

Seleksi Bagian A

Laboratorium Sistem Terdistribusi

"Warisan Fragmentasi"

Dipersiapkan oleh:

妹ラボラトリー

(Asisten Laboratorium Sistem Terdistribusi)

Sister; Lab²²

Dikerjakan oleh:

M. Rayhan Farrukh - 13523035

Waktu Mulai :

Minggu, 20 Juli 2025, 01.35 WIB

Waktu Akhir :

Kamis, 31 Juli 2024, 23.59 WIB

Tahap II

I. Jaringan Komputer



Wireshark

App

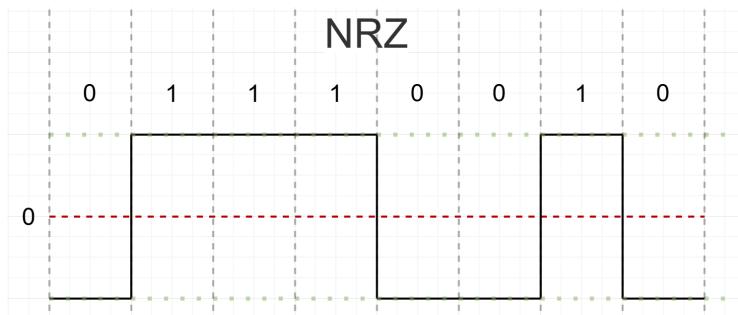
<https://youtube.com/@samekosaba>

1. (2.5 poin)

- a. NRZ, bit 1 direpresentasikan dengan level tegangan konstan (umumnya positif) dan bit 0 dengan level tegangan konstan lain (umumnya negatif), tanpa adanya kondisi netral (tegangan tidak pernah bernilai 0).

Kekurangannya pada sinkronisasi, jika bit sama terlalu panjang (000000 atau 111111), maka tidak ada transisi, sehingga sulit melakukan sinkronisasi.

Ilustrasi NRZ untuk 01110010:

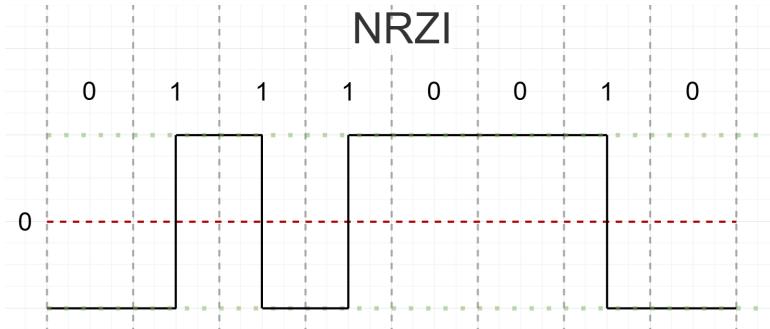


- b. NRZI, bit 1 direpresentasikan dengan perubahan level tegangan (transisi), dan bit 0 dengan tidak adanya perubahan.

Kelebihannya adalah mengangani masalah sinkronisasi pada NRZ untuk bit 1 yang panjang.

Kekurangannya masalah sinkronisasi masih ada untuk bit 0 yang panjang.

Ilustrasi NRZI untuk 01110010:

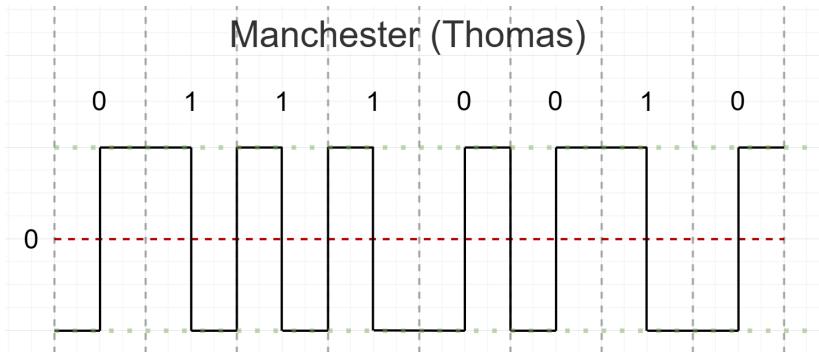


- c. Manchester, kedua *bit state* direpresentasikan dengan transisi tegangan di tengah bit, misal *high-to-low* untuk 1 dan *low-to-high* untuk 0 (konvensi G.E. Thomas).

Kelebihannya menghilangkan masalah sinkronisasi yang ada pada NRZ dan NRZI.

Kekurangannya membutuhkan sekitar dua kali lipat *bandwidth* dari NRZ karena terjadi dua kali lipat transisi tegangan, dan jarak transmisi lebih pendek.

Ilustrasi Manchester untuk 01110010:

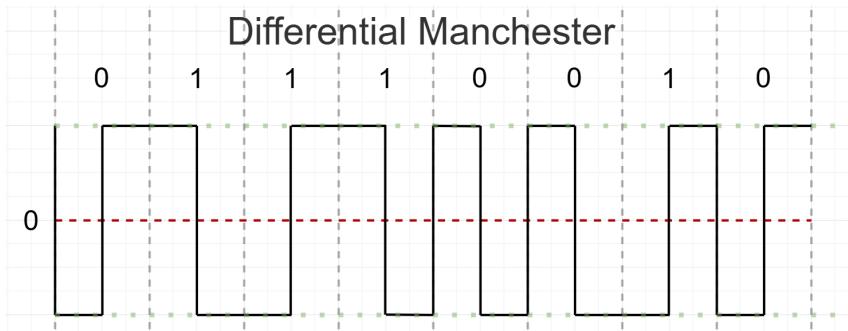


- d. Differential Manchester, sama seperti manchester selalu memiliki transisi di tengah bit. Namun perbedaan bit bukan pada transisi *high-to-low* atau *low-to-high* di tengah, melainkan pada transisi yang terdapat di awal bit (0) dan tidak adanya transisi tersebut (1).

Kelebihannya lebih tahan terhadap *noise*, interferensi dan perubahan kutub (positif/negatif).

Kekurangannya menambah kompleksitas implementasi.

Ilustrasi Differential Manchester untuk 01110010:

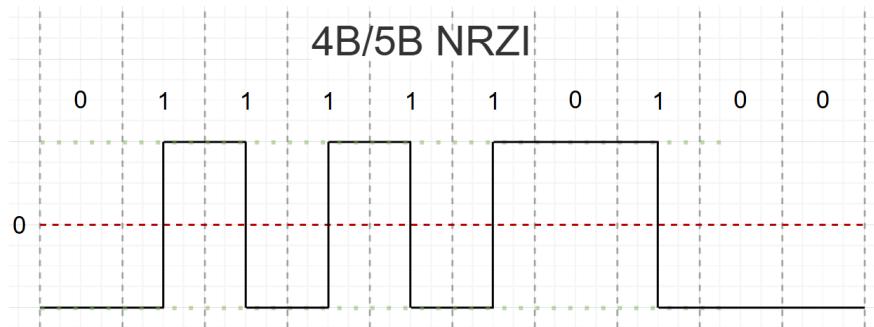


- e. 4B/5B merupakan *block code encoding* yang tidak mentranslasikan data *bit-by-bit* secara langsung seperti metode sebelumnya. 4B/5B mengkodekan 4 bit data menjadi 5 bit data baru. Pengkodean ini dibuat sedemikian sehingga jumlah bit 0 yang bersampingan sedikit mungkin. Pengkodean ini mengatasi masalah sinkronisasi bit 0 panjang yang dimiliki NRZI, dan karena itu 4B/5B di transmisikan menggunakan *encoding* NRZI.

Kelebihannya efisien secara *bandwidth* dan lebih simpel dari Differential Manchester (karena menggunakan NRZI)

Kekurangannya efisiensi data 80% karena 4 bit diubah menjadi 5 bit.

Ilustrasi 4B/5B untuk 01110010:



01110010 dikodekan menjadi 0111110100 sesuai *encoding table*, kemudian hasil tersebut di-*encode* menggunakan NRZI.

- [1] W. G. Wong, "What's the Difference Between NRZ, NRZI, and Manchester Encoding?," *Electronic Design*, July 1, 2024. [Online]. Available: <https://www.electronicdesign.com/article/21802271/electronic-design-whats-the-difference-between-nrz-nrzi-and-manchester-encoding>
- [2] GeeksforGeeks, "Difference Between Manchester and Differential Manchester Encoding," GeeksforGeeks, August 5, 2024. [Online]. Available: <https://www.geeksforgeeks.org/computer-networks/difference-between-manchester-and-differential-manchester-encoding>
- [3] L. Peterson and B. Davie, "2.2: Encoding," *Engineering LibreTexts, Computer Networks - A Systems Approach* (Peterson and Davie). [Online]. Available: [https://eng.libretexts.org/Bookshelves/Computer_Science/Networks/Computer_Networks_-_A_Systems_Approach\(Peterson_and_Davie\)/02%3A_Direct_Connections/2.02%3A_Encoding](https://eng.libretexts.org/Bookshelves/Computer_Science/Networks/Computer_Networks_-_A_Systems_Approach(Peterson_and_Davie)/02%3A_Direct_Connections/2.02%3A_Encoding)

2. (2.5 poin)

- a. *Collision* atau tabrakan pada transmisi data bermasalah karena terjadinya inferensi antara sinyal dari kedua pihak yang bertabrakan, sehingga terjadinya *packet loss*. Algoritma yang digunakan adalah CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*)

Cara Kerja:

Carrier Sense artinya *sender* mendeteksi voltase pada sebuah *channel/kabel* yang merupakan sinyal atau data yang sedang dikirim. Sebelum mengirim data, *sender* akan mengecek apakah sedang ada transmisi data pada *channel*, jika ada maka tunggu hingga transmisi tersebut selesai. Jika tidak, data dikirim langsung.

Selagi data dikirim, *sender* akan terus menerus memonitor *channel* untuk mengecek keberadaan data lain yang dikirim. Jika di tengah pengiriman terdeteksi *collision* (voltase abnormal), maka pengiriman data dihentikan dan perangkat mengirim sinyal untuk mengabari ke perangkat lainnya bahwa terjadi *collision*.

Setelah itu, *sender* akan menunggu beberapa saat sebelum data di-retransmisi. *Delay retransmisi* dihitung menggunakan algoritma *Binary Exponential Backoff*, dimana *delay* dipilih secara acak pada suatu *range*. *Range* akan membesar secara eksponensial seiring terjadinya *collision* sekvensial.

- b. Ketika perangkat *wireless* mentransmisi data, sinyal yang sedang dikeluarkan juga diterima oleh *receiver*-nya sendiri, dan sinyal ini akan lebih kuat dari sinyal yang berasal dari perangkat lain. Oleh karena itu, tidak bisa mendeteksi sinyal lain sekaligus mengirimkan sinyal sendiri sehingga *collision detection* tidak bisa dilakukan.
- c. *Wireless Network* menggunakan CSMA/CA (*CSMA with Collision Avoidance*)

Cara Kerja:

Bagian awal sama seperti CSMA/CD, yaitu *sender* mendeteksi apakah terdapat sinyal *wireless* pada jaringan, dan akan menunggu jika terdeteksi ada sinyal dari perangkat lain. Namun, jika tidak terdeteksi sinyal lain, *sender* akan menunggu beberapa saat yang disebut *Inter-frame gap* (IFG) sebelum mengirim data.

Setelah mengirim, *sender* akan menunggu ACK dari *receiver/target*. Jika tidak ada ACK setelah *timeout* maka *collision* terjadi dan *sender* akan menunggu sesuai algoritma *Backoff* sebelum me-retransmisi data.

Selain itu, pada CSMA/CA juga dapat menggunakan RTS/CTS untuk mengatasi *hidden node problem*. Mekanismenya, *sender* mengirim *frame Request to Send* (RTS) kepada *receiver*, jika *channel* tidak sedang sibuk, *receiver* akan mengirim *Clear to Send* (CTS) kepada *sender*, yang menandakan aman untuk mengirim data. Perangkat lain pada jaringan yang “mendengar” CTS akan diam dan memberi jalan untuk *sender* pertama menyelesaikan transmisi.

- [1] Coursera Staff, "What is CSMA?," Coursera, Oct. 7, 2024. [Online]. Available: <https://www.coursera.org/articles/csma>
- [2] Wikipedia contributors, "Carrier-sense multiple access with collision avoidance," Wikipedia, The Free Encyclopedia, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Carrier-sense_multiple_access_with_collision_avoidance&oldid=1173678070

3. (3 poin)

- a. Informasi yang dibutuhkan adalah alamat IP dan nomor port, dimana alamat IP dibungkus pada *network layer* (IP) dan nomor port pada *transport layer* (TCP/UDP).

UDP disebut *simple demultiplexer* karena berfungsi mengarahkan paket-paket yang tiba ke proses yang sesuai **hanya menggunakan** nomor port tujuan dari masing-masing paket, tanpa fungsi tambahan seperti *flow control/ordered delivery*.

- b. Alamat IP disimpan pada *network layer* (Protokol IP) bukan pada *network layer* (TCP/UDP). Ini karena pembagian tugas *layer* pada OSI model.

Pada model OSI, *transport layer* menjembatani *application layer* (atau *session*, *presentation*, dan *application* yang disatukan pada model TCP/IP) dengan *network layer*. Layer ini bertanggung jawab untuk komunikasi process-to-process menggunakan nomor port dan meneruskan data ke *network layer*.

Pada *network layer*, data dari *transport layer* akan dienkapsulasi dengan informasi tambahan, termasuk alamat IP tujuan. *Network layer* hanya perlu mengetahui alamat IP untuk mengirim paket dan tidak perlu mengetahui nomor port.

Ketika paket sampai di tujuan, data didekapsulasi dimana *network layer* akan meneruskan data ke *transport layer*, yang kemudian menggunakan nomor port untuk meneruskannya ke proses yang benar, dimana untuk melakukan ini *transport layer* tidak perlu mengetahui alamat IP.

- c. Penambahan *pseudo-header* (yang mengandung informasi IP) penting untuk memastikan data yang disampaikan utuh dan sampai di tujuan yang benar. Jika *pseudo-header* tidak ditambahkan pada TCP, maka *error detection* TCP hanya dapat mendekripsi kerusakan data saja. Sedangkan pada level IP *checksum* untuk data TCP tidak disimpan, sehingga hanya mampu mendekripsi kerusakan pada *header IP* saja.

- [1] L. Peterson and B. Davie, "5: End-to-End Protocols," *Engineering LibreTexts, Computer Networks - A Systems Approach* (Peterson and Davie). [Online]. Available: [https://eng.libretexts.org/Bookshelves/Computer_Science/Networks/Computer_Networks_-_A_Systems_Approach\(Peterson_and_Davie\)/05%3A_End-to-End_Protocols](https://eng.libretexts.org/Bookshelves/Computer_Science/Networks/Computer_Networks_-_A_Systems_Approach(Peterson_and_Davie)/05%3A_End-to-End_Protocols)
- [2] GeeksforGeeks, "Calculation of TCP Checksum," GeeksforGeeks, last updated July 11, 2025. [Online]. Available: <https://www.geeksforgeeks.org/computer-networks/calculation-of-tcp-checksum>

4. (4 poin)

- a. Kurir mencerminkan pendekatan QUIC dengan menunjukkan bahwa QUIC melewati prosedur panjang untuk *establish connection* dengan langsung mengenkripsi dan mengirim data dalam satu *trip*. Sehingga "penjaga" (misalnya *router*) tidak sempat mengenali data dan asalnya.

Pada koneksi biasa (misalnya TCP), sebelum data dikirimkan perlu ada *establishing connection* melalui *three-way handshake*, serta *TLS handshake* untuk keperluan enkripsi. Ini seperti kurir yang harus membuka gerbang, membuka pintu lalu masuk ke rumah sebelum paket diletakkan.

- b. Seperti yang disinggung pada cerita, penjaga tidak bisa mengetahui apakah kurir tersebut berhak masuk atau tidak, sehingga menimbulkan potensi dimana penyerang melakukan *masquerading* sebagai kurir dan mengantarkan paket yang berbahaya.

Protokol QUIC membuat *firewall* tidak dapat menginspeksi paket yang dikirim karena semua detail paket tersebut dienkripsi dari awal sehingga menimbulkan *blind spot* pada keamanan.

- c. Skenario pada soal merujuk pada fitur *Connection Migration* pada QUIC, dimana paket yang dikirim dapat berpindah jaringan (misal dari WIFI ke jaringan seluler) tanpa harus membuat koneksi baru dari awal.

Keuntungan utamanya adalah meningkatkan kecepatan dan stabilitas koneksi, terlebih jika sering berpindah jaringan.

Tantangannya adalah masalah kompatibilitas dengan *hardware* atau *software* lama yang mungkin tidak bisa meng-*handle* metode ini.

- [1] S. Petryschuk, "What is QUIC? Everything You Need to Know," Auvik - Auvik Networks, last updated Oct. 23, 2024. [Online]. Available: <https://www.auvik.com/franklyit/blog/what-is-quic-protocol>
- [2] A. Buchet and C. Pelsser, "An Analysis of QUIC Connection Migration in the Wild," arXiv preprint arXiv:2410.06066, Oct. 2024. [Online]. Available: <https://arxiv.org/html/2410.06066v1>

5. (2.5 poin)

Bapak, Ibu, jadi *reverse proxy* ini perlu ditambahkan ke sistem kita karena akan sangat bermanfaat. *Reverse proxy* bertindak seperti satpam gerbang yang mengatur semua lalu lintas yang masuk ke sistem kita. Ini akan membuat sistem kita lebih aman karena tidak perlu berhubungan langsung sama dunia luar, satpam inilah yang akan berkomunikasi dan juga sekaligus melakukan *filtering* untuk lalu lintas yang tidak aman. Selain itu, satpam ini juga mencatat hal-hal yang sering digunakan oleh pengguna sehingga bisa cepat tanggap melayani permintaan tersebut tanpa harus diproses oleh sistem kita. Ini akan meningkatkan kecepatan aplikasi kita.

Kemudian ketika banyak yang memakai aplikasi kita, satpam ini akan mengarahkan pengguna ke *server* yang kosong sehingga bebannya seimbang dan aplikasi kita tidak menjadi lambat. Selain itu, kalo ada bagian sistem kita rusak, kita jadi mudah mengelolanya karena satpam hanya perlu mengarahkan pengguna ke *server* yang tidak rusak.

- [1] Cloudflare Learning Center, "What is a reverse proxy?" [Online]. Available: <https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy>.

6. (2.5 poin)

Mekanisme koneksi *proxy* SOCKS5 dan komunikasi *client-server*:

- Pertama, *client* memulai *handshake* ke *proxy server* untuk memulai koneksi ke *target server*. *Proxy server* kemudian mengirim konfirmasi ke *client* yang dapat berisikan permintaan ke *client* untuk melakukan autentikasi.
- Setelah itu, *client* mengirim permintaan untuk membuat koneksi ke *target server*. *Proxy server* akan mencoba membuat koneksi ke target dengan menggunakan alamat IP *proxy server* sebagai *source address*, serta *port number* unik untuk menandakan *client*.
- Setelah itu, *proxy server* akan mengirim pesan khusus yang berisi status keberhasilan. Jika tidak berhasil, maka koneksi *client-proxy* ditutup dan sesi dianggap selesai. Jika berhasil, *proxy server* akan mulai menjadi perantara untuk semua pesan yang dikirim antara *client* dan *target server*.

SOCKS5 memiliki protokol tambahan untuk menambah fleksibilitas dalam meng-*handle network traffic*. Contohnya, karena protokol tambahan SOCKS5 dapat memiliki banyak metode autentikasi dan dapat meng-*handle* lebih banyak protokol selain TCP (misalnya UDP).

Fundamental Advantages:

- Privasi dengan menyembunyikan alamat IP *client*
- *Bypass* pemblokir seperti *firewall* atau *ISP content filtering*
- Fleksibilitas protokol

- [1] C. Gamutan, "What is a SOCKS Proxy and How Does It Work," Geonode, Apr. 26, 2023. [Online]. Available: <https://geonode.com/blog/socks-proxy-guide>
- [2] G. Ayisigi, "What is SOCKS Protocol? (Socket Secure)," Medium, Dec. 3, 2023. [Online]. Available: <https://gokhnayisigi.medium.com/what-is-socks-protocol-socket-secure>

7. (4 poin)

- a. Untuk Klien A, *Distance Vector* lebih cocok karena lebih efisien dan sederhana. Protokol *Link State* membutuhkan lebih banyak sumber daya untuk fitur-fitur yang *overkill* pada jaringan kecil yang dimiliki Klien A, dimana skalabilitas dan pemulihan cepat bukan hal yang kritis.
- b. Untuk Klien B, *Link State* lebih cocok karena secara langsung menjawab kebutuhan pemulihan yang cepat. Karena mekanisme *link state flooding* membuat konvergensi cepat, sehingga jaringan bisa melakukan pemulihan lebih awal. Selain itu, jalur redundan dapat menjadi penyebab *persistent routing loop*, dimana *Link State* kebal terhadap masalah tersebut.
- c. *Count to Infinity* disebabkan oleh komunikasi antar tetangga yang saling memberikan informasi rute yang tidak valid, misal ketika ada *link* yang terputus. Ini tidak mungkin terjadi pada *Link State* karena tidak menggunakan model komunikasi antar tetangga seperti *Distance Vector*. Pada *Link State* setiap *router* membuat peta topologi jaringan, dan dengan itu dapat menghitung rute terpendek sendiri.

8. (3 poin)

VPN bekerja dengan membuat *network tunnel* untuk komunikasi antara *client* dan *vpn server*. Di dalam *tunnel* ini, semua data yang terkirim terenkripsi sehingga DPI tidak dapat mendeteksi permintaan koneksi ke situs-situs yang diblokir sehingga *TCP reset* pun tidak terjadi. Selain itu, *client* sebenarnya tidak pernah berhubungan dengan *target server*, namun hanya dengan *VPN server*, sehingga teknologi pamungkas Kominfo tidak mengetahui bahwa *client* sedang mengakses sebuah situs terlarang, karena *client* memang tidak mengaksesnya.

- [1] Sahil. "How does vpn help bypassing blocked site?," Super User, Jul. 18, 2019. [Online]. Available: <https://superuser.com/questions/1461512/how-does-vpn-help>

9. (2 poin)

- a. ping menggunakan protokol ICMP yang bekerja pada *network layer*.

curl *by default* menggunakan protokol HTTP yang beroperasi pada *application layer*, namun juga mendukung banyak protokol *application layer* lainnya. Dibawah itu, curl menggunakan TCP atau UDP (sesuai kebutuhan protokol *application layer*) pada *transport layer* dan IP pada *network layer*.

- b. *Firewall* pada server memblokir koneksi pada port HTTP/S (port 80 & 443) tapi tidak memblokir ICMP.

Cara menyelidiki: jalankan perintah nmap -p 80,443 hiyoritomoe.com untuk melihat status pemblokiran port.

Web Server pada hiyoritomoe.com tidak berjalan. ping hanya menggunakan protokol *network layer* yang hanya mengetes koneksi ke *server tujuan* dan tidak mengetes koneksi ke port tertentu. Bisa saja *web server* yang *listening* pada port 80/443 sedang tidak berjalan.

Cara menyelidiki: jalankan curl dengan flag -v untuk melihat log lengkap alasan koneksi gagal

- c. Seperti yang dijelaskan pada poin b, konfigurasi *firewall* yang dapat menyebabkan ini pada server adalah pemblokiran *inbound traffic* untuk TCP pada port 80 dan/atau 443, yang berasal dari manapun. Dan tidak ada *rule* lain yang memblokir ICMP.

- d. Seperti kasus sebelumnya, penyebab yang paling mungkin untuk kasus ini adalah *firewall* pada *host* atau antivirus yang memblokir ICMP sedangkan *traffic* HTTP/HTTPS tidak diblokir. Untuk skenario antivirus yang memblokir, ini dilakukan untuk menghindari *network probing* yang dilakukan dengan ping.

- [1] The curl project. "curl man page." [Online]. Available: <https://curl.se/docs/manpage.html>

II. Sistem Paralel dan Terdistribusi



Distributed and Parallel Sisters

1. (2.5 poin)

- a. *Bottleneck* performa *distributed filesystem* (DFS):
 - i. *Network Latency*, akses DFS dilakukan dalam jaringan, sehingga kecepatan jaringan dapat memengaruhi performa, tidak seperti akses *file* pada mesin lokal.
 - ii. *Centralized Metadata*, ketika menggunakan satu titik untuk menyimpan metadata, maka jika ada banyak pengguna yang mengakses *file* secara bersamaan, server metadata akan kewalahan sehingga merusak performa.
- b. Beberapa solusi yang dapat digunakan untuk mengatasi *bottleneck*:
 - Menggunakan ukuran *chunk* yang besar agar tidak terlalu sering meminta data ke server metadata dan *overhead* jaringan lebih sedikit. GFS menggunakan 64 MB dan HDFS menggunakan 128 MB *chunk size*
 - Menggunakan *caching* metadata pada sisi *client*, agar mengurangi interaksi *client* dan server metadata

- [1] F. Ul Haq, "Exploring Distributed File Systems," *System Design Handbook*, Mar. 20, 2025. [Online]. Available: <https://www.systemdesignhandbook.com/blog/exploring-distributed-file-systems/#apache-hadoop-distributed-file-system-hdfs>.
- [2] "Metadata Management in Distributed File Systems," GeeksforGeeks, Jul. 23, 2025. [Online]. Available: <https://www.geeksforgeeks.org/system-design/metadata-management-in-distributed-file-systems/>.
- [3] "Which is faster, and why: transferring several small files or few large files?," Server Fault, May 19, 2009. [Online]. Available:

<https://serverfault.com/questions/9742/which-is-faster-and-why-transferring-several-small-files-or-few-large-files>.

2. (2.5 poin)

Sebuah *vector clock* menyimpan *logical clock* untuk setiap *node* yang ada pada sistem. Kemudian setiap *node* menyimpan vektor miliknya sendiri. Nilai-nilai di dalam vektor akan diperbarui sesuai aturan tertentu, yang menandakan *event-event* pada *node*, sehingga vektor dapat menunjukkan hubungan antar *event* pada semua *node/proses*.

Dua *events* *a* dan *b* dikatakan konkuren jika ada indeks *i* dimana $a[i] < b[i]$, namun ada indeks lain *j* dimana $a[j] > b[j]$.

Dua *events* *a* dan *b* memiliki hubungan kausal jika untuk semua indeks *i*, $a[i] \geq b[i]$ (*a causally follows b*) atau $a[i] \leq b[i]$ (*a causally precedes b*).

Vector clock lebih *powerful* dari *logical clock* biasa (Lamport *clock*) untuk *causal ordering* karena Lamport *clock* hanya menggunakan satu integer per-node yang hanya mampu mencatat kapan suatu *event* terjadi tanpa mengetahui hubungannya dengan *event* lain.

- [1] GeeksforGeeks, "Logical Clock in Distributed System," GeeksforGeeks. Last updated 15 July 2025 [Online]. Available: <https://www.geeksforgeeks.org/computer-networks/logical-clock-in-distributed-system/>.
- [2] S. S. Kumar, "Vector Clocks" Medium. May 17 2020 [Online]. Available: <https://medium.com/big-data-processing/vector-clocks-182007060193>.

3. (2.5 poin)

Tidak adanya *global clock* membuat sinkronisasi menjadi lebih sulit. Menggunakan *local clock* pada masing-masing *node* memunculkan masalah baru seperti *clock drift*. Dua *event* yang terjadi bersamaan pada dua komputer yang berbeda mungkin tercatat pada waktu yang berbeda, atau *event* yang terjadi lebih awal tercatat seolah-olah terjadi belakangan. Selain *clock drift* ada beberapa masalah lain. Oleh karena itu, setiap *node* harus berkomunikasi satu sama lain dengan lebih terlibat, contohnya penggunaan *logical clock* yang dibahas pada soal nomor 2.

4. (3 poin)

False Sharing adalah kejadian dimana dua *thread* tanpa sengaja menurunkan performa satu sama lain karena dua variabel berbeda yang terletak pada *cache line* yang sama.

Ini disebut "*silent performance killer*" karena sulit menentukan apakah *false sharing* akan terjadi dari melihat kode nya.

Follow Up

Kejadian tersebut (*false sharing*) menurunkan performa karena *MESI protocol*. *Cache line* adalah unit data terkecil yang bisa dikelola oleh *cache*. Walaupun kedua *thread* (misal **A** dan

B) mengakses variabel yang berbeda, mereka tetap mengakses *cache line* yang sama. *False sharing* terjadi ketika **A** mengakses *cache line* dan menandakannya sebagai M (*modified*), sesuai MESI *protocol* maka *cache line* sama yang diakses **B** menjadi *invalid*. Ketika *cache line* menjadi *invalid*, **B** harus mengambil versi terbaru dari *cache line* tersebut yang sedang dipegang **A**, operasi ini membutuhkan waktu. Setelah mengambil dari **A**, **B** mengakses *cache line* tersebut dan menandakannya juga sebagai M, mengulangi proses invalidasi *cache line*, kejadian ini terjadi berulang-ulang hingga salah satu *thread* selesai mengakses variabel. Inilah yang menyebabkan penurunan performa.

- [1] Alibaba Cloud, "False Sharing: The Silent Performance Killer of Concurrent Programming," *Alibaba Cloud Topic*. Last updated August 24 2016 [Online]. Available: <https://topic.alibabacloud.com/a/false-sharing-the-silent-performance-killer>.

5. (3 poin)

Masing-masing layanan berinteraksi dengan melakukan *RPC call* pada layanan yang dibutuhkannya. Misalnya pada layanan **Feed** ketika menampilkan *post* pada laman *feeds*. **Feed** akan melakukan *RPC call* ke layanan **Image**, **Comment**, **Like**, dan **Metadata** untuk mendapatkan semua informasi yang diperlukan untuk menayangkan *post-post* yang akan ditayangkan pada laman *feeds*. Semua informasi kemudian dibungkus **Feed** untuk ditampilkan pada *client side*. Begitu juga untuk layanan-layanan yang di-invokasi oleh **Feed** akan melakukan *RPC call* ke layanan lain yang dibutuhkannya.

Ketika melakukan **Fanout**, misalnya akun dengan 10 juta *follower* membuat suatu *post*, maka *post* tersebut akan ditayangkan ke laman *feeds* untuk 10 juta akun. Tentu saja ini akan memakan waktu yang cukup lama. Jika dilakukan secara sinkron dengan *RPC* maka pembuat *post* harus menunggu hingga **Fanout** selesai untuk 10 juta akun sebelum bisa lanjut menggunakan aplikasi. Oleh karena itu, sebaiknya **Fanout** dilakukan secara asinkron dengan *message queue*.

- [1] C. Penmetsa, "Remote Procedure Call (RPC) API Explained," Medium, Jan. 18, 2024. [Online]. Available: <https://medium.com/codenx/remote-procedure-call-rpc-api-explained-3d4a494ff28b>.

6. (3 poin)

intinya gini cuy, *distributed system* sama *decentralized system* itu bedanya di fokusnya. *Distributed system* itu murni soal arsitektur teknisnya, di mana sistemnya jalan di banyak komputer yang tersebar biar nggak gampang tumbang kalau satu mesin mati, sama bisa bagi-bagi kerjaan ke komputernya biar lebih kencang.

Nah, kalau *decentralized* tuh soal politik dan kekuasaan, tapi fungsi utamanya mirip dengan *distributed system*. *Decentralized system* itu berarti nggak ada satu bos atau pihak yang bisa seenaknya ngubah aturan atau matiin sistemnya. Jadi, yang bikin orang suka kripto tu karena desentralisasinya ngebikin nilainya susah dimanipulasi oleh satu pihak.

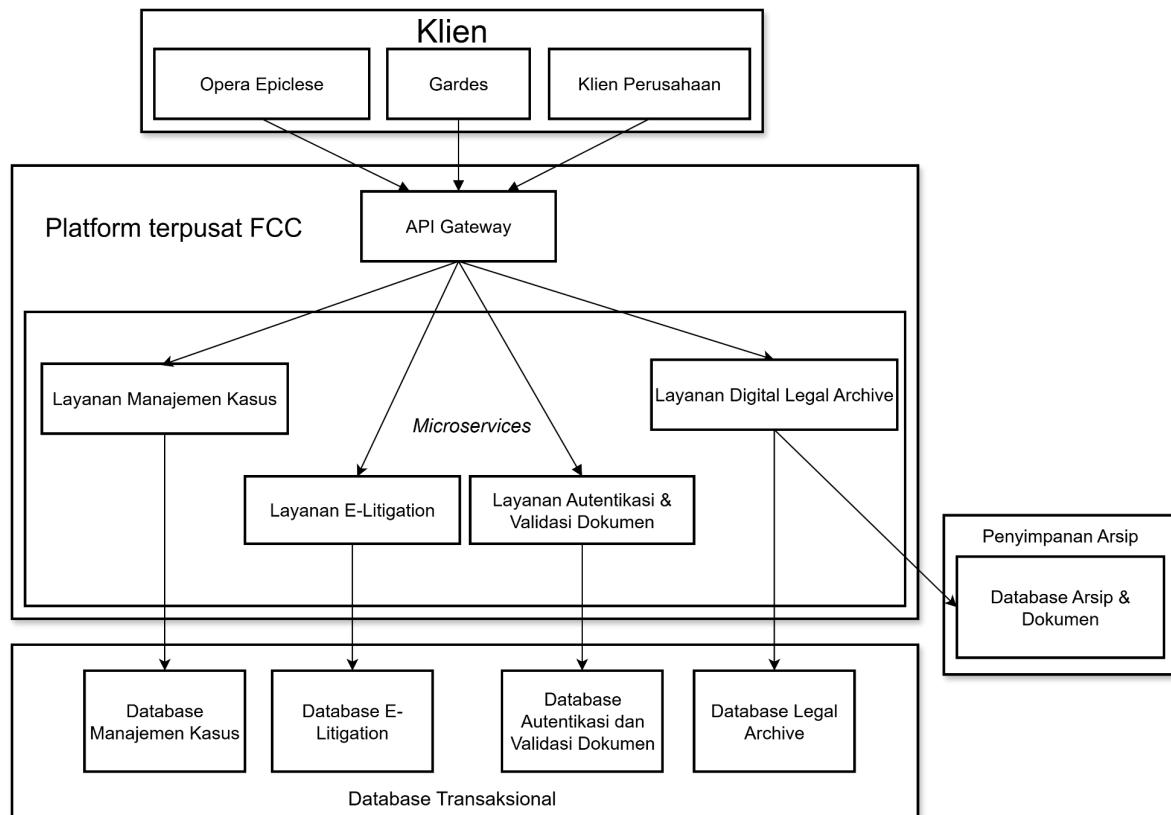
III. Teknologi Sistem Terintegrasi



Integrating the freeze team wheelchair

1. (20 poin)

- a. Untuk memenuhi kebutuhan, FCC sebaiknya membuat sistem dengan arsitektur *microservice*. Diagram dari arsitektur tersebut adalah sebagai berikut:



Arsitektur *microservice* dipilih karena memungkinkan setiap layanan untuk berjalan dan dikelola secara independen. Ini membuat layanan dapat di-*scale-out* tanpa mengganggu layanan lainnya. Kemudian untuk interoperabilitas, *layanan-layanan tersebut akan diakses melalui API gateway* untuk memudahkan klien mengakses masing-masing layanan.

Pada diagram, terlihat bahwa klien akan berinteraksi dengan *platform* melalui API *gateway* yang akan meneruskan permintaan mereka ke layanan yang relevan. Kemudian masing-masing layanan akan terhubung kepada basis data transaksional yang terpisah untuk keperluan penyimpanan data sehari-hari. Lalu untuk layanan *Digital legal archive* terdapat basis data khusus untuk penyimpanan arsip dan dokumen.

- b. Arsitektur sistem harus dibangun untuk dapat mengelola data dan menjaganya dari *tampering*. Data yang dihasilkan IoT harus dapat dikelola dengan baik agar tidak terjadi instabilitas pada sistem. Ini dapat dilakukan dengan sistem *stream processing* terdistribusi untuk menampung data dari banyak sumber kemudian menyalirkannya ke sistem dengan teratur. Sistem ini harus bekerja dengan latensi yang sangat rendah untuk memastikan data dapat diakses dan digunakan secara *real-time*.

Kemudian untuk menjaga keabsahan data, sistem dapat menggunakan *anti-tampering* berbasis kriptografi dan *blockchain*. Setiap data dari IoT diamankan dengan *digital signature* dan *hashing* untuk memastikan integritas data, kemudian *hashnya dicatat* pada *blockchain* untuk memastikan tidak dapat diubah. Selain *blockchain*, sistem dapat menggunakan *ledger database* dan mengintegrasikan fitur *trusted timestamping* jika dinilai biaya implementasi *blockchain* terlalu mahal.

- c. Dari aspek keamanan, penambahan banyak perangkat IoT yang tersebar akan memperbesar *attack vector*, menciptakan banyak titik lemah baru yang bisa dimanfaatkan penyerang untuk mencuri data, atau bahkan menyusup ke sistem internal FCC.

Dari aspek privasi, integrasi IoT berarti terjadi pengumpulan data secara terus-menerus pada ruang publik. Ini mungkin menjadi pengawasan berlebihan, yang beresiko menyebabkan penyalahgunaan data, serta kesulitan dalam mengelola data sensitif.

- d. Solusi yang dibahas pada poin (b) sudah menjawab masalah ini dengan prinsip *zero trust*. Penggunaan *blockchain* memastikan pihak manapun dengan hak kuasa apapun tidak dapat merusak data dari perangkat IoT. Sedangkan *ledger database* dengan *timestamping* memastikan tidak dapat diganggu pihak internal, namun tidak memenuhi prinsip *zero trust* karena masih meletakkan kepercayaan pada penyedia layanan.
- e. FCC sebaiknya menggunakan sistem *tiered storage* dengan membagi data-data yang didapat berdasarkan usia dan keperluan mengaksesnya. Ini dilakukan agar biaya menyimpan data sepadan dengan *value* dari data tersebut. Jenis data akan dibagi menjadi tiga tingkat:
 - *Hot*

Menggunakan *hardware* yang fokus pada kecepatan. Menyimpan data yang perlu diakses dengan cepat, contohnya data kasus hukum yang sedang aktif dan data baru dari IoT.

- *Warm*

Menggunakan *hardware* yang seimbang antara biaya dan kecepatan. Untuk data yang jarang diakses namun berkemungkinan akan dibutuhkan sehingga tetap harus dapat diakses dengan relatif cepat, contohnya data untuk kasus yang baru ditutup.

- *Cold*

Menggunakan *hardware* yang fokus pada ukuran penyimpanan. Untuk menyimpan data arsip jangka panjang, contohnya data untuk kasus yang telah lama selesai atau menyimpan data lainnya yang tidak dibutuhkan sebelum dihapus.

f. **Manfaat**

Integrasi IoT akan meningkatkan *value proposition* perusahaan sebagai layanan hukum digital. Integrasi ini akan menambah sumber bukti objektif yang dimiliki FCC. Ini juga akan meningkatkan efisiensi operasional untuk klien FCC dengan otomatisasi proses pengumpulan data dari IoT.

Risiko

Risiko utama dari integrasi ini adalah yang dibahas pada poin (c), keamanan dan privasi. Tetapi selain itu, risiko lainnya adalah *data tampering*, biaya untuk skalabilitas integrasi serta kerusakan yang terjadi pada *hardware* yang menyebabkan kehilangan data.

Mitigasi

Untuk memitigasi risiko, FCC harus menyesuaikan sistem untuk dapat menghindari risiko-risiko tersebut. Ancaman keamanan dapat dimitigasi dengan menguatkan keamanan perangkat IoT dan hubungannya ke sistem utama FCC, ini bisa dengan enkripsi *end-to-end* transfer data atau mengisolasi jaringan yang terhubung. Dampak privasi dapat dimitigasi dengan menetapkan kebijakan tata kelola data yang ketat.

Risiko data tampering dimitigasi dengan verifikasi data yang dibahas pada poin (b), dan risiko biaya skalabilitas dengan *tiered storage* yang dibahas pada poin (e). Untuk risiko kehilangan data akibat kerusakan *hardware*, dapat dimitigasi dengan melakukan *streaming* sebagai metode perekaman.

g. Bagian yang masih dapat dikembangkan untuk lebih memastikan keamanan sistem adalah kemampuan sistem untuk mencatat setiap aktivitas pada sistem sekaligus menganalisisnya. Ini berarti setiap peristiwa, seperti perubahan atau interaksi yang terjadi di dalam sistem akan tercatat secara otomatis, dan ketika ada aktivitas yang mencurigakan maka tim keamanan sistem bisa menghentikannya lebih awal.

[1] E. Palachi, “System architecture diagram basics & best practices,” vFunction, 2025. [Online]. Available: <https://vfunction.com/blog/architecture-diagram-guide>.

- [2] GeeksforGeeks, "What is the Role of API gateway in Microservices?," GeeksforGeeks, 2025. [Online]. Available: <https://www.geeksforgeeks.org/system-design/the-role-of-api-gateway-in-microservices>.
- [3] Debut Infotech, "How to Make Documents Tamper-Proof Using Blockchain," Debut Infotech. [Online]. Available: <https://www.debutinfotech.com/blog/how-to-make-documents-tamper-proof-using-blockchain>.
- [4] B. Reamico, "Understanding Immutable Ledger Technology: A Beginner's Guide," Medium, Oct. 29 2024. [Online]. Available: <https://medium.com/@benreamico/understanding-immutable-ledger-technology-a-beginners-guide>.
- [5] A. Brand, "Hot Storage vs Cold Storage: Choosing the Right Tier for Your Data," Medium, Jun 20 2022. [Online]. Available: <https://aronbrand.medium.com/hot-storage-vs-cold-storage-choosing-the-right-tier-for-your-data-12fa7c30959d>.

2. (2 poin)

Ada beberapa alasan mengapa integrasi *legacy sistem* sering gagal:

- **Inkompatibilitas format data atau data kotor.** Ini menyebabkan sistem baru tidak mampu mengolah data yang dihasilkan *legacy system* dengan baik, dan membutuhkan upaya tambahan untuk transformasi atau pembersihan data.
- **Arsitektur sistem yang tidak memadai.** Ketika *legacy system* tidak dirancang untuk menangani kecepatan atau volume permintaan yang dibutuhkan aplikasi modern sehingga menimbulkan *bottleneck*.
- **Kurangnya dokumentasi.** Pengembang tidak tahu cara menggunakan API karena sistem cukup kompleks atau terdapat *business logic* yang tersembunyi. API yang "tinggal digunakan" menjadi tidak berguna jika tidak tahu cara menggunakannya.
- **Kurangnya tenaga ahli.** *Legacy code* yang dibuat dengan teknologi atau bahasa pemrograman yang usang dan sudah tidak umum sehingga pengembang kesulitan dalam menggunakan ataupun memeliharanya.

IV. Keamanan Informasi



She doesn't have anything to do with information security, I just really love her - Duke

1. (4 poin)

- a. *Security* - pengamanan suatu sistem atau infrastruktur dari perusakan atau serangan untuk menjaga *triad CIA*.

Privacy - kebebasan suatu pihak dari keadaan diawasi oleh pihak lain baik secara langsung ataupun tidak langsung, serta hak pihak tersebut untuk mengontrol pengelolaan informasi pribadinya.

Keduanya saling melengkapi. Jika keamanan sistem tidak terjaga, maka informasi pribadi dapat diakses pihak lain sehingga merusak *privacy*. Namun, adakalanya *privacy* dikorbankan untuk mendapat *security*, contohnya penggunaan *surveillance camera*.

- b. *Vulnerability* - kelemahan atau celah intrinsik pada suatu sistem yang berpotensi disalahgunakan.

Threat - sesuatu yang berpotensi menyebabkan terjadinya hal yang tidak diinginkan kepada sistem.

Keduanya saling melengkapi untuk menimbulkan risiko atau bahaya. Adanya *vulnerability* bukan menjadi masalah jika tidak ada *threat* yang dapat meng-exploit-nya.

- c. *Incident* - kejadian keamanan yang tidak diinginkan, dimana kebijakan keamanan dilanggar baik secara sengaja ataupun tidak disengaja.

Attack - sebuah upaya yang disengaja untuk merusak atau mengakses suatu sistem secara tidak sah.

Keduanya saling melengkapi. Terjadinya sebuah *attack* yang berhasil merusak sistem merupakan sebuah *incident*. Tetapi, kejadian *incident* bukan berarti hasil dari sebuah *attack*.

- d. *Authentication* - proses keamanan yang memverifikasi kebenaran identitas seseorang.
Authorization - proses keamanan yang memverifikasi hak akses atau izin yang dimiliki seseorang.
Keduanya saling melengkapi. Untuk keamanan sistem, diperlukan keduanya untuk memastikan seorang penyusup tidak dapat menggunakan identitas atau informasi orang lain tanpa izin, serta untuk memastikan seseorang hanya dapat mengakses hal-hal yang berhak dia akses

- [6] V. Vicente, "Privacy vs. Security: Understanding the Difference," AuditBoard, Jun. 19, 2024. [Online]. Available: <https://auditboard.com/blog/privacy-vs-security>.
- [7] C. Kidd, "Vulnerabilities, Threats & Risk Explained," Splunk, Jul. 25, 2024. [Online]. Available: https://www.splunk.com/en_us/blog/learn/vulnerability-vs-threat-vs-risk.html.
- [8] "What is the difference between "Incident", "Attack" and "event"?", Information Security Stack Exchange, Apr. 22, 2018. [Online]. Available: <https://security.stackexchange.com/questions/184318/what-is-the-difference-between-incident-attack-and-event>.
- [9] "Authentication vs. Authorization: What's the Difference?," OneLogin. [Online]. Available: <https://www.onelogin.com/learn/authentication-vs-authorization>.

2. (2 poin)

Peran-peran teknologi:

Wazuh - memonitor keamanan dari *endpoints* (seperti komputer atau server) yang dimiliki organisasi dan menghasilkan *log* dari hasil *monitoring*.

Suricata - memonitor *network traffic* yang masuk dan keluar pada jaringan organisasi dan menghasilkan *log*-nya.

Kafka - mengelola *stream* data-data yang dihasilkan oleh Wazuh dan Suricata dan meneruskannya ke *database*.

Elasticsearch - tempat penyimpanan data dan sekaligus menjadi alat untuk menganalisis data tersebut.

Kibana - sebagai *user interface* untuk Elasticsearch, mempermudah visualisasi dan analisis data.

Aliran data:

Wazuh & Suricata → Kafka → Elasticsearch → Kibana

3. (4 poin)

Kebijakan Penggunaan *Single Sign-On*

Kebijakan

Yayasan Rumah Perapian (YRP) berkomitmen untuk melindungi kerahasiaan, integritas dan ketersediaan seluruh asset informasi pada sistem internal YRP. Untuk memenuhi komitmen ini, ditetapkan bahwa semua akses pengguna ke aplikasi internal YRP wajib diautentikasi menggunakan metode *Single Sign-On* (SSO). Kebijakan ini berlaku untuk semua karyawan, kontraktor, dan pihak ketiga yang diberikan akses ke sistem YRP.

Standar

Untuk menegakkan kebijakan di atas, semua pengguna dan sistem harus mematuhi standar keamanan berikut:

- 1) **Penggunaan Akun:** Seluruh pengguna harus menggunakan akun SSO individual yang telah disediakan untuk mengakses semua aplikasi internal YRP.
- 2) **Multi-Factor Authentication (MFA):** Setiap akun SSO wajib dilindungi dengan MFA. MFA dilakukan melalui aplikasi *authenticator* pada perangkat seluler (contoh: Microsoft Authenticator). Jika berkendala dalam penggunaan aplikasi *authenticator*, hubungi Tim IT YRP untuk mendiskusikan metode alternatif.
- 3) **Kompleksitas Kata Sandi:** Kata sandi SSO harus memenuhi kriteria minimum sebagai berikut:
 - a) Panjang minimal 12 karakter.
 - b) Mengandung kombinasi huruf besar (A-Z), huruf kecil (a-z), angka (0-9), dan simbol (!@#\$%).
 - c) Tidak boleh mengandung nama pengguna atau nama "YRP".
- 4) **Pencatatan Aktivitas (Logging):** Semua upaya login melalui SSO, baik yang berhasil maupun yang gagal, akan dicatat oleh sistem dan *log* tersebut akan disimpan.
- 5) **Larangan Berbagi Akun:** Dilarang keras berbagi atau meminjamkan kredensial akun SSO kepada orang lain, termasuk rekan kerja.,.

Prosedur

• Prosedur Aktivasi Akun

- 1) Pengguna baru akan menerima *email* aktivasi pada alamat *email* resmi YRP milik pengguna.
- 2) Klik tautan aktivasi yang ada pada *email* tersebut.
- 3) Pengguna akan diarahkan untuk membuat kata sandi baru. Masukkan kata sandi yang memenuhi standar yang berlaku.
- 4) Ikuti instruksi di layar untuk mendaftarkan perangkat Anda untuk MFA.
- 5) Masukkan kode 6 digit yang muncul pada aplikasi *authenticator* untuk menyelesaikan proses aktivasi.

- **Prosedur Lupa Kata Sandi**
 - 1) Buka halaman login SSO YRP.
 - 2) Klik tautan "Lupa Kata Sandi" atau "Forgot Password".
 - 3) Masukkan alamat *email* YRP Anda untuk menerima instruksi pembaruan kata sandi.
 - 4) Ikuti petunjuk yang dikirimkan pada *email* dan verifikasi identitas Anda sesuai petunjuk.
 - 5) Buat kata sandi baru yang memenuhi standar yang berlaku.
- **Prosedur Kehilangan Akses Aplikasi Authenticator**
 - 1) Hubungi *IT Helpdesk* YRP melalui telepon atau email untuk melaporkan kehilangan akses ke aplikasi authenticator.
 - 2) Ikuti proses verifikasi identitas yang akan diarahkan oleh Tim *IT Helpdesk*.
 - 3) Setelah verifikasi berhasil, Anda akan diinstruksikan untuk login kembali ke akun SSO YRP hanya menggunakan kata sandi.
- **Prosedur Pelaporan Insiden Keamanan**
 - 1) Jika pengguna mencurigai akun SSO telah diretas, atau ada notifikasi login yang mencurigakan, segera lapor ke *IT Helpdesk* YRP melalui telepon (+62XX) atau email (helpdesk@yrp.co.id).
 - 2) Ikuti arahan lebih lanjut dari Tim IT untuk melakukan pemulihan akun.

Guideline

Untuk meningkatkan keamanan akun pribadi Anda dan organisasi, pengguna sangat disarankan untuk mengikuti panduan berikut:

- Sebaiknya membuat kata sandi SSO dalam bentuk *passphrase*, yaitu *password* yang terdiri dari banyak kata, agar lebih mudah diingat namun tetap memenuhi standar.
- Hindari menggunakan kata sandi akun SSO YRP untuk akun pribadi lainnya untuk mencegah penyalahgunaan akun YRP jika akun lain tersebut diretas.
- Disarankan untuk selalu *Log Out* secara manual dari sesi SSO setelah selesai menggunakan aplikasi, terutama jika bekerja menggunakan komputer yang dapat diakses oleh banyak orang.
- Waspadai upaya *phishing*. Tim IT YRP tidak akan pernah meminta kata sandi atau kode MFA melalui email, *chat*, atau telepon. Jangan klik tautan mencurigakan yang mengatasnamakan YRP.

- [1] "S. Matsemela, "Policies, Standards, Procedures and Guidelines," YouTube, Mar. 10, 2024. [Online]. Available: <http://www.youtube.com/watch?v=8q7e9yfbegQ>.
- [2] "Cyber Security Policies and Standards," SANS Institute, n.d. [Online]. Available: <https://www.sans.org/information-security-policy>.

4. (10 poin)

a. Threat Model

i. Aset

- Data perusahaan
- Data klien dan mitra
- Audit dan *log*
- Kredensial pengguna
- Kode sumber
- *Secrets & Configs* (*session tokens, API keys*, dll.)
- Infrastruktur aplikasi
- *Goodwill/reputasi* perusahaan

Aktor

- Pengguna internal (manajemen dan staf IBS)
- Pengguna eksternal (klien dan mitra kerja IBS)
- *Insider threat*
- *External attacker*
- Layanan *third-party* yang digunakan aplikasi

ii. Metodologi STRIDE

Spoofing

- Penggunaan kredensial pengguna oleh penyerang yang didapat melalui *phishing* atau cara lain
- *Session hijacking* terhadap pengguna
- Pemalsuan *session token* untuk mengakses akun tertentu
- Pembuatan aplikasi palsu oleh penyerang yang bermaksud menipu pengguna untuk mendapat kredensial

Tampering

- *Payload tampering* melalui *Man-in-the-Middle attack* ketika menggunakan fitur yang disediakan layanan *third-party* (misal *payment API*)
- *Code injection* yang merusak alur kerja aplikasi sehingga menyebabkan keuntungan atau kerugian suatu pihak
- *Parameter tampering* untuk mengakses data atau fitur “illegal”
- *File* yang diunggah ke aplikasi dimodifikasi oleh penyerang untuk memasukkan data palsu atau *malware*

Repudiation

- Kurangnya *audit log* yang menyebabkan penyanggahan aktivitas tertentu oleh pengguna
- Manipulasi atau penghapusan *audit log* oleh penyerang untuk menutupi aktivitasnya

Information Disclosure

- Kebocoran data antar pengguna karena akses kontrol lemah

- *Error* yang tidak ditangkap atau ditangani dengan baik membocorkan logika atau detail teknis aplikasi
- Enkripsi lemah yang menyebabkan data yang ditransfer dapat dibaca penyerang
- Data pengguna tidak dienkripsi dan terjadi kebocoran basis data

Denial of Service

- Serangan DDoS, baik yang sengaja atau tidak sengaja
- Upaya gagal *login* yang disengaja untuk mengunci akun pengguna
- Penggunaan *resource* berlebihan oleh satu pengguna karena tidak ada *rate limiting*

Elevation of Privilege

- *Privilege escalation* vertikal, pengguna meng-exploit *vulnerability* sehingga mendapat hak akses level manajemen IBS
- *Privilege escalation* horizontal, klien atau mitra kerja yang berhasil mendapat akses untuk melihat atau memodifikasi data pengguna lain

b. Skenario Kena Hack

i. Top 10 OWASP

Menurut saya kategori OWASP Top 10:2021 yang relevan adalah A01:2021-Broken Access Control, A07:2021-Identification and Authentication Failures dan A09:2021-Security Logging and Monitoring Failures

Broken Access Control

Kategori ini mencakup kegagalan dalam penegakan aturan yang membataskan pengguna agar tidak bertindak diluar izinnya. Kerusakan atas *access control* menyebabkan pengguna bisa mendapatkan hak akses yang lebih tinggi melalui suatu celah pada aplikasi. Pada kasus IBS, ini terjadi ketika *hacker* menggunakan celah pada *endpoint API* untuk menaikkan tingkat akses.

Identification and Authentication Failures

Kategori ini mencakup kelemahan pada proses verifikasi identitas pengguna baik ketika hendak memasuki akun, atau selagi di dalam akun (pengelolaan sesi). Kegagalan pada proses autentikasi memungkinkan penyerang untuk menggunakan akun pengguna yang bisa saja memiliki akses tinggi. Pada kasus IBS, ini terjadi ketika *hacker* berhasil melakukan *brute force* pada aplikasi, dimana seharusnya ada *limit* percobaan masuk agar *brute force* tidak bisa dilakukan.

Security Logging and Monitoring Failures

Kategori ini mencakup kurangnya pencatatan dan pemantauan aktivitas pada aplikasi. Kegagalan dalam memantau sistem membuat serangan menjadi tidak terdeteksi, sehingga penyerang memiliki banyak waktu untuk mengembangkan serangannya. Pada kasus IBS, setelah serangan *brute force*, kegagalan *login* berkali-kali seharusnya memicu alarm tentang kegiatan yang mencurigakan. Sehingga serangan terdeteksi lebih awal dan *privilege escalation* dapat dihindari.

ii. Secure SDLC

Mengikuti proses *Secure SDLC* dengan baik akan mencegah kejadian ini karena dari awal mengintegrasikan pengamanan dan pengembangan aplikasi. Setiap tahap pengembangan dilakukan dengan memperhatikan implikasi keamanannya. Hal ini memastikan potensi ancaman dapat diidentifikasi dan dimitigasi sejak saat mendesain aplikasi.

Kegiatan pada tahap-tahap *Secure SDLC*

- Tahap *Planning/Requirement Gathering*

Kebutuhan keamanan bisa didefinisikan sejak awal. Contohnya batas upaya *login* untuk mencegah *brute force* dan penerapan MFA untuk memperkuat proses autentikasi.

- Tahap *Design*

Melakukan *threat modeling* untuk memetakan *attack vector* dengan baik, serta merancang mitigasi atas *threat* yang diidentifikasi. Contohnya merancang alur autentikasi yang kuat untuk aplikasi dan setiap API yang akan digunakan.

- Tahap *Development*

Menulis kode sesuai *secure coding practices* untuk memenuhi rancangan yang telah dibuat sebelumnya. Pengembang juga dapat menggunakan alat pemindai kode otomatis untuk mencari celah yang ada pada kode sumber.

- Tahap *Testing*

Melakukan *penetration testing* pada aplikasi untuk mencari celah-celah keamanan untuk memastikan tahap pengembangan sudah terselesaikan dengan baik.

- Tahap *Release*

Men-deploy aplikasi pada lingkungan *server* yang aman dan telah dikonfigurasi dengan baik, dan dilanjutkan dengan pemantauan keamanan untuk dapat mendeteksi dan merespons ancaman dengan cepat.

iii. Contoh Kasus Serupa

Uber Breach 2022

Pada 15 September 2022, terjadi serangan pada sistem internal Uber yang menyebabkan *hacker* mendapat akses admin pada beberapa sistem dan layanan yang digunakan Uber.

Penyerangan ini dimulai dengan *hacker* yang mendapat kredensial seorang kontraktor eksternal Uber, yang kemungkinan dibelinya dari “*dark web*”. *Hacker* menggunakan kredensial ini untuk masuk ke akun kontraktor, lalu melakukan *spam* notifikasi MFA, serta mengontak sang kontraktor melalui Whatsapp dengan menyamar sebagai tim IT Uber untuk meyakinkannya menerima permintaan MFA. Ini membuat *hacker* dapat masuk ke akun sang kontraktor.

Akun ini memberi *hacker* akses ke VPN milik Uber, yang memberi akses ke intranet Uber. Kemudian melalui *scanning* intranet *hacker* menemukan sebuah *network share* yang dapat diakses langsung oleh akun tersebut. Di dalam *network share* terdapat skrip PowerShell yang memuat kredensial admin yang di-*hardcode* dalam bentuk *plaintext*. Kredensial ini memberi akses ke *Privileged Access Management* yang pada dasarnya menjadi titik paling fatal. Dengan ini *hacker* mendapat akses ke layanan-layanan yang digunakan Uber seperti Slack dan AWS, serta data finansial Uber. Yang paling fatal adalah akses ke HackerOne yang berisikan laporan *bug bounty* untuk celah keamanan pada sistem Uber yang belum diperbaiki.

Meskipun begitu, tidak ada kebocoran data ke publik atau kerusakan yang serius pada sistem Uber, karena sang *hacker* tidak memiliki niat merusak, namun hanya untuk mencoba melakukan *cyberattack*. Dampak yang paling signifikan dari kasus ini adalah demonstrasi kegagalan keamanan pada sistem Uber.

Pelajaran yang dapat diambil

- Perkuat autentikasi. IBS seharusnya MFA agar *brute force* tidak berguna. Namun, penggunaan MFA juga harus yang lebih kompleks seperti *number matching* atau OTP untuk mempersulit upaya *social engineering*.
- Kelola aktor pada sistem untuk menghindari *social engineering*. Ini bisa dengan melakukan pelatihan untuk *staff* atau pihak internal IBS, dan imbauan untuk pihak pengguna eksternal.
- Terapkan prinsip *least privilege*. Pada kasus Uber, akun kontraktor seharusnya tidak dapat mengakses *network share* tersebut karena di luar *scope* kebutuhannya, sama halnya dengan akun pada sistem IBS. Ini akan meminimalisir kerusakan jika akun sudah terlanjur terkompromi.

- [1] L. Conklin, "Threat Modeling Process," OWASP Foundation, n.d. [Online]. Available: https://owasp.org/www-community/Threat_Modeling_Process
- [2] "OWASP Top 10," OWASP Foundation, n.d. [Online]. Available: <https://owasp.org/Top10>
- [3] "OWASP in SDLC," OWASP Foundation, n.d. [Online]. Available: https://owasp.org/www-project-integration-standards/writeups/owasp_in_sdlc.
- [4] E. Kost, "What Caused the Uber Data Breach in 2022?," UpGuard. Nov 18 2024 [Online]. Available: <https://www.upguard.com/blog/what-caused-the-uber-data-breach>
- [5] "The Uber Breach Case Study: Cybersecurity Lessons Learned," HumanFirewall, [Online]. Available: <https://humanfirewall.io/the-uber-breach-case-study>
- [6] M. Jackson, "Uber Breach 2022 – Everything You Need to Know," GitGuardian Blog, Sep. 16, 2022. [Online]. Available:

V. Miscellaneous



If you don't do these, he'll haunt your dreams tonight

1. (10.0 poin - WAJIB)

a. Landasan Etika:

Sebagai salah satu mahasiswa, terlebih lagi mahasiswa yang terpilih menjadi asisten akademik. Sudah menjadi kewajiban ku untuk memastikan mahasiswa lain dapat mengikuti kegiatan akademik dengan baik dan tanpa dirugikan oleh rekan-rekannya yang melakukan tindak kecurangan.

Landasan Hukum:

Peraturan akademik ITB pasal 35 tentang Pengawas Ujian poin 2, yang berbunyi: "Pengawas ujian wajib melaporkan tindak kecurangan peserta ujian dalam Berita Acara Pelaksanaan Ujian."

Pasal 3 Undang-Undang Nomor 12 Tahun 2012 tentang Pendidikan Tinggi, yang mengatakan bahwa pendidikan tinggi berasaskan 8 hal yang diantaranya adalah kejujuran, keadilan serta tanggung jawab.

b. 3 alat untuk mendeteksi kecurangan:

- Plagiarism detector
- Common sense, analisis sendiri dan mengajak diskusi bersama asisten lain
- AI untuk analisis.

c. Jika ada jawaban yang sangat mirip dengan jawaban orang lain sedangkan jawaban tersebut tidak umum dan subjek tidak dapat menjelaskannya. Dan jika pekerjaan, dalam bentuk kode atau lainnya, dapat ditemukan di internet sedangkan tidak ada *mention* atau *credit* terhadap konten tersebut, serta subjek tidak dapat menjelaskan dengan baik tentang hasil pekerjaan tersebut.

d. Studi Kasus

- i. Mendiskusikan hal tersebut dengan anggota kelompok dan meminta mereka menjelaskannya. Jika penjelasan mereka tidak masuk akal, maka akan saya kurangi nilainya tergantung seberapa "besar" kode yang diambil tersebut.
 - ii. Jika penggunaan kode sumber eksternal tidak diperbolehkan, akan saya lakukan seperti poin (i). Namun jika diperbolehkan, akan ditindaklanjuti dengan mengecek kesesuaian penggunaan kode dengan *license* yang digunakan kode tersebut, serta menilai pemahaman mereka terhadap kode yang tugas besar mereka.
 - iii. Akan saya beri peringatan sekali saja dan mencatat nama dan nim mereka. Jika masih melakukan, maka akan dilaporkan ke dosen
 - iv. Akan saya nilai seberapa mirip jawaban mereka dan mengajak asisten lain menilainya. Bisa saja jawaban yang tidak standar tersebut hasil diskusi atau belajar bareng mereka sebelum ujian. Namun jika terlalu mirip maka akan dilaporkan ke dosen.
 - v. Penyusunan kalimat/*style yapping* tiap orang pasti beda walaupun materi atau tingkat pemahaman sama. Jadi akan diselidiki lebih lanjut seberapa mirip jawaban mereka.
 - vi. Jika diskusi tidak diperbolehkan pada tugas tersebut, akan saya laporkan ke dosen. Namun jika diperbolehkan, mungkin saja terjadi salah submit, sehingga akan diberikan kesempatan kedua.
- e. Iya, karena menurut saya merupakan kewajiban untuk asisten menjaga integritas akademik, sangat tidak profesional jika hanya menjaga reputasi personal ketimbang mendahulukan hal krusial seperti itu. Selain itu, saya rasa kalo reputasi rusak karena hal seperti ini, ga bakal terlalu parah rusaknya, jadi aman aja, kalo orang lain sampai setidak suka itu berarti aneh aja mereka.