BAB II

LANDASAN TEORI

2.1 Machine Learning

Machine learning merujuk pada sebuah metode yang membuat komputer memiliki kemampuan dalam mempelajari dan melakukan sebuah pekerjaan secara otomatis. Proses *machine learning* dilakukan melalui algoritma tertentu, sehingga pekerjaan yang diperintahkan kepada komputer dapat dilakukan secara otomatis (Hairani, 2018).

Machine learning dilakukan melalui 2 fase, yaitu fase *training* dan fase *application*. Fase *training* adalah proses pemodelan dari algoritma yang digunakan akan dipelajari oleh sistem melalui *training* data, sedangkan fase *application* adalah proses pemodelan yang telah dipelajari sistem melalui fase *training* akan digunakan untuk menghasilkan sebuah keputusan tertentu, dengan menggunakan *testing* data. *Machine learning* dapat dilakukan dengan dua cara, yaitu *supervised learning* dan *unsupervised learning*. *Unsupervised learning* adalah pemrosesan *sample* data dilakukan tanpa mewajibkan hasil akhir memiliki bentuk yang sesuai dengan bentuk tertentu, dengan menggunakan beberapa *sample* data sekaligus. Penerapan *unsupervised learning* dapat ditemukan pada proses visualisasi, atau eksplorasi data. *Supervised learning* adalah pemrosesan *sample* data x akan diproses sedemikian rupa, sehingga menghasilkan *output* yang sesuai dengan hasil akhir y. *Supervised learning* dapat diterapkan pada proses klasifikasi (Hairani, 2018).

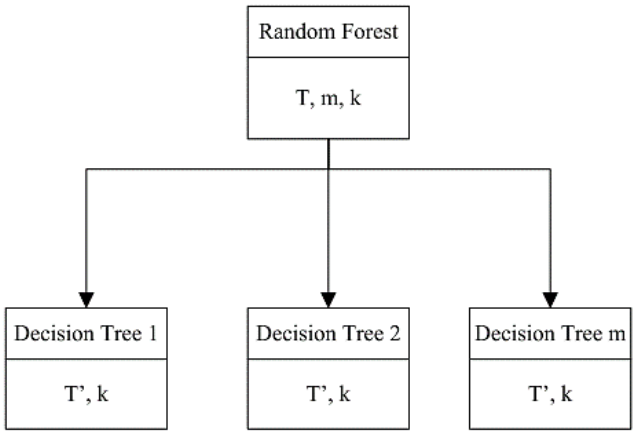
2.2 Random Forest

Random forest merupakan metode *bagging* yaitu metode yang membangkitkan sejumlah *tree* dari data *sample* dimana pembuatan satu *tree* pada saat *training* tidak bergantung pada *tree* sebelumnya kemudian keputusan diambil berdasarkan *voting* terbanyak (Wibowo, Saikhu, & Soelaiman, 2016).

Dua konsep yang menjadi dasar dari *random forest* adalah membangun *ensemble* dari *tree* via *bagging* dengan *replacement* dan penyeleksian fitur secara acak untuk tiap *tree* yang dibangun. Pertama, setiap *sample* yang diambil dari dataset untuk *training tree* bisa dipakai lagi untuk *training tree* yang lain. Kedua, fitur yang digunakan pada saat *training* untuk tiap *tree* merupakan subset dari fitur yang dimiliki oleh dataset (Wibowo et al., 2016).

Klasifikasi berbasis *ensemble* akan mempunyai performa yang maksimal jika antar *basic learner* mempunyai korelasi yang rendah. Sebuah *ensemble* harus membangun *basic learner* yang lemah, karena *learner* yang kuat kemungkinan besar akan mempunyai korelasi yang tinggi dan biasanya juga menyebabkan *overfit*, sedangkan *random forest* meminimalkan korelasi serta mempertahankan kekuatan klasifikasi dengan cara melakukan pengacakan pada proses *training*, yaitu dengan memilih sejumlah fitur secara acak dari semua fitur yang ada pada setiap melakukan *training tree*, kemudian menggunakannya menggunakan fitur-fitur yang terpilih untuk mendapatkan percabangan *tree* yang optimal. Berbeda dengan proses *training tree* pada *decision tree* biasa, proses *training tree* yang menjadi bagian dari *random forest* tidak menggunakan proses *pruning* akan tetapi percabangan akan terus dilakukan sampai ukuran batas *leaf* tercapai (Wibowo et al., 2016).

Random forest mempunyai dua parameter utama, yaitu: m jumlah *tree* yang akan dipakai dan k yaitu maksimal banyaknya fitur yang dipertimbangkan ketikan proses percabangan. Semakin banyak nilai m maka semakin bagus hasil klasifikasi, sedangkan untuk nilai k direkomendasikan sebesar akar kuadrat atau logaritma dari jumlah total fitur (Wibowo et al., 2016).



Gambar 2.1 Ilustrasi *Random Forest*

Sumber: (Wibowo et al., 2016)

Gambar 2.1 menunjukan proses *training* untuk *random forest* menggunakan dataset T dengan sejumlah m *tree* sebagai *basic leaner* dan k fitur yang dipilih secara acak dari total fitur yang ada untuk percabangan pada setiap *tree*. Proses *training* pada setiap *tree* menggunakan dataset T' yang merupakan hasil dari *bootstrap* dari dataset yang dijadikan parameter untuk *random forest*. *Bootstrap* merupakan proses memilih *sample* dari dataset yang akan digunakan proses *training tree*. Metode *ensemble, bootstrap* merupakan proses *sampling* dengan *replacement,* sehingga *sample* yang diambil untuk proses *training tree* yang satu masih bisa dipakai lagi untuk proses *training tree* yang lainnya (Wibowo et al., 2016).

2.3 Metode Anomali

Metode Anomali adalah sebuah pola dari kumpulan data yang tidak memiliki pola yang normal yaitu berbeda dengan data yang lainnya. Pola yang berbeda ini yang disebut dengan anomali atau *outliers*. Metode anomali dapat menemukan jenis *malware* baru karena *malware* tersebut memiliki kebiasaan yang terdeteksi sebagai hal yang unik atau berbeda dengan *malware* yang lainnya. Metode anomali memiliki kelebihan dan kekurangan, ditunjukkan pada tabel 2.4.

Tabel 2.1 Kelebihan dan Kekurangan Metode Anomali (Ismiyushar et al., 2018).

Kelebihan	Kekurangan
- Memajukan pemahaman mengenai environment yang digunakan. - Memberikan kesempatan yang lebih baik dalam menangkap penyerang sebelum melakukan serangan lebih jauh. - Mempersiapkan cara untuk menangani hal yang tidak terduga.	- Dapat terhambat dalam urusan sumber daya intensif - Proses manual yang dilakukan pada metode anomali dapat memperlambat proses yang dilakukan oleh sumber daya yang belum terlatih. - Membutuhkan ruang lingkup yang luas.

2.4 Malware

Malware adalah program komputer yang digunakan untuk melakukan aktivitas yang sifatnya merusak. Tujuan dari pembuat *malware* ini yaitu memasang *malware* pada target tujuan dan mendapatkan kontrol penuh terhadap *device* tersebut. *Malware* ini umumnya disebarkan menggunakan beberapa cara, diantaranya : *social engineering attack, email phishing* dan *file download fraud*. Cara tersebut bertujuan untuk membuat korban tertipu dan menyebabkan sistemnya terinfeksi *malware*. Target dari penyebaran *malware* ini, diantaranya: pencurian data rahasia,

mencari *username* dan *password*, *DDoS* dan *spam email*. *Impact* dari suatu *malware* terhadap korban yaitu gangguan pada *service* yang menyebabkan hilangnya produktivitas dan dapat menyebabkan hilangnya penghasilan.

2.5 Mirai

Mirai adalah malware yang mengubah perangkat yang terinfeksi menjadi *bot* untuk mengeksekusi serangan *DDoS*. *Mirai* menginfeksi perangkat *IoT* dengan akses jarak jauh yang diaktifkan melalui *telnet* dan nama pengguna dan kata sandi standar disimpan. *Mirai* dibagi menjadi tiga bagian, *Server CNC* menyediakan terminal virtual untuk pengguna *botnet*, menyimpan bukti *bot* terdaftar dan meneruskan perintah serangan kepada mereka. *Loader* mengunggah dan mengeksekusi *malware* pada perangkat yang dilaporkan rentan. *Bot* mencari target yang rentan dan mengeksekusi serangan *DoS* pada permintaan (Sinanovic & Mrdovic, 2017).

2.5.1 Bagian-bagian *Malware Mirai*

Berikut ini merupakan beberapa bagian *malware mirai*, diantaranya :

a. Bot

Bot ditulis seluruhnya dalam bahasa pemrograman *C*. Mulai dari *file main.c*, dapat dilihat bahwa *Mirai* menghapus *file exe*-nya setelah dijalankan dan akan tersimpan di *RAM*, hal tersebut merupakan salah satu cara untuk menghindari deteksi. *Malware* menonaktifkan pengawas waktu pada perangkat yang terinfeksi, mencegahnya dari *restart*. *Bot* tersebut memeriksa dan mematikan *malware* lain yang sudah berjalan pada perangkat yang sama. Nama acak untuk prosesnya dibuat

untuk membuat proses deteksi lebih sulit. *Bot* tersebut kemudian menjalankan fungsi *fork() system call* beberapa kali untuk membuat proses untuk setiap modul sampai *bot* itu terhubung ke *server CNC* dan menunggu perintah untuk dieksekusi. Modul lainnya berjalan di samping proses utama dan cara kerja modul tersebut ditunjukkan pada tabel 2.2.

Tabel 2.2 Cara Kerja Modul Bot Mirai (Sinanovic & Mrdovic, 2017)

Modul	Cara Kerja
Attack	*Attack* memecah modul perintah ketika diterima dan meluncurkan serangan *DoS*. Sepuluh metode serangan *DoS* diimplementasikan dalam sepuluh fungsi yang berbeda. Modul memutuskan fungsi mana yang dijalankan berdasarkan perintah yang dikeluarkan, dan menghentikan eksekusi begitu waktu durasi habis.
Killer	*Killer* mematikan proses yang menahan *port* 22, 23, 80, dan mencadangkan *port-port* ini untuk mencegah *restart* aplikasi yang dimatikan. *Killer* juga terus memindai memori yang mencoba menemukan dan mematikan *malware* serupa yang dibuat dan dijalankan oleh penyerang lainnya
Scanner	*Scanner* menggunakan *telnet* dan alamat IP publik yang dibuat secara acak untuk memeriksa perangkat *IoT* rentan lainnya. Nama pengguna dan kata sandi *Telnet* diambil dari tabel yang berisi 62 kombinasi standar pabrik, jika koneksi dengan perangkat acak berhasil dibuat, alamat IP perangkat *IoT* yang rentan dikirim ke *server* pelaporan dengan nama pengguna dan kata sandi yang cocok.

b. CNC Server

CNC server ditulis dalam bahasa pemrograman *Google Go* yang pertama terhubung ke *database MySql* menggunakan kredensial yang telah ditentukan. *CNC Server* menciptakan dua soket pendengaran, soket pertama mengambil *port* 23

untuk telnet dan soket lainnya mengambil *port* 101 untuk *API*. Koneksi dibuat oleh pengendali awal dan memutuskan koneksi tersebut dari pengguna terdaftar *CNC* atau registrasi *bot* baru (Sinanovic & Mrdovic, 2017).

c. Loader

Loader ditulis dalam bahasa pemrograman *C*. *Loader* pertama kali membuat *server* untuk mengunduh muatan yang telah dikompilasi untuk berbagai arsitektur menggunakan *wget* atau *TFTP* dari *busybox*. *Loader* kemudian mulai bertindak seperti *server* pelaporan, mendengarkan perangkat *IoT* yang rentan yang dapat dikompromikan, kemudian setelah informasi tentang target potensial diterima, *loader* akan menghubungkannya melalui *telnet*, mengunduh dan menjalankan *payload* terhadap perangkat yang dikompromikan, sehingga mengubahnya menjadi *bot* baru (Sinanovic & Mrdovic, 2017).

2.5.2 Serangan *DDoS Mirai*

Mirai botnet dan variannya melakukan puluhan ribu serangan *DDoS*, strategi di balik serangan ini mengkarakterisasi target mereka, dan menyoroti studi kasus tentang target *profil* tinggi seperti *Krebs on Security, Dyn, dan Lonestar Cell Liberia*. *Mirai* memiliki kemiripan dengan layanan *booter* (yang memungkinkan pelanggan membayar serangan *DDoS* terhadap target yang diinginkan), dengan beberapa operator *Mirai* menargetkan *platform game* populer seperti *Steam*, *Minecraft*, dan *Runescape* (Antonakakis et al., 2017).

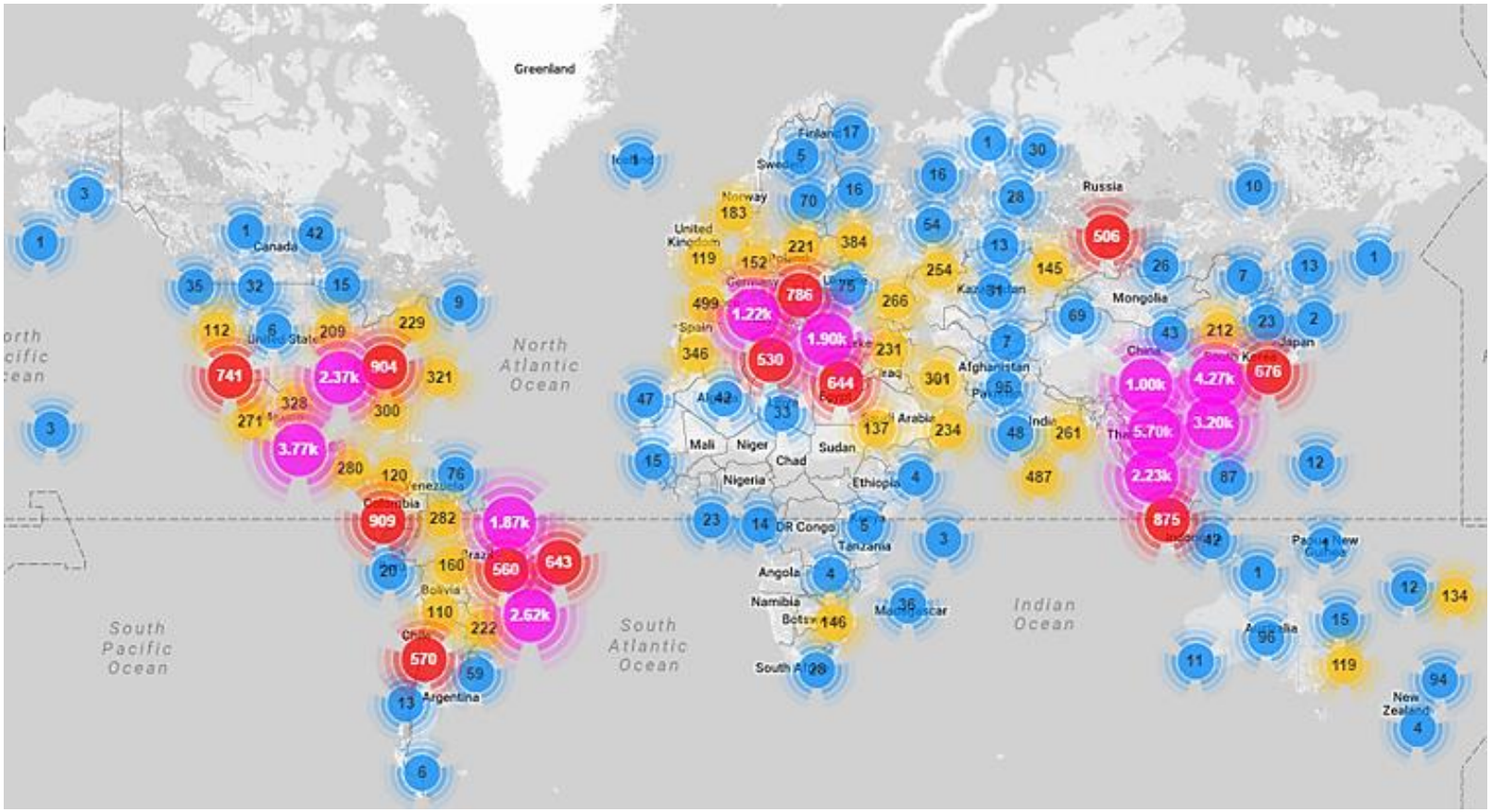
2.5.3 Jenis Serangan Mirai

Operator *Mirai* mengeluarkan 15.194 perintah serangan *DDoS*, tidak termasuk serangan duplikat. Serangan-serangan ini menggunakan strategi menghabiskan sumber daya yang berbeda : 32,8% adalah volumetrik, 39,8% adalah *TCP*, dan 34,5% adalah serangan lapisan aplikasi. *Mirai* meluncurkan 15.194 serangan antara 27 September 2016 - 28 Februari 2017, termasuk serangan *[A]pplication-layer*, *[V]olumetric*, dan menghabiskan *TCP [S]tate* exhaustion, yang semuanya merata. Rincian jenis serangannya ditunjukkan pada tabel 2.3.

Tabel 2.3 Jenis Serangan (Antonakakis et al., 2017)

Attack Type	*Attacks*	*Targets*	*Class*
HTTP flood	2,736	1,035	A
UDP-PLAIN flood	2,542	1,278	V
UDP flood	2,440	1,479	V
ACK flood	2,173	875	S
SYN flood	1,935	764	S
GRE-IP flood	994	587	A
ACK-STOMP flood	830	550	A
VSE flood	809	550	A
DNS flood	417	173	A
GRE-ETH flood	318	210	A

Alamat IP perangkat yang terinfeksi *Mirai* terlihat di 164 negara. Dibuktikan dengan peta pada Gambar 2.2, IP *botnet* tersebar di berbagai negara, dengan keterangan warna sebagai berikut : biru < 99 IP, kuning 100 < 499 IP, merah 500 < 999 IP dan merah muda > 1000 IP yang terinfeksi (Angrishi, 2017).



Gambar 2.2 Distribusi Mirai Botnet hingga 26 Oktober 2016

Sumber: (Angrishi, 2017)

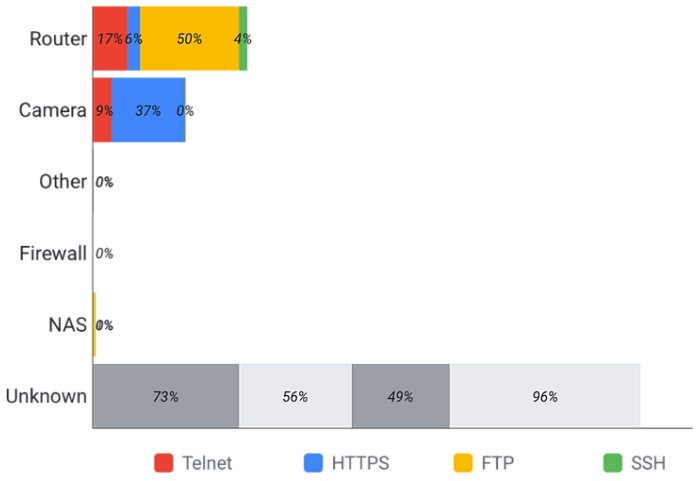
2.6 Internet of Thing

Internet of Things (IoT) adalah langkah evolusi besar berikutnya di dunia internet. *IoT* merupakan kunci dalam dunia digital kehidupan yang sudah saling terkoneksi (Angrishi, 2017). *IoT* sangat erat hubungannya dengan komunikasi mesin dengan mesin (M2M) tanpa campur tangan manusia ataupun komputer. Istilah *IoT* mulai dikenal tahun 1999 yang saat itu disebutkan pertama kalinya dalam sebuah presentasi oleh Kevin Ashton, *cofounder and executive director of the Auto-ID Center* di *MIT* (Limantara, Cahyo, Purnomo, & Mudjanarko, 2017).

Tujuan utama *Internet of Things* adalah untuk memungkinkan kehidupan yang lebih aman dan pengurangan risiko pada berbagai tingkat kehidupan. Daya tarik futuristik untuk membuat hidup sedikit lebih menyenangkan dalam rutinitas sehari-hari yang sibuk. Munculnya ponsel pintar, televisi pintar, dan perangkat pintar lainnya seperti *Amazon echo* dengan *Alexa* atau *Google Home*, sebagian besar ide di atas bukan bagian dari mimpi fiksi ilmiah lagi melainkan sudah menjadi kenyataan. Perangkat *IoT* memiliki beragam aplikasi, terutama dalam otomatisasi rumah (*smart home*), perawatan kesehatan, solusi energi pintar, kendaraan yang terhubung secara otonom dan sistem kontrol industri (Angrishi, 2017).

Ancaman terhadap *Internet of Things* menargetkan berbagai perangkat keras, termasuk kamera IP, *router* rumah, dan perangkat pintar. Ancaman ini umumnya mempengaruhi sistem berbasis *Linux* (Andrews, 2018). *Malware mirai* telah menginfeksi lebih dari 65.000 perangkat *IoT*. Mirai juga sudah menyumbang separuh dari semua pemindaian *telnet* internet, seperti yang ditunjukkan pada Gambar 2.3. Puncaknya pada November 2016 *Mirai* telah menginfeksi lebih dari

600.000 perangkat IoT. Layanan perangkat yang terinfeksi menggunakan pemindaian Internet *Censys* mengungkapkan bahwa sebagian besar perangkatnya adalah *router* dan kamera. *Mirai* yang aktif menghapus identifikasi yang menjelaskan tidak bisa melakukan identifikasi sebagian besar perangkat.



Gambar 2.3 Perangkat IoT yang Terinfeksi *Mirai*

Sumber: (Cloudflare, 2017)

Ancaman terhadap Internet of Things menargetkan berbagai perangkat keras, termasuk kamera IP, *router* rumah, dan perangkat pintar. Ancaman ini umumnya mempengaruhi sistem berbasis *Linux*.

Karakteristik dari *malware IoT* yang digunakan untuk mengatur serangan *DDoS* adalah sebagai berikut:

a. *Malware IoT* sebagian besar berbasis *Linux*.
b. *Malware IoT* memiliki efek samping pada kinerja *host*. *Malware IoT* menjadi aktif dan melakukan *DDoS* pada perintah tertentu dari para penggembala *botnet*.
c. *Malware IoT* berada di memori sementara (*RAM*) perangkat *IoT*.

d. Perangkat *IoT* biasanya tidak menggunakan teknik refleksi atau amplifikasi untuk meluncurkan serangan, sehingga sangat sulit untuk mengenali dan mengurangi serangan menggunakan metode konvensional.
e. Volume membanjiri lalu lintas yang dihasilkan oleh *bot IoT* sangat tinggi, dalam urutan 100 Gbps atau lebih tinggi, dibandingkan dengan *botnet PC* konvensional.
f. Lokasi perangkat *IoT* yang terinfeksi didistribusikan di seluruh dunia.
g. Selain membanjiri lalu lintas yang umum digunakan, yaitu, *HTTP*, *TCP*, lalu lintas *UDP*, beberapa *botnet IoT* menghasilkan lalu lintas tidak konvensional seperti lalu lintas *GRE* dan menggunakan teknik "*DNS water torture*" yang tidak biasa selama serangan *DDoS*.

2.7 Kajian Penelitian Terdahulu

Penelitian sebelumnya melakukan pendeteksian *botnet* menggunakan algoritma Algoritma *K-Nearest Neighbor* (k-NN). Penelitian tersebut menghasilkan akurasi tertinggi dari pengujian pada data *training* skenario 9 sebesar 97,27% dan menghasilkan akurasi terendah yaitu pengujian pada data *training* skenario 3 dengan nilai 84,84%. Persentase rata-rata dari enam pengujian adalah sebesar 92,57% (Hairani, 2018).

Penelitian lainnya mengenai perbandingan pendekatan pembelajaran mesin untuk mendeteksi lalu lintas botnet dengan membandingkan algoritma *Logistic Regression (LR), Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF)* dan *Neural Networks (NN)*. Penelitian tersebut menghasilkan

kesimpulan bahwa *Random Forest* adalah model yang superior dan terbukti lebih kuat daripada model lain dengan Skor F1 0.99 dan memiliki kinerja terbaik untuk deteksi anomali (Abraham et al., 2018).

Penelitian ini bertujuan untuk melakukan pendeteksian serangan *Malware Mirai (Scan, ACK, SYN, UDP, UDPplain)* terhadap arsitektur perangkat *IoT (security camera*) menggunakan metode *machine learning* dengan algoritma *Random Forest* untuk deteksi anomali.

Berikut merupakan *literature reviews* dari penelitian sebelumnya yang bersangkutan pada deteksi *botnet* ditunjukan pada tabel 2.4.

Tabel 2.4 Penelitian Terkait

No	**Peneliti**	**Metode / *Framework***	**Domain Penelitian**	**Hasil Penelitian**
1	Yonathan Satrio Nugroho Irwan Sembiring (2016)	Metode *Network Forensics*	Analisa *Network Forensics* Pada Serangan Botnet	Faktor yang mempengaruhi pola serangan botnet Zeus adalah banyaknya jumlah unique domain, intensitas domain host atau server, dan interval waktu.
2	Erick Lamdompak S (2016)	Algoritma *Support Vector Machine* (SVM)	Klasifikasi *Malware Trojan Ransomware* Dengan Algoritma *Support Vector Machine* (SVM)	N-grams mendeteksi malware yang di ektraks berdasarkan kode operasional yang sering muncul. Algoritma SVM menghasilkan klasifikasi *Malware* dengan normal file yang di pisahkan berdasarkan garis *Hyperlane*.

Tabel 2.4 Penelitian Terkait (lanjutan 1)

No	Peneliti	Metode / *Framework*	Domain Penelitian	Hasil Penelitian
3	Abdul Haris Muhammad Bambang Sugiantoro Ahmad Luthfi (2017)	Metode Analisis Dinamis dan Statis	Metode Klasifikasi Dan Analisis Karakteristik *Malware* Menggunakan Konsep Ontologi	Penerapan ontologi sebagai knowledge base dasar sangat dibutuhkan dalam melakukan analisis karakteristik *malware*.
4	Alexandre Dulaunoy Gérard Wagener Sami Mokaddem (2017)	Metode Analisis Statis	*An Extended Analysis of An Iot Malware from A Blackhole Network*	Menggali lebih dalam lalu lintas *blackhole*, pengamatan dapat dilakukan pada perilaku jenis malware IoT.
5	Georgios Kambourakisa Constantinos Koliasa Angelos Stavrou (2017)	Metode Analisis Statis	*The Mirai Botnet and the IoT Zombie Armies*	Mirai mengubah korbannya menjadi zombie yang melumpuhkan perangkat sampai pengguna memperhatikannya dan melakukan reboot.
6	Hamdija Sinanovi´c Sasa Mrdovic (2017)	Metode Analisis Dinamis	*Analysis of Mirai Malicious Software*	Perilaku jaringan yang sederhana, memungkinkan untuk membuat *signature* IDS pada Mirai. Hal tersebut menjadi cara terbaik dan termudah untuk mendeteksi dan menghentikan Mirai.
7	Lakshya Mathurb Mayank Rahejab Prachi Ahlawat (2018)	Metode Analisis Statis	*Botnet Detection via Mining of Network Traffic Flow*	Merekomendasikan Regresi Logistik karena dapat melakukan tugas-tugas dalam jumlah waktu yang menguntungkan dengan akurasi tinggi.

Tabel 2.4 Penelitian Terkait (lanjutan 2)

No	Peneliti	Metode / *Framework*	Domain Penelitian	Hasil Penelitian
8	Tika Hairani (2018)	Algoritma K-Nearest Neighbor	Deteksi Botnet Menggunakan Algoritma *K-Nearest Neighbor*	Akurasi tertinggi dari pengujian pada data training skenario 9 sebesar 97,27% dan hasil terendah pada data training skenario 3 dengan nilai 84,84%.
9	Naufal Abrian Ismiyushar M. Teguh Kurniawan Adityas Widjajarto (2018)	Metode Anomali	Analisis Dampak Malware Berdasarkan Api Call Dengan Metode Anomali	Kategorisasi dapat dilakukan dengan menggunakan malicious activity data set menggunakan parameter API calls.
10	B. Abraham A. Mandya Rohan Bapat Fatma Alali Don E. Brown Malathi Veeraraghavan (2018)	Algoritma *LR Naive Bayes SVM Random Forest* dan *Neural Networks*	*A Comparison of Machine Learning Approaches to Detect Botnet Traffic*	Random Forest adalah model superior dengan Skor F1 0.99. Random Forest terbukti lebih kuat daripada model lainnya
11	Yair Meidan Michael Bohadana Yael Mathov Yisroel Mirsky Asaf Shabtai Dominik Breitenbacher Yuval Elovici (2018)	Algoritma *Support Vector Machine (SVM), Local Outlier Factor (LOF*) dan *IsolationForest*	*N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders*	Menghasilkan FPR nol pada sebagian besar perangkat IoT dalam satu set tes.

Tabel 2.4 Penelitian Terkait (lanjutan 3)

No	Peneliti	Metode / *Framework*	Domain Penelitian	Hasil Penelitian
12	Najah Ben Said Fabrizio Biondi Vesselin Bontchev Olivier Decourbe Thomas Given-Wilson AxelLegay Jean Quilbeuf (2018)	Metode Analisis Sintaksis dan Analisis Perilaku	*Detection of Mirai by Syntactic and Behavioral Analysis*	Analisis perilaku ditemukan lebih efektif dalam mendeteksi sampel Mirai daripada analisis sintaksis, dengan analisis perilaku memiliki skor F0,5 hingga 99,41% terhadap skor F0,5. 98,61% dari analisis sintaksis.
13	Denar Regata Akbi Arini R Rosyadi (2018)	Alogritma *k-Nearest Neighbor* (k-NN)	Analisis Klasterisasi *Malware*: Evaluasi Data Training Dalam Proses Klasifikasi *Malware*	Data latih yang dihasilkan melalui proses klastering menggunakan frekuensi system call malware lebih akurat dibandingkan dengan data latih yang dihasilkan dengan menggunakan suatu penamaan malware.
14	K. Nguyen T. Tuan Son Hai Le Anh Phan Viet M. Ogawa N. Minh (2018)	Metode Analisis Statis	*Comparison of Three Deep Learning-based Approaches for IoT Malware Detection*	Pendekatan berbasis CNN bekerja dengan cukup baik, mungkin sebagian karena malware IoT tidak menggunakan teknik kebingungan
15	S Megira A R Pangesti F W Wibowo (2018)	Metode Analisis Dinamis Statis dan Reverse Engineering	*Malware Analysis and Detection Using Reverse Engineering Technique*	Reverse Engineering adalah teknik yang sesuai untuk digunakan dalam menganalisis malware.

Satu dari lima belas penelitian pada tabel 2.4, mendekati penelitian yang akan dilakukan dengan judul "**Pengolahan Data *Traffic* pada Perangkat *Internet of Things* dengan menggunakan Algoritma *Random Forest",*** penetian yang paling mendekati ditunjukkan pada tabel 2.5.

Tabel 2.5 Penelitian yang Mendekati

No	Peneliti	Metode / *Framework*	Domain Penelitian	Hasil Penelitian
1	B. Abraham A. Mandya Rohan Bapat Fatma Alali Don E. Brown Malathi Veeraraghavan (2018)	Algoritma LR Naive Bayes SVM Random Forest dan Neural Networks	*A Comparison of Machine Learning Approaches to Detect Botnet Traffic*	Random Forest adalah model superior dengan Skor F1 0.99. *Random Forest* terbukti lebih kuat daripada model lainnya.
2	Sugih Pamela (2019)	Algoritma Random Forest	Pengolahan Data *Traffic* pada Perangkat *Internet of Things* dengan menggunakan Algoritma *Random Forest*	Mengukur tingkat akurasi *Random Forest* dalam mendeteksi serangan botnet Mirai (*Scan, ACK, SYN, UDP* dan *UDPplain*) pada arsitektur perangkat IOT (*Internet of Things)* berjenis *security camera*

Tabel 2.5 merupakan penelitian terkait yang telah dilakukan sebelumnya mengenai deteksi *botnet.* Penelitian tersebut berjudul "*A Comparison of Machine Learning Approaches to Detect Botnet Traffic*" (Abraham et al., 2018) yaitu penelitian dengan melakukan perbandingan pembelajaran mesin dalam mendeteksi lalu lintas *botnet* menggunakan dataset CTU-13. Lalu lintas jaringan yang digunakan ditangkap dalam bentuk *log* koneksi, yang dihasilkan oleh *framework*

pemantauan jaringan populer yang disebut Bro. Penelitian tersebut membandingkan kinerja beberapa algoritma, yaitu *Logistic Regression (LR), Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF) dan Neural Networks (NN),* algoritma tersebut ditujukan untuk mendeteksi serangan botnet menggunakan metode deteksi anomali. Penelitian tersebut menggunakan 8 jenis *botnet/ malware (Zeus, Conficker, Dridex, Necurs, Miuref, Bunitu, Upatre* dan *Trickbot).* Penelitian tersebut menghasilkan kesimpulan bahwa algoritma terbaik untuk mendeteksi lalu lintas *botnet* adalah menggunakan *Random Forest,* kecuali untuk botnet *Bunitu* dan *Zeus* yang menghasilkan algoritma *Logistic Regression* menjadi algoritma yang paling tepat untuk mendeteksinya.

Penelitian dengan judul *"A Comparison of Machine Learning Approaches to Detect Botnet Traffic*" cukup sebagai acuan dan sudah memenuhi aspek yang dibutuhkan untuk penelitian yang akan dilakukan, terutama penelitiannya cukup mendekati dalam hal teknik dasar yang akan digunakan dengan judul penelitian "Pengolahan Data *Traffic* pada Perangkat *Internet of Things* dengan menggunakan Algoritma *Random Forest*".

Perbedaan dengan penelitian yang akan dilakukan terdapat pada jenis *botnet/ malware* yang digunakan yaitu menggunakan *Mirai,* selain itu penelitian ini menggunakan dataset publik yang berbeda dimana jenis serangannya juga berbeda. Pada penelitian lain menggunakan dataset dari *log* lalu lintas jaringan, sedangkan penelitian yang akan dilakukan datasetnya berupa *log* serangan *botnet* pada arsitektur perangkat *Internet of Things* yaitu berjenis *security camera*. Keterbaruan penelitian yang akan dilakukan, dirangkum pada matrik penelitian dalam tabel 2.6.

Tabel 2.6 Matrik Penelitian

No	Judul Jurnal	Ruang Lingkup Penelitian																				Penulis
		Malware/ Botnet								Algoritma							Dataset		Deteksi			
		Zeus	***Conficker***	***Bunitu***	***Trickbot***	***Neris***	***Virut***	***NSIS.ay***	***Mirai***	***LR***	***NB***	***SVM***	***RF***	***NN***	***k-NN***	***gSpan***	***CTU-13***	***UCI***	***Signatured***	***Specification***	***Anomaly***	
1	Analisa Network Forensics Pada Serangan Botnet																					Yonathan Satrio Nugroho (2016)
2	Klasifikasi Malware Trojan Ransomware Dengan Algoritma SVM											✓										Erick Lamdompak S (2016)
3	Metode Klasifikasi Dan Analisis Karakteristik Malware Menggunakan Konsep Ontologi																		✓			Abdul Haris Muhammad (2017)
4	An Extended Analysis of An Iot Malware from A Blackhole Network								✓													Alexandre Dulaunoy (2017)
5	The Mirai Botnet and the IoT Zombie Armies								✓													Georgios Kambourakisa (2017)
6	Analysis of Mirai Malicious Software								✓										✓			Hamdija Sinanovi´c (2017)
7	Botnet Detection via mining of network traffic flow									✓							✓					Lakshya Mathurb (2018)

Tabel 2.6 Matrik Penelitian (lanjutan 1)

No	Judul Jurnal	Ruang Lingkup Penelitian																				Penulis
		Malware/ Botnet								Algoritma							Dataset		Deteksi			
		Zeus	***Conficker***	***Bunitu***	***Trickbot***	***Neris***	***Virut***	***NSIS.ay***	***Mirai***	***LR***	***NB***	***SVM***	***RF***	***NN***	***k-NN***	***gSpan***	***CTU-13***	***UCI***	***Signatured***	***Specification***	***Anomaly***	
8	Deteksi Botnet Menggunakan Algoritma k-NN					✓	✓	✓							✓		✓					Tika Hairani (2018)
9	Analisis Dampak Malware Berdasarkan Api Call Dengan Metode Anomali																				✓	Naufal Abrian Ismiyushar (2018)
10	A Comparison of Machine Learning Approaches to Detect Botnet Traffic	✓	✓	✓	✓					✓	✓	✓	✓	✓			✓				✓	Brendan Abraham (2018)
11	N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders								✓			✓									✓	Yair Meidan (2018)
12	Detection of Mirai by Syntactic and Behavioral Analysis								✓							✓			✓			Najah Ben Said (2018)
13	Analisis Klasterisasi Malware: Evaluasi Data Training Dalam Proses Klasifikasi Malware														✓				✓	✓	✓	Denar Regata Akbi (2018)
14	Comparison of Three Deep Learning-based Approaches for IoT Malware Detection																		✓			Khanh Duy Tung Nguyen (2018)

Tabel 2.6 Matrik Penelitian (lanjutan 2)

No	Judul Jurnal	Ruang Lingkup Penelitian																				Penulis
		Malware/ Botnet								Algoritma							Dataset		Deteksi			
		Zeus	***Conficker***	***Bunitu***	***Trickbot***	***Neris***	***Virut***	***NSIS.ay***	***Mirai***	***LR***	***NB***	***SVM***	***RF***	***NN***	***k-NN***	***gSpan***	***CTU-13***	***UCI***	***Signatured***	***Specification***	***Anomaly***	
15	Malware Analysis and Detection Using Reverse Engineering Technique																		✓			S Megira (2018)
16	Pengolahan Data *Traffic* pada Perangkat *Internet of Things* dengan menggunakan Algoritma *Random Forest*								✓				✓					✓			✓	Sugih Pamela (2019)

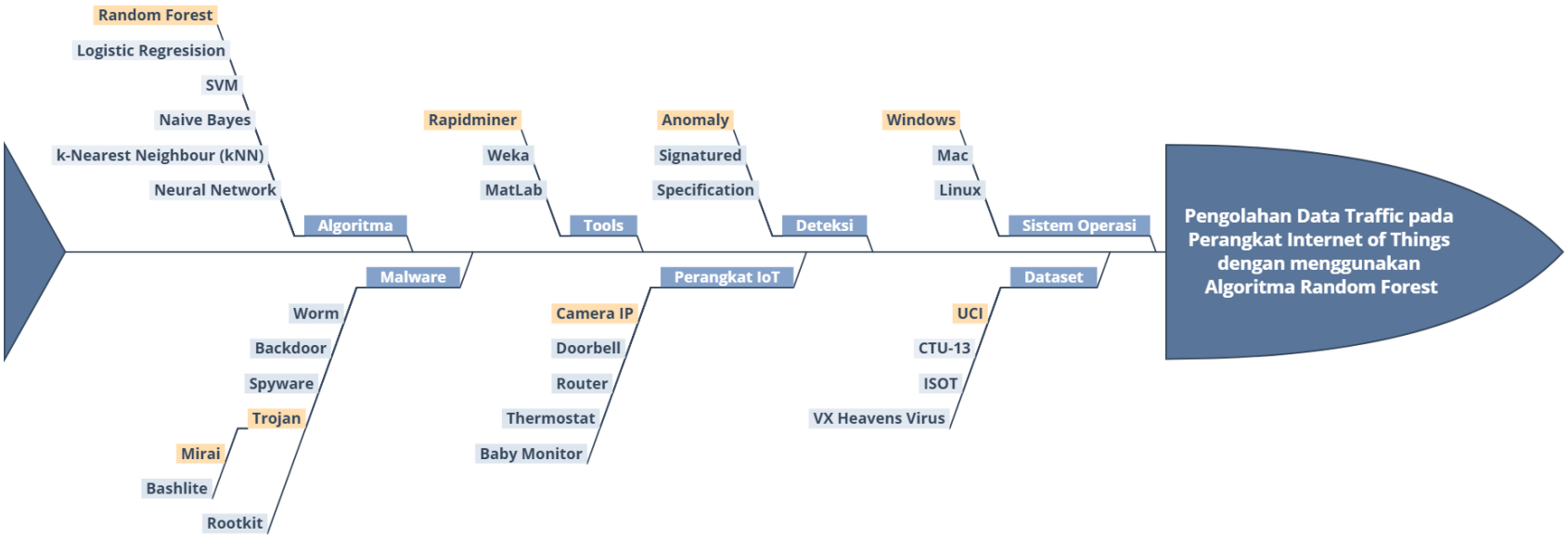
Tabel 2.6 menunjukkan matrik penelitian terkait mengenai deteksi malware yang telah dilakukan sebelumnya. Terdapat 16 matriks yang disusun secara berurut berdasarkan tahun jurnal tersebut dipublikasi. Ruang lingkup penelitian dibagi menjadi 4 aspek, yaitu : *Malware/ Botnet* (*Zeus, Conficker, Bunitu, Trickbot, Neris, Virut, NSIS.ay, Mirai*) Algoritma (*Logistic Regression, Naive Bayes, Support Vector Machine, Random Forest, Neural Network, gSpan*), Dataset (CTU-13, UCI) dan Deteksi (*Signatured, Specification, Anomaly*).

Tabel 2.7 Matrik Penelitian Terdekat

No	Judul Jurnal	Ruang Lingkup Penelitian																				Penulis
		Malware/ Botnet								Algoritma							Dataset		Deteksi			
		Zeus	***Conficker***	***Bunitu***	***Trickbot***	***Neris***	***Virut***	***NSIS.ay***	***Mirai***	***LR***	***NB***	***SVM***	***RF***	***NN***	***k-NN***	***gSpan***	***CTU-13***	***UCI***	***Signatured***	***Specification***	***Anomaly***	
1	*A Comparison of Machine Learning Approaches to Detect Botnet Traffic*	✓	✓	✓	✓					✓	✓	✓	✓	✓			✓				✓	Brendan Abraham (2018)
2	Pengolahan Data *Traffic* pada Perangkat *Internet of Things* dengan menggunakan Algoritma *Random Forest*								✓				✓					✓			✓	Sugih Pamela (2019)

Tabel 2.7 menunjukkan matrik penelitian yang terdekat. Penelitian yang dilakukan mempunyai kesamaan dalam aspek algoritma dan pendeteksiannya, yaitu menggunakan Algoritma *Random Forest* dan *anomaly detection*. Keterbaruan dari penelitian yang dilakukan terdapat pada jenis *Mawlare/ Botnet* dan sumber dataset yang berbeda, yaitu penelitiannya menggunakan *Malware Mirai* dan menggunakan dataset *UCI Repository*.

Berikut ini gambar diagram *fishbone* pada penelitian ini:



Gambar 2.5 Diagram *Fishbone*

Gambar 2.5 menunjukkan diagram tulang ikan (*fishbone*) penelitian, diagram ini menggambarkan hal-hal yang terkait dengan penelitian dan rencana *output* penelitian. Klasifikasi *malware* yang akan diteliti adalah *malware* berjenis *Trojan* yaitu *botnet Mirai*, yaitu serangan pada perangkat *Internet of Things* berjenis *security camera*, pendeteksian yang dilakukan menggunakan teknik *anomaly detection*, pengujiannya dilakukan pada dataset dari *UCI Repository*. Algoritma *Random Forest* digunakan untuk proses klasifikasinya menggunakan aplikasi *Rapidminer* pada sistem operasi *Windows*.