.25) ...
Purging configuration files for php7.3-common (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-xmlrpc (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-tidy (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-curl (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-opcache (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-xml (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-gd (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-bcmath (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-mysql (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-cli (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-readline (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-ldap (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-fpm (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-intl (7.3.33-1+ubuntu18.04.1+deb.sury.org+1) ...
Purging configuration files for php7.3-imagick (3.6.0-4+ubuntu18.04.1+deb.sury.org+10) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.56) ...
Processing triggers for ufw (0.36-0ubuntu0.18.04.2) ...
Rules updated for profile 'Nginx Full'
Skipped reloading firewall
```

Gambar 4.82 Hasil perintah `apt purge`

C. Identification and Authentication Failures

SSH Login

Dalam pengujian pertama, salah satu serangan yang dilakukan adalah dengan melakukan percobaan *login* pada ssh secara berulang-ulang. Maka dari itu, implementasi *hardening* yang dilakukan adalah dengan membatasi dan melakukan *block* permintaan pada layanan ssh.

Perangkat lunak yang digunakan untuk melakukan hal tersebut adalah `fail2ban`. *Software* tersebut akan melakukan pemindaian pada berkas log seperti `/var/log/auth.log` dan akan melakukan pemblokiran alamat ip yang melakukan kesalahan autentikasi terlalu banyak. Pada server Sekawan v2, *software* fail2ban belum terpasang, sehingga untuk menggunakannya pengguna perlu melakukan instalasi pada server Sekawan v2 dengan perintah Gambar 4.83 di terminal:

```
sudo apt install fail2ban
```

Gambar 4.83 Perintah apt untuk memasang *software* fail2ban pada server Sekawan V2

Setelah program fail2ban terpasang dalam server, dilanjutkan dengan melakukan konfigurasi lanjutan berupa membuat salinan berkas `fail2ban.local` dari `fail2ban.conf` dan `jail.local` dari `jail.conf`. Pembuatan berkas tersebut bertujuan untuk menghindari kesalahan pada saat terjadi *upgrade* (Yarikoptic, Mikes, Lostcontrol, Henryut, & Jensck, 2013). Untuk membuat fail2ban mengawasi layanan ssh, berkas konfigurasi `jail.local` dimodifikasi dengan menambahkan kode seperti pada Gambar 4.84.

```
[ssh]
Enabled=true
Port=2421
Logpath=/var/log/auth.log
Filter=sshd
Maxretry=5
Bantime=3600
Ignoreip=127.0.0.1
```

Gambar 4.84 Pengaturan ssh pada fail2ban di berkas `jail.local`

Dapat dilihat pada Gambar 4.84, terdapat tujuh pengaturan yang ditambahkan dalam konfigurasi ssh pada fail2ban. Pada pengaturan *jail* tersebut, fail2ban akan: mengaktifkan pemindaian untuk ssh, mengawasi pada *port* 2421, mengawasi berkas `auth.log`, menerapkan filter yang terdapat pada `/etc/fail2ban/filter.d/sshd.conf`, membatasi percobaan *login* sebanyak lima kali, melakukan pemblokiran IP selama 3600 detik atau satu jam apabila terbukti melanggar, dan mengabaikan alamat IP dari 127.0.0.1 atau *localhost*.

Setelah konfigurasi diterapkan pada berkas *jail.conf*, layanan fail2ban memerlukan *restart service* yang bertujuan untuk mengaplikasikan pengaturan Gambar 4.85. Pada fail2ban untuk melakukan *restart* layanan dapat menggunakan perintah:

```
Sudo fail2ban-client restart
```

Gambar 4.85 Perintah terminal untuk restart layanan fail2ban

4.6 Validasi Hasil Hardening

Validasi Hasil Hardening hanya akan dilakukan apabila pada pengujuan pertama sistem terbukti belum aman. Skenario uji yang dilakukan sama dengan tahap identifikasi kerentanan atau sedikit berbeda untuk mengetahui hasil dari implementasi *hardening*. Validasi hanya dilakukan dengan pengujian luar untuk mensimulasikan serangan dari luar. Tahap ini akan melakukan validasi celah kerentanan yang telah dilakukan proses hardening pada tiga komponen Sekawan V2, yaitu: Sekawan-API, Sekawan-FE, dan Server.

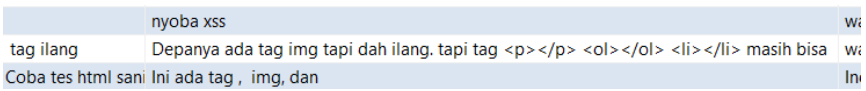
4.6.1 Sekawan-API

a. Injections

Dalam pengujian ini akan dilakukan hal yang sama dengan pengujian pertama, yakni dengan membuat *request* dengan perangkat lunak Postman. *Payload* atau data yang dikirimkan akan berisi *tag* html yang tidak diperbolehkan sesuai dengan aturan yang telah ditetapkan pada Gambar 4.66. Tujuan dari pengujian ini adalah untuk memeriksa status aplikasi setelah proses implementasi *hardening* dengan pembuatan fungsi sanitasi dilakukan. Keseluruhan *payload* dapat dilihat dalam Gambar 4.86

Gambar 4.86 Postman membuat permintaan ke *endpoint* API dengan *malicious payload*

Hasil dari pengujian tersebut dapat dilihat di Gambar 4.87. Pada penyimpanan basis data yang dibuka dengan perangkat lunak DBeaver terlihat bahwa *request* berhasil disimpan. Di dalamnya *payload* yang telah dikirimkan yaitu tag html <script>, <img>, dan <iframe> telah berhasil dihapus. Dengan demikian proses *hardening* telah berhasil dilakukan.



Gambar 4.87 Hasil dari *request* Postman dengan *malicious payload*

4.6.2 Sekawan-FE

1. Security Misconfiguration

HTTP *Response Header* dan *Clickjacking*

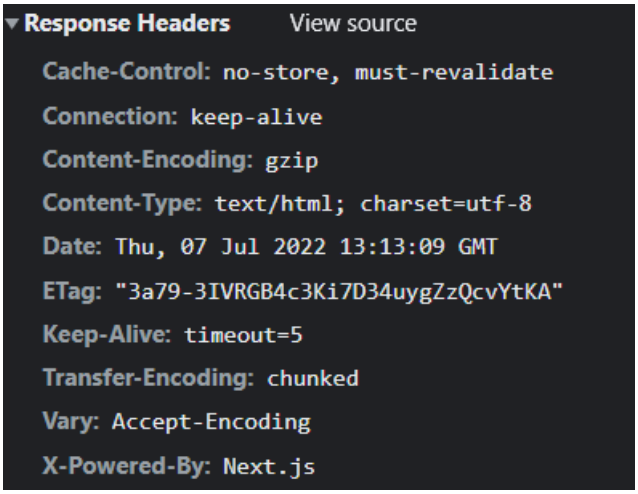
Salah satu proses *hardening* yang telah dilakukan adalah menambahkan *security http response header* tambahan. *Header* tersebut digunakan pada saat server memberikan respon pada *request* dengan metode http `GET` yang dibuat oleh pengguna (hamishwillee, et al., 2022).

Pengujian yang dilakukan adalah dengan memeriksa status *response header* tersebut setelah proses *hardening* dan melakukan perbandingan dengan hasil sebelum *hardening*. Proses yang dilakukan untuk memeriksa hal tersebut adalah dengan membuka *Developer Tools* pada peramban yang digunakan dan membuka tab *networks* untuk memantau aliran data pada *browser*. Langkah selanjutnya adalah melakukan *request* halaman web ke alamat IP Sekawan v2, dalam hal ini adalah `localhost:3000`.

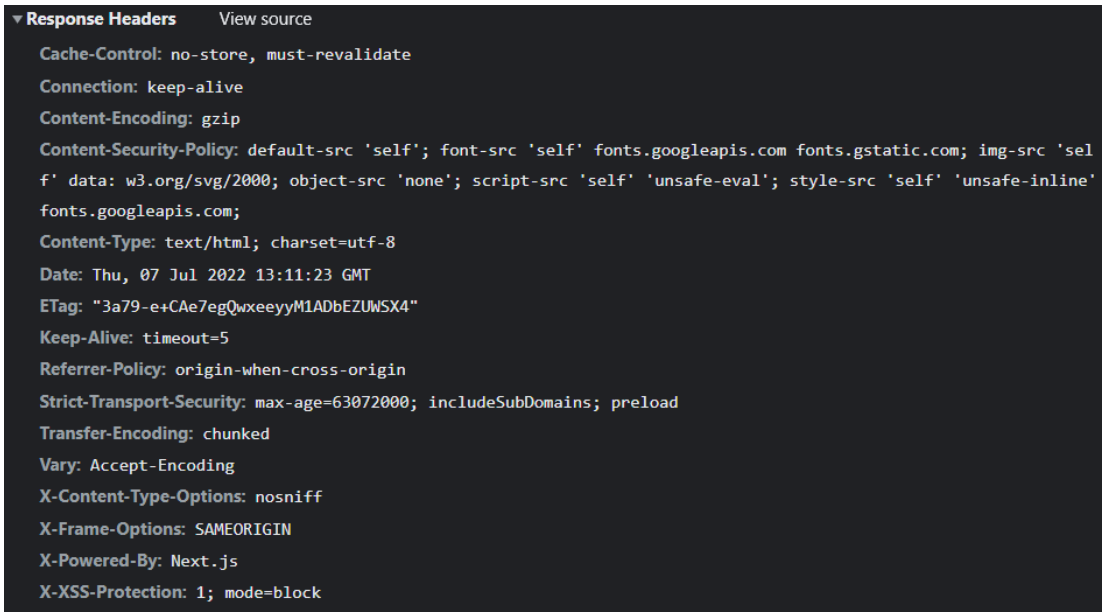
Name	Status	Type	Initiator	Size	Time
localhost	200	document	Other	3.0 kB	104 ms
data:image/svg+xml,...	200	svg+xml	(index)	(memory ...	0 ms
data:image/gif;base...	200	gif	(index)	(memory ...	0 ms
data:image/svg+xml,...	200	svg+xml	(index)	(memory ...	0 ms
css2?family=Rubik:wght...	200	stylesheet	(index)	(disk cac...	2 ms
webpack.js?ts=16572000...	200	script	(index)	9.2 kB	27 ms
react-refresh.js?ts=16572...	200	script	(index)	24.6 kB	36 ms
main.js?ts=1657200061296	200	script	(index)	1.1 MB	322 ms
_app.js?ts=1657200061296	200	script	(index)	328 kB	116 ms
index.js?ts=1657200061296	200	script	(index)	99.5 kB	63 ms
_buildManifest.js?ts=1657...	200	script	(index)	633 B	29 ms
_ssgManifest.js?ts=16572...	200	script	(index)	411 B	16 ms
_middlewareManifest.js?ts...	200	script	(index)	427 B	13 ms
hero-image-min.dca016c...	200	png	(index)	480 kB	9 ms
iJWKBXyIfDnIV7nBrXw.wo...	200	font	css2?family=Rubik...	(disk cac...	2 ms

Gambar 4.88 Berkas yang dikirim dalam respon dari *localhost*:3000

Pada Gambar 4.88 terdapat berkas bernama *localhost*, pilih *file* tersebut untuk memeriksa *response header* yang digunakan. Pada Gambar 4.89 dan Gambar 4.90 terlihat perbandingan antara sebelum dan sesudah dilakukannya proses pengerasan keamanan dengan *security http response header*. Gambar 4.89 menunjukkan bahwa sebelum proses *hardening* dilakukan hanya terdapat sepuluh response header yang digunakan, sedangkan Gambar 4.90 menunjukan kondisi *response header* setelah melalui proses tersebut berjumlah 16. Dengan begitu proses pengerasan keamanan telah berhasil dilakukan.



Gambar 4.89 *Response header* pada berkas *localhost* sebelum proses *hardening*

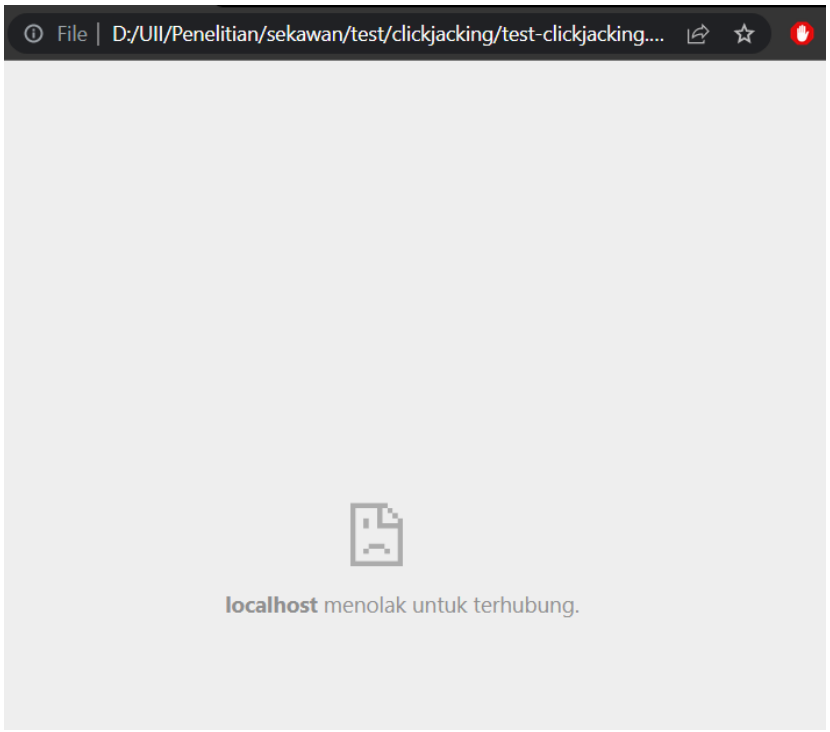


Gambar 4.90 *Response header* pada berkas *localhost* sesudah proses *hardening*

Clickjacking

Response header yang telah diimplementasikan dapat terlihat pada Gambar 4.90. Salah satu *http response header* yang ditambahkan dalam proses pengerasan keamanan adalah `X-Frame-Options` yang berfungsi untuk menentukan penggunaan *tag* HTML iframe dalam halaman web. *Header* yang ditambahkan memiliki nilai `SAMEORIGIN` yang mengindikasikan bahwa *tag* HTML iframe hanya dapat dimuat apabila berasal dari *origin* yang sama (Perrier, et al., X-Frame-Options - HTTP | MDN, 2022). Gambar 4.91 adalah halaman web penyerang yang akan memuat halaman web Sekawan v2 setelah melalui proses pengerasan keamanan. Terlihat bahwa halaman web yang disematkan tidak dapat dimuat oleh halaman lain. Dengan

begitu dapat dikatakan bahwa proses penyerangan gagal dan proses pengerasan keamanan berhasil dilakukan.



Gambar 4.91 Hasil pengujian *clickjacking* setelah proses *hardening*

4.6.3 Server

a. Identification and Authentication Failures

SSH *Bruteforce*

Pengujian *bruteforce* layanan ssh kedua dilakukan dilakukan secara manual, tidak seperti pengujian pertama dimana perangkat lunak *Hydra* digunakan untuk melakukan otomasi serangan. Hal tersebut dikarenakan untuk menghindari pemblokiran alamat IP oleh *firewall* yang terdapat pada jaringan UII. Sebelum melakukan pengujian, terlihat pada Gambar 4.93 layanan fail2ban diperiksa pada server Sekawan v2 untuk memastikan bahwa layanan tersebut dan *jail* yang diterapkan telah berjalan. Gambar 4.92 perintah untuk memeriksa layanan status fail2ban:

```
Sudo fail2ban-client status ssh
```

Gambar 4.92 Perintah terminal untuk memeriksa status fail2ban

```
penelitian-18523107@informatics-sekawan:~$ sudo fail2ban-client status ssh
Status for the jail: ssh
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
```

Gambar 4.93 Status fail2ban sebelum pengujian sesudah proses *hardening*

Pada Gambar 4.93 terlihat bahwa pada status *jail* ssh belum ada alamat IP yang diblokir oleh fail2ban dan jumlah total alamat IP yang telah diblokir. Fail2ban juga telah mengawasi berkas `/var/log/auth.log`, di mana *file* tersebut merupakan berkas log yang digunakan oleh Linux Ubuntu untuk mencatat autentikasi yang terjadi dalam server.

```
Lenovo LEGION@Legion MINGW64 ~
$ ssh informaticssekawan@103.220.113.178 -p 2421
informaticssekawan@103.220.113.178's password:
Permission denied, please try again.
informaticssekawan@103.220.113.178's password:
Permission denied, please try again.
informaticssekawan@103.220.113.178's password:
informaticssekawan@103.220.113.178: Permission denied (publickey,password).

Lenovo LEGION@Legion MINGW64 ~
$ ssh informaticssekawan@103.220.113.178 -p 2421
informaticssekawan@103.220.113.178's password:
Permission denied, please try again.
informaticssekawan@103.220.113.178's password:
Permission denied, please try again.
informaticssekawan@103.220.113.178's password:
informaticssekawan@103.220.113.178: Permission denied (publickey,password).

Lenovo LEGION@Legion MINGW64 ~
$ ssh informaticssekawan@103.220.113.178 -p 2421
informaticssekawan@103.220.113.178's password:
Permission denied, please try again.
informaticssekawan@103.220.113.178's password:
Connection reset by 103.220.113.178 port 2421
```

Gambar 4.94 Percobaan *bruteforce* manual setelah proses *hardening*

Terlihat pada Gambar 4.94, percobaan *login* dengan kredensial yang salah dilakukan sebanyak delapan kali. Pada percobaan ke delapan, koneksi yang dilakukan ke layanan ssh telah direset oleh server Sekawan V2 sehingga pengujian *bruteforce* manual tidak dapat dilakukan kembali. Hal tersebut mengindikasikan bahwa perangkat lunak fail2ban telah

berhasil melakukan pemblokiran alamat IP. Untuk memastikan bahwa IP benar-benar diblokir, maka dilakukan kembali percobaan login sebanyak satu kali. Dapat dilihat pada Gambar 4.95, layanan ssh memberikan respon "*Connection timed out*". Dengan begitu hasil dari proses implementasi *hardening* ssh dengan fail2ban berhasil dilakukan.

```
Lenovo LEGION@Legion MINGW64 ~
$ ssh penelitian-18523107@103.220.113.178 -p 2421
ssh: connect to host 103.220.113.178 port 2421: Connection timed out
```

Gambar 4.95 Percobaan *login* ke server sekawan V2

Untuk memeriksa kembali status *software* fail2ban setelah pengujian, diperlukan kembali akses ke dalam server Sekawan V2. Maka dari itu, proses login dilakukan dengan jaringan yang berbeda. Untuk memeriksa status dari fail2ban dapat menggunakan perintah yang terdapat pada Gambar 4.92. Hasil dari *jail* yang telah diterapkan di layanan ssh dapat dilihat pada Gambar 4.96 di mana terdapat satu alamat IP yang telah diblokir oleh sistem dan terdapat total satu alamat IP yang telah diblokir dengan rincian alamat IP tersebut. Dengan ip penyerang yang telah berhasil diblokir, maka proses hardening telah berhasil dilakukan.

```
penelitian-18523107@informatics-sekawan:~$ sudo fail2ban-client status ssh
Status for the jail: ssh
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   103.95.7.69
```

Gambar 4.96 Status fail2ban setelah pengujian sesudah proses *hardening*

4.7 Analisis Validasi Hasil Hardening

Berdasarkan proses identifikasi kerentanan telah ditemukan celah keamanan dan perbaikan keamanan pada Sistem Sekawan v2. Terdapat total empat jenis celah keamanan OWASP Top 10 yang telah ditemukan pada Sekawan-API, Sekawan-FE, dan server. Masing-masing komponen *microservice* Sekawan v2 mempunyai dua kerentanan dengan berbagai celah keamanan. Celah keamanan yang perlu diperbaiki dengan peringkat paling tinggi berada pada Sekawan-API dengan kategori *injection* dan celah keamanan dengan peringkat paling rendah terdapat pada server yakni dengan kategori *identification and authentication failures*. Dengan melihat Tabel 4.5, maka dapat dikatakan bahwa Sekawan-API adalah aplikasi dengan

kemungkinan tertinggi mendapatkan ancaman siber. Detail celah keamanan pada komponen-komponen Sekawan V2:

A. Sekawan-API
    2. *Injection*: Stored XSS.
    3. *Vulnerable and Outdated Components*: dependensi usang go.

B. Sekawan-FE
    A. *Security Misconfiguration*: *missing http response header* dan *clickjacking*.
    B. *Vulnerable and Outdated Components*: dependensi rentan npm.

C. Server
    a. *Security Misconfiguration*: *user management* dan *unused software*.
    b. *Identification and Authentication Failures*: ssh *bruteforce*.

Proses implementasi *hardening* dan Validasi Hasil Hardening telah berhasil dilakukan dan berhasil memitigasi celah keamanan pada seluruh kerentanan komponen Sekawan V2. Pada Aplikasi Sekawan-API telah melakukan implementasi fungsi sanitasi terhadap *input* ke basis data dan pembaruan versi dependensi go. Perbaikan keamanan Sekawan-FE dilakukan dengan menambahkan security http response header tambahan dan pembaruan dependensi pada kerangka kerja Nextjs. Di sisi server, *hardening* dilakukan dengan menambahkan kata sandi pada akun *root*, menghapus *software* yang tidak digunakan, dan pencegahan serangan *bruteforce* dengan melakukan pemblokiran alamat IP.

Dalam upaya untuk memitigasi risiko keamanan, peneliti telah berhasil melaksanakan implementasi hardening dan validasi hasilnya. Langkah-langkah yang diambil menunjukkan komitmen terhadap meningkatkan keamanan komponen Sekawan V2 dari berbagai sisi, baik aplikasi, antarmuka pengguna, maupun sisi server. Pendekatan ini secara signifikan dapat mengurangi potensi risiko keamanan, termasuk celah input yang dapat dimanfaatkan oleh serangan berbahaya. Oleh karena itu, upaya yang dijalankan dapat dianggap sebagai langkah yang sangat penting dalam menjaga integritas, kerahasiaan, dan ketersediaan data serta menjaga tingkat kepercayaan dari pengguna dan pemangku kepentingan.

BAB V
KESIMPULAN

5.1 Kesimpulan

Penelitian dengan topik Hardening Sistem Informasi Sekawan v2 Menggunakan Framework OWASP telah berhasil melakukan pengerasan terhadap keamanan sistem informasi tersebut. Penelitian ini bertujuan untuk mengamankan sistem Sekawan v2 dari *bugs* atau kesalahan dalam kode dan konfigurasi yang dapat mengakibatkan terjadinya celah keamanan serta serangan dari pihak luar. Proses penelitian dilakukan dengan menggunakan OWASP Top 10 2021 dan OWASP WSTG-Stable: During Development. Berdasarkan seluruh proses penelitian yang telah dilakukan, dapat ditarik kesimpulan:

1. Kondisi keamanan Sistem Sekawan V2 sebelum proses *hardening* tidak sepenuhnya aman. Hal tersebut dikuatkan dengan penemuan berbagai kerentanan baik di aplikasi *frontend, backend,* maupun server.
2. Terdapat empat kategori kerentanan dalam OWASP Top 10 2021 yang terdapat pada Sistem Sekawan V2, antara lain: *injection, security misconfiguration, vulnerable and outdated components,* dan *identification and authentication failures.*
3. Proses perbaikan keamanan Sekawan V2 dilakukan dengan cara pembaruan dependensi, pembuatan fungsi sanitasi, penambahan perangkat lunak, dan pemeriksaan konfigurasi/perangkat lunak.
4. Proses *hardening* berhasil meningkatkan keamanan Sekawan V2. Celah keamanan yang ditemukan pada proses *code review* dan pengujian pertama berhasil diperbaiki.
5. Penggunaan OWASP Top 10 2021 cocok digunakan untuk membantu melakukan pencarian celah-celah keamanan yang terdapat dalam sistem. Kerentanan yang ditemukan pada Sekawan v2 sesuai dengan yang terdapat pada subbagian deskripsi, pencegahan, dan contoh skenario serangan pada OWASP Top 10. Dengan pembaruan secara berkala, menjadikan konten dari publikasi tersebut akan selalu *up to date*.
6. Penggunaan OWASP The Web Security Testing Guide (WSTG)-Stable: During Development sesuai dengan proses yang dilakukan pada Sistem Informasi Sekawan v2 yakni pengembangan perangkat lunak. Proses *Code Walkthrough* berhasil mendapatkan informasi mengenai sistem dan cara kerjanya. Tahap *Code*

Review telah berhasil melakukan investigasi terhadap kode sumber untuk mencari kerentanan pada kode ataupun konfigurasi yang digunakan.

7. Penelitian berhasil melakukan peningkatan keamanan sesuai dengan *Triad* CIA (*Confidentiality*, *Integrity*, dan *Availability*).
8. Tidak semua kategori OWASP Top 10 ditemukan pada tiap-tiap komoponen Sekawan V2.

5.2 Saran

Berdasarkan penelitian yang telah dilaksanakan, terdapat beberapa saran yang dapat dilakukan untuk melakukan pengerasan keamanan lebih baik lagi:

a. Perlu melakukan proses *hardening* kembali dengan metode yang sama maupun lain yang serupa.
b. Melakukan pengujian menggunakan alat pengujian otomatis seperti OWASP ZAP ataupun Arachni untuk memaksimalkan temuan kerentanan.

DAFTAR PUSTAKA

Alemi, K. P. (2021). Input Validation and Input Sanitization for Web Applications. *Dissertation*. doi:http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-458680

Baars, N., M, R., Paris, Y., Khanna, V., Grossman, J., Ladeia, B., . . . Zanni, A. (2022, February 22). *Cross-Site Scripting (XSS) | OWASP Foundtaion*. Retrieved July 20, 2022, from OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation: https://owasp.org/www-community/attacks/xss/

Badan Pengembangan dan Pembinaan Bahasa, Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia. (n.d.). *Hasil Pencarian Perangkat Keras - KBBI Daring*. Retrieved July 14, 2022, from KBBI Daring: https://kbbi.kemdikbud.go.id/entri/perangkat%20keras

Badan Pengembangan dan Pembinaan Bahasa, Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia. (n.d.). *Hasil Pencarian Perangkat Lunak - KBBI Daring*. Retrieved July 17, 2022, from KBBI Daring: https://kbbi.kemdikbud.go.id/entri/perangkat%20lunak

Barker, E., Smid, M., & Branstad, D. (2015). *A Profile for U.S. Federal Cryptographic Key Management Systems.* Gaithersburg: National Institute of Standards and Technology. doi:http://dx.doi.org/10.6028/NIST.SP.800-152

Blankenship, H., & M, R. (2020, June 14). *Vulnerabilites | OWASP Foundation*. Retrieved from OWASP Github Official Repository: https://github.com/OWASP/www-community/blob/master/pages/vulnerabilities/index.md

Budiarto, R. (2017, May 15-17). Penerapan Metode FMEA Untuk Keamanan Sistem Informasi (Studi Kasus: Website POLRI). *2nd Seminar Nasional IPTEK Terapan (SENIT), 2*(1), 73-78. Retrieved from https://ejournal.poltektegal.ac.id/index.php/SENIT2017

Canonical Ltd. (2022, April 28). *Installation | Ubuntu*. Retrieved July 6, 2022, from Enterprise Open Source and Linux | Ubuntu: https://ubuntu.com/server/docs/installation

DBeaver. (2022, July 6). *About | Dbeaver Community*. Retrieved from DBeaver Community | Free Universal Database Tool: https://dbeaver.io/about/

Docile, E. (2018, November 29). *How to create systemd service unit in Linux*. Retrieved from Linux Tutorial - Learn Linux Configuration: https://linuxconfig.org/how-to-create-systemd-service-unit-in-linux

Fontanilla, M. V. (2020). Cybersecurity Pandemic. *Eubios Journal of Asian and International Bioethics*, 161-165. Retrieved from https://www.researchgate.net/journal/Eubios-journal-of-Asian-and-international-bioethics-EJAIB-1173-2571

g0tmi1k. (2022, September 9). *What is Kali Linux? | Kali Linux Documentation*. Retrieved September 13, 2022, from Kali Linux | Penetration Testing and Ethical Hacking Distribution: https://www.kali.org/docs/introduction/what-is-kali-linux/

Gite, V. (2022, April 18). *Understanding /etc/shadow file format on Linux | nixCraft*. Retrieved July 6, 2022, from nixCraft - Linux Tips, Hacks, Tutorials, And Ideas In Blog: https://www.cyberciti.biz/faq/understanding-etcshadow-file/

Google LLC. (2022, April 13). *Avoiding SQL injection risk*. Retrieved from The Go Programming Language: https://go.dev/doc/database/sql-injection

Google LLC. (2022, July 14). *Google Chrome - Download the Fast, Secure Browser from Google*. Retrieved from Google: https://www.google.com/intl/id_id/chrome/

Google LLC. (n.d.). *Go Modules Reference - The Go Programming Language*. Retrieved June 6, 2022, from https://go.dev/ref/mod#go-list-m

hamishwillee, bershanskiy, mfuji09, participator, discodanser, teoli, & fscholz. (2022, February 18). *Response Header - MDN Web Docs Glossary: Definitions of Web-related terms | MDN*. Retrieved July 7, 2022, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Glossary/Response_header

Kasper, J., Robinson, L., Neutkens, T., D, L. A., Steven, Dias, F. B., . . . Scott, T. (2022, February 18). *next.config.js: Custom Headers | Next.js*. Retrieved July 4, 2022, from Next.js by Vercel - The React Framework: https://github.com/vercel/next.js/blob/canary/docs/api-reference/next.config.js/headers.md

Kurniawan, E., & Riadi, I. (2018, February 23). Analisis Tingkat Keamanan Sistem Informasi Akademik Berdasarkan Standard ISO/IEC 27002:2013 Menggunakan SSE-CMM. *INTENSIF Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi, 2*(1), 12-23. doi:https://doi.org/10.29407/intensif.v2i1.11830

Lallie, H. S., Sheperd, L. A., Nurse, J. R., Erola, A., Epiphaniou, G., Maple, C., & Bellekens, X. (2021, June). Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. *Computers & Security, 105*(102248), 1-20. doi:https://doi.org/10.1016/j.cose.2021.102248

M, R., Alon, S., Mavrakis, C., Blankenship, H., & Buckley, N. (2022, February 8). *Clickjacking | OWASP Foundation*. Retrieved from OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation: https://github.com/OWASP/www-community/blob/master/pages/attacks/Clickjacking.md

Martin, M. (2022, July 23). *Difference between Website and Web Application (Web App)*. Retrieved August 5, 2022, from Guru99: https://www.guru99.com/difference-web-application-website.html

Microcosm. (2022, July 1). *Bluemonday Repository*. Retrieved from Microcosm Official Github Repository: https://github.com/microcosm-cc/bluemonday

Microsoft. (2022, July 4). *Documentation for Visual Studio Code*. Retrieved from Visual Studio: https://code.visualstudio.com/docs

Microsoft. (2022, July 5). *microsoft/vscode: Visual Studio Code*. Retrieved from Microsoft Github Official Repository: https://github.com/Microsoft/vscode/

Monteith, S., Bauer, M., Arda, M., Geddes, J., Whybrow, P. C., & Glenn, T. (2021). Increasing Cybercrime Since the Pandemic: Concerns for Psychiatry. *Current Psychiatry Reports, 23*(18), 1-7. doi:https://doi.org/10.1007/s11920-021-01228-w

Nur, R. M., Na'am, J., Nurcahyo, G. W., & Arlis, S. (2019). Peningkatan Keamanan Website Menggunakan Metode XML dengan Framework. *Indonesian Journal of Computer Science, 8*(2), 156-163. doi:https://doi.org/10.33022/ijcs.v8i2.188

Olenčin, M., & Perháč, J. (2021). Automated Hardening of a Linux Web Server. 2-7.

Pawlicka, A., Michał , Pawlicki, M., & Kozik , R. (2021). A $10 million question and other cybersecurity-related ethical dillemas amid the COVID-19 pandemic. *Business Horizons, 64*(6), 730-734. doi:https://doi.org/10.1016/j.bushor.2021.07.010

Perrier, J.-Y., & Sam, A. (2021, October 3). *Referrer-Policy - HTTP | MDN*. Retrieved July 13, 2022, from MDN Web Docs: https://github.com/mdn/content/blob/main/files/en-us/web/http/headers/referrer-policy/index.md?plain=1

Perrier, J.-Y., A1lo, wbamberg, queengooborg, Legner, S., Schonning, N., . . . Magyar, M. (2022, May 13). *X-Frame-Options - HTTP | MDN*. Retrieved July 6, 2022, from MDN Web Docs: https://github.com/mdn/content/blob/main/files/en-us/web/http/headers/x-frame-options/index.md

Perrier, J.-Y., Ruikar, O., queengooborg, & Pardal, D. (2022, May 10). *X-XSS-Protection - HTTP | MDN*. Retrieved from MDN Web Docs:

https://github.com/mdn/content/blob/main/files/en-us/web/http/headers/x-xss-protection/index.md?plain=1

Perrier, J.-Y., Ruikar, O., queengooborg, Willee, H., & Sam, A. (2022, June 11). *X-Content-Types-Options*. Retrieved July 13, 2022, from MDN Web Docs: https://github.com/mdn/content/blob/main/files/en-us/web/http/headers/x-content-type-options/index.md?plain=1

Perrier, J.-Y., Schonning, N., wbamberg, Ruikar, O., queengooborg, Gupta, G., . . . Bershanskiy, A. (2022, May 22). *Strict-Transport-Security - HTTP | MDN*. Retrieved July 14, 2022, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security

Perrier, J.-Y., Smith, M., & Baska, E. (2022, May 13). *HTTP request Method - HTTP | MDN*. Retrieved July 14, 2022, from MDN Web Docs: https://github.com/mdn/content/blob/main/files/en-us/web/http/methods/index.md?plain=1

Perrier, J.-Y., yoshinorin, wbamberg, queengooborg, Smith, M., Willee, H., . . . Medeiros, L. (2022, July 14). *Content-Security-Policy - HTTP | MDN*. Retrieved July 2022, 2022, from MDN Web Docs: https://github.com/mdn/content/blob/main/files/en-us/web/http/csp/index.md?plain=1

Postman, Inc. (2022, July 3). *Postman API Platform*. Retrieved from Postman: https://www.postman.com/product/what-is-postman/

Prabowo, M. A., Darusalam, U., & Ningsih, S. (2020, July). Perancangan Keamanan Server Linux Dengan Metode Hardening Pada Layer 1 dan Layer 7. *JURNAL MEDIA INFORMATIKA BUDIDARMA, 4*(3), 591-603. doi:http://dx.doi.org/10.30865/mib.v4i3.2157

Program Studi Informatika Program Sarjana. (2021, 12 25). *Homepage: SEKAWAN UII*. Retrieved from SEKAWAN Informatika UII: https://informatics.uii.ac.id/sekawan/

Rahardjo, B. (2005). *Keamanan Sistem Informasi.* Bandung.

Rahayu, S. (2016, February 16). *MACAM – MACAM PERANGKAT KERAS PADA KOMPUTER (HARDWARE) – sejarah komputer*. Retrieved July 14, 2022, from #blogUNNES: https://blog.unnes.ac.id/srirahayu/2016/02/19/macam-macam-perangkat-keras-pada-komputer-hardware/

Sanjaya, I. A., Sasmita, G. M., & Arsa, D. M. (2020, July 18). Evaluasi Keamanan Website Lembaga X Melalui Penetration Testing Menggunakan Framework ISSAF. *Jurnal*

Ilmiah Merpati (Menara Penelitian Akademika Teknologi Informasi), 113-124. doi:https://doi.org/10.24843/JIM.2020.v08.i02.p05

Shinde, P. S., & Ardhapurkar, S. B. (2016). Cyber Security Analysis using Vulnerability Assessment and Penetration Testing . *World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)* (pp. 1-5). Coimbatore: IEEE. doi:10.1109/STARTUP.2016.7583912

Steven, Robbins, S., Robinson, L., Mendez, J., Patel, A., Hols, M., . . . Mittal, A. (2022, February 2). *Advanced Features: Security Headers*. (Vercel) Retrieved June 10, 2022, from https://github.com/vercel/next.js/blob/canary/docs/advanced-features/security-headers.md

Susanti, M. (2016, April). PERANCANGAN SISTEM INFORMASI AKADEMIK BERBASIS WEB PADA SMK PASAR MINGGU JAKARTA. *Jurnal Informatika, 3*(1), 91-99. doi:https://doi.org/10.31294/ji.v3i1

TechTarget Contributor. (2019, August). *What is Web Application (Web Apps) and its Benefits*. Retrieved August 5, 2022, from TechTarget: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

the International Electrotechnical Commission, & the International Organization for Standardization. (2015, May). ISO/IEC 2382:2015 Information technology — Vocabulary. 4. Retrieved July 14, 2022, from https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en

Thomson, E. (2020, September 23). *Documentation about audit reports*. Retrieved from NPM Github Official Repository: https://github.com/npm/documentation/blob/main/content/packages-and-modules/securing-your-code/about-audit-reports.mdx

Torsten, McCamon, M., Blankenship, H., van der Stock, A., Sulzbach, B., Falk, A., & Morla, A. (2021, October 1). *OWASP Top Ten*. Retrieved from OWASP Official Github Repository: https://github.com/OWASP/www-project-top-ten/blob/master/index.md

van der Stock, A., Glas, B., Torsten, Smithline, N., Schmitt, S., Pezzè, A., & Humblot, N. (2021, November 16). *OWASP Top 10:2021*. Retrieved 2021, from OWASP Github Official Repository : https://github.com/OWASP/Top10/blob/master/2021/docs/index.md

wstgbot. (2020, December 3). *The Web Security Testing Framework*. Retrieved from OWASP Official Github Repository: https://github.com/OWASP/www-project-web-security-testing-guide/blob/master/stable/3-The_OWASP_Testing_Framework/0-The_Web_Security_Testing_Framework.md

wstgbot. (2020, April 22). *WSTG - v4.1 Testing for Clickjacking | OWASP Foundation*. Retrieved from OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation: https://github.com/OWASP/www-project-web-security-testing-guide/blob/master/v41/4-Web_Application_Security_Testing/11-Client_Side_Testing/09-Testing_for_Clickjacking.md

wstgbot, M, R., & ThunderSon. (2022, March 8). *WSTG - Latest | Test HTTP Methods*. Retrieved November 27, 2022, from OWASP Foundation: https://github.com/OWASP/www-project-web-security-testing-guide/blob/master/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/06-Test_HTTP_Methods.md

Yarikoptic, Mikes, Lostcontrol, Henryut, & Jensck. (2013, May 25). *MANUAL 08 - Fail2ban*. Retrieved September 21, 2022, from Fail2ban: https://www.fail2ban.org/wiki/index.php/MANUAL_0_8

LAMPIRAN

a. Sekawan v2 *Security Checklist*

SEKAWAN V2 *SECURITY CHECKLIST*

Broken Access Control

a. Deskripsi

Kontrol akses memberlakukan kebijakan yang mengatur bahwa pengguna tidak dapat bertindak di luar izin yang telah diterapkan. Kegagalan kontrol akses dapat mengakibatkan kebocoran data, modifikasi atau penghapusan data, dan melakukan tindakan di luar batas pengguna. Kegagalan kontrol akses yang umum meliputi:

A. Pelanggaran prinsip *least privilege* atau hak akses minimum, di mana akses pengguna seharusnya diberikan hanya berdasarkan kemampuan, *role*, atau pengguna tertentu namun tersedia untuk semua.
B. Melewati pemeriksaan kontrol akses dengan memodifikasi URL (pengubahan parameter atau penjelajahan paksa), status aplikasi internal, atau halaman HTML, atau dengan menggunakan alat penyerang yang memodifikasi permintaan API.
C. Mengizinkan melihat atau mengedit akun orang lain, dengan memberikan pengenal uniknya (referensi objek langsung yang tidak aman)
D. Akses API tanpa akses kontrol seperti POST, GET, DELETE.
E. *Elevation of privilege* atau meningkatkan hak akses. Bertindak sebagai pengguna tanpa *login* atau bertindak sebagai admin saat *login* sebagai pengguna.
F. Manipulasi metadata, seperti memutar ulang atau merusak token kontrol akses JSON Web Token (JWT), atau *cookie* atau bidang tersembunyi yang dimanipulasi untuk meningkatkan hak istimewa atau menyalahgunakan pembatalan JWT.
G. Kesalahan konfigurasi CORS memungkinkan akses API dari sumber yang tidak sah/tidak dipercaya.
H. Memaksa untuk melakukan navigasi ke halaman yang diautentikasi sebagai pengguna yang belum diautentikasi atau ke halaman yang khusus sebagai pengguna standar.

b. Pencegahan

a. Terapkan mekanisme kontrol akses sekali dan gunakan kembali di seluruh aplikasi, termasuk meminimalkan penggunaan Cross-Origin Resource Sharing (CORS).
b. Kontrol akses model harus menerapkan kepemilikan *record* daripada menerima bahwa pengguna dapat *create*, *read*, *update*, atau *delete record* apa pun.

c. Nonaktifkan daftar direktori server web dan pastikan metadata file (mis., .git) dan file cadangan tidak ada dalam *root* web.
d. Catat kegagalan kontrol akses, informasikan kepada admin jika perlu (contoh: kegagalan berulang).
e. Menerapkan *rate limit* pada API dan akses pengontrol untuk meminimalkan bahaya dari alat serangan otomatis.
f. Pengidentifikasi *stateful session* harus invalidasi di server setelah *logout*. Token JWT *stateless* sebaiknya berumur pendek sehingga jendela peluang bagi penyerang diminimalkan. Untuk JWT yang lebih tahan lama, sangat disarankan untuk mengikuti standar OAuth untuk mencabut akses.
g. Contoh Skenario Serangan
    1. Aplikasi menggunakan data dalam pemanggilan SQL yang belum diverifikasi untuk mengakses informasi akun:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Penyerang cukup memodifikasi parameter acct pada peramban untuk mengirim nomor akun yang diinginkan. Jika tidak dilakukan verifikasi dengan benar, penyerang dapat mengakses akun manapun.

```
https://example.com/app/accountInfo?acct=notmyacct
```

    2. Penyerang cukup melakukan navigasi ke URL target. Walaupun hak admin diperlukan untuk akses ke halaman admin.

```
https://example.com/app/getappInfo
https://example.com/app/admin_getappInfo
```

Jika pengguna non-admin dapat mengakses halaman admin, hal tersebut adalah celah keamanan. Apabila pengguna yang tidak diautentikasi dapat mengakses halaman pengguna normal dan admin, hal tersebut juga merupakan celah keamanan.

Cryptographic Failures

1. Deskripsi

Hal pertama yang harus dilakukan adalah menentukan kebutuhan proteksi data *in transit* dan *at rest.* Seperti *password*, nomor kartu kredit, riwayat kesehatan, informasi pribadi, dan rahasia bisnis membutuhkan perlindungan ekstra terutama jika data-data tersebut di bawah perlindungan undang-undang privasi seperti: EU *General*

Data Protection Regulation (GDPR), data keuangan PCI-DSS. Data akan menjadi rentan jika pada data-data tersebut:

1. Pengiriman data secara *clear text* (tanpa enkripsi).
2. Menggunakan algoritma lama atau lemah
3. Menggunakan kunci kripto *default*, lemah, atau menggunakan kembali kunci lama
4. Tidak mengutamakan penggunaan protokol keamanan (contoh: https).
5. Menggunakan algoritma *hash* yang telah usang (MD5 dan SHA1)
6. Kata sandi digunakan sebagai kunci kriptografi tanpa adanya fungsi derivasi kunci berbasis kata sandi.
7. Pencegahan

a. Mengklasifikasikan data yang diproses, disimpan, atau dikirimkan oleh suatu aplikasi. Identifikasi data mana yang sensitif menurut undang-undang privasi, persyaratan peraturan, atau kebutuhan bisnis.
b. Jangan menyimpan data sensitif yang tidak perlu.
c. Melakukan enkripsi data *at rest* (data yang disimpan).
d. Menggunakan protokol keamanan yang *up-to-date* kuat.
e. Mengenkripsi data yang sedang ditransfer (data *in transit*) dengan protokol seperti SSL/TLS.
f. Nonaktifkan fungsi *caching* untuk data sensitif.
g. Gunakan akses kontrol keamanan sesuai dengan klasifikasi data.
h. Simpan *password* dengan algoritma dan fungsi *hash* dan *salt*.

8. Contoh Skenario Serangan

a. Aplikasi mengenkripsi nomor kartu kredit dalam *database* menggunakan enkripsi database otomatis. Namun, data ini secara otomatis didekripsi ketika diambil, memungkinkan kerentanan SQL *injection* untuk mengambil nomor kartu kredit dalam *cleartext*.
b. Situs tidak menggunakan atau menerapkan TLS untuk semua halaman atau mendukung enkripsi yang lemah. Penyerang memantau lalu lintas jaringan (misalnya, di jaringan *wireless* yang tidak aman), menurunkan versi protokol koneksi dari HTTPS ke HTTP, memotong permintaan, dan mencuri *cookie* sesi pengguna. Penyerang kemudian menggunakan ulang *cookie* ini dan membajak *session* pengguna (yang telah diautentikasi), mengakses atau memodifikasi data pribadi pengguna.

Alih-alih di atas, mereka dapat mengubah semua data yang diangkut, misalnya, penerima transfer uang.

c. *Database* kata sandi menggunakan *hash* tanpa fungsi *salt* atau sederhana untuk proses penyimpanan. Kerentanan fitur unggah *file* memungkinkan penyerang untuk mengambil basis data kata sandi. Semua *hash* tanpa tambahan fungsi *salt* dapat diekspos dengan *rainbow tabel* dari *hash* yang telah dihitung. *Hash* yang dihasilkan oleh fungsi *hash* sederhana dapat dipecahkan oleh GPU, meskipun telah ditambahkan fungsi *salt*.

Injection

1. Deskripsi

Aplikasi web akan menjadi rentan terhadap serangan injeksi apabila:

1. Data yang diberikan oleh pengguna tidak melalui proses validasi, filtrasi, atau sanitasi oleh aplikasi.
2. *Query* dinamis atau pemanggilan nonparameter tanpa konteks digunakan langsung dalam interpreter.
3. Data berbahaya digunakan dalam parameter pencarian *mapping* relasional objek (ORM) untuk mengekstrak catatan sensitif tambahan.
4. Data berbahaya langsung digunakan atau digabungkan. SQL atau perintah berisi struktur dan data berbahaya dalam *dynamic query*, perintah, atau prosedur tersimpan.

Beberapa serangan injeksi yang lebih umum adalah injeksi SQL, NoSQL, OS *command*, *Object Relational Mapping* (ORM), LDAP, dan Expression Language (EL) atau Object Graph Navigation Library (OGNL). Konsepnya identik di antara semua intepreter. Tinjauan *source code* adalah metode terbaik untuk mendeteksi jika aplikasi rentan terhadap injeksi. Pengujian otomatis dari semua parameter, *header*, URL, *cookie*, JSON, SOAP, dan input data XML sangat dianjurkan. Organisasi dapat menyertakan alat pengujian keamanan aplikasi statis (SAST), dinamis (DAST), dan interaktif (IAST) ke dalam pipa CI/CD untuk mengidentifikasi kelemahan injeksi yang diperkenalkan sebelum *deployment* produksi.

5. Pencegahan

a. Opsi yang lebih baik adalah menggunakan API yang aman, yang menghindari penggunaan *intepreter* sepenuhnya, menyediakan *interface* berparameter, atau bermigrasi ke *Object Relational Mapping Tools* (ORM).

Catatan: Bahkan ketika menggunakan parameter, prosedur tersimpan masih dapat terjadi SQL *Injection* jika fungsi SQL menggabungkan kueri dan data atau mengeksekusi data yang berbahaya dengan EXECUTE IMMEDIATE atau exec().

b. Untuk setiap kueri dinamis, keluarkan karakter khusus menggunakan *escape syntax* khusus untuk interpreter tersebut.
c. Gunakan LIMIT dan kontrol SQL lainnya dalam kueri untuk mencegah kebocoran massal *record* jika terjadi injeksi SQL.

6. Contoh Skenario Serangan

1. Aplikasi menggunakan data yang tidak tepercaya dalam pembuatan SQL *call* rentan berikut ini:

```
String query = "SELECT \* FROM accounts WHERE
custID='" + request.getParameter("id") + "'";
```

2. Demikian pula, *blind trust* aplikasi dalam kerangka kerja dapat mengakibatkan kueri yang masih rentan, (mis., *Hibernate Query Language* (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts
WHERE custID='" + request.getParameter("id") + "'");
```

Dalam kedua kasus, penyerang memodifikasi nilai parameter 'id' di *browser* mereka untuk mengirim: ' atau '1'='1. Sebagai contoh:

```
http://example.com/app/accountView?id=' or '1'='1
```

Insecure Design

1. Deskripsi

Desain tidak aman adalah kategori luas yang mewakili kelemahan yang berbeda, dinyatakan sebagai "ketiadaan atau tidak efektifnya desain kontrol". Terdapat perbedaaan antara desain dan implementasi yang tidak aman. Desain yang aman masih dapat memiliki cacat implementasi yang mengarah ke kerentanan yang dapat dieksploitasi. Desain yang tidak aman tidak dapat diperbaiki dengan implementasi yang sempurna karena menurut definisi, kontrol keamanan yang diperlukan tidak pernah dibuat untuk bertahan dari serangan tertentu. Salah satu faktor yang berkontribusi pada desain yang tidak aman adalah kurangnya profil risiko bisnis yang melekat pada perangkat lunak atau sistem yang sedang dikembangkan, dan dengan demikian kegagalan untuk menentukan tingkat desain keamanan apa yang diperlukan.

2. Pencegahan

a. Terapkan dan gunakan siklus pengembangan yang aman dengan profesional AppSec untuk membantu mengevaluasi dan merancang kontrol terkait keamanan dan privasi.
b. Gunakan *thread modeling* untuk autentikasi kritis, kontrol akses, logika bisnis, dan *key flow*.
c. Integrasikan pemeriksaan masuk akal di setiap tingkat aplikasi Anda (dari *frontend* ke *backend*).
d. Tulis pengujian unit dan integrasi untuk memvalidasi bahwa semua *critical flow* tahan terhadap model ancaman. Kompilasi kasus penggunaan dan kasus penyalahgunaan untuk setiap tingkat aplikasi.
e. Pisahkan lapisan tingkat pada lapisan sistem dan jaringan tergantung pada kebutuhan eksposur dan perlindungan.
f. Batasi konsumsi sumber daya oleh pengguna atau layanan

3. Contoh Skenario Serangan

a. Alur kerja pemulihan kredensial mungkin menyertakan "pertanyaan dan jawaban", yang dilarang oleh NIST 800-63b, OWASP ASVS, dan OWASP Top 10. Pertanyaan dan jawaban tidak dapat dipercaya sebagai bukti identitas karena lebih dari satu orang dapat mengetahui jawabannya, itulah sebabnya mereka dilarang. Kode tersebut harus dihapus dan diganti dengan desain yang lebih aman.
b. Jaringan bioskop memungkinkan diskon pemesanan grup dan memiliki maksimal lima belas peserta sebelum meminta deposit. Penyerang dapat melakukan eksploitasi model aliran ini dan menguji apakah mereka dapat memesan enam ratus kursi dan semua bioskop sekaligus dalam beberapa permintaan, menyebabkan hilangnya pendapatan secara besar-besaran.
c. Situs web *e-commerce* ritel tidak memiliki perlindungan terhadap bot yang dijalankan oleh calon yang membeli kartu video kelas atas untuk menjual kembali situs lelang. Ini menciptakan publisitas yang buruk bagi para pembuat kartu video dan pemilik rantai ritel, dan bertahan lama dengan para penggemar yang tidak dapat memperoleh kartu-kartu ini dengan harga berapa pun. Desain anti-bot yang cermat dan aturan logika domain, seperti pembelian yang dilakukan dalam beberapa detik setelah ketersediaan, dapat mengidentifikasi pembelian yang tidak autentik dan menolak transaksi tersebut.

Security Misconfiguration

1. Deskripsi

Aplikasi web dapat menjadi rentan terhadap serangan apabila:

1. Terdapat fitur terpasang atau aktif yang tidak digunakan (*port* terbuka, *services*, akun, atau hak akses).
2. Akun dan *password* belum diubah dan aktif (*default credential*).
3. Penanganan *error* membocorkan jejak atau informasi sensitif kepada pengguna.
4. Pada sistem yang telah di-*upgrade*, fitur keamanan terbaru belum dilakukan konfigurasi secara aman.
5. Pengaturan keamanan pada aplikasi server, kerangka kerja aplikasi, *libraries*, basis data, dll. tidak menggunakan nilai yang aman.
6. Server tidak mengirim *header* keamanan atau *directives*, atau hal tersebut belum menggunakan *value* yang aman.
7. Perangkat lunak telah usang atau rentan.

Tanpa proses konfigurasi keamanan aplikasi yang berulang dan terpadu, sistem berada pada risiko yang lebih tinggi.

8. Pencegahan

1. Proses *hardening* secara berulang hal tersebut cepat dan mudah untuk menerapkan lingkungan lain. Lingkungan pengembangan, QA, dan produksi semuanya harus dikonfigurasi secara identik, dengan kredensial berbeda yang digunakan di setiap lingkungan. Proses ini harus diotomatisasi untuk meminimalkan upaya yang diperlukan untuk menyiapkan lingkungan baru yang aman.
2. Platform minimal tanpa fitur, komponen, dokumentasi, dan sampel yang tidak perlu. Hapus atau jangan instal fitur dan kerangka kerja yang tidak digunakan.
3. Lakukan tinjauan dan memperbarui konfigurasi yang sesuai untuk semua catatan keamanan, pembaruan, dan *patch* sebagai bagian dari proses manajemen *patch*.
4. Arsitektur aplikasi tersegmentasi memberikan pemisahan yang efektif dan aman antara komponen atau penyewa, dengan segmentasi, kontainerisasi, atau grup keamanan cloud (ACL).
5. Mengirim *directives* keamanan ke klien, (contoh: *header* keamanan).
6. Proses otomatis untuk memverifikasi efektivitas konfigurasi dan pengaturan di semua lingkungan.

7. Contoh Skenario Serangan

1. *Server* dilengkapi dengan aplikasi sampel yang tidak dihapus dari *server* produksi. Contoh aplikasi tersebut telah mempunyai kelemahan keamanan yang telah

diketahui dan digunakan penyerang untuk menyusup ke server. Misalkan salah satu aplikasi adalah konsol admin, dan akun *default* yang tidak diubah. Dalam hal ini, penyerang masuk dengan kata sandi *default* dan mengambil alih.

2. *Directory listing* tidak dinonaktifkan di *server*. Seorang penyerang menemukan bahwa mereka dapat dengan mudah melihat daftar direktori. Penyerang menemukan dan mengunduh kelas Java yang dikompilasi, yang mereka dekompilasi dan merekayasa balik untuk melihat kodenya. Penyerang kemudian menemukan kelemahan kontrol akses yang parah dalam aplikasi.
3. Penyedia layanan cloud (CSP) memiliki izin berbagi *default* yang terbuka ke Internet oleh pengguna CSP lainnya. Ini memungkinkan data sensitif yang disimpan dalam penyimpanan cloud untuk diakses.

Vulnerable and Outdated Components

a. Deskripsi

Aplikasi web dapat menjadi rentan terhadap serangan apabila:

1. Jika pengembang tidak mengetahui versi semua komponen yang Anda gunakan (baik sisi klien maupun sisi server). Termasuk komponen yang digunakan secara langsung serta *nested dependency*.
2. Jika perangkat lunak rentan, tidak didukung, atau telah usang. Hal ini termasuk OS, server web/aplikasi, sistem manajemen basis data (DBMS), aplikasi, API dan semua komponen, lingkungan *runtime*, dan *library*.
3. Jika tidak melakukan perbaikan atau pembaruan platform, kerangka kerja, dan dependensi yang mendasarinya secara tepat waktu dan berbasis risiko. Ini biasanya terjadi di lingkungan ketika *patching* adalah tugas bulanan atau triwulanan di bawah kendali perubahan, membuat organisasi menjadi terancam kerentanan tetap yang tidak perlu.
4. Jika pengembang perangkat lunak tidak menguji kompatibilitas *library* yang diperbarui, ditingkatkan, atau ditambal.
5. Jika tidak melakukan pengamanan konfigurasi komponen.
6. Pencegahan

1. Hapus dependensi yang tidak digunakan, fitur yang tidak perlu, komponen, file, dan dokumentasi.

2. Hanya gunakan komponen dari sumber resmi melalui tautan aman. Utamakan paket yang telah ditandatangani secara digital untuk mengurangi kemungkinan menyertakan komponen berbahaya yang dimodifikasi.
3. Pantau *libraries* dan komponen yang tidak dirawat atau tidak membuat *patch* keamanan untuk versi yang lebih lama. Jika *patch* tidak memungkinkan, pertimbangkan untuk menggunakan *patch* virtual untuk memantau, mendeteksi, atau melindungi dari masalah yang ditemukan.
4. Contoh Skenario Serangan

1. Komponen biasanya berjalan dengan hak akses yang sama dengan aplikasi, sehingga kekurangan pada komponen apa pun dapat mengakibatkan dampak yang serius. Cacat tersebut dapat terjadi secara tidak sengaja (misalnya, kesalahan pengkodean) atau disengaja (misalnya, *backdoor* dalam suatu komponen). Beberapa contoh kerentanan komponen yang dapat dieksploitasi yang ditemukan adalah: CVE-2017-5638, kerentanan eksekusi kode jarak jauh Struts 2 yang memungkinkan eksekusi kode berbahaya di server, telah ditemukan bersalah atas pelanggaran yang signifikan.
2. Perangkat internet of things (IoT) seringkali sulit atau tidak mungkin untuk ditambal, pentingnya menambal kerentanan dapat memberikan dampak positif (mis., Perangkat biomedis).

Identification and Authentication Failures

a. Deskripsi

Konfirmasi identitas pengguna, otentikasi, dan manajemen sesi sangat penting untuk melindungi dari serangan terkait otentikasi. Autentikasi aplikasi web dapat menjadi rentan terhadap serangan apabila:

1. Mengizinkan serangan otomatis seperti mengisi kredensial pengguna pada saat *login*, di mana penyerang memiliki daftar nama pengguna dan kata sandi yang valid.
2. Aplikasi mengizinkan serangan *bruteforce* atau serangan otomatis lainnya.
3. Mengizinkan sandi *default*, lemah, atau terkenal, seperti "Password1" atau "admin/admin".
4. Menggunakan pemulihan kredensial yang lemah atau tidak efektif dan proses lupa sandi, seperti "jawaban berbasis pengetahuan", yang tidak dapat dibuat aman.

5. Menggunakan penyimpanan data kata sandi dengan *plaintext*, terenkripsi secara buruk, atau algoritma *hash* yang lemah.
6. Memiliki otentikasi multifaktor yang hilang atau tidak efektif.
7. Mengekspos *session identifier* di URL.
8. Gunakan kembali *session identifier* setelah proses login berhasil.
9. Tidak melakukan invalidasi ID sesi dengan benar. Sesi pengguna atau token autentikasi (terutama token *single sign-on* (SSO)) tidak divalidasi dengan benar selama *logout* atau periode tidak aktif.
10. Pencegahan

1. Jika memungkinkan, terapkan autentikasi multifaktor untuk mencegah isian kredensial otomatis, *bruteforce*, dan serangan penggunaan kembali kredensial yang telah dicuri.
2. Hindari mengirim atau menyebarkan dengan kredensial *default* apa pun, terutama untuk pengguna admin.
3. Terapkan pemeriksaan kata sandi yang lemah, seperti menguji kata sandi baru atau yang diubah terhadap 10.000 daftar kata sandi terburuk.
4. Batasi atau semakin tunda upaya *login* yang gagal, tetapi berhati-hatilah untuk tidak membuat skenario *denial of service*. Catat semua kegagalan dan beri tahu administrator ketika isian kredensial, *bruteforce*, atau serangan lainnya terdeteksi.
5. Gunakan pengelola sesi built-in sisi server, aman, yang menghasilkan ID sesi acak baru dengan entropi tinggi setelah login. Pengidentifikasi sesi tidak boleh ada di URL, disimpan dengan aman, dan tidak valid setelah logout, idle, dan waktu tunggu absolut.
6. Gunakan pengelola *session built-in* pada sisi server yang aman dan menghasilkan ID sesi acak baru dengan entropi tinggi setelah *login*. *Session identifier* tidak boleh ada di URL, disimpan dengan aman, dan melakukan invalidasi *session* setelah *logout*, *idle*, dan waktu tunggu absolut.
7. Contoh Skenario Serangan

1. Pengisian kredensial, penggunaan daftar kata sandi yang diketahui, adalah serangan umum. Misalkan aplikasi tidak menerapkan perlindungan ancaman atau isian kredensial otomatis. Dalam hal ini, aplikasi dapat digunakan sebagai sumber kata sandi untuk menentukan apakah kredensial itu valid.

2. Sebagian besar serangan otentikasi terjadi karena penggunaan kata sandi yang yang sama terus-menerus. Setelah dianggap sebagai praktik terbaik, rotasi kata sandi dan persyaratan kompleksitas mendorong pengguna untuk menggunakan dan menggunakan kembali kata sandi yang lemah. Organisasi disarankan untuk menghentikan praktik ini per NIST 800-63 dan menggunakan otentikasi multi-faktor.
3. Batas waktu sesi aplikasi tidak disetel dengan benar. Seorang pengguna menggunakan komputer publik untuk mengakses aplikasi. Alih-alih memilih "*logout*", pengguna cukup menutup tab *browser* dan pergi. Penyerang menggunakan *browser* yang sama satu jam kemudian, dan pengguna masih diautentikasi.

Software and Data Integrity Failures

a. Deskripsi

Kegagalan integritas perangkat lunak dan data berhubungan dengan kode dan infrastruktur yang tidak melindungi dari pelanggaran integritas. Contohnya adalah saat aplikasi bergantung pada *Content Delivery Network*, *libraries*, atau modul dari sumber, repositori, dan jaringan pengiriman konten (CDN) yang tidak tepercaya. Pipa CI/CD yang tidak aman dapat menimbulkan potensi akses tidak sah, kode berbahaya, atau akses ilegal sistem. Banyak aplikasi sekarang menyertakan fungsionalitas pembaruan otomatis, di mana pembaruan diunduh tanpa verifikasi integritas yang memadai dan diterapkan ke aplikasi tepercaya sebelumnya. Penyerang berpotensi mengunggah pembaruan mereka sendiri untuk didistribusikan dan dijalankan di semua instalasi. Contoh lain adalah di mana objek atau data dikodekan atau diserialkan ke dalam struktur yang dapat dilihat dan dimodifikasi oleh penyerang dan rentan terhadap deserialisasi yang tidak aman.

4. Pencegahan

1. Gunakan tanda tangan digital atau mekanisme serupa untuk melakukan verifikasi perangkat lunak atau data dari sumber yang dipercaya dan belum diubah.
2. Pastikan pustaka dan dependensi, seperti npm atau Maven, menggunakan repositori tepercaya.
3. Pastikan bahwa terdapat proses peninjauan perubahan kode dan konfigurasi untuk meminimalkan kemungkinan kode atau konfigurasi berbahaya yang dapat dimasukkan ke dalam *pipeline* perangkat lunak.

4. Pastikan bahwa *pipeline* CI/CD perangkat lunak memiliki pemisahan, konfigurasi, dan kontrol akses yang tepat untuk memastikan integritas kode yang masuk melalui proses *build* dan *deploy*.
5. Pastikan bahwa data serial yang tidak ditandatangani atau tidak dienkripsi tidak dikirim ke klien yang tidak dipercaya tanpa beberapa bentuk pemeriksaan integritas atau tanda tangan digital untuk mendeteksi gangguan atau menggunakan ulang data.
6. Contoh Skenario Serangan

1. Banyak *router* rumah, dekoder, *firmware* perangkat, dan lainnya tidak memverifikasi pembaruan melalui *firmware* yang telah ditandatangani. *Firmware* yang tidak ditandatangani adalah target yang sedang populer bagi penyerang dan diperkirakan akan semakin buruk. Hal ini seharusnya menjadi perhatian utama karena sering kali tidak ada mekanisme untuk memulihkan selain memperbaiki di versi mendatang dan menunggu versi sebelumnya usang.
2. Pada banyak negara telah terjadi serangan pada mekanisme pembaruan, dengan serangan penting baru-baru ini adalah serangan SolarWinds Orion. Perusahaan yang mengembangkan perangkat lunak memiliki proses integritas *build* dan *update* yang aman. Namun, ini dapat ditumbangkan, dan selama beberapa bulan, perusahaan mendistribusikan pembaruan berbahaya yang sangat ditargetkan ke lebih dari 18.000 organisasi, di mana sekitar 100 atau lebih terpengaruh. Ini adalah salah satu pelanggaran paling luas dan paling signifikan dari sifat ini dalam sejarah.
3. Aplikasi React memanggil satu set *microservices* Spring Boot. Sebagai pemrogram fungsional, mereka mencoba memastikan bahwa kode mereka tidak dapat diubah. Solusi yang mereka temukan adalah membuat serialisasi status pengguna dan meneruskannya bolak-balik dengan setiap permintaan. Penyerang memperhatikan tanda tangan objek Java "rO0" (di base64) dan menggunakan alat *Java Serial Killer* untuk mendapatkan eksekusi kode jarak jauh di server aplikasi.

Security Logging and Monitoring Failures

a. Deskripsi

Kategori *Security Logging and Monitoring Failures* digunakan untuk membantu mendeteksi, mengeskalasi, dan menanggapi pelanggaran aktif. Tanpa pencatatan dan pemantauan, pelanggaran tidak dapat dideteksi. Pencatatan, deteksi, pemantauan, dan respons aktif yang tidak memadai terjadi kapan saja:

1. Peristiwa pada perangkat lunak yang dapat diaudit, seperti *login*, *login* yang gagal, dan transaksi bernilai tinggi, tidak dicatat.
2. Peringatan dan kesalahan tidak menghasilkan pesan log yang tidak memadai, atau tidak jelas.
3. Log aplikasi dan API tidak dipantau untuk aktivitas yang mencurigakan.
4. Log hanya disimpan secara lokal.
5. Ambang batas peringatan yang tepat dan proses eskalasi respons tidak ada atau tidak efektif.
6. Pengujian penetrasi dan pemindaian oleh alat pengujian keamanan aplikasi dinamis (DAST) (seperti OWASP ZAP) tidak memicu peringatan.
7. Aplikasi tidak dapat mendeteksi, eskalasi, atau memperingatkan serangan aktif secara *real-time* atau mendekati *real-time*.
8. Pencegahan

1. Pastikan semua kegagalan *login*, kontrol akses, dan validasi *input* sisi server dapat dicatat dengan konteks pengguna yang memadai untuk mengidentifikasi akun yang mencurigakan atau berbahaya dan ditahan untuk waktu yang cukup untuk memungkinkan analisis forensik yang tertunda.
2. Pastikan bahwa log dibuat dalam format yang dapat digunakan oleh solusi manajemen log dengan mudah.
3. Pastikan data log dikodekan dengan benar untuk mencegah injeksi atau serangan pada sistem *logging* atau pemantauan.
4. Pastikan transaksi penting memiliki jejak audit dengan kontrol integritas untuk mencegah gangguan atau penghapusan, seperti tabel database hanya-tambah atau serupa.
5. Tim DevSecOps harus membuat pemantauan dan peringatan yang efektif sehingga aktivitas mencurigakan terdeteksi dan ditanggapi dengan cepat.
6. Tetapkan atau adopsi rencana respons dan pemulihan insiden, seperti National Institute of Standards and Technology (NIST) 800-61r2 atau yang lebih baru.
7. Contoh Skenario Serangan

1. Operator situs web penyedia paket kesehatan anak tidak dapat mendeteksi pelanggaran karena kurangnya pemantauan dan pencatatan. Pihak eksternal memberi tahu penyedia rencana kesehatan bahwa penyerang telah mengakses dan memodifikasi ribuan catatan kesehatan sensitif lebih dari 3,5 juta anak. Tinjauan

pasca-insiden menemukan bahwa pengembang situs web tidak mengatasi kerentanan yang signifikan. Karena tidak ada pencatatan atau pemantauan sistem, pelanggaran data dapat berlangsung sejak 2013, periode lebih dari tujuh tahun.

2. Sebuah maskapai penerbangan besar India mengalami pelanggaran data yang melibatkan data pribadi jutaan penumpang selama lebih dari sepuluh tahun, termasuk data paspor dan kartu kredit. Pelanggaran data terjadi di penyedia *hosting cloud* pihak ketiga, yang memberi tahu maskapai tentang pelanggaran tersebut setelah beberapa waktu.
3. Sebuah maskapai penerbangan besar Eropa mengalami pelanggaran *General Data Protection Regulation* (GDPR) yang dapat dilaporkan. Pelanggaran itu dilaporkan disebabkan oleh kerentanan keamanan aplikasi pembayaran yang dieksploitasi oleh penyerang, yang mengumpulkan lebih dari 400.000 catatan pembayaran pelanggan. Maskapai ini didenda 20 juta *pound* sebagai akibat dari regulator privasi.

***Server-Side Request Forgery* (SSRF)**

a. Deskripsi

Cacat SSRF terjadi setiap kali aplikasi web mengambil sumber daya jarak jauh tanpa memvalidasi URL yang disediakan pengguna. Ini memungkinkan penyerang untuk memaksa aplikasi mengirim permintaan yang dibuat ke tujuan yang tidak terduga, bahkan ketika dilindungi oleh *firewall*, VPN, atau teknologi lain dari daftar kontrol akses jaringan (ACL).

Karena aplikasi web modern menyediakan fitur yang nyaman bagi pengguna akhir, mengambil URL menjadi skenario umum. Akibatnya, kejadian SSRF meningkat. Juga, tingkat keparahan dampak serangan SSRF menjadi lebih tinggi karena layanan *cloud* dan kompleksitas arsitektur.

4. Pencegahan

1. Pada Lapisan Jaringan: Segmentasikan fungsionalitas akses sumber daya jarak jauh di jaringan terpisah untuk mengurangi dampak SSRF, Terapkan kebijakan *firewall* "tolak secara *default*" atau aturan kontrol akses jaringan untuk memblokir semua kecuali lalu lintas intranet yang penting.
2. Pada Lapisan Aplikasi: Membersihkan dan memvalidasi semua data input yang disediakan klien, terapkan skema URL, *port*, dan tujuan dengan *whitelist*, jangan mengirim *raw response* ke klien, nonaktifkan pengalihan HTTP,

3. Contoh Skenario Serangan

1. Port scan server internal – Jika arsitektur jaringan tidak tersegmentasi, penyerang dapat memetakan jaringan internal dan menentukan apakah *port* terbuka atau tertutup pada server internal dari hasil koneksi atau waktu yang telah berlalu untuk menghubungkan atau menolak koneksi *payload* SSRF.
2. Akses penyimpanan metadata layanan cloud – Sebagian besar penyedia *cloud* memiliki penyimpanan metadata seperti http://169.254.169.254/. Penyerang dapat membaca metadata untuk mendapatkan informasi sensitif.
3. Eksposure data sensitif – Penyerang dapat mengakses *file* lokal atau layanan internal untuk mendapatkan informasi sensitif seperti file:///etc/passwd dan http://localhost:28017/.
4. Akses ilegal layanan internal – Penyerang dapat menyalahgunakan layanan internal untuk melakukan serangan lebih lanjut seperti Remote Code Execution (RCE) atau Denial of Service (DoS).