DETEKSI SERANGAN MALWARE PADA CLOUD SERVER MENGGUNAKAN METODE ANOMALY BASED

SKRIPSI

Oleh:
NAUFAL ANDRIANTO NURFAUZI
NIM. 15650052

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2022

DETEKSI SERANGAN MALWARE PADA CLOUD SERVER MENGGUNAKAN METODE ANOMALY BASED

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
NAUFAL ANDRIANTO NURFAUZI
NIM. 15650052

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2022

HALAMAN PERSETUJUAN

DETEKSI SERANGAN *MALWARE* PADA *CLOUD SERVER* MENGGUNAKAN METODE *ANOMALY BASED*

SKRIPSI

Oleh :
NAUFAL ANDRIANTO NURFAUZI
NIM. 15650052

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 17 Juni 2022

Pembimbing I,	Pembimbing II,
Dr. M. Amin Hariyadi, M.T	Fresy Nugroho, M. T
NIP. 19670018 200501 1 001	NIP. 19710722 201101 1 001

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan S.T., M.MT., IPM
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

DETEKSI SERANGAN *MALWARE* PADA *CLOUD SERVER* MENGGUNAKAN METODE *ANOMALY BASED*

SKRIPSI

Oleh :
NAUFAL ANDRIANTO NURFAUZI
NIM. 15650052

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal : 17 Juni 2022

Susunan Dewan Penguji

Penguji Utama	:	Dr. Cahyo Crysdian, MCS NIP. 19740424 200901 1 008	( )
Ketua Penguji	:	A'la Syauqi,, M.Kom NIP. 19771201 200801 1 007	( )
Sekretaris Penguji	:	Dr. M. Amin Hariyadi, M.T NIP. 19670018 200501 1 001	( )
Anggota Penguji	:	Fresy Nugroho, M. T NIP. 19710722 201101 1 001	( )

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan ST., M.MT ., IPM
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Naufal Andrianto Nurfauzi
NIM	: 15650052
Fakultas / Jurusan	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: **Deteksi Serangan *Malware* pada *Cloud Server* Skripsi Menggunakan Metode *Anomaly Based***

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 17 Juni 2022
Yang membuat pernyataan,

Naufal Andrianto Nurfauzi
NIM.15650052

MOTTO

“Hidup adalah harapan, maka jagalah harapanmu jika ingin terus hidup.**”**

HALAMAN PERSEMBAHAN

Karya ilmiah ini penulis persembahkan untuk ibu, adik, dan istri saya. Karena dengan cinta, kasih sayang, motivasi, perjuangan serta perngorbanan mereka, dan tidak lupa do'a yang selalu mereka panjatkan menjadikan penulis mampu menyelesaikan karya ilmiah ini.

Terima kasih pula kepada seluruh dosen beserta seluruh staff Teknik Informatika yang membantu dengan sepenuh hati. Khususnya Bapak M. Amin Hariyadi selaku pembimbing pertama, Bapak Fresy Nugroho selaku pembimbing kedua, Bapak Cahyo Crysdian selaku penguji utama, dan Bapak A'la Syauqi selaku ketua penguji, Bapak Fachrul Kurniawan selaku Ketua Jurusan, Bapak Yunifa Miftachul Arif. Dan juga terima kasih khususnya kepada Bu Nia selaku staf administrasi jurusan Teknik Informatika yang selalu membantu kelancaran administrasi penulis.

Tidak lupa terima kasih penulis sampaikan kepada teman-teman yang juga ikut membantu, serta orang-orang yang tidak dapat disebutkan satu per satu yang juga ikut membantu sehingga karya ilmiah ini dapat terselesaikan.

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Puji syukur penulis limpahkan ke hadirat Allah SWT atas limpahan rahmt dan karunia-Nya sehingga penulis dapat menyelesaikan studi dan skripsi. Shalawat serta salam kepada Nabi Muhammad SAW, yang telah mempin umat manusia dari jalan yang gelap menuju jalan yang terang benderang.

Penulis mendapatkan banyak sekali dukungan dari banyak pihak selama proses pengerjaan studi dan lebih khususnya pada proses penyusunan karya ilmiah skripsi ini. Penulis haturkan banyak terimakasih kepada:

1. Ibu, istri dan adik dari penulis yang telah memberikan dukungan baik secara moral hingga material sehingga penulis dapat menyelesaikan skripsi ini.
2. Prof. DR. H. M. Zainuddin, MA selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang beserta jajarannya.
3. Dr. Sri Harini, M.Si, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang beserta jajarannya.
4. Dr. Fachrul Kurniawan ST., M.MT ., IPM, selaku Kepala Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang beserta jajarannya.
5. Dr. M. Amin Hariyadi, M.T, selaku Dosen Pembimbing I yang telah

memberikan ilmu dalam proses pengarahan hingga penyelesaian skripsi ini.

6. Fresy Nugroho, M.T, selaku Dosen Pembimbing II yang telah memberikan ilmu dalam proses pengarahan hingga penyelesaian skripsi ini.
7. Sahabat seperjuangan di Jurusan Teknik Informatika angkatan 2015 yang telah membantu untuk bertukar pikiran dan telah memberikan dukungannya kepada penulis dalam proses penyelesaian skripsi ini.
8. Seluruh keluarga besar Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
9. Seluruh teman-teman yang telah memberikan support dan semangat kepada penulis.
10. Pak Hertanto dan rekan-rekan kerja GL++ yang telah mendukung dan membantu kelancaran dalam menyelesaikan skripsi.

Penulis menyadari bahwa dalam penyusunan karya ilmiah skripsi ini masih terdapat kekurangan dan penulis berharap semoga skripsi ini dapat memberikan manfaat termasuk penulis sendiri.

Malang, 17 Juni 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Nurfauzi, Naufal Andrianto. 2022. **Deteksi Serangan Malware Pada Cloud Server Menggunakan Metode Anomaly Based**. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. M. Amin Hariyadi, M.T, (II) Fresy Nugroho, M.T

Kata Kunci: Anomaly Based, Isolation Forest, Prediksi.

Malware merupakan salah satu jenis serangan yang paling umum dan berbahaya di internet dengan berbagai variasi dan jenisnya. Dengan masifnya perkembangan malware diperlukan sistem deteksi dengan metode yang efektif dan akurat untuk mendeteksi dan mengantisipasi serangan yang datang. Oleh karena itu, dalam penelitian ini bertujuan untuk melakukan implementasi pendekatan terbaru dalam mendeteksi serangan malware pada cloud server dengan tingkat akurasi serta efisiensi sumber daya yang baik. Metode yang diusulkan adalah Anomaly Based yang didukung dengan Isolation Forest sebagai model sistem pakar sehingga mampu meningkatkan akurasi dan efisiensi sumber daya dalam untuk mempercepat waktu komputasi. Data yang digunakan yaitu dataset MTA-KDD'19 oleh Ivan Letteri et al (2020), yang dibagi untuk data latih dan data uji. Proses uji menggunakan 5 fitur, dengan memperoleh tingkat akurasi sebesar 46,67%, presisi 93%, recall 47,05%, dan nilai f-measure 62%. Dari hasil percobaan tersebut, dapat disimpulkan bahwa Anomaly Based dengan model Isolation Forest memiliki kemampuan deteksi yang kurang baik terhadap serangan malware pada cloud server.

ABSTRACT

Nurfauzi, Naufal Andrianto. 2022. **Detecting Malware Attacks on Cloud Servers Using Anomaly Based Method**. Undergraduate Thesis. Department of Informatics Engineering Faculty of Science and Technology Maulana Malik Ibrahim State Islamic University Malang. Supervisor: (I) Dr. M. Amin Hariyadi, M.T, (II) Fresy Nugroho, M.T

Keywords: Anomaly Based, Isolation Forest, Prediction.

Malware is one of the most common and dangerous types of attacks on the internet, with various variations and types. With the massive development of malware, a detection system with effective and accurate methods is needed to detect and anticipate incoming attacks. Therefore, this study aims to implement the latest approach to seeing malware attacks on cloud servers with good accuracy and resource efficiency. The proposed method is Anomaly Based, which is supported by Isolation Forest as an expert system model to increase the accuracy and efficiency of internal resources to speed up computing time. The data used is the MTA-KDD'19 dataset by Ivan Letteri et al. (2020), divided into training and test data. The test process uses five features, obtaining an accuracy rate of 46.67%, 93% precision, 47.05% recall, and 62% f-measure value. From the results of these experiments, it can be concluded that Anomaly Based with the Isolation Forest model has poor detection capabilities against malware attacks on cloud servers.

الملخص

نور فوزي، نوفل أندريانتو. 2022م. **اكتشاف هجوم البرمجيات الخبيثة على الخادم السحابي باستخدام الطريقة المستندة إلى الشذوذ.** البحث العلمي. قسم الهندسة المعلوماتية كلية العلوم والتكنولوجيا جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. تحت إشراف: () الدكتور محمد أمين هاريادي، الماجستير، () فرسي نوغراهو، الماجستير.

الكلمات الرئيسية: المستند إلى الشذوذ، غابة العزلة، التنبؤ

تعد البرمجية الخبيثة نوعا من أنواع الهجوم الأكثر شيوخا وخطيرة على الإنترنت مع كل تنوعها. وبسبب سرعة تطور البرمجيات الخبيثة، فهناك حاجة إلى نظام الكشف بالمنهج الفعالي والصحيح لاكتشاف وتوقع الهجوم القادمة. ولذلك، يهدف هذا البحث إلى تطبيق المدخل الجديد لاكتشاف هجوم البرمجيات الخبيثة على الخادم السحابي مع درجة الدقة وكفاءة الموارد الجيدة. والطريقة المقترحة هي الطريقة المستندة إلى الشذوذ تدعمها غابة العزلة كنموذج نظام الخبراء لكي تقدر على تحسين الدقة وكفاءة الموارد لتسريع وقت الحوسبة. والبيانات المستخدمة هي مجموعة بيانات MTA-KDD'19 لإيفان لتري و وآخرين (2020م) التي تنقسم إلى بيانات التدريب وبيانات الاختبار. استخدمت عملية الاختبار 5 وظائف وحصلت على درجة الدقة 46،67%، والدقة 93%، والاستدعاء 47،05%، وقيمة قياس-ف 62%. استنادا إلى هذه نتائج الاختبار، ففي الختام، إن الطريقة المستندة إلى الشذوذ مع نموذج غابة العزلة لها قدرة اكتشاف هجوم البرمجيات الخبيثة غير جيدة على الخادم السحابي.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Jaringan internet dibangun pada tahun 1969 oleh Departemen Pertahanan Amerika Serikat yang bekerja sama dengan beberapa laboratorium ilmu komputer di Amerika Serikat. Pada mulanya jaringan internet ini bernama Advanced Research Projects Agency Network (ARPANET), dimana saat itu ARPANET dapat menghubungakn antara dua node dimana satu node yang ada di University of California, Los Angeles dapat mengirim pesan ke node yang ada di Stanford Research Institute (Leiner dkk, 1997). Kemunculan istilah internet sendiri dimulai pada tahun 1990 dimana muncul pengganti dari ARPANET yang sistemnya komersil yaitu Internet Service Porvider (ISP). Melalui ISP ini lah pada akhirnya berkembang jaringan Internet yang jauh meluas dari yang pada awalnya hanya nasional hingga menjadi internasional dan mencakup keseluruhan negara walaupun pada awalnya kualitas internetnya masih lambat. Semakin tahun dengan perkembangan sumber daya manusia dan sumber daya infrastruktur yang semakin optimal dalam membangun jaringan internet, menjadikan internet sebagai salah satu kebutuhan utama manusia terhadap komunikasi dan informasi, sehingga muncullah era dimana yang dinamakan era digital pada awal abad 21 ini.

Negara Indonesia sendiri menjadi salah satu negara yang berkembang dan sangat ramai dalam hal perkembangan tekonologi. Dan juga Indonesia menjadi negara yang mendapatkan serangan siber dengan intensitas yang lumayan tinggi bahkan dibanding negara-negara di eropa dan amerika. Di tahun 2012-2014 saja

Indonesia menjadi Top 5 peringkat serangan siber dunia (Lim, 2016). Dalam data yang didapat oleh Kominfo menggunakan sensor public honeynet di 21 titik di Indonesia menunjukkan dalam 1 bulan dapat terjadi sekitar 276.000 serangan dari 5 negara penyerang Indonesia terbanyak (Kominfo, 2018). Kebanyakan dari serangan siber yang mengarah ke Indonesia adalah serangan malware.

Malware telah menjadi ancaman keamanan informasi bagi individu maupun organisasi, dimana setiap waktunya malware menjadi semakin meningkat baik secara kuantitas maupun kualitas kecanggihan dalam menyerang sistem komputer. Apalagi dengan semakin beragamnya perangkat komputer seperti tablet, smartphone, hingga IoT (Internet of Things) membuat malware semakin bervariasi juga. Serangan malware sangatlah mudah menyebar dalam kehidupan sehari-hari kita, dimana banyak sekali jalur maupun cara yang digunakan orang-orang yang tidak bertanggung jawab dalam menyebarkan malware yang tanpa kita sadari media, aplikasi, bahkan sistem yang kita gunakan setiap harinya telah dipasang malware di dalamnya. Contohnya saja adalah melalui email, email menjadi media penyebaran malware yang cukup sering ditemui dalam kehidupan sehari-hari manusia. Seringkali dijumpai email-email spam maupun dalam kotak masuk yang berisi email dari orang yang tidak diketahui dan dicurigai mengandung malware. Biasanya malware disebarkan dalam bentuk link ataupun data yang diberikan oleh pengirim, ketika link dan data tersebut diakses oleh pemilik email tersebut, secara tidak langsung akan mengaktifkan malware yang telah dikirim oleh penyerang. Selain itu ada juga serangan yang berupa kiriman di media sosial yang sangat sering dijumpai, contohnya di media sosial Facebook

yang paling sering dijumpai penyebaran malware berupa link-link yang merugikan orang banyak. Ada juga serangan yang mengarah ke server baik melalui jaringan lokal maupun serangan dari luar untuk menyebarkan malware. Semua serangan tersebut bertujuan untuk mencuri data yang bukan miliknya untuk dimanfaatkan bagi keuntungan para penyerang.

Allah SWT telah berfirman dalam al Qur'an surat Al Baqarah ayat 188 yang menjelaskan bahwasanya

وَلَا تَأْكُلُوٓا اَمْوَالَكُمْ بَيْنَكُمْ بِالْبَاطِلِ وَتُدْلُوْا بِهَآ اِلَى الْحُكَّامِ لِتَأْكُلُوْا فَرِيْقًا مِّنْ اَمْوَالِ النَّاسِ بِالْاِثْمِ وَاَنْتُمْ تَعْلَمُوْنَ

Artinya: "Dan janganlah sebahagian kamu memakan harta sebahagian yang lain di antara kamu dengan jalan yang bathil dan (janganlah) kamu membawa (urusan) harta itu kepada hakim, supaya kamu dapat memakan sebahagian daripada harta benda orang lain itu dengan (jalan berbuat) dosa, Padahal kamu *mengetahui"*.

Berdasarkan tafsir Jalalain pada ayat tersebut dijelaskan bahwa larangan untuk orang-orang yang memakan harta orang-orang yang lain melalui jalan yang batil, yang adalah jalan haram menurut syariat islam, seperti mencuri milik orang lain, melakukan intimidasi dan sebagainya. Dan larangan untuk membawa urusan harta tersebut ke pengadilan dengan melakukan penyuapan terhadap hakin untuk mempermudah dan meloloskan diri dari hukuman. Sedangkan menurut tafsir Ibnu Katsir yang mana beliau menafsirkan bahwasanya tidak ada yang bisa merubah hukum sesuatu yang haram menjadi halal ataupun halal menjadi haram, namun hakim memiliki ikatan terhadap apa yang tampak darinya, dan siapapun yang melakukan tipu muslihat akan mendapatkan dosa. Dari beberapa tafsir tersebut

dapat dikatakan bahwa bagaimanapun juga memanfaatkan hak orang lain demi keuntungan sendiri adalah hal yang tidak diperbolehkan.

Maka untuk mempertahankan data-data penting dari serangan malware, sistem keamanan juga semakin ditingkatkan kecanggihannya serta mengembangkan metode-metode keamanan yang baru. Agar tidak terjadi serangan-serangan yang merugikan orang banyak seperti ransomware wannacry yang menyerang hampir di seluruh belahan dunia. Dalam mengamankan sistem dari serangan yang berbahaya terutama malware, dapat dilakukan pendeteksian dengan memanfaatkan proses log jaringan pada server. Log jaringan memiliki banyak parameter yang dapat digunakan untuk mendeteksi perilaku anomali yang menjadi dasar bahwa proses tersebut dicurigai sebagai malware.

Berdasarkan pemaparan pada latar belakang tersebut, peneliti berusaha untuk mengimplementasikan metode anomaly based dalam intrusion detection system pada cloud server sebagai pendeteksi malware yang akan melakukan deteksi malware berdasarkan anomali dari log jaringan server.

1.2 Pernyataan Masalah

1. Berapakah tingkat akurasi deteksi serangan malware dengan metode anomaly based?
2. Apakah efektif dan efisiens penggunaan sumber daya sistem dalam proses deteksi serangan malware?

1.3 Tujuan Penelitian

1. Memahami tingkat akurasi deteksi serangan malware dengan metode anomaly based.

2. Mengetahui efektifitas dan efisiensi penggunaan sumber daya sistem dalam proses deteksi serangan malware.

1.4 Manfaat Penelitian

1. Adapun manfaat yang dapat diperoleh dari penelitian ini bagi system administrator atau security engineer adalah dapat digunakan sebagai bahan peritmbangan dalam peningkatan keamanan sistem cloud dari serangan berbagai bentuk malware.
2. Manfaat bagi peneliti adalah malware yang terdeteksi dapat digunakan sebagai bahan perhitungan akurasi deteksi, dan juga dalam prosesnya dapat diketahui tingkat efektifitas dan efisiensi dalam penggunaan sumber daya sistem.

1.5 Batasan masalah

Agar permasalahan yang diteliti lebih terfokuskan, serta mencegahnya malware yang menyebar melalui penelitian ini maka:

1. Implementasi akan dilakukan menggunakan dataset MTA-KDD'19.
2. FItur lalu lintas jaringan pada dataset yang akan digunakan adalah TCP RST FLAG.
3. Penentuan anomali menggunakan algoritma Isolation Forest.

1.6 Sistematika Penulisan

Laporan dari penelitian ini mencakup atas lima bab, lima bab tersebut terdiri dari :

BAB I : PENDAHULUAN

Memuat mengenai latar belakang dari permasalahan yang akan diteliti, tujuan dan keguaan atau manfaat penelitian dari penelitian, batasan yang mempersempit bahasan masalah pada penelitian, metodologi dalam melakukan penelitian, serta sistematika penulisan dari laporan penelitian ini.

BAB I I: STUDI PUSTAKA

Pada bab ini mencakup penjelasan terkait penelitian terdahulu yang pernah dilakukan maupun teori dasar dan data-data yang berkaitan dengan malware, anomaly based detection, dan isolation forest sebagai model dari metode deteksi anomaly based malware.

BAB III : METODE PENELITIAN

Bab ini mencakup mengenai prosedur dari penelitian dan rancangan sistem untuk deteksi malware dengan metode anomaly based.

BAB IV : HASIL DAN PEMBAHASAN

Bab ini mencakup skenario uji coba beserta hasil uji coba dari sistem deteksi dengan anomaly based yang ada pada cloud server. Deteksi malware akan dilakukan dengan menerapkan model isolation forest terhadap data yang didapatkan, yang nantinya akan dihitung akurasi deteksi dan juga efisiensi penggunaan sumber daya dalam proses deteksi.

BAB V : PENUTUP

Dalam bab ini mencakup penjelasan terhadap kesimpulan dari proses penelitian yang telah dilakukan beserta saran dan kritik dari peneliti terkait penelitian ini agar dapat dikembangkan dengan lebih baik pada penelitian selanjutnya.

BAB II

STUDI PUSTAKA

2.1 Dasar Teori

2.1.1 Cloud Computing

Pada tahun 2011, US National Institute of Standards and Technology (NIST), mendefinisikan komputasi awan sebagai model yang memungkinkan akses jaringan sesuai permintaan dari mana saja dan di mana saja dan juga nyaman, menuju ke kumpulan sumber daya komputasi yang dapat dikonfigurasi (misalnya, jaringan, server, penyimpanan, aplikasi , layanan, dll.) yang dapat dengan cepat disediakan dan diberikan dengan meminimalisir beban manajemen atau interaksi penyedia layanan. Cloud computing memiliki ciri lima atribut yang menjadi identifikasi khusus, yaitu:

a. Layanan mandiri sesuai permintaan.
b. Akses jaringan luas.
c. Pengumpulan sumber daya.
d. Elastisitas cepat.
e. Layanan terukur.

Cloud computing selain memiliki atribut ciri khusus, juga umumnya dibagi atas dasar fokus layanannya, yaitu:

a. DBaaS, Database as a Service berfokus pada database sebagai layanan tambahan yang lebih baru untuk tiga model layanan cloud yang lain.
b. SaaS, Software as a Service, ayanan yang diberikan kepada pengguna untuk menggunakan aplikasi milik penyedia layanan yang dapat beroperasi pada

infrastruktur cloud. Aplikasi dapat diakses dari berbagai jenis perangkat pengguna melalui antarmuka seperti web (seperti email berbasis web). Pengguna tidak mengelola atau mengendalikan infrastruktur cloud yang mendasar termasuk jaringan, server, sistem operasi, penyimpanan, atau pada tingkat develop aplikasi itu sendiri, dengan kemungkinan pengecualian yang terbatas terhadap beberapa pengaturan dan konfigurasi pada aplikasi pengguna tertentu. Seperti pada Google Apps, SalesForce dan aplikasi jejaring sosial seperti Facebook dan Twitter

c. PaaS, Platform as a Service, layanan yang diberikan kepada pengguna untuk beerbagi aplikasi yang dibuat oleh pengguna atau diperoleh pengguna ke infrastruktur cloud computing menggunakan bahasa pemrograman dan peralatan yang didukung oleh provider. Pengguna tidak mengelola atau mengendalikan infrastruktur cloud yang mendasar termasuk jaringan, server, sistem operasi, atau penyimpanan, namun memiliki kontrol atas aplikasi didevelop dan disebarkan yang memungkinkan aplikasi dapat melakukan konfigurasi hosting. Contohnya yang sudah mengimplementasikan ini adalah Force.com dan Microsoft Azure Investment.

d. IaaS, Infrastructure as a Service, layanan yang diberikan kepada pengguna untuk memproses, menyimpan, berjejaring, serta sumber komputasi penting yang lainnya, dimana pengguna dapat dengan mudah menyebarkan dan menjalankan perangkat lunak secara bebas, yang dapat mencakup pada keseluruhan sistem operasi dan juga aplikasi. Pengguna tidak mengelola atau mengendalikan infrastruktur cloud yang mendasar tetapi memiliki kontrol

atas sistem operasi, penyimpanan, serta aplikasi yang disebarkan, dan juga memungkinkan kontrol terbatas terhadap komponen jaringan yang dipilih dan diizinkan oleh penyedia layanan (seperti firewall host). Contohnya seperti Amazon Elastic Compute Cloud dan Simple Storage Service.

Era Cloud computing dimulai pada tahun 2006 ketika Amazon menawarkan Elastic Cloud Computing (EC2) dan Simple Storage Service (S3), layanan awal yang disediakan oleh Amazon Web Services (AWS). Lima tahun kemudian, pada 2012, EC2 digunakan oleh bisnis di 200 negara. S3 telah melampaui dua triliun objek dan secara rutin menjalankan lebih dari 1,1 juta permintaan puncak per detik. The Elastic Map Reduce telah meluncurkan 5,5 juta cluster sejak dimulainya layanan pada Mei 2010 (ZDNet 2013). Jangkauan layanan yang ditawarkan oleh Cloud Service Provider (CSP), dan jumlah pengguna cloud telah meningkat secara dramatis selama beberapa tahun terakhir.

Gerakan cloud computing diinisiasi oleh gagasan bahwa pemrosesan dan penyimpanan data dapat dilakukan dengan lebih efisien pada sistem komputasi, dan juga penyimpanan dengan ukuran besar yang dapat diakses melalui jaringan internet. Computer clouds mendukung perubahan dan pergeseran paradigma dari komputasi lokal ke network centric dan jaringan content centric, di mana pusat data yang jauh menyediakan sumber daya komputasi dan penyimpanan. Dalam paradigma baru ini, pengguna melepaskan kendali atas data dan kode mereka kepada Cloud Service Provider.

Cloud computing menawarkan komputasi dan layanan penyimpanan yang tehitung dan fleksibel. Sumber daya yang digunakan untuk layanan ini dapat

dihitung dan pengguna hanya dapat dikenakan biaya untuk sumber daya yang mereka gunakan. Cloud computing adalah realitas bisnis karena sejumlah besar organisasi telah mengadopsi paradigma ini.

2.1.2 Konsep Kemanan Sistem Komputer

Pada dasarnya sistem komputer sama seperti sistem real seperti sistem administratif ataupun sistem dengan prosedur-prosedur lainnya yang juga memperhatikan sisi keamanan. Jika pada sistem komputer bagian autentikasi, autorisasi, dan nonrepudiation termasuk dalam alat-alat yang dapat digunakan oleh developer maupun analis sistem komputer untuk menjaga keamanan sistem komputer yang masih sehubungan dengan kerahasiaan informasi, integritas data serta sistem, dan ketersediaan informasi. Ada 6 konsep yang menjadi dasar pondasi dalam sistem keamanan komputer, yaitu Autentikasi, Autorisasi, Nonrepudiation, Kerahasiaan, Integritas, dan Ketersediaan. Memahami masing-masing dari enam konsep ini dan bagaimana masing-masing komponen konsep saling berhubungan satu sama lain, dapat membantu profesional keamanan merancang dan mengimplementasikan sistem yang aman. Dan setiap komponen sangat penting dalam membangun sistem keamanan secara keseluruhan, bilamana ada kegagalan pada salah satu kompone, maka akan mengakibatkan potensi kesalah pahaman dan tidak sinkronisasinya sistem.

Ada tiga konsep kunci, yang dikenal sebagai triad CIA, yang harus dipahami oleh siapa pun yang sedang melakukan pengamanan terhadap sistem informasi, yaitu kerahasiaan, integritas, dan ketersediaan. Setiap profesional keamanan sistem informasi memiliki dedikasi untuk memastikan perlindungan secara

prinsipal ini untuk setiap sistem yang mereka lindungi. Selain itu, ada tiga konsep kunci yang harus dipahami oleh para profesional keamanan untuk menegakkan prinsip-prinsip CIA dengan benar, yaitu autentikasi, autorisasi, dan nonrepudiasi. Pada bagian ini, dijelaskan terkait masing-masing konsep ini dan bagaimana mereka saling berhubungan satu dengan yang lain dalam bidang keamanan informasi komputer. Semua definisi yang digunakan dalam bagian ini berasal dari National Information Assurance Glossary (NIAG) yang diterbitkan oleh Komite Amerika Serikat untuk Sistem Keamanan Nasional.

a. Autentikasi

Autentikasi menjadi komponen penting untuk setiap sistem yang menginginkan peningkatan keamanan, karena autentikasi merupakan kunci untuk memverifikasi sumber dari pesan yang berada dalam lalu lintas jaringan atau mengenai keterkaitan seseorang yang mengklaim pesan tersebut. NIAG mendefinisikan autentikasi sebagai ukuran keamanan yang dirancang untuk menetapkan validitas transmisi, pesan, atau pengguna asli, atau sarana untuk memverifikasi autorisasi individu untuk menerima kategori informasi yang telah ditentukan. Ada banyak metode yang tersedia untuk melakukan autentikasi pada seseorang. Dalam setiap metode, autentikator mengeluarkan tantangan atau tugas yang harus dikerjakan dan dijawab dengan benar oleh seseorang. Tantangan dan tugas ini biasanya terdiri dari permintaan terhadap potongan-potongan informasi yang hanya dapat diberikan dan diketahui pasti oleh pengguna asli. Potongan-potongan informasi ini termasuk dalam tiga klasifikasi yang dikenal sebagai faktor autentikasi. Ketiga faktor tersebut adalah

1) Sesuatu yang anda ketahui

   Informasi yang diasumsikan oleh sistem sebagai informasi yang tidak diketahui oleh orang lain; informasi ini mungkin tergolong rahasia, seperti kata sandi atau kode PIN, atau hanya potongan informasi kecil yang kebanyakan orang tidak mengetahuinya, seperti nama gadis ibu pengguna, nama hewan peliharaan pertama kali, atau boneka yang pertama kali dibeli.

2) Sesuatu yang anda miliki

   Sesuatu yang dimiliki pengguna yang hanya dia saja yang memiliki seperti Radio Frequency Identity (RFID) badge, One Time Password (OTP) yang menghasilkan token dengan batas waktu atau temporary, kunci fisik berupa kartu, atau kuncu yang sesungguhnya.

3) Sesuatu yang mencerminkan diri anda

   Informasi yang sangat menunjukkan identitas anda dan sangat identik, tidak dimiliki orang lain, seperti sidik jari, keluaran suara, atau pemindaian retina mata, yang mana faktor ini keamanan ini dikenal sebagai keamanan biometrik.

Ketika sistem autentikasi memerlukan lebih dari satu faktor tersebut, komunitas keamanan mengklasifikasikannya sebagai sistem yang membutuhkan autentikasi multifaktor. Dua contoh dari faktor yang sama, seperti kata sandi yang digabungkan dengan nama gadis ibu pengguna, bukanlah autentikasi multifaktor karena masih memiliki jenis faktor yang sama, tetapi menggabungkan pemindaian sidik jari dan nomor identifikasi pribadi (PIN) adalah termasuk autentikasi multifaktor, karena memvalidasi sesuatu yang dimiliki pengguna (pemilik sidik jari tersebut) dan sesuatu yang diketahui pengguna (PIN).

Autentikasi juga berlaku untuk melakukan validasi terhadap sumber pesan, seperti pada paket jaringan atau email. Pada tingkatan yang rendah, sistem autentikasi pesan tidak dapat menggunakan sistem faktor yang sama yang berlaku untuk autentikasi manusia. Sistem autentikasi pesan sering menggunakan tanda tangan digital seperti kriptografi, yang terdiri dari inti atau hash dari pesan yang dihasilkan dengan meggunakan kunci rahasia. Karena hanya satu orang yang memiliki akses ke kunci yang menghasilkan tanda tangan, penerima dapat memvalidasi pengirim pesan. Tanpa sistem autentikasi yang baik, tidak mungkin untuk mempercayai bahwa pengguna adalah seseorang yang benar seperti yang dia katakan, atau bahwa pesan berasal dari siapa yang mengklaimnya benar.

b. Autorisasi

Sementara autentikasi berkaitan dengan verifikasi identitas, autorisasi berfokus pada penentuan izin apa yang dimiliki pengguna untuk mengakses program atau proses. NIAG mendefinisikan autorisasi sebagai hak akses yang diberikan kepada pengguna, program, atau proses. Setelah sistem yang aman mengautentikasi pengguna, sistem juga harus memutuskan hak istimewa seperti apa yang mereka miliki. Misalnya, aplikasi perbankan online akan mengautentikasi pengguna berdasarkan kredensialnya, tetapi kemudian harus menentukan akun yang dapat diakses pengguna tersebut. Selain itu, sistem harus menentukan tindakan apa yang dapat dilakukan pengguna terkait akun tersebut, seperti melihat saldo dan melakukan transfer.

c. Nonrepudiation

Bayangkan sebuah skenario di mana seorang pembeli membeli mobil dari seorang pedagang dan kemudian menandatangani kontrak yang menyatakan bahwa dia akan membayar sejumlah uang tunai untuk mobil tersebut dan akan mengambil alih kepemilikannya pada hari Kamis. Jika pembeli tersebut kemudian memutuskan untuk tidak membeli mobil, dia mungkin mengklaim bahwa seseorang memalsukan tanda tangannya dan bahwa dia tidak bertanggung jawab atas kontrak tersebut. Untuk membantah klaimnya, sang penjual dapat menunjukkan bahwa notaris memverifikasi identitas pembeli dan mencap dokumen untuk menunjukkan verifikasi ini. Dalam hal ini, stempel notaris telah memberikan kontrak properti nonrepudiation, yang didefinisikan oleh NIAG sebagai kepastian pengirim data, diberikan bukti pengirim dan penerima dengan bukti identitas pengirim, sehingga tidak dapat kemudian menyangkal telah memproses data.

Dalam dunia komunikasi digital, tidak ada notaris yang dapat mencap setiap pesan yang dikirimkan, tetapi nonrepudiation tetap diperlukan. Untuk memenuhi persyaratan ini, sistem yang aman biasanya mengandalkan kriptografi asimetris (atau kunci publik). Sementara sistem kunci simetris menggunakan satu kunci untuk mengenkripsi dan mendekripsi data, sistem asimetris menggunakan pasangan kunci. Sistem ini menggunakan satu kunci (pribadi) untuk menandatangani data dan menggunakan kunci lainnya (publik) untuk memverifikasi data. Jika kunci yang sama dapat menandatangani dan memverifikasi konten pesan, pengirim dapat mengklaim bahwa siapa pun yang memiliki akses ke kunci tersebut dapat dengan mudah memalsukannya. Sistem

kunci asimetris memiliki properti nonrepudiation karena penandatangan pesan dapat merahasiakan kunci pribadinya.

d. Kerahasiaan

Istilah kerahasiaan sudah tidak asing lagi bagi kebanyakan orang, sekalipun mereka yang tidak berkecimpung dalam industri keamanan. NIAG mendefinisikan kerahasiaan sebagai jaminan bahwa informasi tidak diungkapkan kepada individu, proses, atau perangkat yang tidak berwenang.Memastikan bahwa pihak yang tidak berwenang tidak memiliki akses ke suatu informasi adalah tugas yang kompleks dan tidak mudah. Hal ini bisa lebih mudah untuk dipahami ketika dipecah menjadi tiga langkah utama.

Pertama, informasi harus memiliki perlindungan yang mampu mencegah beberapa pengguna mengaksesnya. Kedua, pembatasan harus diterapkan untuk membatasi akses ke informasi hanya bagi mereka yang memiliki autorisasi untuk melihatnya. Ketiga, sistem autentikasi harus ada untuk memverifikasi identitas mereka yang memiliki akses kepada data.

Autentikasi dan autorisasi, yang dijelaskan sebelumnya di bagian ini, sangat penting untuk diterapkan sebagai upaya menjaga kerahasiaan, tetapi konsep kerahasiaan terutama berfokus pada penyembunyian atau perlindungan informasi. Salah satu cara untuk melindungi informasi adalah dengan menyimpannya di lokasi pribadi atau di jaringan pribadi yang dibatasi hanya untuk mereka yang memiliki akses sah ke informasi tersebut. Jika suatu sistem harus mengirimkan data melalui jaringan publik, organisasi harus menggunakan kunci yang hanya diketahui oleh pihak yang berwenang untuk mengenkripsi data. Untuk informasi

yang berjalan melalui Internet, perlindungan ini dapat berarti menggunakan jaringan pribadi seperti virtual private network (VPN), yang mengenkripsi semua lalu lintas antara titik akhir, atau menggunakan sistem email terenkripsi, yang membatasi tampilan pesan ke penerima yang dituju. Jika informasi rahasia secara fisik meninggalkan lokasi yang dilindungi (seperti ketika karyawan mengangkut pita cadangan antar fasilitas), organisasi harus mengenkripsi data jika jatuh ke tangan pengguna yang tidak berwenang.

Kerahasiaan informasi digital juga membutuhkan kontrol di dunia nyata. Shoulder surfing, praktik melihat dari balik bahu seseorang atau mengintip kegiatan seseorang saat berada di layar komputernya, adalah cara nonteknis bagi penyerang untuk mengumpulkan informasi rahasia.

Ancaman fisik, seperti pencurian secara sederhana, juga mengancam kerahasiaan. Konsekuensi dari pelanggaran kerahasiaan bervariasi tergantung pada sensitivitas data yang dilindungi. Pelanggaran nomor kartu kredit, seperti dalam kasus sistem pemrosesan Sistem Pembayaran Heartland di AS pada tahun 2008, dapat mengakibatkan tuntutan hukum dengan pembayaran hingga jutaan dolar.

e. Integritas

Dalam bidang keamanan informasi, integritas biasanya mengacu pada integritas data, atau memastikan bahwa data yang disimpan akurat dan tidak mengandung modifikasi yang tidak sah. Informasi Nasional Assurance Glossary (NIAG) mendefinisikan integritas sebagai berikut:

1) Kualitas dari sebuah Sistem Informasi yang mencerminkan kebenaran logis dan keandalan sistem operasi.
2) Kelengkapan logis dari perangkat keras dan perangkat lunak yang menerapkan mekanisme perlindungan.
3) Konsistensi struktur data dan kemunculan data yang disimpan.

Prinsip ini, yang bergantung pada autentikasi, autorisasi, dan nonrepudiatient sebagai kunci untuk menjaga integritas, mencegah mereka yang tidak memiliki otorisasi untuk memodifikasi data secara tidak sah. Dengan melewati sistem autentikasi atau meningkatkan hak istimewa di luar yang seharusnya diberikan kepada mereka, penyerang dapat mengancam integritas data.

Cacat dan kerentanan perangkat lunak dapat menyebabkan kerugian yang tidak disengaja dalam integritas data dan dapat membuka sistem untuk modifikasi data yang tidak sah. Program biasanya mengontrol dengan ketat ketika pengguna memiliki akses baca- tulis ke data tertentu, tetapi kerentanan perangkat lunak memungkinkan untuk menghindari kontrol itu. Misalnya, penyerang dapat mengeksploitasi kerentanan injeksi Structured Query Language (SQL) untuk mengekstrak, mengubah, atau menambahkan informasi ke database.

Mengganggu integritas data saat berhenti atau dalam pesan dalam perjalanan dapat memiliki konsekuensi serius. Jika dimungkinkan untuk mengubah pesan transfer dana yang lewat antara pengguna dan situs web perbankan online-nya, penyerang dapat menggunakan hak istimewa itu untuk keuntungannya.

Seperti kasus bagaimana penyerang dapat membajak transfer dan mencuri dana yang ditransfer dengan mengubah nomor rekening penerima dana yang

tercantum dalam pesan ke nomor rekening bank penyerang sendiri. Memastikan integritas jenis pesan ini sangat penting untuk sistem yang aman. Perhatikan bahwa, dalam mode keamanan formal, integritas ditafsirkan secara lebih sempit sebagai perlindungan terhadap modifikasi atau penghancuran informasi yang tidak sah

f. Ketersediaan

Sistem informasi harus dapat diakses oleh pengguna agar sistem ini dapat memberikan nilai apa pun. Jika sistem mati atau merespons terlalu lambat, sistem tidak dapat memberikan layanan yang seharusnya. NIAG mendefinisikan ketersediaan sebagai akses tepat waktu dan andal ke layanan data dan informasi untuk pengguna yang berwenang.

Serangan pada komponen ketersediaan agak berbeda dari serangan pada integritas dan kerahasiaan. Serangan paling terkenal pada ketersediaan adalah serangan Denial of Service (DoS). DoS bisa datang dalam berbagai bentuk, tetapi setiap bentuk pasti ditujukan untuk mengganggu sistem dengan cara melakukan pencegahan terhadap pengguna yang sah dari mengakses sistem. Salah satu bentuk DoS adalah kelelahan sumber daya, di mana penyerang membebani sistem hingga tidak lagi merespons permintaan yang sah. Sumber daya yang dimaksud dapat berupa memori, waktu unit pemrosesan pusat (CPU), bandwidth jaringan, dan/atau komponen lain apa pun yang dapat dipengaruhi oleh penyerang. Salah satu contoh serangan DoS adalah flooding jaringan, dimana penyerang mengirimkan begitu banyak lalu lintas jaringan ke sistem yang ditargetkan

sehingga lalu lintas memenuhi jaringan dan tidak ada permintaan yang sah yang dapat melewatinya.

Mitigasi serangan pada jaringan komputer dapat dilakukan mulai dari desain jaringan komputer yang benar serta implementasi aplikasi monitoring jaringan yang mempunyai fitur Intrusion Detection System/Intrusion Prevention System (Ericka J dan Prakarsa W, 2020). Selain itu, ada juga mengenai User Access Control yang dapat dimanfaatkan untuk mengontrol akses dari setiap permintaan kepada sistem (Crysdian C et al, 2000).

2.1.3 Malware

Malware (malicious software) menurut beberapa ahli memiliki beberapa pengertian. Perangkat lunak yang digunakan dengan tujuan untuk mencoba melanggar kebijakan keamanan sistem komputer terkait dengan kerahasiaan, integritas, atau ketersediaan (Sharp, 2009). Selain itu, jika dilihat dari sudut organisasi ekonomi malware adalah istilah umum untuk perangkat lunak yang dimasukkan ke dalam sistem informasi untuk menyebabkan kerusakan pada sistem itu atau sistem lain, atau untuk menumbangkannya untuk digunakan selain dari yang dimaksudkan oleh pemiliknya (OECD, 2007). Istilah malware digunakan untuk berbagai bentuk permusuhan atau intrusif perangkat lunak (Schultz dkk, 2001). Malware berbahaya karena mereka memiliki terlalu banyak keterbatasan pada mesin yang tidak teratur seperti menonaktifkan pendeteksi malware atau anti virus yang dipasang untuk tujuan keamanan (Jain and Bajaj, 2014).

Menurut Jain and Bajaj (2014), malware dapat digolongkan menjadi beberapa golongan.

a. Virus

Konsep virus dan malware telah mendampingi perjalanan teknologi selama beberapa dekade, seiring dengan perkembangan teknologi pendeteksian. Disini akan dijelaskan perbedaan antara virus dan jenis malware lain yang dapat menginfeksi pengguna dan organisasi. Internet menampung banyak bentuk perangkat lunak berbahaya, juga dikenal sebagai malware, yang berbeda dalam fungsi dan tujuannya. Seringkali terjadi, deskripsi malware, apa pun jenisnya, salah mengklasifikasikan perangkat lunak berbahaya sebagai virus. Selama bertahun-tahun, istilah virus komputer telah menjadi istilah yang mencakup semua perangkat lunak berbahaya, namun dalam dunia keamanan komputer, virus mengacu pada jenis malware tertentu yang menyebar dengan menginfeksi file lain dengan muatan berbahayanya. Orang awam sering salah menyebut semua jenis malware sebagai virus, padahal sebenarnya belum tentu benar, bisa berarti bahwa malware tersebut adalah Trojan horse (Trojan) atau worm. Trojan adalah bagian dari perangkat lunak berbahaya yang terlihat serupa dengan aplikasi yang sah. Trojan berjalan pada sistem yang terinfeksi seolah-olah itu adalah aplikasi dengan tujuan yang bermanfaat. Worm adalah jenis malware lain yang merupakan executable mandiri yang menyebar melalui berbagi jaringan dan kerentanan sistem.

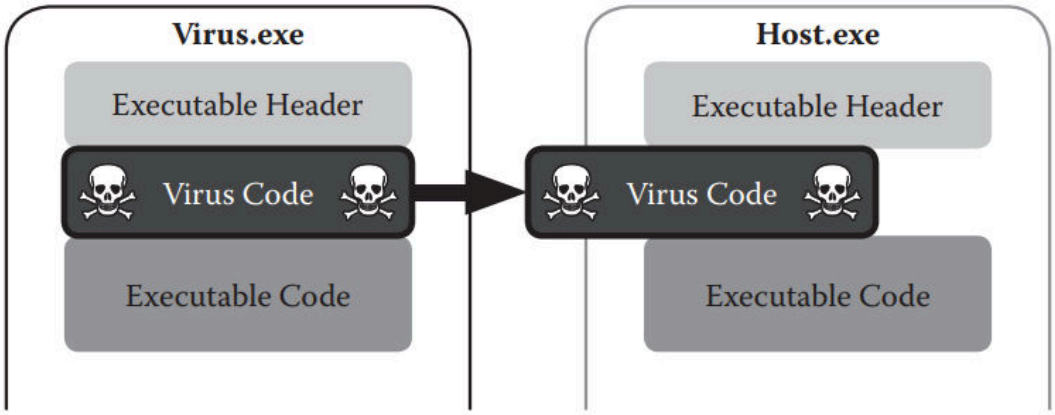
Di sisi lain, virus bekerja tidak mandiri dan membutuhkan infeksi file host untuk menyebar. Virus bersifat parasit, menginfeksi sistem dengan menempelkan

dirinya ke file lain. Virus komputer menyebar dengan cara yang sama seperti virus biologis, yang menyuntikkan DNA ke dalam sel inang untuk mereplikasi dirinya sendiri dan menyebabkan sel meledak, melepaskan virus yang direplikasi untuk menyebar ke sel lain. Virus komputer mencapai kesetaraan teknologi dengan menulis kodenya ke dalam file host. Virus akhirnya berjalan ketika pengguna membuka file host yang terinfeksi. Sekarang perbedaan antara virus dan jenis malware lainnya jelas, sejarah singkat virus komputer akan memberikan beberapa informasi latar belakang yang berguna. Virus berbasis PC IBM pertama yang tercatat, disebut *"*Brain*"* memulai debutnya pada Januari 1986. Brain menyalin dirinya sendiri ke sektor boot floppy disk, ruang pada floppy disk yang digunakan untuk menjalankan kode ketika sistem dimulai. Setelah di memori, ia mencoba untuk menyalin dirinya sendiri ke floppy disk lain, efek samping utama dari infeksi adalah perubahan label volume menjadi *"*(c) Brain*"*.

Virus Brain tidak terlalu merusak tetapi mengambil keuntungan dari penggunaan floppy disk yang berat pada era tersebut. Virus lain, bagaimanapun tidak berbahaya dan menyebabkan kerusakan pada sistem yang terinfeksi. Pada tahun 1987, virus Yerusalem dan variannya mulai menginfeksi sistem. Virus ini berada di memori dan menginfeksi semua file yang dapat dieksekusi (seperti .com dan .exe) pada sistem. Ketika pengguna membuka file yang terinfeksi, virus akan menghapus file yang terinfeksi. Kode virus dalam file host yang terinfeksi sering kali memiliki tiga bagian berbeda: discovery module, replication module, dan payload. Discovery module memungkinkan virus untuk menemukan file host, dan

replication module melakukan infeksi dengan menyalin seluruh kode virus ke file host.

Terakhir, payload berisi kode untuk melakukan tindakan tambahan pada sistem yang terinfeksi selain dari penemuan dan replikasi file. Tindakan spesifik yang dilakukan oleh payload tergantung pada tujuan virus. Payload berkisar dari kode yang tidak berbahaya, seperti virus Cascade yang mengubah teks yang ditampilkan di layar, hingga kode yang merusak, seperti virus Yerusalem yang menghapus file yang terinfeksi.



Gambar 2.1 Jalan kode virus

Dampak virus pada sistem menuntut solusi untuk mendeteksi dan membersihkan infeksi. Produk antivirus mencoba mendeteksi virus dengan mencari file untuk discovery module, replication module, atau payload. Metode deteksi mencakup kecocokan pola tertentu dalam metode yang dapat dieksekusi atau heuristik untuk mendeteksi aktivitas virus. Produk antivirus ini juga berusaha membersihkan infeksi virus dengan menghapus kode virus dan mengembalikan konten file asli. Program antivirus tidak bisa begitu saja menghapus file yang terinfeksi karena hal itu dapat berdampak buruk pada operasi sistem. Antivirus

harus mendeteksi teknik yang digunakan virus untuk mengeksekusi kode virus di dalam file yang terinfeksi

b. Worms

Worms merupakan kelas besar dari malicious code yang menyebar di antara komputer dengan mendistribusikan salinannya sendiri dalam berbagai cara. Worm adalah salah satu bentuk paling awal dari malicious code yang bisa menjadi jinak atau merusak. Malicious code disebut sebagai worm jika menyebar ke sistem lain dengan menggandakan dirinya sendiri tanpa melampirkan file lain. Tidak seperti virus komputer yang menyebar dengan menginfeksi file yang dapat dieksekusi atau jenis file lain, worm menyebar dengan mendistribusikan salinannya sendiri. Salinan mungkin tidak identik dengan worm asli, tetapi mereka memiliki fungsi yang sama dan dapat terus menyebar ke komputer lainnya. Worm Morris, dirilis oleh Robert Morris pada tahun 1988, adalah salah satu worm pertama yang menyebar di Internet. Worm menyebar di Internet dengan mengeksploitasi beberapa kerentanan yang diketahui dalam program UNIX umum. Morris menyatakan bahwa tujuan worm adalah untuk mengukur ukuran Internet pada saat itu, tetapi menyebar begitu cepat sehingga menyebabkan Denial of Service (DoS) yang meluas.

Pada umumnya, worms mempunyai dua peranan, yang pertama adalah menyebar ke komputer lain atau tambahan, naum sebagian besar juga memiliki tugas sekunder yang dikenal sebagai payload. Worm memuat kode khusus yang diprogram oleh penyerang untuk berjalan ketika setelah worm menyebar. Dalam kasus Worm Morris, tujuannya adalah untuk mengukur ukuran internet, tetapi

kebanyakan worm memiliki muatan yang jauh lebih berbahaya. Muatan tersebut dapat mencakup serangan Denial of Service (DDoS), distribusi spam, kejahatan dunia maya, atau apa pun yang dipilih penyerang. Pada tahun-tahun sejak program Morris lepas kendali, lebih banyak worm yang berkembang telah menyebar di internet. Banyak worm menargetkan kerentanan dalam layanan jaringan populer seperti server HTTP dan NetBIOS. Namun, banyak yang tidak menggunakan kerentanan untuk menyebar, alih-alih menggunakan surel, jaringan peer-to-peer (P2P), jaringan sosial, dan protokol komunikasi perangkat seluler. Teknik propagasi ini mengandalkan bagaimana menipu pengguna untuk menjalankan program dan tidak dapat menyebar tanpa interaksi manusia. Worm tidak terbatas pada satu metode propagasi tetapi dapat menggunakan salah satu atau semua metode ini sekaligus.

Untuk mengurangi ancaman dari worm komputer, administrator harus melindungi sistem dari semua teknik propagasi. Langkah-langkah berikut ini akan mengurangi kemungkinan infeksi worm dalam jaringan:

1) Gunakan produk antivirus untuk memindai e-mail masuk dan pesam dari messenger terhadap tautan yang berbahaya.
2) Nonaktifkan fungsi autorun untuk perangkat USB baik flashdrive maupun smartphone.
3) Terapkan patch untuk kerentanan dalam layanan jaringan pada waktu yang tepat.
4) Nonaktifkan akses ke jaringan P2P.

5) Memberikan pendidikan terhadap pengguna tentang bahaya worm yang menggunakan teknik rekayasa sosial (social engineering).

c. Spyware

Spyware adalah perangkat lunak yang diinstal tanpa persetujuan anda, baik itu komputer tradisional, aplikasi di browser web anda, atau aplikasi seluler yang berada di perangkat anda. Singkatnya, spyware mengkomunikasikan informasi pribadi dan rahasia tentang anda kepada penyerang. Informasi tersebut mungkin berupa laporan tentang kebiasaan atau pembelian browsing online anda, tetapi juga dapat dimodifikasi untuk mencatat hal-hal seperti penekanan tombol pada keyboard, informasi kartu kredit, kata sandi, atau kredensial login.

d. Adware

Adware adalah nama yang diberikan untuk program yang dirancang untuk menampilkan iklan di komputer anda, mengarahkan permintaan pencarian anda ke situs web periklanan dan mengumpulkan data jenis pemasaran tentang anda, misalnya, jenis situs web yang anda kunjungi, apa saja yang anda akses, jenis informasi apa yang sering anda cari, sehingga iklan yang disesuaikan dapat ditampilkan.

e. Trojan

Trojan horse adalah program jahat yang disamarkan untuk mengelabui pengguna tanpa dicurigai untuk mengunduh dan menginstalnya. Setelah ini terjadi, malware dengan sengaja melakukan aksi atau tindakan yang tidak diharapkan pengguna . Ini sering kali melibatkan penyediaan akses jarak jauh ke mesin yang terinfeksi, yang memungkinkan penyerang mencuri data, menginstal

malware tambahan, atau memonitor aktivitas pengguna. Trojan tidak mereplikasi (seperti worm), dan juga tidak menginfeksi file lain (seperti virus), tetapi bisa juga merusak.

f. Botnet

Botnet adalah jaringan komputer yang saling terhubung, biasanya dikontrol oleh sebuah komputer pusat yang menjadi pusat perintah untuk menyelesaikan tugas-tugas tertentu. Meskipun tidak selalu berbahaya, mereka sering digunakan untuk kegiatan terlarang. Di mana bot digunakan untuk pengejaran ilegal, botnet biasanya dioperasikan oleh hacker, yang kemudian mengendalikan komputer lain dalam jaringan, yang disebut sebagai zombie. Komputer yang merupakan bagian dari botnet digunakan untuk mengirim spam, melakukan serangan denial of service, dan bahkan mentransfer dana untuk kegiatan kriminal. Peretas juga dapat menjual jasa botnet untuk mengirim spam. Ini memungkinkan spammer untuk menghindari deteksi (email tidak berasal dari server mereka), serta mengurangi biaya, karena pemilik komputer yang menggunakan jasa botnet tersebut cukup membayar biaya jasa botnet tanpa perlu menyediakan kebutuhan lainnya, namun hal itu juga cukup berbahaya karena bisa saja sang penjual jasa melakukan penipuan atau sejenisnya.

2.1.4 Anomaly Based (Outlier Based) Detection

Deteksi intrusi berbasis anomali terdiri dari proses pengamatan dan mengenali penyimpangan dari perilaku normal, yang telah ditangkap dan di maintain dalam sebuah profil elektronik. Secara umum dipahami bahwa batasan utama dari pendekatan deteksi berbasis anomali adalah bahwa ia menghasilkan

tingkat alarm palsu yang lebih tinggi daripada pendekatan deteksi misuse (Hall dkk, 2005). Sistem deteksi anomali dapat mendeteksi serangan baru dengan memantau penyimpangan yang signifikan dalam perilaku program normal (Murtaza dkk, 2013). IDS berbasis anomali bekerja dengan terlebih dahulu membangun model statistik dari pola penggunaan yang menggambarkan perilaku normal dari sumber daya yang dipantau. Setelah fase training data awal ini, sistem menggunakan kesamaan matrik untuk membandingkan permintaan input baru dengan model yang sudah ada, dan menghasilkan peringatan untuk mereka yang mengalami deviasi secara signifikan, mengingat mereka anomali. Pada dasarnya, serangan dideteksi karena menghasilkan perilaku yang berbeda, yaitu, anomali dari apa yang diamati ketika membuat model training data. Keuntungan utama dari sistem berbasis anomali adalah kemampuannya untuk mendeteksi serangan yang sebelumnya tidak dikenal ketika muncul (Nascimento & Correia, 2014).

2.1.5 Isolation Forest

Menurut Chun-Hui Xiao et al. (2018) dalam penelitian yang berjudul Anomaly Detection in Network Management System Based on Isolation Forest disebutkan bahwa Isolation Forest adalah Unsupervised Machine Learning yang digunakan untuk mendeteksi anomali. Isolation Forest membangun sebuah ensemble dari iTrees yang memiliki struktur setara dengan pohon pencarian biner. Tony Liu F et al. (2009) dalam penelitiannya yang berjudul Isolation Forest menyatakan bahwa Isolation Forest atau iForest, membangun ensemble iTrees untuk kumpulan data tertentu, maka anomali adalah instance yang memiliki panjang jalur rata-rata pendek pada iTrees. Hanya ada dua variabel pada Isolation

Forest ini, yaitu jumlah dari pohon yang dibangun dan ukuran sub-sampling. Berikutnya, Yufei Song et al. (2019) dalam penelitian mereka yang berjudul Isolation Forest based Detection for False Data Attacks in Power Systems menjelaskan prinsip dasar dari deteksi Isolation Forest ini adalah dengan memisahkan ruang data menjadi satu set subruang dengan pola yang acak. Data anomali biasanya terletak pada data yang jarang ada pada suatu dataset, algoritma ini dapat mengidentifikasi titik anomali dengan sangat cepat.

2.2 Penelitian Terkait

Pada penelitian sebelumnya, (Chun-Hui et al, 2018) mengusulkan mekanisme pendeteksian anomali pada Network Management System Based dengan menggunakan metode perhitungan statistik untuk mengekstrak fitur dan menggunakan Isolation Forest, semacam algoritma machine learning untuk membuat dua klasifikasi, kombinasi antara teknologi tradisional dan tekonolgi terbaru. Menurut mereka deteksi menggunakan Isolation Forest beberapa kelebihan, yaitu

(i) Memiliki kompleksitas waktu linier dan membutuhkan memori yang lebih rendah.
(ii) Bekerja dengan baik pada data multi dimensi sehingga dapat melakukan pendeteksian anomali pada gabungan beberapa dataset terkait.
(iii) Dapat mengidentifikasi anomali bahkan ketika tidak ada data anomali di dalam data pelatihan.

Jyothsna dan Prasad (Jyothsna & Prasad, 2019) melakukan penelitian Intrusion Detection System dengan metode anomaly based yang dioptimasikan

dengan menggunakan algoritma Feature Association Impact Scale dalam proses training datasetnya. Hal ini dilakukan untuk mengoptimalkan tingkat akurasi deteksi terhadap intrusi jaringan berbasis anomali. Penelitian ini menghasilkan tingkat akurasi deteksi sebesar 88% dengan menggunakan dataset NSL-KDD.

Pada penelitian sebelumnya, (Tao et al., 2018) mengusulkan algoritma deteksi anomali lalu lintas jaringan paralel SPIF yang didasarkan pada iForest dan Spark. Dengan memanfaatkan algoritma Isoaltion Forest untuk pendeteksian anomali lalu lintas jaringan dan teknologi Spark untuk pemrosesan data besar, SPIF dapat melakukan secara paralel proses pemodelan dan proses evaluasi anomali dengan baik.

Kumar dan Sharma (Kumar & Sharma, 2018) melakukan penelitian hybrid intrusion detection system yang mengkombinasikan metode anomaly based dengan signature based yang dapat digunakan pada cloud pribadi. Efisiensi dari algoritma yang diimplementasikan cukup baik dalam hal kompleksitas serta dapat mudah diimplementasikan dengan berbagai teknologi open source seperti Java, Python, dll.

Letteri et al. (2020) menyajikan dataset baru bernama MTA-KDD'19 yang dibuat untuk memudahkan pengujian dan perbandingan dari analisa algoritma malware traffic berbasis machine learning. Dataset ini dibuat untuk digunakan dalam berbagai penelitian ke depannya yang lebih kompleks.

Menurut Hu (2010) berfokus pada penelitian Hybrid Intrusion Detection System yang menggabungkan berbagai mesin deteksi yang menghasilkan skema

dan kerangka kerja deteksi sistem yang baru. Metode yang diintegrasikan diantaranya adalah Hidden Markov Model dan anomaly based.

BAB III

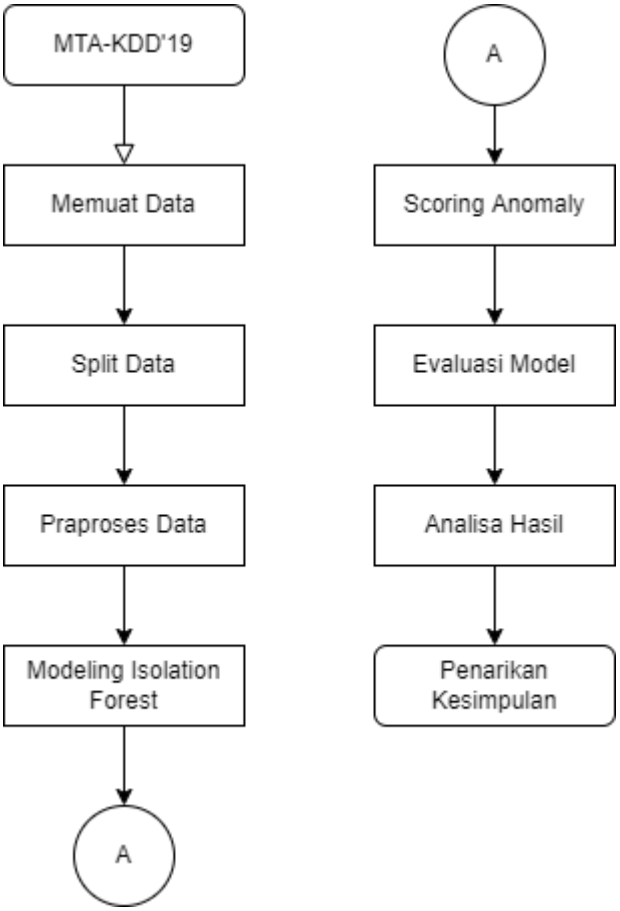
METODE PENELITIAN

Dalam bab ini akan dijelaskan beberapa hal yang mencakup dengan proses penelitian ini, yaitu tahapan penelitian yang akan dilakukan, kebutuhan sistem untuk mengolah dataset data latih dengan machine learning dan penyelesaian masalah terkait dengan deteksi malware dengan anomaly based.

Penelitian ini akan menguji data malware traffic dengan metode anomaly based model Isolation Forest untuk melakukan deteksi malware dan melakukan uji akurasi serta efisiensi penggunaan resource.

3.1 Prosedur Penelitian

Dalam menjalankan proses penelitian deteksi serangan malware dengan anomaly based, peneliti membuat acuan prosedur penelitian agar penelitian lebih sistematis dan terarah. Acuan yang dibuat berupa diagram prosedur penelitian yang ada pada Gambar 3.1



Gambar 3.1 Diagram Prosedur Penelitian

Berdasarkan gambar 3.1 terdapat delapan proses penelitian yang dilakukan untuk menyelesaikan penelitian ini. Penelitian diawali dengan melakukan pemuatan dataset MTA-KDD'19 yang terdiri dari data malware dan legitimate/normal. Berikutnya dataset yang telah dimuat akan dibagi menjadi data latih dan data uji dengan perbandingan 80:20 untuk data latih berbanding data uji. Setelah dataset terbagi menjadi data latih dan data uji, maka dataset akan diolah dengan melakukan tahapan praproses data demgan tujuan meningkatkan kualitas data yang akan diolah. Yang kemudian dilanjutkan dengan melakukan modeling algoritma Isolation Forest menggunakan dataset data latih. Output dari hasil pemodelah Isolation Forest menggunakan data latih dataset MTA-KDD'19

berikutnya akan menjadi standar acuan dalam proses fitting serta prediksi untuk mendapatkan skor terhadap data-data yang terindikasi anomali dan normal. Setelah semua data dilakukan scoring maka model akan dilakukan evaluasi yaitu pencarian akurasi model Isolation Forest dan perhitungan efisiensi resource dalam melakukan proses mulai dari memuat data hingga evaluasi model.

3.1.1 Memuat Data

Dalam sub bab ini peneliti akan menjelaskan proses sebelum melakukan praproses terhadapa data, maka peneliti harus memuat data yang digunakan sebagai objek dalam peelitian. Data ini diakuisisi dari dataset penelitian berjudul MTA-*KDD'19: A Dataset for Malware Traffic Detection* oleh Ivan Letteri et al (2020). Data yang digunakan dalam penelitian ini terdiri dari dua jenis data, yaitu data Legitimate (Normal) dan data Malware Attack yang memiliki 33 fitur. Beberapa fitur seperti FinFlagDist, SynFlagDist, RstFlagDist, PshFlagDist, dan AckFlagDist adalah beberapa fitur yang dapat digunakan untuk menganalisa malware traffic. Data Legitimate dan Malware Attack akan digabungkan menjadi satu data kombinasi sebagai objek penelitian.

Jumlah data yang dimuat sebagai objek penelitian ini berjumlah 64.554 data. Adapun fitur-fitur yang ada pada dataset MTA-KDD'19 yaitu sebanyak 33 fitur seperti pada tabel 3.1

Tabel 3.1 FItur-fitur MTA-KDD'19

No.	Nama Fitur	No.	Nama Fitur
1	FinFlagDist	18	1stPktLen
2	SynFlagDist	19	MaxLenrx
3	RstFlagDist	20	MinLenrx
4	PshFlagDist	21	StdDevLenrx
5	AckFlagDist	22	AvgLenrx
6	DNSoverIP	23	MinIATrx
7	TCPoverIP	24	AvgIATrx
8	UDPoverIP	25	NumPorts
9	MaxLen	26	FlowLEN
10	MinLen	27	FlowLENrx
11	StdDevLen	28	repeated_pkts_ratio
12	AvgLen	29	NumCon
13	MaxIAT	30	NumIPdst
14	MinIAT	31	Start_flow
15	AvgIAT	32	DeltaTimeFlow
16	AvgWinFlow	33	HTTPpkts
17	PktsIOratio		

Untuk mengetahui secara rinci data yang akan dimuat, peneliti menggunakan fungsi describe() dari Pandas untuk menampilkan rincian data secara tabel dengan beberapa penjelasan posisi nilai setiap fiturnya, yaitu:

1) Count, menunjukkan jumlah total data dari fitur.
2) Mean, menunujukkan sebagai nilai rata-rata dari fitur.
3) Std, menunjukkan nilai standar deviasi dari fitur.
4) Min, sebagai nilai terkecil dari fitur.
5) 25%, menampilkan nilai pada 25% dari total nilai fitur.
6) 50%, menampilkan nilai pada 50% dari total nilai fitur.
7) 75%, menampilkan nilai pada 75% dari total nilai fitur.
8) Max, menampilkan nilai terbesar dari fitur.

Dalam implementasinya terhadap dataset MTA-KDD'19, fungsi describe() menunjukkan secara detail rincian data terhadap 33 fitur yang ada seperti pada Gambar 3.2 untuk 5 fitur FinFlagDist, SynFlagDist, RstFlagDist, PshFlagDist, dan AckFlagDist serta 6 fitur pada Gambar 3.3 yaitu DNSoverIP, TCPoverIP, UDPoverIP, MaxLen, MinLen, StdDevLen.

```
39 dfComplete.describe()
```

	FinFlagDist	SynFlagDist	RstFlagDist	PshFlagDist	AckFlagDist
count	6.455400e+04	6.455400e+04	6.455400e+04	6.455400e+04	6.455400e+04
mean	1.253150e-13	5.244941e-14	5.619304e-14	2.021604e-14	-2.403196e-15
std	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00
min	-1.204652e+00	-1.407562e+00	-7.169369e-01	-1.350512e+00	-2.047207e+00
25%	-1.204652e+00	-2.161944e-01	-7.169369e-01	-7.651495e-01	-7.270884e-01
50%	-3.112059e-02	-2.161944e-01	-7.169369e-01	1.112646e-01	5.924374e-02
75%	7.778985e-01	6.363655e-01	1.191003e+00	6.974030e-01	6.917300e-01
max	2.548173e+00	2.748681e+00	1.893017e+00	2.665581e+00	2.988854e+00

Gambar 3. 2 Deskripsi Dataset 5 Fitur

	DNSoverIP	TCPoverIP	UDPoverIP	MaxLen	MinLen	StdDevLen
6.455400e+04	6.455400e+04	6.455400e+04	6.455400e+04	6.455400e+04	6.455400e+04	6.455400e+04
9.621050e-14	6.330776e-14	4.506690e-14	5.814569e-14	3.070946e-08	-2.484344e-14	
1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	1.000008e+00	
-1.545158e-01	-7.182148e+00	-1.569913e-01	-1.493266e+00	-4.121468e+00	-1.194269e+01	
-1.545158e-01	1.547113e-01	-1.569913e-01	-1.226459e+00	-1.028609e+00	-7.762913e-01	
-1.545158e-01	1.547113e-01	-1.569913e-01	6.527169e-01	6.286402e-01	1.747893e-01	
-1.545158e-01	1.547113e-01	-1.569913e-01	7.226693e-01	6.286402e-01	7.970619e-01	
6.778562e+00	1.547113e-01	6.836627e+00	6.820353e+00	3.289629e+00	1.365945e+01	

Gambar 3. 3 Deskripsi Dataset 6 Fitur

Untuk menjadi data yang akan dimuat sepenuhnya pada proses berikutnya, fitur-fitur yang tidak diperlukan pada proses pelatihan maupun pengujian perlu dihapus dari dataset untuk lebih mengefisiensi proses menggunakan fungsi drop() pada Pandas, adapun fitur-fitur yang dihapus seperti pada Tabel 3.2.

Tabel 3.2 Daftar fitur yang dihapus

No.	Nama Fitur
1	DNSoverIP
2	TCPoverIP
3	UDPoverIP
4	MaxLen
5	MinLen
6	StdDevLen
7	AvgLen
8	MaxIAT
9	MinIAT
10	AvgIAT
11	AvgWinFlow
12	PktsIOratio
13	1stPktLen
14	MaxLenrx
15	MinLenrx
16	StdDevLenrx
17	AvgLenrx
18	MinIATrx
19	AvgIATrx
20	NumPorts
21	FlowLEN
22	FlowLENrx
23	repeated_pkts_ratio
24	NumCon
25	NumIPdst
26	Start_flow
27	DeltaTimeFlow
28	HTTPpkts

3.1.2 Split Data

Dataset yang telah dimuat perlu dipisahkan menjadi data latih dan data uji sebagai bahan dari proses membangun model anomaly based Isolation Forest dan bahan uji metode anomaly based. Data latih nantinya digunakan dalam proses

pemodelan algoritma Isolation Forest yang menjadi dasar dalam perbandingan data yang akan membedakan data normal dan data anomali pada data uji. Sedangkan data uji digunakan sebagai data yang akan diolah pada proses uji coba.

Pembagian data kombinasi menjadi data latih dan data uji dilakukan dengan perbandingan 80% data latih dan 20% data uji dengan memanfaatkan fungsi train_test_split() dari Scikit-Learn. Dari total 64.554 data pada dataset MTA-KDD'19 jika dibagi 80% data latih dan 20% data uji, maka dihitung jumlahnya adalah sebanyak 51.643 data latih dan 12.911 data uji. Dalam implementasinya, melakukam split data dengan fungsi train_test_split() akan menghasilkan 4 jenis data uji dan data latih, yaitu

- X_train untuk menampung data source yang akan dilatih.
- X_test untuk menampung data source yang akan diuji coba.
- y_train untuk menampung data target yang akan dilatih.
- y_test untuk menampung data target yang akan diuji coba.

Dalam melakukan split data juga harus diperhatikan 4 variabel utama dalam fungsi train_test_split(), yaitu arrays, test_size, dan random_state.

```
X_train, X_test, y_train, y_test = train_test_split(df_train,
                                                    df_classes,
                                                    test_size=0.2,
                                                    random_state=25)
```

Gambar 3. 4 Implementasi split data

Hasil pemisahan data latih dan data uji menghasilkan data dengan rincian untuk data X_train seperti pada Gambar 3.5.

	FinFlagDist	SynFlagDist	RstFlagDist	PshFlagDist	AckFlagDist
count	51643.000000	51643.000000	51643.000000	51643.000000	51643.000000
mean	-0.000961	-0.002743	0.001871	-0.004700	-0.004362
std	0.999294	0.999870	1.000313	0.999521	1.000207
min	-1.204652	-1.407562	-0.716937	-1.350512	-2.047207
25%	-1.204652	-0.216194	-0.716937	-0.765149	-0.727088
50%	-0.031121	-0.216194	-0.716937	0.111265	0.059244
75%	0.777899	0.636365	1.191003	0.697403	0.685326
max	2.548173	2.748681	1.893016	2.665581	2.988854

Gambar 3. 5 Rincian data X_train

Kemudian rincian untuk data X_test seperti pada Gambar 3.6.

	FinFlagDist	SynFlagDist	RstFlagDist	PshFlagDist	AckFlagDist
count	12911.000000	12911.000000	12911.000000	12911.000000	12911.000000
mean	0.003846	0.010971	-0.007484	0.018799	0.017446
std	1.002886	1.000524	0.998789	1.001770	0.999058
min	-1.204652	-1.407562	-0.716937	-1.350512	-2.047207
25%	-1.204652	-0.216194	-0.716937	-0.765149	-0.727088
50%	-0.031121	-0.216194	-0.716937	0.111265	0.083924
75%	0.777899	0.636365	1.191003	0.744905	0.704255
max	2.533023	2.728170	1.893017	2.589989	2.954863

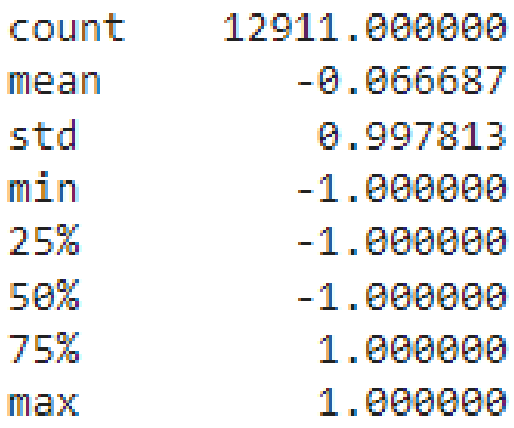
Gambar 3. 6 Rincian Data X_test

Untuk rincian data y_train seperti pada Gambar 3.7

```
count    51643.000000
mean        -0.063571
std          0.997987
min         -1.000000
25%         -1.000000
50%         -1.000000
75%          1.000000
max          1.000000
```

Gambar 3. 7 Rincian Data y_train

Dan rincian pada data y_test seperti pada Gambar 3.8



Gambar 3. 8 Rincian Data y_test

3.1.3 Praproses Data

Peneliti melakukan tahap yang penting yaitu praproses data, yang bertujuan untuk mengolah data agar lebih mudah diproses dalam sistem. Karena umumnya data yang diaplikasikan dalam analisis adalah data yang memiliki kekurangan atau tidak semputna dan tidak memiliki konsistesi. Maka diperlukan adanya tahap melakukan praproses terhadap data yang memiliki sasaran untuk meningkatkan kualitas data penelitian, tahapan tersebut dapat meningkatkan efisiensi hasil yang didapatkan serta meningkatkan akurasi.

a. Pembersihan Data

Tahap awal yang dilakukan dalam praproses data adalah melakukan pembersihan atau seleksi data. Pembersihan atau seleksi data diperlukan karena umumnya diketahui adanya data yang hilang atau mengalami kerusakan. Setelah pada proses sebelumnya peneliti telah mendapatkan 4 jenis data melalui proses split data, maka berikutnya data harus dipastikan bersih dan konsisten, maka data harus diperiksa apakah memiliki data yang berisi NaN atau bernilai Infinity. Jika

pada data diketahui nilai NaN atau Infinity, kemudian langkah praproses tidak dapat dilanjutkan. Sehingga perlu dilakukan pemeriksaan terlebih dahulu. Untuk memeriksa ada atau tidaknya nilai NaN dan Infinity tersebut, maka digunkana fungsi isna().any() dari Pandas. Dalam fungsi isna().any() tersebut akan memeriksa fitur dataset terkait adanya data yang rusak, yang akan menghasilkan output nilai True jika fitur ada nilai NaN dan Infinity dan False jika tidak ada nilai NaN dan Infinity di dalam fitur. Pada Gambar 3.9 hingga Gambar 3.12 memberikan hasil pemeriksaan terhadap nilai yang menghasilkan NaN serta Infinity di dalam data latih dan data uji.

```
44 X_train.isna().any()

FinFlagDist    False
SynFlagDist    False
RstFlagDist    False
PshFlagDist    False
AckFlagDist    False
```

Gambar 3. 9 Keluaran pembersihan data X_train

```
44 X_test.isna().any()

FinFlagDist    False
SynFlagDist    False
RstFlagDist    False
PshFlagDist    False
AckFlagDist    False
```

Gambar 3. 10 Hasiil pembersihan data X_test

```
46 y_train.isna().any()
```

```
False
```

Gambar 3. 11 Hasil pembersihan data y_train

```
44 y_test.isna().any()
```

```
False
```

Gambar 3. 12 Hasil pembersihan data y_test

Dari hasil pemeriksaan pada data latih baik pada X_train dan y_train, dapat dilihat bahwa semua fitur tidak mengandung nilai NaN dan nilai Infinity. Dan pada hasil pemeriksaan data uji pada data X_test dan y_test juga menghasilkan hasil yang sama, yaitu tidak ada fitur yang mengandung nilai NaN dan juga Infinity. Dengan hasil ini maka langkah praproses dapat dilanjutkan.

b. Standarisasi Atribut

Setelah data dibersihkan dari data rusak maupun data hilang, maka tahapan selanjutnya adalah melakukan standarisasi terhadap atribut yang ada pada data baik numerik maupun kategorik. Pada data MTA-KDD'19 semua data beratribut numerik. Atribut numerik memiliki rentang yang berbeda-beda sehingga perlu dilakukan standarisasi. Standarisasi atribut numerik dilakukan dengan melakukan perhitungan rata-rata yang kemudian juga deviasi standar pada setiap fitur, berikutnya fitur dilakukan standarisasi berdasarkan rumus 3.1:

$$Standarisasi = \frac{x - \mu}{\sigma} \quad (3.1)$$

Standarisasi atribut ini dijalankan supaya nilai tersebut bisa lebih serupa tanpa mengubah substansi dalam nilai tersebut. Sehingga bisa membantu lebih baik dalam proses analisis data. Dalam implementasinya terhadap data MTA-KDD'19 standarisasi atribut dilakukan dengan memanfaatkan fungsi StandardScaler() dari Scikit-Learn. Dari proses standarisasi, data akan mengalami perubahan menjadi variabel scaled_tr_x sebagai data latih yang berasal dari data X_train yang telah distandarisasi, scaled_te_x sebagai data uji yang berasal dari data X_test yang telah distandarisasi, sedangkan untuk data y_train dan x_train tidak perlu dilakukan standarisasi atribut karena hanya terdiri dari 1 dimensi array saja. Proses implementasi standarisasi atribut dengan menggunakan fungsi StandardScaler() seperti pada Gambar 3.13, sedangkan hasil dari standarisasi atribut untuk data X_train dan data X_test seperti pada Gambar 3.14 dan Gambar 3.15.

```
tr_x = StandardScaler().fit_transform(X_train.values)
scaled_tr_x = pd.DataFrame(tr_x, index=X_train.index,
                           columns=X_train.columns)

te_x = StandardScaler().fit_transform(X_test.values)
scaled_te_x = pd.DataFrame(te_x, index=X_test.index,
                           columns=X_test.columns)
```

Gambar 3. 13 Proses Standarisasi Atribut

	FinFlagDist	SynFlagDist	RstFlagDist	PshFlagDist	AckFlagDist
count	5.164300e+04	5.164300e+04	5.164300e+04	5.164300e+04	5.164300e+04
mean	3.253943e-17	1.031906e-18	5.764913e-17	8.255245e-18	1.155734e-17
std	1.000010e+00	1.000010e+00	1.000010e+00	1.000010e+00	1.000010e+00
min	-1.204553e+00	-1.405016e+00	-7.185901e-01	-1.346471e+00	-2.042442e+00
25%	-1.204553e+00	-2.134815e-01	-7.185901e-01	-7.608215e-01	-7.225841e-01
50%	-3.018074e-02	-2.134815e-01	-7.185901e-01	1.160210e-01	6.359277e-02
75%	7.794176e-01	6.391979e-01	1.188771e+00	7.024459e-01	6.895513e-01
max	2.550960e+00	2.751809e+00	1.890572e+00	2.671586e+00	2.992625e+00

Gambar 3. 14 Standarisasi atribut X_train

	FinFlagDist	SynFlagDist	RstFlagDist	PshFlagDist	AckFlagDist
count	1.291100e+04	1.291100e+04	1.291100e+04	1.291100e+04	1.291100e+04
mean	-3.302034e-18	-3.302034e-17	-1.430882e-17	-1.045644e-17	-1.788602e-17
std	1.000039e+00	1.000039e+00	1.000039e+00	1.000039e+00	1.000039e+00
min	-1.205067e+00	-1.417846e+00	-7.103401e-01	-1.366944e+00	-2.066680e+00
25%	-1.205067e+00	-2.270556e-01	-7.103401e-01	-7.825931e-01	-7.452653e-01
50%	-3.486702e-02	-2.270556e-01	-7.103401e-01	9.230626e-02	6.654356e-02
75%	7.718550e-01	6.250910e-01	1.199986e+00	7.248517e-01	6.874828e-01
max	2.521996e+00	2.715881e+00	1.902879e+00	2.566747e+00	2.940301e+00

Gambar 3. 15 Standarisasi atribut X_test

3.1.4 Modeling Isolation Forest

Secara teoritis Isolation Forest dibangun berbasis decision trees dan termasuk dari unsupervised model karena tidak memiliki label. Isolation Forest sendiri memiliki dua tahap proses yaitu pelatihan dan pengujian. Pada bab ini yang akan dibahas adalah proses pelatihan Isolation Forest yang dibangun dengan memanfaatkan sub sample dari data latih. Pada tahap pelatihan ini, Isolation Forest dibangun dengan mempartisi secara rekursif datset pelatihan yang

diberikan hingga instance terisolasi atau ketinggian pada pohon tertentu telah tercapai yang menghasilkan model parsial. Dan batas tinggi pohon l secara otomatis diatur oleh ukuran sub-sampling ψ: l = ceiling(log2 ψ), yang kira-kira merupakan tinggi pohon rata-rata. Untuk melakukan pemodelan Isolation Forest, digunakan dua algoritma untuk melakukan pelatihan model yaitu iForest dan iTrees seperti yang ada pada gambar 3.2 dan 3.3.

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - sub-sampling size

Output: a set of t *iTrees*

1: **Initialize** $Forest$
2: set height limit $l = ceiling(\log_2 \psi)$
3: **for** $i = 1$ to t **do**
4: $X' \leftarrow sample(X, \psi)$
5: $Forest \leftarrow Forest \cup iTree(X', 0, l)$
6: **end for**
7: **return** $Forest$

Gambar 3.16 Algoritma 1 iForest

Pada algoritma yang pertama yaitu algoritma iForest, ada beberapa langkah yang perlu dilalui, langkah-langkah tersebut adalah

1) Menentukan inputan yang akan diproses oleh dalam algoritma iForest. Pada algoritma iForest terdapat 3 inputan yang diperlukan, yaitu
    a) Pertama, inputan dengan variabel X sebagai data masukan yaitu dataset training untuk pelatihan.
    b) Kedua, adalah inputan dengan variabel t sebagai jumlah dari tree yang akan digunakan dalam pemodelan sebagai parameter yang menentukan dan mengontrol estimasi jumlah pohon ensemble yang digunakan. Secara

default nilai trees adalah 100, karena menurut Liu et al, panjang path yang baik ditemukan sebelum nilai t = 100.

c) Ketiga, inputan dengan variabel ψ sebagai inputan untuk sub sampling size yang menjadi ukuran yang digunakan sebagai kontrol ukuran data pelatihan. Secara empiris umumnya berada di angka 2^8 atau 256 yang secara umum memberikan detail yang cukup untuk menjalankan anomaly detection pada data yang luas.

2) Langkah berikutnya adalah menginisiasi atau mendefnisikan fungsi dari iForest.

3) Menentukan batas dari tinggi pohon isolation forest, dengan rumus 3.2

$$l = ceiling(log2\,\psi) \quad (3.2)$$

4) Melakukan perulangan yang dimulai dari 1 hingga sejumlah variabel t yang adalah jumlah tree berdasarkan inputan di awal. Dalam perulangan tersebut terdapat proses sampling data pada data inputan X menggunakan ukuran sub sampling ψ yang menghasilkan data *X'*

5) Variable Forest mendapatkan hasil dari gabungan antara nilai pada variabel forest sebelumnya dan juga hasil dari algoritma iTrees dengan variabel inputan data *X',0,*l

6) Perulangan selesai dan mengembalikan nilai Forest

Algorithm 2 : $iTree(X, e, l)$

Inputs: X - input data, e - current tree height, l - height limit

Output: an iTree

1: **if** $e \geq l$ or $|X| \leq 1$ **then**
2: return *exNode*$\{Size \leftarrow |X|\}$
3: **else**
4: let Q be a list of attributes in X
5: randomly select an attribute $q \in Q$
6: randomly select a split point p from max and min values of attribute q in X
7: $X_l \leftarrow filter(X, q < p)$
8: $X_r \leftarrow filter(X, q \geq p)$
9: return *inNode*$\{Left \leftarrow iTree(X_l, e + 1, l),$
10: $Right \leftarrow iTree(X_r, e + 1, l),$
11: $SplitAtt \leftarrow q,$
12: $SplitValue \leftarrow p\}$
13: **end if**

Gambar 3.17 Algoritma 2 iTree

Pada algoritma yang kedua, yaitu algoritma iTrees yang menjadi salah satu fungsi yang dibutuhkan pada alforitma iForest untuk mencari melakukan proses olah data. Alur proses dari algoritma iTrees ini adalah

1) Menentukan data inputan untuk diolah pada algoritma iTrees ini, ada 3 variabel inputan yang dibutuhkan ileh iTrees, yaitu
    a) Pertama, variabel X sebagai variabel yang mewakili dataset yang diinput dan akan diproses.
    b) Kedua, variabel e sebagai variabel nilai ketinggian pohon saat ini, ini mentnukan posisi node ketika proses sedang berlangsung.
    c) Ketiga, variabel l sebagai batas ketinggian yang ditentukan sebelumnya pada algoritma iForest.

2) Langkah berikutnya adalah masuk ke dalam fungsi iTrees ini, pada fungsi iTrees ini intinya adalah terkait dengan kondisi if else.
3) Jika ketinggian pohon saat ini (e) lebih besar dari sama dengan batas ketinggian pohon (l) atau jika nilai |X| lebih kecil dari sama dengan 1 maka iTrees akan mengembalikan nilai dari *exNode{Size* ← *|X|}*.
4) Kondisi lainnya akan melakukan proses dimana akan ada variabel Q yang menjadi variabel baru sebagai daftar dari atribut pada variabel X
5) Selanjutnya melakukan pemilihan secara acak atribut q di mana q adalah element dari variabel Q.
6) Berikutnya secara acak memilih sebuah titik pemisah variabel p dari nilai max dan min dari atribut q dalam variabel X.
7) Memisahkan nilai variabel X untuk node kiri dan node kanan, dimana nilai pada variabel X_l didapatkan dari filter data X dan nilai q kurang dari nilai p. untuk variabel X_r didapatkan dari nilai filter data X dan nilai q lebih besar dari sama dengan nilai p.
8) Langkah terakhir adalah mengembalikan nilai kepada node. Yang pertama memisahkan untuk node kiri, dimana mengambil nilai hasil dari iTree dengan inputan untuk variabel (X) mengambil variabel X_l kemudian ketinggian pohon saat ini (e) mengambil nilai e + 1, dan untuk variabel l sebagai batas ketinggian teteap mengambil dari nilai awal l. Kedua, memisahkan untuk node kanan dengan variabel (X) mengambil dari variabel X_r, untuk ketinggian pohon saat ini (e) mengambil nilai dari variabel e + 1, dan untuk variabel l sama dengan node kiri yaitu mengambil nilai dari variabel l di awal.

Selanjutnya menentukan atribut untuk split adalah variabel q dan nilai untuk split adalah variabel p.

Proses pemodelan dari Isolation Forest jika diimplementasikan ke dalam kode python, maka digunakan fungsi IsolationForest() dari sklearn. Fungsi IsolationForest() memiliki 9 parameter yang diperlukan untuk menjalankan fungsinya, yaitu:

1) n_estimators, jumlah estimasi dari total pohon pada Isolation Forest, secaraa default nilai n_estimators adalah 100.
2) max_samples, jumlah maksimal dari data yang diambil dari variabel X untuk melakukan pelatihan di setiap n_estimators. Nilai default dari max_samples adalah auto atau jika dilihat secara nilai adalah min(256, n_samples). Jika nilai max_samples bertipe data integer maka akan langsung mengambil jumlah sesuai dengan nilainya. Sedangkan jika nilai max_samples bertipe data float maka data yang diambil adalah max_samples * X.shape[0].
3) contamination, besaran dari contamination dataset, yaitu proporsi dari nilai outlier dalam dataset. Digunakan saat melakukan penyesuaian untuk menentukan ambang batas pada skor data sample. Nilai default dari contamination adalah auto, nilai auto sendiri bernilai 0.1. Sedangkan jika nilai contamination bertipe data float, maka nilai contamination ada di antara nilai 0 dan 0.5.
4) max_features, jumlah maksimal fitur yang digunakan dari data X pada tiap n_estimators. Nilai default dari max_features adalah 1.0. Jika nilai bertipe data int maka nilai langsng diambil dari nilai max_features. Sedangkan jika

nilai max_reatures bertipe data float, maka nilai didapatkan dari max_features * x.shape[1].

5) bootstrap, jika bernilai true pohon individu sesuai dengan subsets acak dari sample data latih dengan pergantian. Jika bernilai false maka menjalankan sampling tanpa pergantian. Nilai defaultnya adalah false.
6) n_jobs, jumlah pekerjaan yang berjalan secara parallel untuk semua fungsi fit dan predict. Nilai none berarti 1 kecuali dalam sebuah joblib.parallel_backend context. Nilai -1 berarti menggunakan semua processor. Nilai defaultnya adalah none.
7) random_state, mengontrol pseudo-randomness dari pemilihan fitur dan membagi nilai untuk setiap proses percabangan dan setiap pohon pada forest. Nilai defaultnya adalah none.
8) verbose, mengontrol verbosity dari proses pembangunan trees. Nilai defaultnya adalah 0.
9) warm_start, dimana nilai diatur true, akan digunakan lagi solusi dari variabel call sebelumnya agar menyesuaikan dan menambahkan lebih banyak estimator ke dalam ansambel, apabila tidak, cukup sesuaikan seluruhnya dengan forest yang baru.

Pada implementasi penelitian ini, peneliti menggunakan konfigurasi parameter dengan nilai n_estimators adalah 100, nilai max_samples adalah auto, nilai contamintaion adalah 0.07, dan nilai max_features adalah 1.0 seperti pada Gambar 3.18.

```
model = IsolationForest(n_estimators=100,
                        max_samples='auto',
                        contamination=0.07,
                        max_features=float(1.0))
```

Gambar 3. 18 modeling Isolation Forest

Setelah melakukan menjalankan fungsi IsolationForest(), antara data latih dengan hasil output dari fungsi IsolationForest() perlu disesauaikan agar menjadi model sistem pakar yang baik menggunakan fungsi fit(). Fungsi fit() diimplementasikan oleh estimator dan juga memiliki parameter masukan untuk sample data latih (X). Dan apabila digunakan pada model supervised, fungsi fit juga dapat menggunakan parameter argumen untuk label (y). Dan juga ada pilihan parameter opsional seperti bobot, dll. Jika didetailkan fungsi fit() ini sebagai berikut.

fit(X, y, sampel_weight=None)

X - Vektor pelatihan dimana n_samples adalah jumlah sampel dan n_features adalah fumlah fitur.

y – Nilai target, digunakan jika menggunakan jenis machine learning supervised.

sample_weight – Bobot tiap sampel.

3.1.5 Anomaly Scoring

Memberikan skor atau nilai pada data sebagai penanda bahwa data tersebut termasuk dalam golongan anomali atau bukan. Skor anomali ini dibutuhkan pada metode deteksi anomali apapun. Kesulitan dalam menurunkan skor dari h(x)

adalah bahwa sementara tinggi maksimum yang mungkin terjadi pada iTree tumbuh dalam orde n, sedangkan tinggi rata-rata tumbuh dalam orde log n. Karena iTrees memiliki kesamaan struktur dengan Binary Search Tree (BST) maka estimasi rata-rata h(x) untuk pemberhentian external node sama dengan pencarian yang gagal pada BST seperti pada tabel Skor ini didapatkan dari fungsi PathLength dimana skor anomali s diturunkan dari panjag jalur yang diharapkan E(h(x)) untuk setiap instance pengujian. (h(x)) diturunkan dengan melewati banyak instance melalui setiap fungsi iTree di iForest. Dengan menggunakan fungsi PathLength, panjang jalur tunggal h(x) diturunkan dengan menghitung jumlah tepi e dari root node ke terminating node saat instance x melintasi melalui fungsi iTree. Ketika instance x diakhiri pada external node, di mana Size > 1, nilai return adalah e ditambah penyesuaian c(Size). Penyesuaian perhitungan dilakukan untuk subtree yang tidak dibangun di luar batas ketinggian pohon. Ketika h(x) diperoleh untuk setiap pohon dari ensemble, skor anomali dihasilkan dengan menghitung s(x, ψ) dalam rumus 3.3.

Tabel 3.3 Tabel perbandingan iTree dan BST

iTree	BST
Binary tree yang layak	Binary tree yang layak
Pemberhentian external node	Pencarian gagal
Tidak berlaku	Pencarian berhasil

Kompleksitas proses evaluasi adalah O(nt log ψ), di mana n adalah pengujian ukuran data. Detail fungsi PathLength dapat dilihat pada Gambar. Untuk

menemukan m anomali teratas, cukup mengurutkan data menggunakan s dalam urutan menurun. Instance m pertama adalah anomali m teratas.

Algorithm 3 : $PathLength(x, T, e)$

Inputs : x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called
Output: path length of x

1: **if** T is an external node **then**
2: return $e + c(T.size)$ $\{c(.)$ is defined in Equation 1$\}$
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, e + 1)$
7: **else** $\{x_a \geq T.splitValue\}$
8: return $PathLength(x, T.right, e + 1)$
9: **end if**

Gambar 3. 19 Algoritma PathLength

Jika dilihat secara detail algoritma dari fungsi PathLength, maka dapat dipahami alur fungsi mulai dari variabel input yang dibutuhkan ada 3, yaitu variabel x berisi instance, variabel T berisi sebuah hasil iTree, dan variabel e berisi panjang path saat ini. Kemudian output yang duhasilkan oleh fungsi PathLength adalah panjang path dari x. Untuk prosesnya sendiri, fungsi PathLength dapat dijabarkan sebagai berikut

1) Jika kondisi T adalah external node maka,
2) Mengembalikan nilai e + c(T.size) dimana perhitungan c(.) didapatkan dari rumus 3.3.

$$c(n) = 2H(n-1) - (2(n-1)/n) \quad (3.3)$$

3) Kondisi if selesai.

4) Memasukkan nilai ke variabel a dimana nilai variabel a didapat dari T dikali dengan variabel splitAtt dari fungsi iTree.
5) Jika x_a kurang dari T.splitValue maka,
6) Mengembalikan PathLength dengan inputan x sebagai variabel x, T.left sebagai variabel T, dan e+1 sebagai variabel e.
7) Jika tidak, dan jika x_a lebih besar dari T.splitValue maka.
8) Mengembalikan PathLength dengan inputan x sebagai variabel x, T.right sebagai variabel T, dan e+1 sebagai variabel e.
9) Kondisi if selesai dan fungsi PathLength telah selesai.

Pada fungsi PathLength, dalam proses pengambilan decision dan predicition oleh model yang telah dilatih, secara matematis, rumus dari anomaly score dapat dituliskan dengan rumus berikut.

$$S(x,n) = 2\text{^}(-E(h(x))/c(n)\ ) \quad (3.3)$$

Dimana variabel h(x) adalah path length dan E(h(x)) adalah rata-rata path length dari setial iTrees (Isolation Tree). Path length sendiri adalah jarak yang mengukur titik dimana data tidak dapat dibagi lagi dengan root node dari struktur algoritma decision tree. Dan c(n) adalah rata-rata dari h(x) dengan nilai yang diketahui variabel n. Dari hasil persamaan, anomaly score bisa dibagi menjadi 3 bagian.

a. Ketika nilai E(h(x)) sama dengan c(n), maka variabel score (s) bernilai 0.5.
b. Ketika nilai E(h(x)) sama dengan 0, maka variabel score (s) bernilai 1.
c. Ketika nilai E(h(x)) sama dengan n - 1, maka variabel score (s) bernilai 0.

Skor anomali memiliki rentang nilai antara 0 hingga 1. Secara umum interpretasi dari skor anomali yang diberikan dapat dijelaskan sebagai berikut.

1) Ketika Nilai skor anomali (s) sama dengan 1 atau mendekati nilai 1 menunjukkan bahwa data tersebut adalah data **anomali** dan maknanya memiliki path length yang pendek.
2) Ketika Nilai skor anomali (s) lebih kecil dari 0.5 menunjukkan bahwa data tersebut termasuk cukup aman jika digolongkan ke dalam data **normal** dan maknanya memiliki path length yang panjang.
3) Nilai skor anomali mendekatim atau kurang lebih hampir sama dengan nilai 0.5 menunjukkan bahwa data tersebut bisa dikatakan tidak ditemukan adanya anomali ataupun perbedaan data yang cukup jauh satu sama lain, sehingga dikatakan data tergolonh sangat normal.

BAB IV

UJI COBA DAN PEMBAHASAN

4.1 Skenario Uji Coba

a. Scoring Anomaly

Setelah model berhasil didefinisikan dan disesuaikan, berikutnya data akan dicari kolom skor dan kolom anomali. Pencarian nilai dari kolom skor dengan memanggil fungsi decision_function() dari data yang telah dilatih dan memasukkan fitur RstFlagDist sebagai parameter scoring. Serupa dengan scoring, pencarian nilai dari kolom anomali dilakukan juga menggunakan RstFlagDist sebagai parameter, namun menggunakan fungsi yang berbeda, yaitu menggunakan fungsi predict() dan menggunakan model yang sudah dilatih sebelumnya.

Dua kolom scoring dan kolom anomali akan ditambahkan ke dataframe data uji untuk diolah pada langkah berikutnya. Setelah menambahkan kedua kolom ini kemudian melakukan pemeriksaan dataframe dari data uji. Jika dua kolom ini ditambahkan maka seharusnya data memiliki total 29 kolom yang sebelumnya 27 kolom seperti pada Gambar 4.1.

```
51 scaled_te_st['scores'] = model.decision_function(scaled_te_st[['RstFlagDist']])
52 scaled_te_st['anomaly_score'] = model.predict(scaled_te_st[['RstFlagDist']])
53 scaled_te_st.shape
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not
  "X does not have valid feature names, but"
(12911, 29)
```

Gambar 4.1 Scoring dan Anomaly Scoring

Nilai negatif pada kolom score dan nilai -1 pada kolom anomali menunjukkan adanya anomali pada dataframe. Dan nilai 1 pada kolom anomali menunjukkan bahwa data tersebut adalah normal. Pada dataframe data uji yang

mengandung data anomali seperti pada Gambar 4.2 dan yang mengandung data normal seperti pada Gambar 4.3.

```
53 scaled_te_st[scaled_te_st['anomaly_score']==-1].head()
```

AvgIAT	AvgWinFlow	PktsIOratio	1stPktLen	MaxLenrx	MinLenrx	StdDevLenrx	AvgLenrx	MinIATrx	AvgIATrx	NumPorts	FlowLEN	FlowLENrx	scores	anomaly_score
2.066127	1.242616	2.390851	-0.189419	-1.419016	0.416642	-1.019947	-0.858655	1.891900	2.128432	1.655589	-0.710494	-0.057114	-0.041313	-1
-0.508948	0.969220	0.232336	0.943188	1.212826	0.113088	1.916948	1.805018	-0.124110	-0.699231	1.285934	1.072902	0.359156	-0.041313	-1
1.981289	1.242616	0.100560	-0.189419	-1.419016	0.416642	-1.019277	-0.858655	0.445111	2.072437	1.285934	-0.568092	-0.058739	-0.055521	-1
1.053309	-0.185020	-0.339621	-0.752028	0.746929	-0.821048	0.546615	0.552593	-0.123144	1.023865	1.103713	0.338521	-0.053077	-0.027138	-1
1.947065	-0.159017	-0.700619	-0.189419	-0.376685	0.113088	-0.452817	-0.167232	-0.123935	2.070869	1.285934	0.304093	-0.054233	-0.032051	-1

Gambar 4.2 Data Anomali pada Data Uji

```
53 scaled_te_st[scaled_te_st['anomaly_score']==1].head()
```

AvgIAT	AvgWinFlow	PktsIOratio	1stPktLen	MaxLenrx	MinLenrx	StdDevLenrx	AvgLenrx	MinIATrx	AvgIATrx	NumPorts	FlowLEN	FlowLENrx	scores	anomaly_score
-0.036198	0.839299	0.883385	0.943188	1.212826	1.013027	1.937272	1.757291	-0.124107	-0.429433	1.655589	0.121714	-0.030837	0.156740	1
-0.895652	-1.747168	-1.612505	-0.752028	-0.557097	-0.821048	-0.891503	-1.134331	-0.124089	-0.725170	0.285557	1.832186	-0.030680	0.168507	1
-1.116718	-1.683839	-1.523735	-0.189419	-0.088144	0.113088	-0.424488	-0.644445	-0.123915	0.599280	-0.891764	0.854350	-0.057625	0.168507	1
-1.125162	1.242616	0.100560	-0.189419	-1.419016	1.013027	-0.907510	-0.824985	-0.124111	-1.245102	-0.891764	-1.533662	-0.060639	0.156740	1
-1.056761	-1.200880	0.459578	1.404163	1.212826	1.013027	1.137156	0.654241	-0.124105	-1.172154	1.409367	0.896462	-0.042522	0.168507	1

Gambar 4.3 Data Normal pada Data Uji

Setiap data point pada data uji diberi skor anomali oleh Isolation Forest, dimana setiap batas contamination pada model sebelumnya yang mempengaruhi dari batas data dinilai sebagai data anomali. Contamination dapat disesuaikan untuk menguji kesesuaian data sebagai anomali.

b. Evaluasi Model

Pada sub bab ini dijelaskan proses untuk mendapatkan nilai akurasi, nilai presisi, nilai recall, serta nilai f-measure pada hasil prediksi serta mencari tingkat efisiensi resource dari penggunaan algoritma isolation forest dalam melakukan deteksi anomali. Pada perhitungan terhadap akurasi, presisi, recall, serta nilai f-measure peneliti memanfaatkan output dari hasil deteksi Isolation Forest.

Nilai akurasi didapatkan dari perhitungan nilai persentase dari data percobaan yang dibandingkan dengan prediksi benar terhadap keseluruhan data uji coba yang

dilakukan. Peneliti menghitung nilai akurasi sesuai dengan rumus 4.1 (Hariyadi, 2010).

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} x100\% \quad (4.1)$$

Presisi menghitung persentase perbandingan antara prediksi benar positif dengan keseluruhan hasil yang diprediksi positif. Perhitungan presisi sesuai dengan rumus 4.2 (Hariyadi, 2010)

$$Presisi = \frac{TP}{TP + FP} x100\% \quad (4.2)$$

Recall menghitung persentase prediksi benar positif yang dibandingkan dengan semua kemungkinan prediksi data yang positif. Perhitungan nilai recall sesuai dengan rumus 4.3 (Hariyadi, 2010).

$$Recall = \frac{TP}{TP + FN} x100\% \quad (4.3)$$

f-measure menghitung kefektifan pada pengujian dengan mencermati inlai presisi dan recall. Perhitungan nilai f-measure sesuai dengan rumus 4.4 (Hariyadi, 2010).

$$f - measure = 2.\frac{presisi.recall}{presisi + recall} \quad (4.4)$$

Penjelasan terkait pengertian dalam istilah-istilah yang disebutkan di atas seperti dijelaskan berikut ini.

- Disebut True Positive (TP) apabila data anomali dan diperlakukan sebagai anomali.
- Disebut False Positive (FP) apabila data normal namun diperlakukan sebagai anomali.
- Disebut True Negative (TN) apabila data normal dan diperlakukan sebagai normal.
- Disebut False Negative (FN) apabila data anomali namun diperlakukan sebagai normal.

Berikut ini adalah data confusion matrix sebagai panduan untuk lebih memahami perbandingan data aktual dan data prediksi seperti pada tabel 4.1.

Tabel 4. 1 Confusion Matrix

Aktual / Prediksi	Anomali	Normal
Anomali	TP	FP
Normal	FN	TN

4.2 Hasil Uji Coba

Dalam tahapan hasil uji coba, peneliti menunjukkan hasil dari uji coba sesuai dengan proses dan tahapan skenario uji coba yang sudah dijelaskan pada sub bab sebelumnya. Uji coba data Evaluasi model dilakukan untuk mengukur akurasi dari model Isolation Forest, serta mengukur efisiensi resource yang digunakan ketika menjalankan model Isolation Forest.

a. Menghitung Akurasi Model Isolation Forest

Dalam tabel 4.2 ditunjukkan hasil uji coba yang menghsilkan nilai skor anomali pada data testing yang dibandingkan dengan data aktual sebagai pembanding hasil untuk mendapatkan akurasi, precision, recall, dan f1-score.

Tabel 4. 2 Hasil Uji Coba Data Dengan Uji

No. Baris Data	Data Prediksi	Data Aktual
18655	1	1
26860	1	-1
5324	1	-1
18318	1	-1
15985	1	1
...	...	...
19306	1	-1
9174	1	-1
10768	1	1
12968	1	1
27512	1	-1

Akurasi: 0,4730075129734335

Data dari hasil uji coba tersebut kemudian digunakan untuk mencari nilai akurasi, presisi, recall, dan f-measure.

1) $Akurasi = \frac{TP+TN}{TP+TN+FP+FN} x100\%$

$= \frac{5596+511}{5596+511+6375+429} x100\%$

$= 46{,}67\%$

2) $Presisi = \frac{TP}{TP+FP} x100\%$

$= \frac{5596}{5596+6375} x100\%$

$= 93\%$

3) $Recall = \frac{TP}{TP+FN} x100\%$

$$= \frac{5596}{5596+6375} x100\%$$

$$= 47{,}05\%$$

4) $f - measure = 2.\frac{presisi.recall}{presisi+recall}$

$$= 2.\frac{0{,}93.0{,}47}{0{,}93+0{,}47}$$

$$= 0{,}62$$

b. Mencari Efisiensi Resource

Hasil evaluasi model dalam mencari resource menggunakan fungsi Trace Memory Allocation untuk menghitung penggunaan memori sepanjang kode program dijalankan, serta menggunakan timer default dari google collab yang menghitung lama waktu eksekusi kode program.

```
Current memory usage is 33.741979MB; Peak was 62.158969MB
```

Gambar 4. 4 Hasil Evaluasi Penggunaan RAM Saat Proses Running Code

✓ 5s completed

Gambar 4.5 Hasil Evaluasi Lama Waktu Running Code

4.3 Pembahasan

Setelah tahapan dan proses uji coba selesai secara keseluruhan, maka akan terlihat hasil uji coba data menggunakan dataset MTA-KDD'19. Berikut adalah confusion matrix serta classification report dari hasil uji coba yang ditunjukkan pada Gambar 4.15.

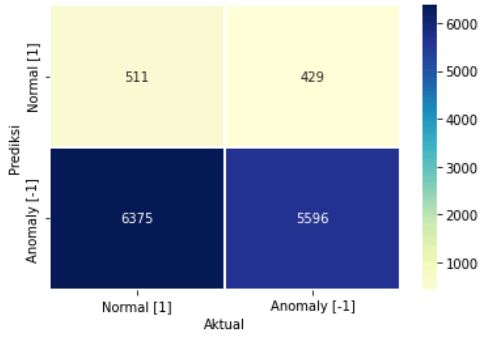
```
Isolation Forest model

Akurasi:  0.4730075129734335
Confusion Matrix:
[[ 511 6375]
 [ 429 5596]]

              precision    recall  f1-score   support

          -1       0.07      0.54      0.13       940
           1       0.93      0.47      0.62     11971

    accuracy                           0.47     12911
   macro avg       0.50      0.51      0.38     12911
weighted avg       0.87      0.47      0.59     12911
```



Gambar 4.6 Confusion Matrix dan Classification Report

Dari confusion matrix pada Gambar 4.15 di atas, hasil uji coba menunjukkan nilai TP sejumlah 5596; FP sejumlah 429; FN sejumlah 6375; dan TN sejumlah 511. Dan berdasarkan classification repor tersebut juga didapatkan data akurasi, precision, recall, dan f1-score. Akurasi sebesar 47,30%; precision sebesar 93,47; recall sebesar 47,05%; dan f1-score sebesar 62,59%. Dari hasil yang ditunjukkan oleh akurasi, precision, recall, dan f1-score, serta didukung oleh hasil pengujian penggunaan resource yang digunakan ketika proses deteksi menunjukkan bahwa pengujian deteksi malware menggunakan metode anomaly based dengan model isolation forest cukup baik, walaupun tingkat akurasi dan recallnya di bawah 50%.

Peretasan atau hacking dengan tujuan untuk mencuri, memanipulasi, maupun tindak kejahatan digital yang lain sangatlah merugikan dan meresahkan, terutama

bagi kalangan yang awam dengan teknologi. Melakukan peretasan untuk keuntungan pribadi sama saja dengan memakan harta orang lain secara ilegal, atau dalam istilah islam memakan harta dengan jalan yang batil. Sehingga perlu Sadanya solusi-solusi pencegahan dan penanganan terkait dengan tindak kejahatan peretasan tersebut. Sejalan dengan dengan bahasan penelitian terkait deteksi Malware pada Cloud Server, di mana malware menjadi salah satu cara peretasan yang umum dilakukan, maka perlu dilakukan usaha solutif yang serius terhadap peretasan, utamanya pada cloud server sebagai wadah penyedia layanan-layanan di internet yang diakses oleh banyak masyarakat.

Melakukan deteksi adalah proses pengelompokan atau pemisahan antara data yang normal dan tidak normal. Dalam penelitian ini, data tidak normal adalah data yang anomali, atau data yang menyimpang dari data normal. Hal ini sejalan dengan ajaran islam yang menjelaskan tentang adanya hal yang baik (haq) dan hal yang buruk (batil). Data traffic normal menunjukkan suatu hal yang baik (haq), sedangkan traffic malware menunjukkan hal yang buruk (batil). Dalam islam melakukan kejahatan seperti mencuri adalah hal yang batil, apalagi jika orang tersebut mengetahui bahwa hal tersebut adalah batil dan tetap melaukannya, maka itu juga semakin menambah dosa orang tersebut dan menimbulkan kerugian yang lebih pada orang lain. Hal ini disampaikan pada Al Qur'an Surat Al Baqarah ayat 188 sebagai berikut.

وَلَا تَأْكُلُوْٓا اَمْوَالَكُمْ بَيْنَكُمْ بِالْبَاطِلِ وَتُدْلُوْا بِهَآ اِلَى الْحُكَّامِ لِتَأْكُلُوْا فَرِيْقًا مِّنْ اَمْوَالِ النَّاسِ بِالْاِثْمِ وَاَنْتُمْ تَعْلَمُوْنَ

188. Janganlah kamu makan harta di antara kamu dengan jalan yang batil dan (janganlah) kamu membawa (urusan) harta itu kepada para hakim dengan

maksud agar kamu dapat memakan sebagian harta orang lain itu dengan jalan dosa, padahal kamu mengetahui.

Berdasarkan tafsir Jalalain dijelaskan dalam ayat tersebut memiliki makna larangan bagi manusia untuk memakan harta manusia yang lain melalui jalan yang batil, jalan yang haram menurut syariat islam, seperti melakukan pencurian, perampokan, mengintimidasi, menipu dan semacamnya. Sera larangan untuk menyuap hakim di pengadilan terhadap urusan harta tersebut, yang mana engkau mengetahui bahwa hal tersebut adalah sebuah dosa.

Menurut tafsir Ibnu Katsir, beliau menafsirkan bahwa tidak ada yang bisa merubah hukum sesuatu yang haram menjadi halal ataupun sebaliknya halal menjadi haram, namun hakim memiliki ikatan terhadap apa yang tampak darinya, dan siapapun yang melakukan tipu muslihat akan mendapatkan dosa.

Oleh karena itu, dengan adanya penelitian ini, menjadikan sebuah bentuk upaya untuk melakukan pencegahan pada serangan peretasan pada sistem cloud server. Harapan peneliti terhadap sistem yang telah dibuat agar menjadi suatu pilihan dalam upaya penanganan tepat terhadap pemasalahan peretasan jaringan komputer.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dari penelitian yang telah dilakukan, dapat diambil kesimpulan bahwa deteksi malware yang dilakukan dengan metode anomali dan menggunakan algoritma Isolation Forest dapat melakukan pendeteksian dengan cukup baik. Data yang digunakan dalam melakukan analisis adalah data terbaru dari dataset lalu lintas malware yaitu MTA-KDD'19 yang merupakan hasil penelitian Ivan Letteri et al berdasarkan data asli milik organisasi Stratosphere IPS. Data MTA-KDD'19 terdiri dari data legitimate dan data malware yang digabungkan menjadi satu data utuh. Data tersebut kemudian dibagi menjadi data latih dan data uji sebagai bahan pemodelan Isolation Forest. Dalam penelitian ini dapat diambil dua kesimpulan yang diuraikan sebagai berikut:

1. Dari hasil uji coba yang dilakukan terhadap data uji didapatkan hasil akurasi sebesar 47,30%; precision sebesar 93,47; recall sebesar 47,05%; dan f1-score sebesar 62,59%.
2. Sumber daya yang digunakan pada proses deteksi rata-rata menggunakan 33mb dan titik tertinggi penggunaan memori pada 64mb. Dan juga durasi dalam proses menjalankan deteksi adalah kurang lebih 5 detik.

5.2 Saran

Peneliti sadar jika di dalam tahapan-tahapan penelitian dari awal sampai akhir masih banyak terdapat kekurangan, baiknya dilakukan pengembangan serta perbaikan lebih lanjut sehingga bisa menghasilkan proses serta hasil yang lebih

baik. Berikut adalah saran dari peneliti berdasarkan proses penelitian yang dapat memiliki manfaat bagi pihak yang bersangkutan, yaitu:

1. Dalam pengolahan data serta pemilhan fitur diharapkan bisa lebih menganalisis dengan detail dan memperhatikan jenis serangan apa yang akan dideteksi dan memperhitungkan keseimbangan dataset.
2. Lebih memperhitungkan terkait variabel dari isolation forest seperti contamination, estimators, dan variabel lainnya agar lebih mampu meningkatkan kualitas pembacaan terhadap anomali.

DAFTAR PUSTAKA

Canzanese, R., Mancoridis, S., & Kam, M. (2012). System Call-based Detection of Malicious Processes.

Chun-Hui, X., Chen, S., Cong-Xiao, B., & Xing, L. (2018). Anomaly Detection in Network Management System Based on Isolation Forest. Proceedings - 2018 4th Annual International Conference on Network and Information Systems for Computers, ICNISC 2018, 56–60. https://doi.org/10.1109/ICNISC.2018.00019

Crysdian, C., Selamat, H., & Noor Md Sap, M. (2000). USER ACCESS CONTROL AND SECURITY MODEL.

Ericka, J., & Prakasa, W. (2020). Peningkatan Keamanan Sistem Informasi Melalui Klasifikasi Serangan Terhadap Sistem Informasi. Jurnal Ilmiah Teknologi Informasi Asia, 14(2).

Gao, R., Zhang, T., Sun, S., & Liu, Z. (2019). Research and Improvement of Isolation Forest in Detection of Local Anomaly Points. Journal of Physics: Conference Series, 1237(5). https://doi.org/10.1088/1742-6596/1237/5/052023

Graham, J., Howard, R., & Olson, R. (2011). CYBER SECURITY ESSENTIALS.

Hu, J. (2010). Host-Based Anomaly Intrusion Detection. Handbook of Information and Communication Security, 235–255. https://doi.org/10.1007/978-3-642-04117-4_13

Hall, J., Barbeau, M., & Kranakis, E. (2005). Anomaly-based intrusion detection using mobility profiles of public transportation users. 2005 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob'2005, 2, 17–24. https://doi.org/10.1109/wimob.2005.1512845

Hariyadi, M. A., Purnomo, M. H., Hariadi, M., Ketut, I., & Purnama, E. (2010). Lung Segmentation at Image X-ray for Detecting Cardio Thorax Ratio Using Max-Tree Filtering and Geometric Active Contour. Journal of Mathematic and Technology, ISSN, 2078-0257.

Hariyadi, M. A. (2017). Segmentasi Paru-Paru pada Citra X-Ray Thorax Menggunakan Distance Regularized Levelset Evolution (DRLSE). MATICS, 9(1), 48. https://doi.org/10.18860/mat.v9i1.4130

IEEE Power & Energy Society, IEEE Industry Applications Society, & Institute of Electrical and Electronics Engineers. (2019). Isolation Forest based Detection for False Data Attacks in Power Systems. 2019 IEEE PES Innovative Smart Grid Technologies Asia978-1-7281-3520-5/19/$31.00 ©2019 IEEE.

Jain, M., & Bajaj, P. (2014). Techniques in Detection and Analyzing Malware *Executables : A Review*. 3(5), 930–935.

Kozlak, J. (2007). Probabilistic Anomaly Detection Based On System Calls Analysis. Computer Science, 8(1), 93–93. https://doi.org/10.7494/csci.2007.8.3.93

Kumar, R., & Sharma, D. (2018). Signature-Anomaly Based Intrusion Detection Algorithm. Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018, (ICECA), 836–841. https://doi.org/10.1109/ICECA.2018.8474781

Letteri, I., Penna, G. della, di Vita, L., & Grifa, M. T. (2020). MTA-*KDD'19: A* Dataset for Malware Traffic Detection *.

Lim, C., & Lim, C. (2016). Sistem Pemantauan Ancaman Serangan Siber di Indonesia Generasi Baru IDSECCONF 2016.

Marinescu, D. C. (2018). Cloud Computing.

Murtaza, S. S., Khreich, W., & Hamou-lhadj, A. (2013). A host-based anomaly detection approach by representing system calls as states of kernel modules A Host-based Anomaly Detection Approach by Representing System Calls as States of Kernel Modules. (March 2015). https://doi.org/10.1109/ISSRE.2013.6698896

Nascimento, G., & Correia, M. (2014). Anomaly-based intrusion detection in software as a service Anomaly-based Intrusion Detection in Software as a Service. (June 2011). https://doi.org/10.1109/DSNW.2011.5958858

Schultz, M. G., Eskin, E., Zadok, E., & Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, 38–49. https://doi.org/10.1109/secpri.2001.924286.

Sharp, R. (2009). An Introduction to Malware Classification of Malware. Network, 1–28. https://doi.org/10.1017/CBO9780511623806.

Shetty, A. (n.d.). Two-layered anomaly based system implemented on a cloud server. 4(3), 821–826.

Somwanshi, A. A., & Joshi, P. S. A. (2016). Implementation of Anomaly baseds for Server Security. 285–288.

Tao, X., Peng, Y., Zhao, F., Zhao, P., & Wang, Y. (2018). A parallel algorithm for network traffic anomaly detection based on Isolation Forest. International Journal of Distributed Sensor Networks, 14(11). https://doi.org/10.1177/1550147718814471

Tony Liu, F., Ming Ting, K., & Zhou, Z.-H. (n.d.). Isolation Forest.

Vakili, M., Ghamsari, M., & Rezaei, M. (2020). Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification.

Vivek, S. (2012). Use of Anomaly baseds to Increase Awareness regarding Network Security. International Journal of Recent Technology and Engineering (IJRTE), 1(2), 171–175.

Zulfikar, A., Ariq Rahmani, F., & Azizah Kantor Wilayah Direktorat Jenderal Perbendaharaan Provinsi Bangka Belitung, N. (n.d.). DETEKSI ANOMALI MENGGUNAKAN ISOLATION FOREST PADA BELANJA BARANG PERSEDIAAN SATKER POLRI.