DIREKTORAT OPERASI KEAMANAN SIBER
NATIONAL CSIRT OF INDONESIA
Id-SIRTII/CC
INDONESIA SECURITY INCIDENT RESPONSE TEAM ON INTERNET INFRASTRUCTURE COORDINATION CENTER

PANDUAN KEAMANAN

YARA RULES

Direktorat Operasi Keamanan Siber
Badan Siber dan Sandi Negara
2022

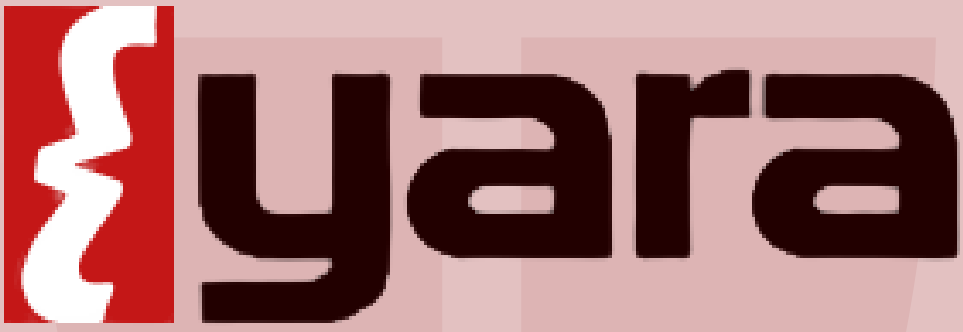
TLP: WHITE

DAFTAR ISI

YARA

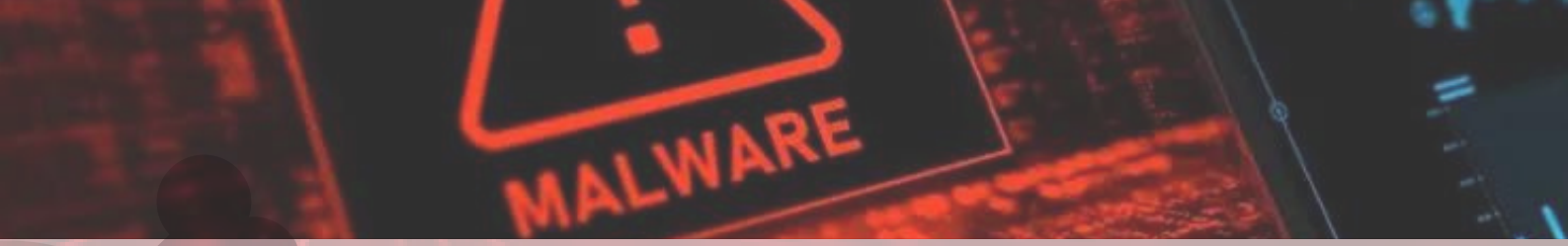
Tentang

YARA merupakan singkatan dari *Yet Another Recursive Acronym* atau *Yet Another Ridiculous Acronym* berupa *tools* yang bersifat *Open Source* yang dibuat oleh Victor Alvarez dari VirusTotal. YARA digunakan untuk mengidentifikasi dan mengklasifikasikan *malware* melalui penggunaan aturan berbasis *signature* dan karakteristik target lainnya yang dapat dijalankan pengguna terhadap *file* dan YARA *rules* yang lebih maju, seperti mencari data di alamat *virtual memory* tertentu dalam proses yang sedang berjalan.



Kegunaan

Dengan YARA, pengguna dapat membuat deskripsi *malware family* (atau apa pun yang ingin digambarkan/klasifikasikan) berdasarkan pola tekstual atau biner. Setiap deskripsi, identitas, rules, terdiri dari serangkaian *string* dan ekspresi *boolean* yang menentukan logikanya.



FUNGSI YARA

1. *Filtrasi: Sorting*
sejumlah *file malware* untuk memudahkan pencarian

2. Analisis Email
Yara *rules* dapat digunakan untuk melakukan deteksi terhadap *email malicious*

3. SIEM
Yara *Rules* merupakan salah satu metode yang tepat untuk mendeteksi, mengklasifikasikan, dan mencari *malware* di SIEM

4. *Threat Hunting*
Yara *Rules* dapat digunakan untuk melakukan analisis *file*, ekstraksi *Indicator of Compromise* yang terdapat pada *file* berbahaya, biner, dan program yang dicurigai

5. Analisis *Memory*
Yara *Rules* dapat mengidentifikasi proses yang mencurigakan atau berbahaya yang terdapat di memori komputer

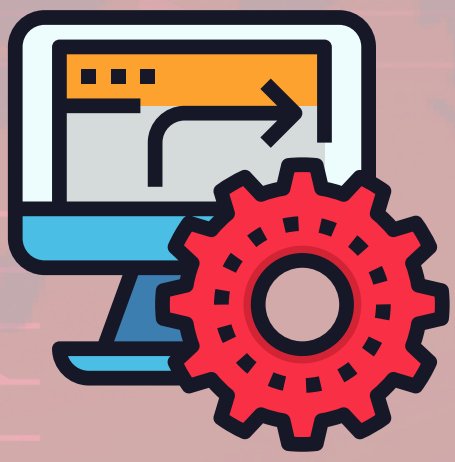
JENIS YARA

MANUAL DEPLOY

Pengguna dapat menentukan segala sesuatu yang terdapat pada YARA *Tools* untuk melakukan identifikasi, analisis maupun melakukan fungsi dari YARA *Tools* lainnya. Pengguna dapat mengatur *rules* sesuai dengan kebutuhan yang diinginkan ketika hendak menjalankan YARA *Tools*.

AUTOMATICALLY GENERATE RULES

Terdapat *rules generator* antara lain yargen, yara *generator*, yabin, dan joe sandbox. Yargen merupakan *tools* berbasis python untuk menghasilkan yara *rules* yang dikembangkan oleh Florian Roth. *Tools* ini menghasilkan yara *rules* dengan menggunakan teknik fuzzy dan naïve bayes.



MANUAL DEPLOY

MEMBUAT FILE YARA RULES

Yara *rules* dapat disimpan dalam *file* teks biasa yang dapat dibuat menggunakan berbagai macam editor teks.

File Yara *rules* memenuhi persyaratan berikut sebelum dimasukkan ke *virtual machine* :

- Nama *file* harus unik
- Konten *file* tidak boleh kosong

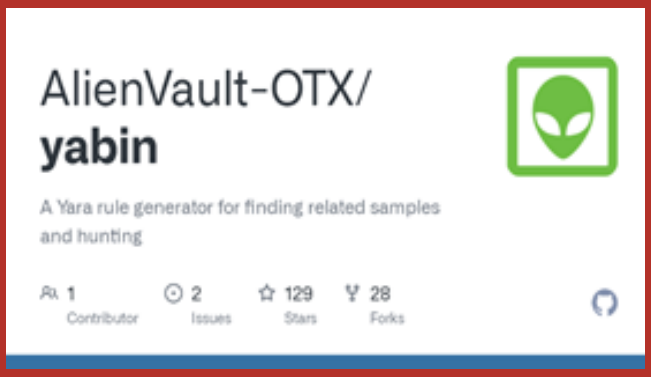
CONTOH SEDERHANA YARA RULES

```
rule NumberOne
{
meta:
desc = "Sonala"
weight = 10
strings:
$a = {6A 40 68 00 30 00 00 6A 14 8D 91}
$b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
$c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
condition:
$a or $b or $c
}
```

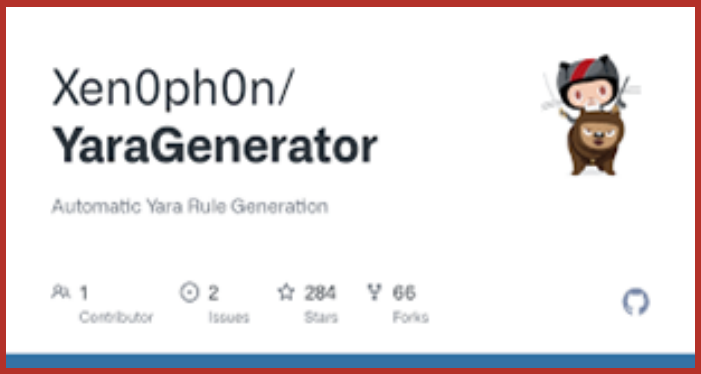
Tata Cara :

1. Klik tambahkan pada **tambahkan *file* YARA *rules***
   tambahkan YARA *rules* akan muncul
2. Setelah windows tambahkan YARA *rules* terbuka, konfigurasikan berikut ini :
   Rules File : Jelajahi dan pilih YARA *rules* untuk ditambahkan
   Analisis *File* : Tentukan jenis *file* yang diproses *Virtual Machine*.
3. Klik tambahkan apabila menyetujui YARA *rules* dan jenis *file* yang dianalisis.

AUTOMATICALLY GENERATE RULES



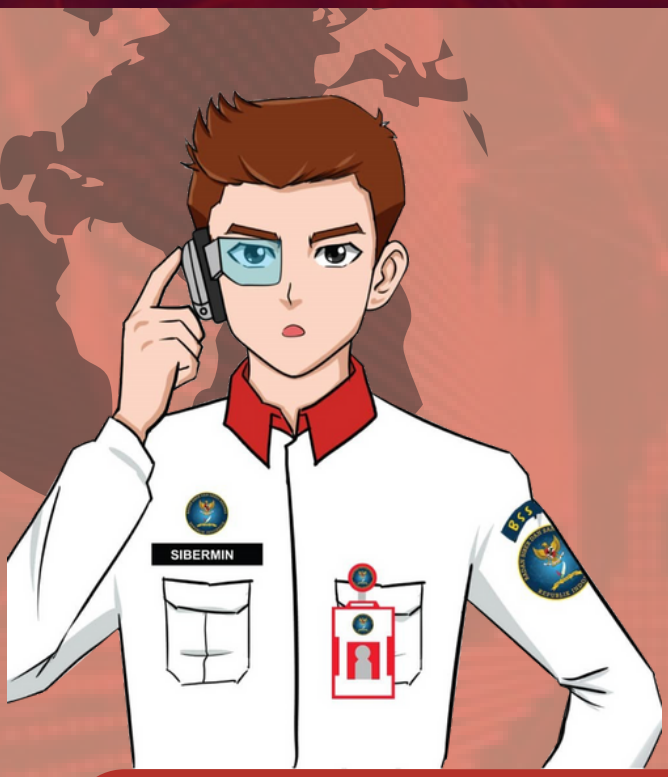
Yabin adalah *tools* berbasis Python untuk menghasilkan YARA *rules* yang dikembangkan oleh komunitas Alien Vault *Open Threat Exchange* (OTX). *Tools* ini menghasilkan YARA *Rules* dengan menemukan fungsi dalam sampel *malware* atau *family* dari sebuah *malware*.



YaraGenerator merupakan *tools* berbasis python untuk pengembangan yara *rules* yang diciptakan oleh Chris Clark. *Tools* ini menghasilkan yara *rules* dengan *signature* untuk berbagai jenis *file* seperti EXE, PDF, dan *Email* menggunakan logika *prioritization logic* dan *code refactoring*.

Joe Sandbox Cloud adalah penganalisis *malware* yang mengeksekusi *file* sampel di lingkungan yang dikontrol. *Tools* ini akan memantau perilaku dari *file* sampel tersebut untuk kemudian menghasilkan laporan komprehensif dalam berbagai format.

MALWARE



MALICIOUS SOFTWARE

Pengertian

"Perangkat lunak yang dibuat dengan tujuan memasuki, mengganggu, merusak hingga melakukan eksploitasi sistem, jaringan, atau server tanpa diketahui oleh pemiliknya yang dapat menyebabkan kerugian."

Beberapa jenis *malware* dapat mencari jalan sendiri untuk menginfeksi perangkat dan beberapa lagi harus dikirimkan ke perangkat tersebut. Tujuan adanya serangan *malware* ini pun bervariasi dari mulai melacak data hingga dapat menyandera perangkat demi tebusan atau merusaknya tanpa alasan yang jelas. Jumlah *malware* saat ini sangat banyak hingga nyaris tak terhingga.

malware

JENIS MALWARE

1. *Virus*

"Suatu program yang memiliki kemampuan untuk merusak *file* dalam sistem komputer. Virus biasanya tidak dapat menginfeksi komputer dengan sendirinya tetapi membutuhkan pengguna untuk menjalankan program yang disusupinya seperti melalui *email* untuk menyebarkan virus ke komputer lain, atau bahkan menghapus apa pun yang ada di dalam *harddisk*. Kerusakan yang disebabkan virus berkisar dari gangguan kecil hingga besar atau kehilangan data secara keseluruhan."

2. *Worm*

"Program yang memiliki kemampuan untuk mereplikasi diri. Tujuan dari worm adalah meningkatkan populasi dan mentransfer dirinya ke komputer lain melalui internet, media penyimpanan. Worm bertujuan untuk memperlambat kinerja dari sistem kemudian menyebabkan kerusakan langsung pada sistem."

3. *Trojan Horse*

"Perangkat lunak yang dapat merusak sebuah sistem atau jaringan. Tujuan dari trojan adalah memperoleh informasi dari target (*password*, kebiasaan *user* yang tercatat dalam *log system*, data, dan lain-lain), membuat *backdoor*, serta mendapatkan kendali aksesnya untuk masuk ke dalam sistem."

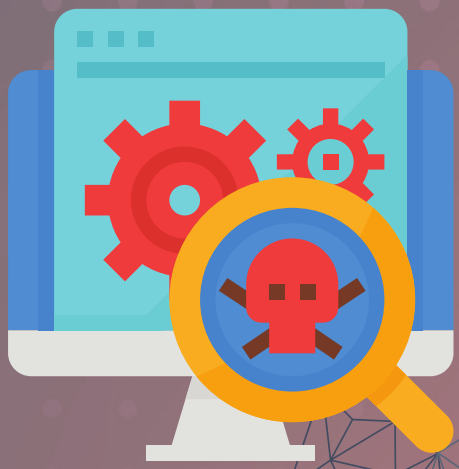
JENIS MALWARE

4. *Ransomware*

"Malware yang dapat membatasi pengguna untuk melakukan pengoperasian dan mengunci/mengenkripsi sistem. Ransomware akan menampilkan pesan peringatan untuk meminta uang tebusan dengan jaminan terbukanya enkripsi sehingga komputer bisa kembali ke kondisi normal. Malware ini menginfeksi melalui kerentanan pada sistem."

5. *Rootkit*

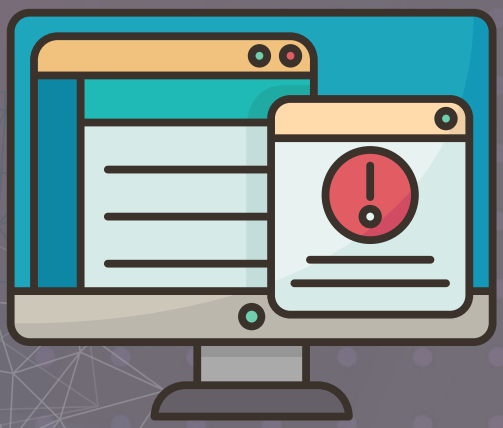
"Kumpulan perangkat lunak yang dirancang untuk memberikan akses tidak sah ke komputer dengan cara menutupi keberadaannya dan bertindak seolah-olah proses yang terpercaya sehingga memungkinkan proses tersebut tetap berjalan tanpa terdeteksi sebagai aktivitas yang berbahaya."



6. *Spyware*

"*Malware* yang mampu mengumpulkan informasi tanpa sepengetahuan target dan mengirimkan informasi tersebut ke pengguna jarak jauh tanpa otorisasi dari target. *Malware* ini juga dapat mengunduh dan menginstal aplikasi berbahaya lain tanpa otorisasi dari target."

JENIS MALWARE



7. *Adware*

"Adware merupakan perangkat lunak berbahaya yang berfokus kepada menghasilkan uang daripada mengancam komputer target. Perangkat lunak ini aktif menampilkan banner iklan di situs web dan jendela aplikasi. Gejala utamanya adalah munculnya iklan *pop-up* di desktop atau program, situs web atau aplikasi yang sebelumnya tidak menampilkannya."

8. *Keylogger*

"Keylogging merupakan proses melacak tombol yang ditekan pengguna dalam menemukan kata sandi sensitif ataupun memantau komunikasi pribadi. Berbagai jenis keylogger seperti *stalkerware* atau *hardware* tentunya membuat *malware* ini sangat sulit untuk ditemukan."

1. Melakukan pengecekan *source code malware* tanpa mencoba untuk mengeksekusi/menjalankan.
2. Menggunakan pendekatan *signature-based* untuk melakukan analisis.
3. Melakukan analis berdasarkan *fingerprint* dari *file*, *virus scanning*, menganalisis artefak memori, dan mendeteksi *obfuscation*.
4. Efektif dalam menganalisis *malware* yang sudah pernah terdeteksi sebelumnya.
5. *Tools* : Virustotal, Hybrid Analysis, String Analysis, dan Yara Rules.

Metode Analisis *Malware*

Analisis Statis

Analisis Dinamis

1. Melakukan eksekusi *source code* melalui virtual sandbox.
2. Menggunakan pendekatan *behaviour* untuk melakukan analisis.
3. Efektif menganalisis semua *malware* baik yang sudah terdeteksi dan belum terdeteksi.
4. Melakukan analisis melalui perubahan *registry, network* dan *system call,* API *call,* dan *memory write.*
5. *Tools* : Regshot, Process Monitor, Process Explorer, IDA Pro, CaptureBAT.

PENYEBARAN MALWARE

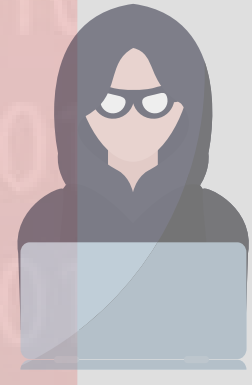
EMAIL

Metode penyebaran paling populer adalah melalui *email*. Spam sederhana maupun serangan *phishing* canggih masih dapat mengelabui orang untuk mengklik tautan atau mengunduh lampiran berisi *malware*.

PENJELAJAHAN

Jika mengklik *pop-up* atau iklan yang ditemui saat menjelajahi internet, maka akan ada kemungkinan masuk ke situs web berbahaya dan mengunduh *malware* di latar belakang dan keylogger/trojan akan menyusup ke perangkat.

REKAN KERJA



Beberapa *malware* menyebar dari perangkat satu ke perangkat lainnya melalui jaringan internal yang dipakai oleh setiap komputer di ruangan tersebut.

PERANGKAT LUNAK

Malware dikenal melakukan *piggybacking* (menyusup dalam aplikasi). Seseorang bisa saja tanpa sadar menginstal *malware* bersama dengan perangkat lunak yang sah.

IMPLEMENTASI YARA

YARA DENGAN ZEEK

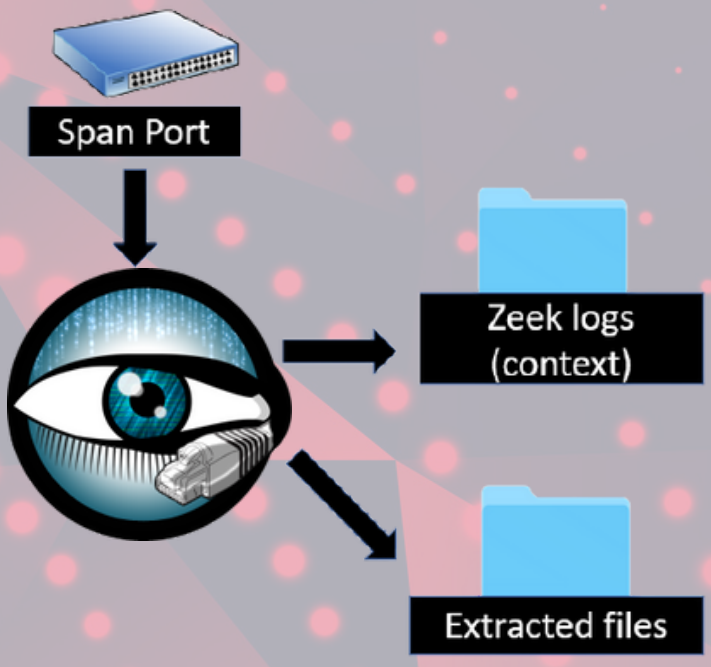
Zeek merupakan *tools open source* yang dapat digunakan untuk mendukung YARA dalam melakukan deteksi terhadap file berbahaya (malware) karena Zeek dapat melakukan deteksi berbasis jaringan.

Zeek Netwrok Security Monitoring yang sebelumnya dikenal dengan istilah Bro memiliki sensor untuk mendeteksi berbagai macam *event* dan dapat membuat *log* pada lalu lintas jaringan. Selain itu, Zeek dapat melakukan ekstraksi file secara langsung dari jaringan.

INTEGRASI YARA DENGAN ZEEK

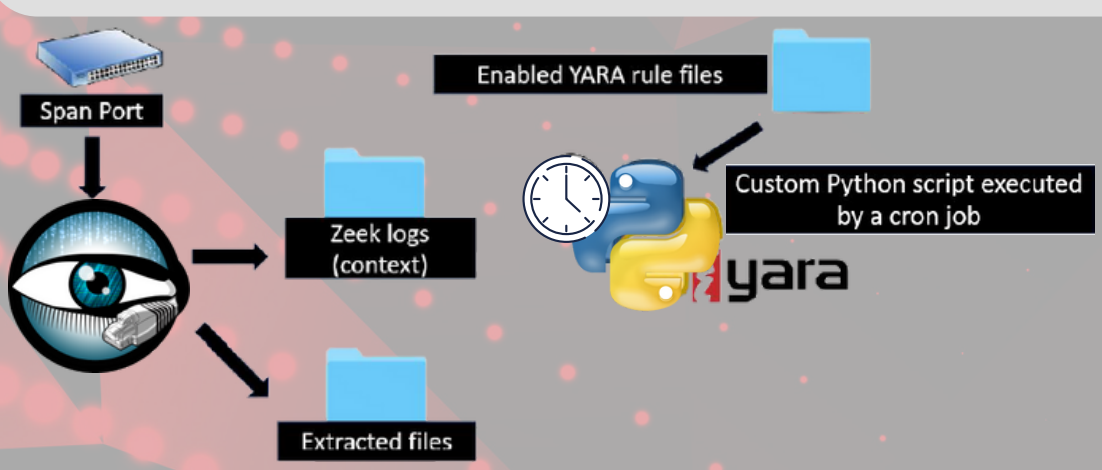
Menurut David Bernal Michelena penerapan Zeek dan Yara dapat didefinisikan sebagai sistem untuk mendeteksi *malware* di jaringan. Skema yang dilakukan adalah Zeek akan bertindak sebagai IDS *tools* atau sensor dan Yara yang akan melakukan deteksi terhadap serangan dengan menggunakan berbagai macam rules yang diterapkan. Langkah kerja rincinya sebagai berikut:

1. Komponen pertama adalah sensor Zeek. Sensor Zeek akan menerima lalu lintas untuk diperiksa, kemudian menghasilkan *log* jaringan dan mengekstrak file ke folder tertentu.



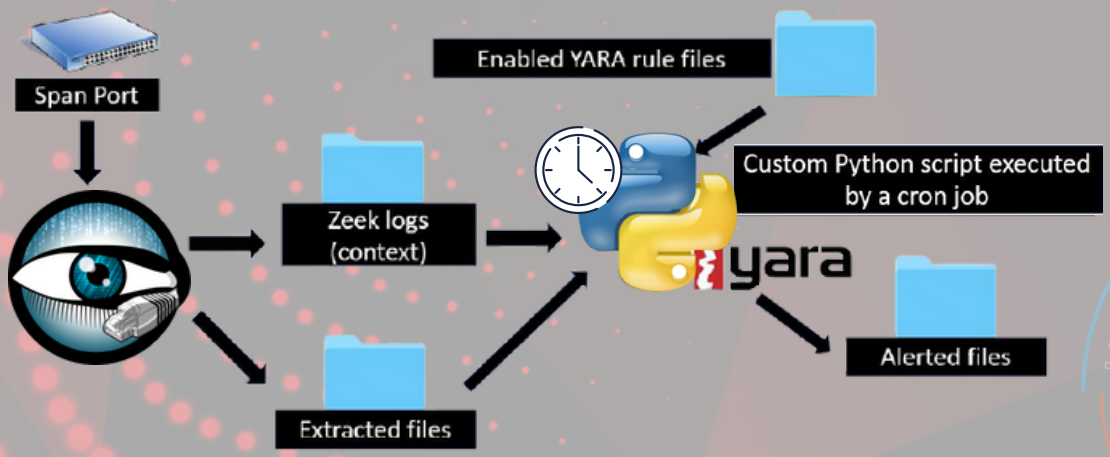
2

Terdapat cron yang akan menjalankan skrip python khusus zeekYaraAlert.py yang dikembangkan. Kemudian, akan mengambil semua rules YARA yang diaktifkan sebagai masukan. Pada dasarnya cron akan menggabungkan semua rules YARA yang terletak di folder rules YARA menjadi satu file. File tersebut akan digunakan untuk memindai semua file yang diekstrak oleh Zeek ke dalam folder file hasil ekstraksi. Dengan catatan rules YARA tidak boleh diduplikasi.



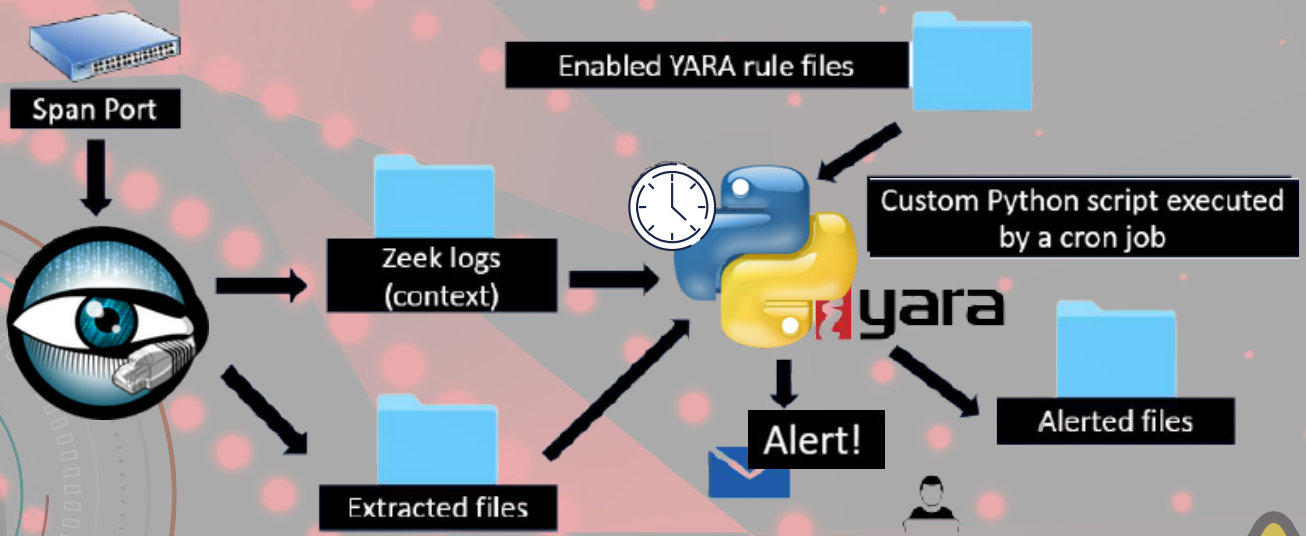
3

Jika terdapat file yang didefinisikan sebagai *alert*. Hal tersebut akan dicatat sebagai id dari file tersebut dan digunakan untuk mencari *log* file Zeek lainnya untuk mendapatkan lebih banyak konteks, seperti alamat IP yang terlibat, detail tingkat aplikasi, seperti URL HTTP, agen pengguna, Dengan menggunakan metode HTTP dan menyalin file yang memicu rules YARA ke folder file Alerted.



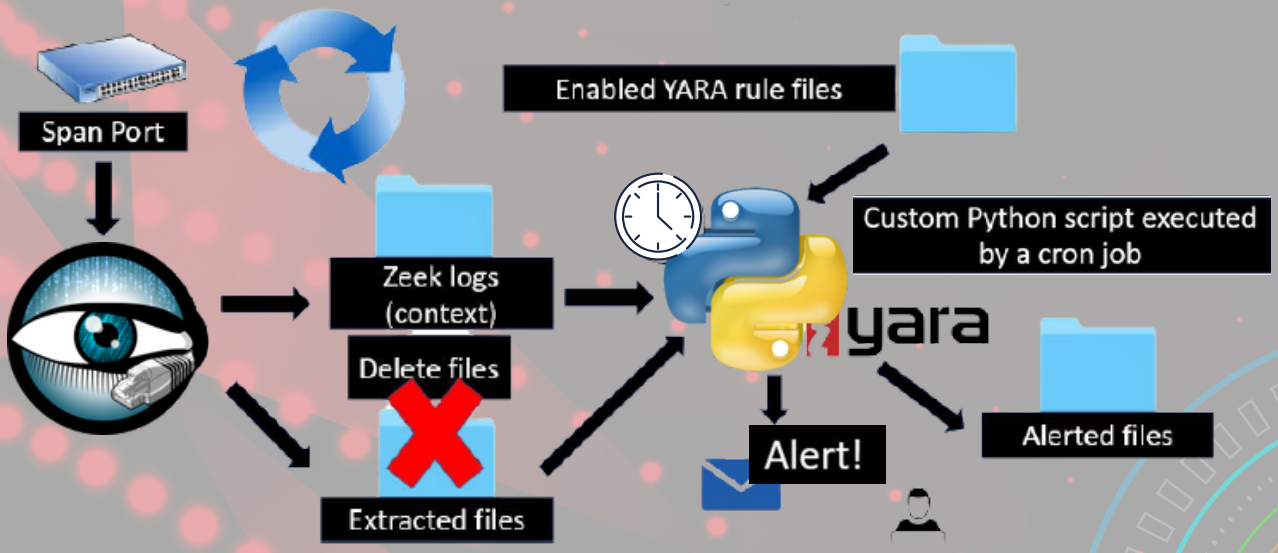
4

Skrip akan mengirimkan peringatan email dengan contoh file yang memicu rules YARA yang dikompresi dengan kata sandi "infected", jika di bawah batas file yang ditetapkan dalam skrip yang secara default berukuran 10 MB.



5

Terakhir, skrip akan menghapus semua file pada folder file yang diekstraksi dan akan terus berjalan secara *default* diatur dalam durasi satu menit pada skrip. Karena beberapa file diekstraksi per menit, pemindaian akan menjadi relatif cepat dibandingkan dengan melakukan pemindaian penuh dengan menggunakan rules YARA pada *hard drive* di endpoint.



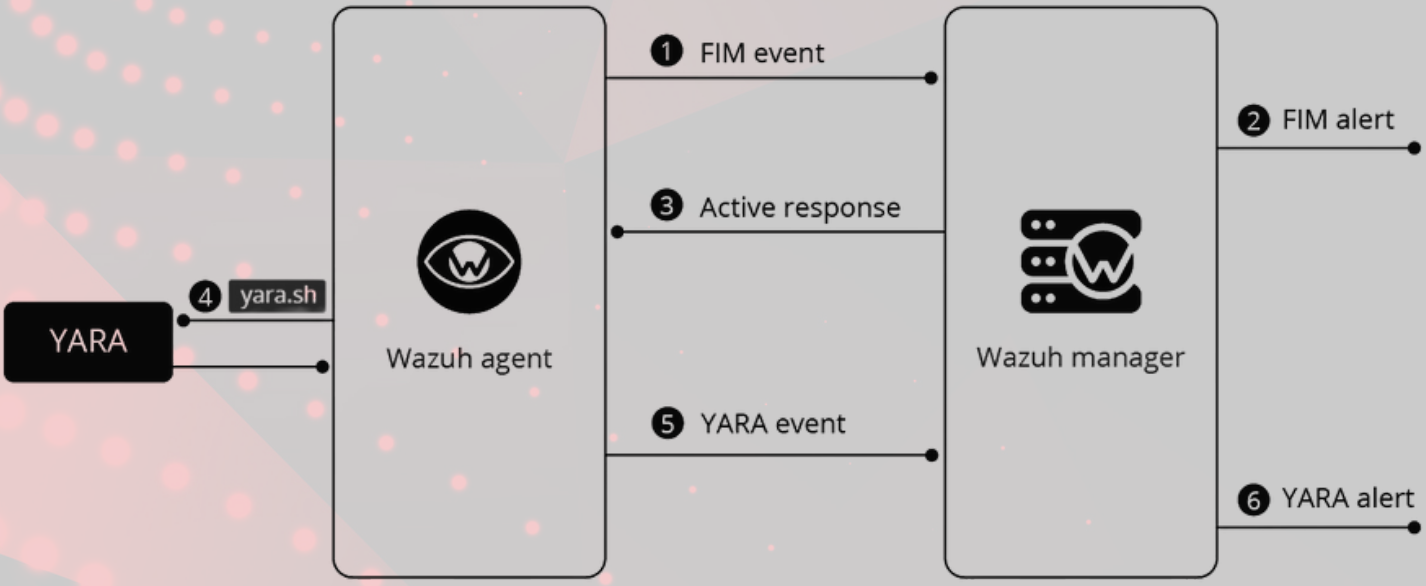
IMPLEMENTASI YARA

YARA DENGAN WAZUH

Wazuh merupakan perangkat lunak berbasis *open source* yang berfungsi sebagai sistem deteksi intrusi berbasis *host*. Wazuh melakukan analisis *log*, pemeriksaan integritas, pemantauan registri Windows, dan deteksi *rootkit*.

Wazuh dan YARA dapat diintegrasikan melalui berbagai cara salah satunya dengan pemindaian YARA secara otomatis menggunakan ***active response module*** ketika Wazuh FIM *alert* dipicu.

Diagram di bawah ini mengilustrasikan alur proses dan interaksi antar komponen yang berperan dalam pengintegrasian YARA dengan Wazuh.



Pada panduan ini, akan diberikan penjelasan terkait dengan implementasi YARA pada Wazuh dengan menggunakan *stateless active response* yang akan dijalankan secara lokal pada agen yang menghasilkan peringatan.

Langkah pertama yang dilakukan adalah mengonfigurasikan lingkungan implementasi untuk menguji PoC. Berikut merupakan langkah-langkah implementasinya:

1. Menambahkan *rules* ke dalam file 'local_rules.xml' pada Wazuh manager yang terletak pada direktori berikut:

   `/var/ossec/etc/rules/local_rules.xml`

   Hal ini digunakan untuk memberikan peringatan apabila terdapat *file* yang diubah atau ditambahkan pada direktori berikut:

   `/tmp/yara/malware/`

2. Menambahkan *dekoder* ke dalam file 'local_decoder.xml' pada Wazuh manager yang terletak pada direktori berikut:

   `/var/ossec/etc/decoders/local_decoder.xml`

   Hal ini bertujuan untuk mengekstrak hasil pemindaian YARA.

3. Melakukan konfigurasi pada file 'ossec.conf' yang terletak pada direktori berikut:

   `/var/ossec/etc/ossec.conf`

4. Restart Wazuh manager untuk menerapkan perubahan konfigurasi dengan *command* berikut:

   `systemctl restart wazuh-manager`

5. *Compile* dan *install* YARA.

6. *Download* YARA *rules.*

7. *Download* sampel *malware* dan jalankan YARA *scan.*

8. Buat *script* 'yara.sh' pada direktori berikut:

```
/var/ossec/active-response/bin/
```

Mengubah *file owner* dari 'yara.sh' ke root:wazuh dan *file permissions*-nya menjadi 0750.

Jalankan *command* berikut untuk memproses *input* JSON:

```
apt install -y jq
```

9. Mengubah pengaturan *file integrity monitoring* pada file 'ossec.conf' dengan *command* di bawah ini untuk memantau direktori /tmp/yara/malware secara *real time.* Setelah itu, *restart* Wazuh Agent.

```
<syscheck><directories
whodata="yes">/tmp/yara/malware</directories>
</syscheck>
```

IMPLEMENTASI YARA

YARA DENGAN KRAKEN

Kraken merupakan pemindai Yara yang dapat digunakan berbagai platform sederhana seperti Windows, Mac, FreeBSD, dan Linux. Kraken digunakan untuk respons insiden, penelitian, dan pendeteksian secara ad-hoc (bukan untuk perlindungan endpoint).

Unduh aturan Yara yang telah diperbarui pada **https://github.com/botherder/kraken**.

Kraken dapat diluncurkan tanpa *command* apapun dan melakukan pemindaian secara otomatis. Kraken tidak mengkomunikasikan hasil apapun kepada server.

Kraken dapat berjalan menggunakan *command* berikut ini:

- Menggunakan kraken --backend example.com akan menimpa BACKEND *default* yang disediakan selama waktu pembuatan.

- Menggunakan kraken --report akan membuat Kraken melaporkan *autoruns* atau deteksi apa pun ke server *backend* yang dikonfigurasi.

- Meluncurkan kraken --daemon akan menjalankan pemindaian pertama dan kemudian berjalan terus menerus. Dalam *mode* daemon, Kraken akan memantau setiap pembuatan proses baru dan memindai biner dan memorinya, serta memeriksa secara teratur setiap entri baru yang terdaftar untuk *autorun*.

- Mengaktifkan --daemon akan secara otomatis mengaktifkan --report juga, meskipun tidak ditentukan secara eksplisit.

- Mengaktifkan --debug hanya akan menampilkan semua pesan log debug, sebagian besar termasuk detail tentang file dan proses yang sedang dipindai.

- Menggunakan --no-autoruns, --no-filesystem atau --no-process akan menonaktifkan pemindaian *autoruns*, *file* yang disimpan di *disk*, dan proses yang berjalan, masing-masing. Catatan: *flag* ini tidak memengaruhi perilaku kraken saat dijalankan dalam *mode* daemon.

- Jika pemindaian sistem *file* diaktifkan, Kraken akan memindai seluruh folder *root* secara rekursif (/ pada sistem *nix dan semua *drive* tetap yang terpasang pada sistem Windows). Menggunakan --folder dapat menentukan folder tertentu yang ingin dipindai.

- Opsi --rules memungkinkan penentuan ke *file* atau *folder* yang berisi aturan Yara yang ingin digunakan untuk pemindaian. Jika kompilasi salah satu aturan ini gagal (misalnya, karena menyertakan modul yang tidak diaktifkan di pustaka Yara *default*), eksekusi akan dibatalkan.

Jika tidak ada --rules opsi yang ditentukan, Kraken akan mencoba memuat *file* aturan yang dikompilasi menggunakan urutan berikut:

1. Mencari *file* aturan yang dikompilasi di direktori kerja saat ini.
2. Mencari *file* aturan yang dikompilasi di folder penyimpanan Kraken lokal, jika itu berjalan dalam mode daemon.
3. Mencoba untuk mengekstrak *file* aturan yang dikompilasi dari aset tertanam yang dihasilkan pada waktu pembuatan (seperti yang dijelaskan di bagian Membangun).

Jika tidak ada aturan kompilasi *file* yang ditemukan, pemindai Yara Kraken akan dinonaktifkan dan eksekusi akan dilanjutkan tanpanya.

KONFIGURASI

Ketika kraken diluncurkan dalam *mode* daemon, kraken akan mencari *file* konfigurasi di direktori kerja atau di direktori persisten. *File* konfigurasi ini sebagian besar digunakan untuk mencari nama host dari *backend* yang harus dihubungkan oleh Kraken. Jika *file* konfigurasi tidak ada, *file* konfigurasi akan dibuat menggunakan parameter default yang disediakan selama waktu pembuatan (terutama BACKEND).

Jika kraken diluncurkan dalam *mode* normal, kraken masih akan mencari *file* konfigurasi apa pun, tetapi tidak akan menulis satu ke *disk* jika tidak ada. Jika tidak ada *file* konfigurasi yang ditemukan, itu akan menggunakan parameter *default* yang disediakan selama waktu pembuatan (sekali lagi, BACKEND).

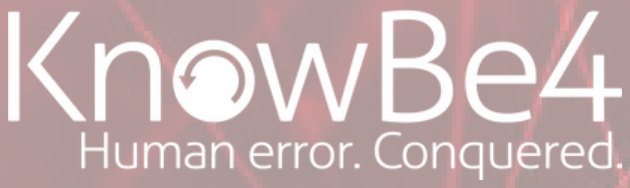
Untuk memberikan parameter yang berbeda, Kraken membuat *file* config.yaml di direktori yang sama dengan biner kraken menggunakan format berikut:

```
domain_dasar: <nilai>
```

Atau, menentukan backend khusus dari baris perintah menggunakan kraken --backend example.com.

PENGGUNA YARA

PENGGUNA YARA



REFERENSI

·Michelena, D. B. (2019). Detecting Malicious Files with YARA Rules as They Traverse the Network. Black Hat USA 2019, August.

·Naik, N., Jenkins, P., Cooke, R., Gillett, J., & Jin, Y. (2020). Evaluating Automatically Generated YARA Rules and Enhancing Their Effectiveness. 2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, 1146–1153. https://doi.org/10.1109/SSCI47803.2020.9308179.

·R. E. Indrajit, "Skenario Kombinasi Tools yang Efektif dalam Analisis Malware," Am. J. Appl. Sci., vol. 9, no. 3, pp. 1–22, 2011.

·N. Naik, P. Jenkins, N. Savage, L. Yang, K. Naik, and J. Song, "Embedding Fuzzy Rules with YARA Rules for Performance Optimisation of Malware Analysis," IEEE Int. Conf. Fuzzy Syst., vol. 2020-July, 2020, doi: 10.1109/FUZZ48607.2020.9177856.

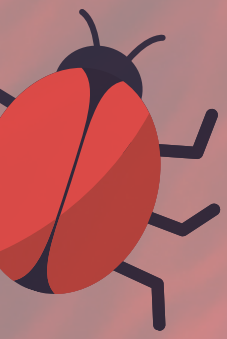
2017, "Deep Discovery Analyzer 5.8 Online Help," Trend Micro Incorporated, https://docs.trendmicro.com/all/ent/ddan/v5.8/en-us/ddan_5.8_olh/Adding-a-YARA-Rule-F.html#GUID-0E10540D-DC10-4FFC-A871-3C243ACFA2CF.

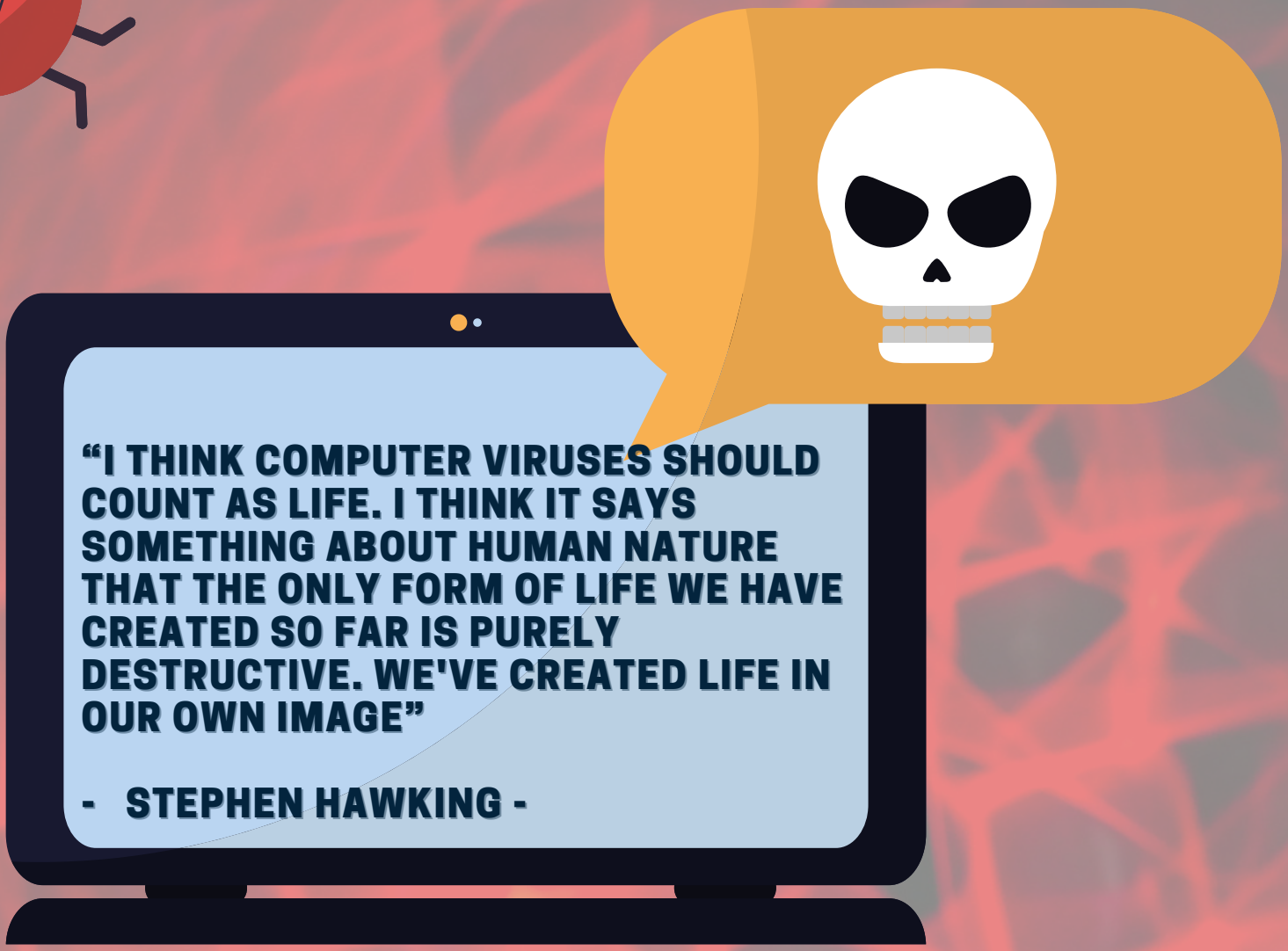
2022. "Malware Analysis : How to use Yara Rules to Detect Malware," Blue Teams Academy, https://www.blueteamsacademy.com/yara/.

2020, "Kraken - Cross Platform Yara Scanner in Go Source",eForensic Magazine, https://hakin9.org/kraken-cross-platform-yara-scanner-written-in-go-resources/.

NordVPN, "Cara Malware Menyebar,", https://nordvpn.com/id/cybersecurity/what-is-malware/

2020, "How to integrate YARA with Wazuh," wazuh, https://wazuh.com/blog/how-to-integrate-yara-with-wazuh/




“I THINK COMPUTER VIRUSES SHOULD COUNT AS LIFE. I THINK IT SAYS SOMETHING ABOUT HUMAN NATURE THAT THE ONLY FORM OF LIFE WE HAVE CREATED SO FAR IS PURELY DESTRUCTIVE. WE'VE CREATED LIFE IN OUR OWN IMAGE”
- STEPHEN HAWKING -

DIREKTORAT OPERASI KEAMANAN SIBER

NATIONAL CSIRT OF INDONESIA

Id-SIRTII/CC

INDONESIA SECURITY INCIDENT RESPONSE TEAM ON INTERNET INFRASTRUCTURE COORDINATION CENTER

TLP: WHITE