Algoritma Shikata Ga Nai untuk Meningkatkan Kinerja Antivirus

Wildan Zaim Syaddad - 13518068 (*Author*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13518068@std.stei.itb.ac.id

***Abstract*— Shikata Ga Nai merupakan salah satu algoritma encoding yang digadang-gadang menjadi algoritma yang paling baik untuk melakukan penyembunyian shellcode atau malicious code. Namun, berbagai antivirus maupun firewall yang ada masih kesulitan untuk mendeteksi shellcode tersebut dikarenakan antivirus atau firewall tersebut hanya mendeteksi vanilla implementation. Untuk itu diperlukan kombinasi dari berbagai cara yang ada untuk melakukan deteksi terhadap malicious file. Malicious atau shellcode yang mampu bypass antivirus dapat menyebabkan hacktivism, remote code execution, dan berbagai hal berbahaya lainnya.**

Keywords—shikata ga nai, shellcode, antivirus, firewall, deteksi, polymorphic

I. PENDAHULUAN

Dewasa ini perkembangan teknologi sangat berkembang dengan pesat. Mulai dari teknologi di bidang informasi, pangan, mesin, industri, arsitektur, dan lain sebagainya. Salah satu perkembangan teknologi yang paling populer dan berdampak adalah perkembangan dari komputer yang sudah sangat biasa digunakan oleh manusia dalam kehidupan sehari hari. Hampir seluruh kegiatan manusia saat ini terbantu oleh keberadaan komputer seperti pekerjaan, edukasi atau pendidikan, hiburan, komunikasi, dan lain sebagainya.

Dengan kemajuan teknologi ternyata memiliki dampak positif dan negatif. Selain dengan meningkatnya dan mempermudah berbagai pekerjaan manusia, ternyata juga meningkatkan teknologi kriminal atau cybercrime. Cybercrime atau kejahatan siber merupakan tindakan kejahatan yang dilakukan dengan teknologi digital atau lebih tepatnya adalah tindakan ilegal yang dilakukan dengan menggunakan teknologi komputer dan jaringan internet guna melakukan serangan terhadap sistem informasi target. Contoh dari cybercrime adalah peretasan, pembobolan perangkat teknologi, pencurian data korban, dan lain sebagainya. Cybercrime diatur dalam Undang Undang Nomor 11 tahun 2008 tentang Informasi dan Transaksi Elektronik (UU ITE).

Salah satu kejahatan yang paling populer adalah peretasan atau Hacking yang bertujuan jahat. Pada dasarnya hacking tidak selalu jahat, karena bisa jadi digunakan untuk keperluan pengetesan terhadap suatu sistem lalu diharapkan mampu melakukan mitigasi atau perbaikan dari sistem tersebut. Hal ini biasa disebut sebagai penetration testing.

Dalam prakteknya, kegiatan hacking menggunakan berbagai metode dan perantara. Metode tersebut bisa melalui perantara komputer atau bahkan manusia. Peretasan dengan melalui manusia biasanya melakukan rekayasa terhadap pikiran dan perasaan dari manusia tersebut. Hal ini biasa disebut *social engineering* atau rekayasa sosial yang memanfaatkan perasaan manusia yang tidak ada pada komputer. Sedangkan peretasan yang menggunakan perantara komputer tersebut seperti Man in the Middle attack, shellcode, web hacking, dan lain sebagainya.

Salah satu perantara yang paling umum untuk melakukan peretasan yang berdampak bahaya adalah dengan menggunakan perantara Shellcode. Shellcode ini digunakan oleh pelakunya untuk mendapatkan akses secara *remote* dari komputer korban atau target. Shellcode tersebut dikirim kepada komputer korban dengan harapan korban mampu menjalankan shellcode tersebut. Ketika shellcode sudah dijalankan maka secara tidak sadar komputer korban sudah dapat diambil alih oleh penyerang.

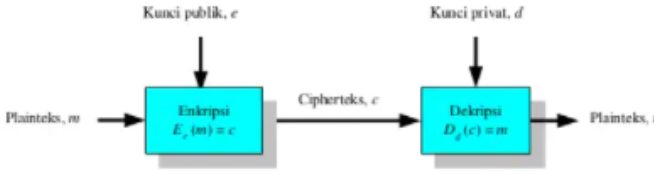
Tentunya supaya shellcode tersebut bisa mengambil alih komputer, haruslah tidak terdeteksi oleh sistem keamanan komputer maupun oleh korban. Terdapat berbagai cara yang biasa dilakukan oleh peretas untuk menyembunyikan shellcode sehingga mampu melewati antivirus dan kesadaran dari korban, salah satunya adalah dengan cara melakukan encoding dengan berbagai algoritma yang ada. Salah satu yang digadang-gadang sebagai algoritma terbaik adalah algoritma Shikata Ga Nai. Algoritma Shikata Ga Nai merupakan algoritma yang rumit dan mampu melewati atau melakukan *bypassing* dari antivirus atau firewall dan mampu membuat korban tidak sadar bahwa terdapat file yang merupakan shellcode. Shellcode tersebut berjalan pada *background* sehingga tidak disadari, serta mampu menempel pada proses atau perangkat lunak lainnya. Untuk itu, dibutuhkan dan diharapkan antivirus atau firewall untuk bisa mendeteksi kecanggihan shellcode yang menggunakan algoritma Shikata Ga Nai.

II. LANDASAN TEORI

A. *Algoritma Kriptografi Kunci Publik*

Algoritma Kriptografi Kunci Publik merupakan suatu algoritma yang berbeda dari sistem kriptografi kunci simetri. Apabila kriptografi kunci simetri menggunakan kunci yang sama baik untuk melakukan enkripsi maupun dekripsi, algoritma kriptografi kunci publik menggunakan kunci yang berbeda untuk melakukan enkripsi maupun dekripsi. Ide kriptografi kunci-nirsimetri ini muncul pada tahun 19767 dengan makalah yang ditulis oleh Diffie-Hellman dengan judul makalah "*New Directionss in Cryptography*".

Pada algoritma kunci publik atau biasa disebut kriptografi kunci-nirsimetri masing masing pengirim dan penerima memiliki sepasang kunci yaitu kunci publik dan kunci privat. Kunci publik digunakan untuk melakukan enkripsi pesan, sedangkan kunci privat digunakan untuk melakukan dekripsi pesan.



Gambar 1. Proses Enkripsi Kunci-nirsimetri

Dengan mekanisme ini terdapat keuntungan atau menutupi kekurangan pada mekanisme kunci simetri, yaitu tidak ada kebutuhan untuk mengirimkan kunci rahasia. Penerima pesan cukup membiarkan kunci publiknya diketahui oleh pengirim sehingga pengirim pesan menggunakan kunci tersebut untuk melakukan enkripsi. Penerima pesan memiliki kunci privat yang digunakan untuk melakukan dekripsi setelah pesan tersebut diterima. Hal inilah yang menjadikan bahwa kunci enkripsi tidak harus dikirim melalui saluran yang aman.

Dengan adanya kriptografi kunci publik ini setidaknya memiliki keuntungan bahwa tidak diperlukan pengiriman kunci rahasia enkripsi serta dapat menekan jumlah kunci. Hal ini tentunya didasarkan pada fakta bahwa komputasi untuk enkripsi atau dekripsi pesan mudah dilakukan serta secara komputasi hampir tidak mungkin menurunkan kunci privat apabila diketahui kunci publik.

Namun hal ini tidak berarti algoritma kunci publik tidak memiliki kekurangan. Kekurangan dari kriptografi kunci publik adalah proses enkripsi maupun dekripsi umumnya lebih lambat ketimbang kunci simetris sebab menggunakan operasi yang melibatkan operasi perpangkatan yang besar, selain itu ukuran cipherteks yang lebih besar daripada plainteks serta ukuran kunci relatif lebih besar ketimbang pada kunci simetri. Selain itu terdapat kekurangan lainnya seperti dari cipherteks tidak dapat memberikan informasi mengenai otentikasi pengirim sebab kunci publik yang disebar atau diketahui secara luas, lalu tidak ada algoritma kunci publik yang terbukti aman. kebanyakan algoritma mendasarkan keamanannya pada sulit mudahnya memecahkan persoalan aritmatik yang menjadi dasar dari pembangkitan kunci seperti pemfaktoran, logaritmik, pemangkatan, fungsi atau operasi tertentu, dan lain sebagainya.

B. *Algoritma Shikata Ga Nai*

Shikata Ga Nai merupakan salah satu algoritma yang biasa digunakan dalam dunia keamanan siber. Hal ini dilakukan lantaran Shikata Ga Nai menggunakan prosedur yang sangat rumit dalam implementasinya. Shikata Ga Nai memanfaatkan kelemahan berbagai antivirus atau firewall saat ini yang hanya mendeteksi atau melakukan pengecekan secara vanila. Hal ini tentunya sangat rentan, karena seharusnya antivirus atau firewall juga harus berkembang seiring dengan banyaknya malware yang juga berkembang dan terus bervariasi.

Shikata Ga Nai merupakan *polymorphic XOR additive feedback encoder* yang biasa digunakan oleh framework Metasploit. Dalam setiap pembuatan encoded shellcode akan dilakukan secara polimorfik sehingga berbeda untuk setiap pembuatan berikutnya. Encoder ini menawarkan tiga fitur utama yang mampu memberikan perlindungan tingkat lanjut saat digabungkan. Hal inilah yang menjadi penyebab utama kenapa algoritma Shikata Ga Nai sangat rumit dan susah untuk dideteksi.

Fitur utama yang pertama adalah teknik utama yang digunakan oleh *decoder stub generator*. Teknik utama tersebut adalah teknik metamorfik. *Decoder stub generator* digunakan untuk menghasilkan keluaran yang berbeda pada setiap kali pembangkitan melalui penataan ulang kode dan substitusi. Hal ini digunakan untuk menghindari dari pendeteksian digital signature.

Fitur utama yang kedua merupakan penggunaan kunci modifikasi diri yang dirantai melalui *additive feedback*. Algoritma Shikata Ga Nai pada implementasinya menggunakan banyak iterasi yang mana menghasilkan output berikutnya yang berbeda. Fitur utama ini digunakan untuk memastikan apakah output berikutnya tersebut benar atau salah. Output tersebut dikatakan salah apabila input atau kunci juga salah pada setiap iterasi.

Fitur ketiga atau fitur utama terakhir adalah pengaburan. Rintisan decoder itu sendiri sebagian akan dikaburkan melalui modifikasi sendiri dari blok dasar yang dilindungi terhadap emulasi yang menggunakan instruksi FPU. Instruksi FPU (Floating-Point Unit) secara umum akan mengambil dua item pertama dalam stack, lalu menjalankan aksi serta mengembalikan jawabannya pada puncak dari stack.

Selain itu, Shikata Ga Nai juga melalui berbagai macam teknik yang dinamis, seperti dynamic instruction substitution, block ordering, penggantian antar register, pengacakan

pengurutan instruksi, penambahan junk code, random key, dan pengacakan dalam instruksi.

Algoritma Shikata Ga Nai dapat diperinci menjadi sebagai berikut :

1. Inisialisasi kunci
2. Menentukan lokasi relatif EIP (sehingga bisa dilakukan modifikasi instruksi yang berdasar pada alamat). Pada metasploit biasanya digunakan fstenv/fnstenv.
3. Iterasi yang melakukan :
    a. Vanilla Shikata Ga Nai memberikan nilai nol pada register sehingga dapat digunakan sebagai penghitung dan secara eksplisit memindahkan nilai penghitung ke dalam register, sehingga porsi dari loop jelas.
    b. Shikata Ga Nai mendekode instruksi instruksi pada alamat memori yang lebih tinggi (mampu divariasi untuk alamat yang lebih rendah jika diinginkan lebih banyak tipu daya). Hal ini dilakukan dengan menambahkan nilai ke alamat yang disimpan sebelumnya (relatif terhadap EIP) dan melakukan XOR dengan kuncinya.
    c. Alamat dari sebelumnya tersebut (relatif terhadap EIP) kemudian dimodifikasi beserta kuncinya. Hal ini bisa digunakan secara langsung oleh metasploit. Metasploit secara default menambah atau mengurangi nilai intruki di suatu tempat relatif terhadap alamat yang disimpan sebelumnya (yang relatif terhadap EIP).
    d. Iterasi berlanjut sampai seluruh instruksi berhasil didekodekan lalu memindahkan eksekusi ke shellcode. Shellcode ini menggunakan Reverse sehingga komputer penyerang bertindak sebagai listener.
4. Shikata Ga Nai memungkinkan *multiple iteration*. Jika *multiple iteration* digunakan, maka step satu hingga tiga diulang setelah iterasi sekarang selesai.

C. Shellcode

Shellcode merupakan serangkaian instruksi yang didesain atau dirancang untuk melakukan manipulasi eksekusi program dengan cara yang tidak dimaksudkan pada awalnya. Hal ini bertujuan untuk melakukan injeksi kode berbahaya pada suatu proses yang rentan sehingga tidak dapat dideteksi dan mampu melakukan eksekusi ketika file tersebut dieksekusi pada target.

Terdapat berbagai cara untuk membuat shellcode tersebut. Cara manual dan cara yang terautomasi. Untuk membuat shellcode secara manual, dapat dilakukan dengan cara mengambil opcode dari suatu mesin assembler/disassembler seperti MASM (Microsoft Macro Assembler). Opcode yang dihasilkan tersebut masih mentah dan seringkali tidak dapat dieksekusi di luar lingkungan asal. Untuk itu dalam proses manual ini masih diperlukan beberapa penyesuaian yang kompatibel dengan prosesor tempat opcode tersebut dieksekusi. Sedangkan cara terautomasi dapat menggunakan Framework Metasploit. Algoritma Shikata Ga Nai masuk dalam cara yang terautomasi. Shikata Ga Nai menangani ketidaksesuaian opcode dengan prosesor tempat opcode tersebut dieksekusi. Selain itu Shikata Ga Nai juga mampu melakukan pengaburan terhadap shellcode sehingga tidak mudah dideteksi.

D. Antivirus

Antivirus merupakan perangkat lunak yang bertujuan atau berfungsi untuk mendeteksi, memindai, dan bahkan menghapus file file yang mencurigakan dan berbahaya. File file tersebut antara lain seperti virus, trojan, worm, adware, malware, shellcode, dan lain sebagainya. Terdapat banyak antivirus seperti Smadav, Avira, Norton, dan lain sebagainya.

Antivirus memiliki berbagai jenis. Terdapat setidaknya empat jenis antivirus yang ada seperti antiviru gratis seperti smadav dan avast, lalu antivirus trial yang bisa digunakan secara gratis namun terdapat batasan waktu seperti esset antivirus, lalu antivirus donasi yang mengharuskan pengguna berdonasi, lalu antivirus berbayar yang mewajibkan penggunanya untuk membeli lisensi dengan biaya tertentu untuk dapat menggunakan antivirus tersebut.

Antivirus memiliki cara kerja dengan cara melakukan pemindaian file atau kode. Antivirus memiliki basis data sendiri, biasanya perusahaan ini mengkompilasi database ekstensif dari virus dan malware yang sudah dikenal sehingga bisa dilakukan komparasi atau perbandingan.

Ketika suatu file dipindai, antivirus akan membandingkan basis data yang dimiliki untuk menemukan kesamaan atau kecocokan dengan file yang akan dipindai. Apabila ditemukan kemiripan maka file tersebut akan diisolasi bahkan bisa dihapus. Hal ini pada umumnya dapat diatur oleh pengguna jasa dari antivirus ini.

Meskipun dengan cara kerja tersebut, perkembangan antivirus dianggap tidak secepat perkembangan berbagai file jahat yang sangat cepat melakukan modifikasi. Untuk itu salah satu kelemahan antivirus kebanyakan adalah hanya mencocokan dan memindai secara vanila saja.

III. RENCANA PENYELESAIAN MASALAH

Seperti yang sudah dijelaskan pada bab sebelumnya, algoritma Shikata Ga Nai merupakan algoritma yang sangat rumit dan sangat sulit untuk dideteksi. Serta seperti yang sudah dijelaskan sebelumnya bahwa cara kerja dari antivirus ataupun firewall yang ada saat ini kebanyakan hanya melakukan pemindaian secara vanila. Agar perangkat lunak antivirus atau firewall tersebut mampu melakukan pendeteksian terhadap shellcode yang menggunakan algoritma Shikata Ga Nai, tentunya perlu untuk perangkat lunak tersebut

mengenali dan menyesuaikan dengan pola cara kerja dari algoritma Shikata Ga Nai.

Terdapat dua cara untuk melakukan pendeteksian terhadap suatu file apakah sudah dilakukan encoding. Cara pertama adalah dilakukan secara statis (static analysis), dan cara kedua yaitu melakukan reverse file. Analisis statis dilakukan dari encoded shellcode dan biasanya dilakukan dengan cara melakukan komparasi dengan file yang sudah diidentifikasi atau dipastikan menggunakan algoritma Shikata Ga Nai. Cara reverse file dilakukan dengan melakukan analisis terhadap setiap hardcoded block dan perlu diingat bahwa setiap block tersebut mau dimodifikasi seperti apapun akan tetap memiliki instruksi yang sama. Kedua cara ini bisa dijadikan pedoman bagi antivirus atau firewall untuk mendeteksi shellcode yang menggunakan algoritma Shikata Ga Nai.

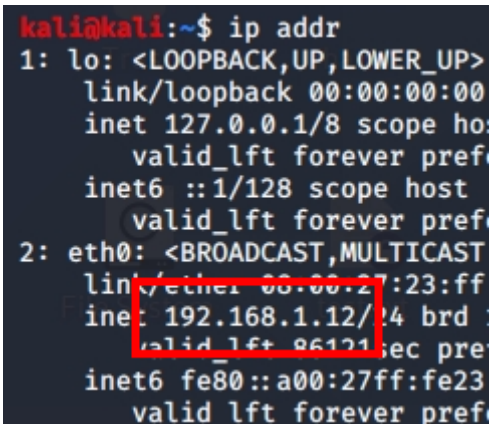
Pada bagian rencana penyelesaian masalah ini akan dibagi menjadi beberapa tahap. Tahap pertama yaitu pembuatan shellcode menggunakan algoritma Shikata Ga Nai, lalu dilakukan analisis dengan kedua cara analisis sehingga diharapkan mampu diimplementasi pada antivirus.

A. *Pembuatan Shikata Ga Nai Encoded Payload*

Penulis menggunakan sistem operasi Linux dengan reverse tcp sehingga komputer penyerang digunakan sebagai listener. Berikut merupakan beberapa tahap atau langkah yang digunakan untuk membuat shellcode Shikata Ga Nai Reverse TCP:

1. Menentukan LHOST dan LPORT

   LHOST merupakan alamat IP dari komputer penyerang. Sedangkan LPORT merupakan port dari komputer penyerang yang akan digunakan sebagai jalan masuk dari komputer target. Pada kasus ini penulis menggunakan LPORT 4444 dan LHOST 192.168.1.12



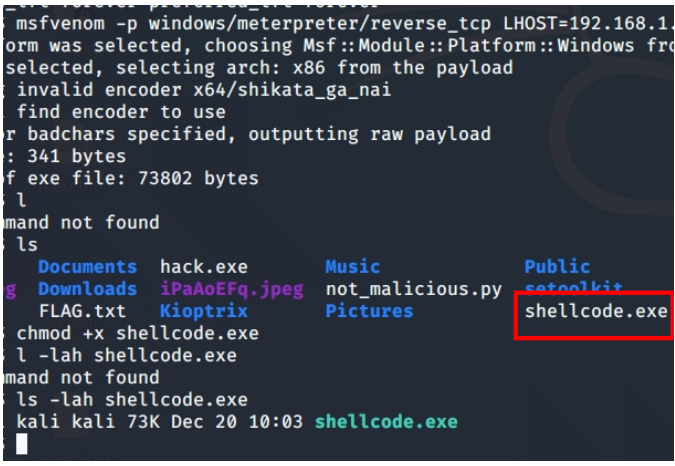
Gambar 2. IP Address Listener

2. Pembuatan dengan MSFVENOM

Shellcode yang digunakan menggunakan framework dari Metasploit. Payload yang digunakan merupakan payload reverse tcp pada windows. Shellcode yang diharapkan memiliki ekstensi exe dengan iterasi sebanyak 10. Hal ini dipilih agar dapat dieksekusi pada sistem operasi target yang mana windows.

berikut merupakan command yang penulis gunakan untuk membuat shellcode.

```
$ msfvenom -p
windows/meterpreter/reverse_tcp
LHOST=192.168.1.12 LPORT=4444 -f exe -e
x64/shikata_ga_nai i 10 > shellcode.exe
```



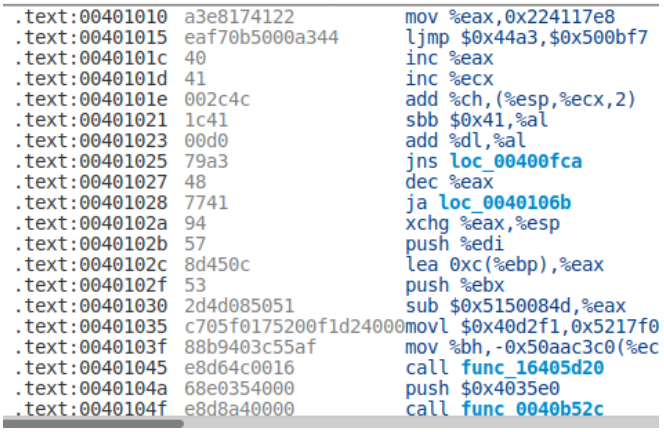
Gambar 3. Pembuatan Shellcode

3. Penyiapan Assembler / Disassembler

   Pada kasus kali ini digunakan diassembler online yang bernama ODA (The Online Disasembler).

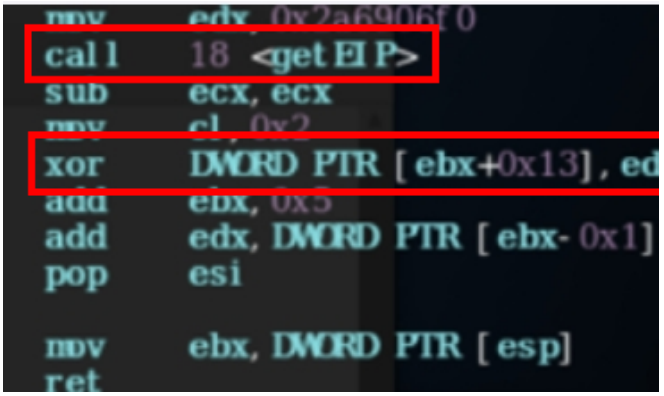
IV. HASIL DAN PEMBAHASAN

Hasil dari pembuatan shellcode tersebut memiliki ekstensi executable file dan bernama shellcode.exe. Dikarenakan hendak dilakukan analisis, maka tidak perlu menggunakan metasploit untuk exploit dari eksekusi file tersebut. Shellcod tersebut kemudian dilakukan diassemble dengan ODA.



Gambar 4. ODA

Melakukan deteksi Shikata Ga Nai dilakukan dengan cara mengetahui bagaimana pola dari algoritma tersebut bekerja. Pola bagaimana algoritma ini bekerja sudah dijelaskan pada Landasan Teori. Terdapat beberapa hal mencolok yang menjadi ciri khas dari algoritma Shikata Ga Nai yaitu pemanggilan atau penggunaan EIP dan XOR. Salah satu kunci untuk dilakukan pendeteksian apakah file tersebut menggunakan shikata ga nai adalah CALL EIP dan XOR dari EIP tersebut seperti pada gambar dibawah ini.



Gambar 5. EIP dan XOR pada Shellcode Shikata Ga Nai

Selain itu untuk analisis statis, dapat digunakan open source Yara. Yara mampu mendeteksi kode dengan masukan rule atau aturan dari Shikata Ga Nai yang melakukan pengambilan kunci, EIP, dekode, dan penyiapan dekode selanjutnya. Seperti contoh rule dari Shikata Ga Nai adalah sebagai berikut.

```
rule SGN
{
meta:
  author = “XX”
  md5 = “42810cb02db3bb2cbb428af0d8b0376e”
strings:
  $s1 = { BA ?? ?? ?? ?? [0-10] E8 ?? ?? ?? ??
[0-10] 31 53 13 [0-10] 83 c3 05 [0-10] 03 53 ff
}
condition:
  any of them
}
```

Selain dari analisis statis, dapat juga dilakukan analisis terkait tingkah laku atau behaviour dari file yang berbahaya tersebut. Hal ini dikarenakan bisa jadi secara statis atau fisik, file tersebut sudah dimodifikasi. Dengan melihat tingkah laku dari file tersebut, analisis dapat mendapatkan pola serangan dari file tersebut sehingga mampu melakukan deteksi bahwa file tersebut menggunakan Shikata Ga Nai. Untuk melakukan analisis terhadap tingkah laku dari file tersebut dapat digunakan Intezer, any run..

V. KESIMPULAN

Algoritma Shikata Ga Nai merupakan algoritma yang sangat rumit sehingga masih sering digunakan oleh pihak baik maupun pihak jahat. Shikata Ga Nai mampu melakukan encoding terhadap shellcode baik untuk meningkatkan tingkat keamanan atau justru digunakan oleh pihak tidak bertanggung jawab untuk melakukan peretasan. Antivirus atau firewall memiliki peranan penting yang diharapkan mampu mendeteksi dari shellcode tersebut. Untuk itu seharusnya antivirus dan firewall tersebut tidak hanya mendeteksi secara vanila, namun juga mampu berkembang. Perkembangan tersebut dapat dilakukan peningkatan kinerja seperti berikut : melakukan analisis statis terhadap ciri khas dari Shikata Ga Nai yaitu penggunaan EIP dan XOR dari EIP tersebut, lalu antivirus atau firewall juga melakukan analisis dari tingkah laku sehingga mampu mengurangi resiko dari file yang ternyata sudah dimodifikasi (bisa dengan bekerja sama dengan intezer).

UCAPAN TERIMAKASIH

Puji syukur Penulis panjatkan kehadirat Tuhan Yang Maha Esa sebab atas berkat rahmat-Nya serta segala izin-Nya penulis mampu menyelesaikan penulisan makalah kriptografi yang berjudul . Terima kasih juga Penulis haturkan kepada dosen mata kuliah Kriptografi Bapak Rinaldi Munir yang sudah berkenan untuk menyampaikan ilmu serta dukungan selama satu semester ini tentang kriptografi. Tak lupa terima kasih juga Penulis haturkan kepada teman teman kelas kriptografi Teknik Informatika tahun ajaran 2021 ini serta keluarga yang senantiasa mendoakan.

REFERENCES

[1] Valle, Marcos. "*Debugging Shikata Ga Nai*" [Online]. Available : https://marcosvalle.github.io/re/exploit/2018/08/25/shikata-ga-nai.html [Accessed: 19-Dec-2021]

[2] Guaran. "*Debugging Shikata Ga Nai"* [Online]. Available : https://github.com/Gu4rana/SLAE-Shikata_Ga_Nai_Decoding- [Accessed: 19-Dec-2021]

[3] Ryan Farley, Xinyuan Wang " *Automatic Extraction of Obfuscated*

[4] *Attack Code from Memory Dump*" [Online]. Available : http://mason.gmu.edu/~rfarley3/2014-ISC-CodeXt.pdf [Accessed: 19-Dec-2021]

[5] STEVE MILLER, EVAN REESE, NICK CARR. "*Shikata Ga Nai Encoder Still Going Strong*". [Online]. Available : https://www.mandiant.com/resources/shikata-ga-nai-encoder-still-going-strong [Accessed: 19-Dec-2021]

[6] Munir, Rinaldi. 2021. Slide Kuliah Kriptografi (Bandung: Institut Teknologi Bandung)

[7] Shella. "*Cara Kerja Software Antivirus*". [Online]. Available: https://ids.ac.id/cara-kerja-software-anti-virus/ [Accessed: 19-Dec-2021]

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Desember 2021

Wildan Zaim Syaddad
13518068