REvil - Sodinokibi

Analisi tecnica e Threat Intelligence report

Analisi a cura di:
Gianfranco Tonello
Michele Zuin
Federico Girotto

CTA-2019-06-24

Last revision: 2019-07-17

Sommario

Introduzione

Il ransomware Sodinokibi, conosciuto anche con il nome di REvil, ha fatto la sua prima apparizione nel mese di aprile 2019, dove sfruttava una particolare vulnerabilità di Oracle WebLogic Server per diffondersi.

Il C.R.A.M. (Centro di Ricerca Anti-Malware) di TG Soft ha analizzato negli ultimi mesi l'evoluzione di questo ransomware.

In Italia la sua prima apparizione risale al 24 maggio 2019 con un attacco RDP, come abbiamo segnalato nel tweet del 28 maggio 2019:

Gli autori del ransomware Sodinokibi, anche se sono alle prime versioni della loro creazione, sembrano avere una lunga esperienza nella creazione di questa tipologia di progetto criminale.

Alcuni ricercatori hanno individuato delle similitudine con il ransomware GandCrab, quest'ultimo ha pubblicato ufficialmente a inizio giugno la chiusura del progetto. Sembrerebbe che il candidato a occupare il vuoto lasciato dal GandCrab sia proprio il ransomware Sodinokibi.

Vettore d'infezione

Il ransomware Sodinokibi usa diverse metodologie per diffondersi:

- Vulnerabilità di Oracle WebLogic Server
- Attacco RDP
- Campagne di Malspam
- Watering hole
- Uso Exploit kit e malvertising

In Italia abbiamo osservato diverse tipologie di diffusione implementate dal ransomware. A parte la vulnerabilità di Oracle WebLogic Server, le altre metodologie elencate sopra sono state riscontrate infezioni in Italia.

Il primo attacco che abbiamo registrato risale al 24 maggio 2019, in questo caso il ransomware Sodinokibi si diffondeva attraverso un attacco RDP. L'attacco via RDP, eseguendo un brute force sulle credenziali, è già stato utilizzato da altri ransomware come il Dharma.

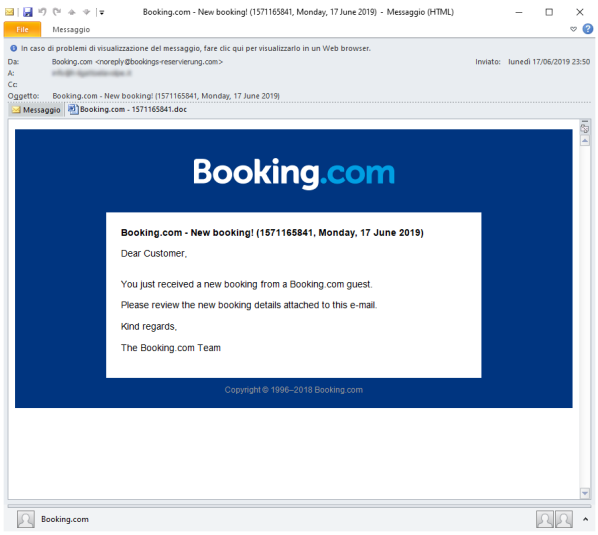
Interessante notare che l'IP utilizzato dall'attaccante per accedere via RDP era 151.106.56[.]254, questo stesso IP è stato individuato in altri attacchi RDP nel mese di giugno 2019.

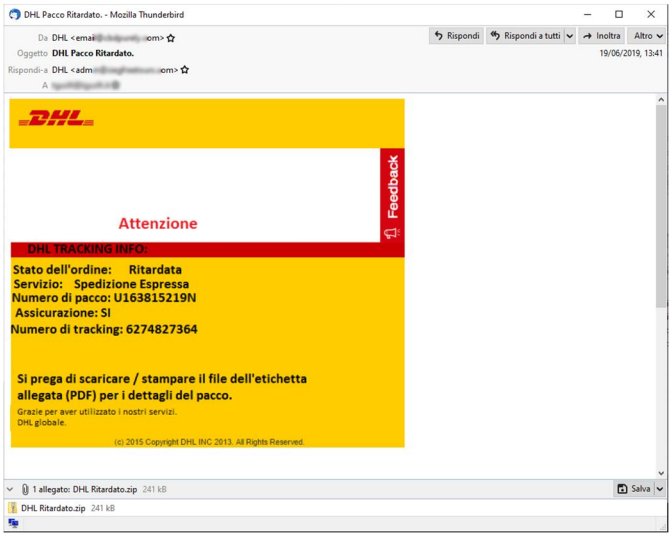
Altri attacchi che abbiamo individuato sono avvenuti attraverso campagne di malspam nel mese di giugno. Queste campagne avevano come tema:

- Booking.com
- DHL

Il tema di "Booking.com" nel mese estivo è molto azzeccato visto l'avvicinarsi delle vacanze, questo può indurre molte persone ad aprire l'allegato infetto.

Nelle figure sottostanti possiamo vedere le due campagne di malspam del Sodinokibi.





In Italia abbiamo avuto il primo caso di watering hole sul sito “winrar.it” del distributore italiano di WinRar. Dal pomeriggio fino a tarda sera del 19 giugno 2019 veniva scaricato un file infetto dal ransomware Sodinokibi al posto del setup di WinRar.

Nel 2016 il sito di “winrar.it” era già stato colpito dall’APT StrongPity, anche in questo caso stiamo parlando di attacco watering hole, dove il setup di WinRar era stato modificato in modo da contenere oltre al programma di installazione il malware spia StrongPity.

Se nel 2016 l’attacco a “winrar.it” era avvenuto da un gruppo professionista di Cyber-spionaggio, nell’attacco del 2019 si sono limitati a sostituire il setup di WinRar con Sodinokibi. Chi ha prelevato nel pomeriggio del 19 giugno WinRar, poteva accorgersi immediatamente che nel file scaricato qualcosa non andava bene, infatti le icone del file scaricato non sono quelle di WinRar ma di un cerchio rosso sbarrato o di un pupazzo di neve, come possiamo vedere in figura:

Inoltre l’esecuzione del file non comportava l’installazione di WinRar, come invece era successo con StrongPity.

Chi ha eseguito l’attacco di watering hole a winrar.it ha sfruttato in malo modo l’occasione.

Altri casi di diffusione in Italia del Sodinokibi sono stati osservati attraverso attacchi di malvertising in data 7 giugno 2019.

Gli autori del Sodinokibi sembrano essere molto attivi nella diffusione della loro creazione.

Analisi ransomware Sodinokibi

Vediamo quindi l'analisi tecnica riferita alla versione 1.1 di Sodinokibi.

Quando viene eseguito un file infetto da Sodinokibi, il ransomware per prima cosa crea un mutex differente per ogni build, come ad esempio:

`Global\D382D713-AA87-457D-DDD3-C3DDD8DFBC96`

A questo punto viene decifrato con RC4 una sezione del file, che contiene la configurazione del malware strutturata in questo modo:

```
{
    "pk": "",
    "pid": "",
    "sub": "",
    "dbg": ,
    "fast": ,
    "wipe": ,
    "wht": {
        "fld": [],
        "fls": [],
        "ext": []
    },
    "wfld": [],
    "prc": [],
    "dmn": "",
    "net": ,
    "nbody": "",
    "nname": "",
    "exp": ,
    "img": ""
}
```

Nella seguente tabella vediamo la descrizione dei campi:

Campo	Descrizione
pk	Chiave pubblica in base64
pid	Identificatore distributore
sub	Identificatore sottoscrizione
dbg	Debug: true/false
fast	True/False
wipe	True/False
wht -> fld	Esclusione cartelle
wht -> fls	Esclusione files
wht -> ext	Esclusione in base alle estensioni
wfld	Wipe folder
prc	Processi da terminare
dmn	Domini C2
net	Cifratura dei file nella rete: true/false
nbody	Istruzioni del riscatto
nname	{EXT}-readme.txt (dove EXT è l'estensione del file cifrato)
exp	Exploit True/False
img	Immagine del desktop con alert della cifratura

Se il campo “exp” è “true” allora viene eseguita una shellcode a 32 o 64 bit con l’exploit CVE-2018-8453 per l’elevazione dei privilegi.

Il passo successivo è la creazione della chiave di registro **REcfg** se non già presente:

`HKEY_LOCAL_MACHINE\SOFTWARE\recfg`

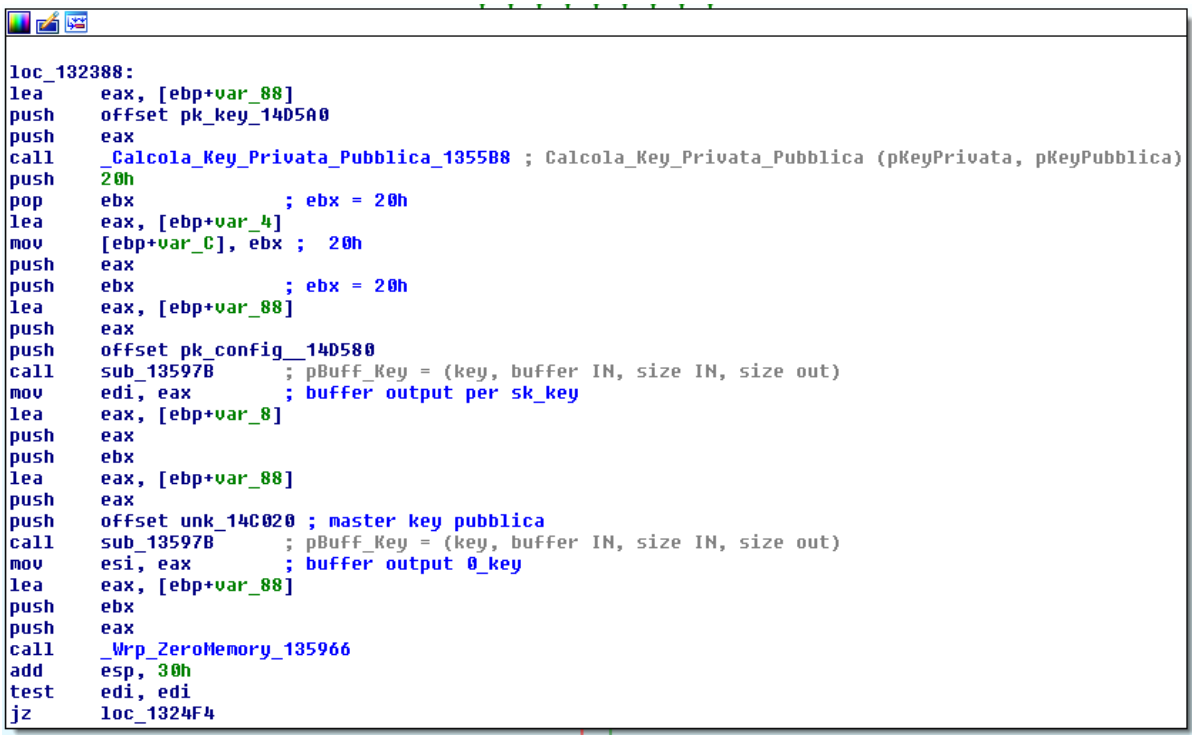
Se non dispone delle autorizzazioni sufficienti, la chiave viene creata in HKEY_CURRENT_USER.

All’interno di **REcfg** vengono creati i seguenti valori:

- pk_key
- sk_key
- 0_key
- rnd_ext
- stat

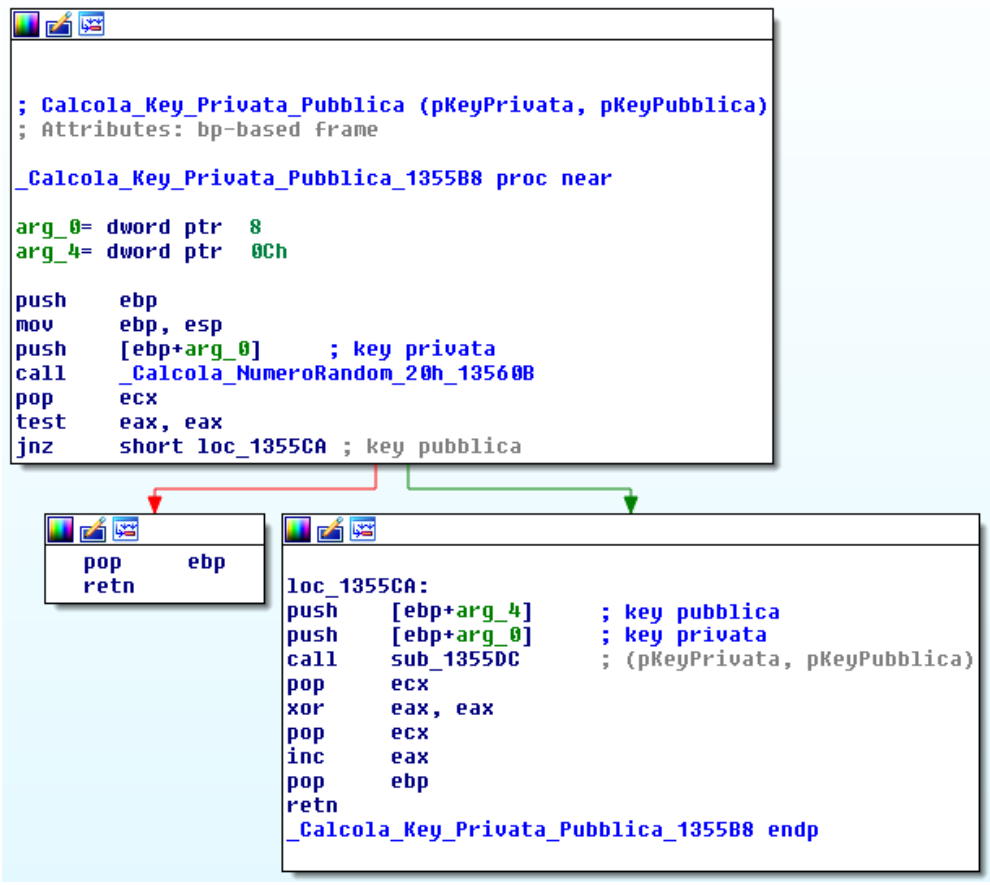
Calcolo delle chiavi private e pubbliche

A questo punto vengono calcolate le chiavi private e pubbliche, come possiamo vedere in figura:



```
loc_132388:
lea     eax, [ebp+var_88]
push    offset pk_key_14D5A0
push    eax
call    _Calcola_Key_Privata_Pubblica_1355B8 ; Calcola_Key_Privata_Pubblica (pKeyPrivata, pKeyPubblica)
push    20h
pop     ebx             ; ebx = 20h
lea     eax, [ebp+var_4]
mov     [ebp+var_C], ebx ;  20h
push    eax
push    ebx             ; ebx = 20h
lea     eax, [ebp+var_88]
push    eax
push    offset pk_config__14D580
call    sub_13597B      ; pBuff_Key = (key, buffer IN, size IN, size out)
mov     edi, eax        ; buffer output per sk_key
lea     eax, [ebp+var_8]
push    eax
push    ebx
lea     eax, [ebp+var_88]
push    eax
push    offset unk_14C020 ; master key pubblica
call    sub_13597B      ; pBuff_Key = (key, buffer IN, size IN, size out)
mov     esi, eax        ; buffer output 0_key
lea     eax, [ebp+var_88]
push    ebx
push    eax
call    _Wrp_ZeroMemory_135966
add     esp, 30h
test    edi, edi
jz      loc_1324F4
```

Le chiavi private e pubbliche vengono calcolate in questo modo:

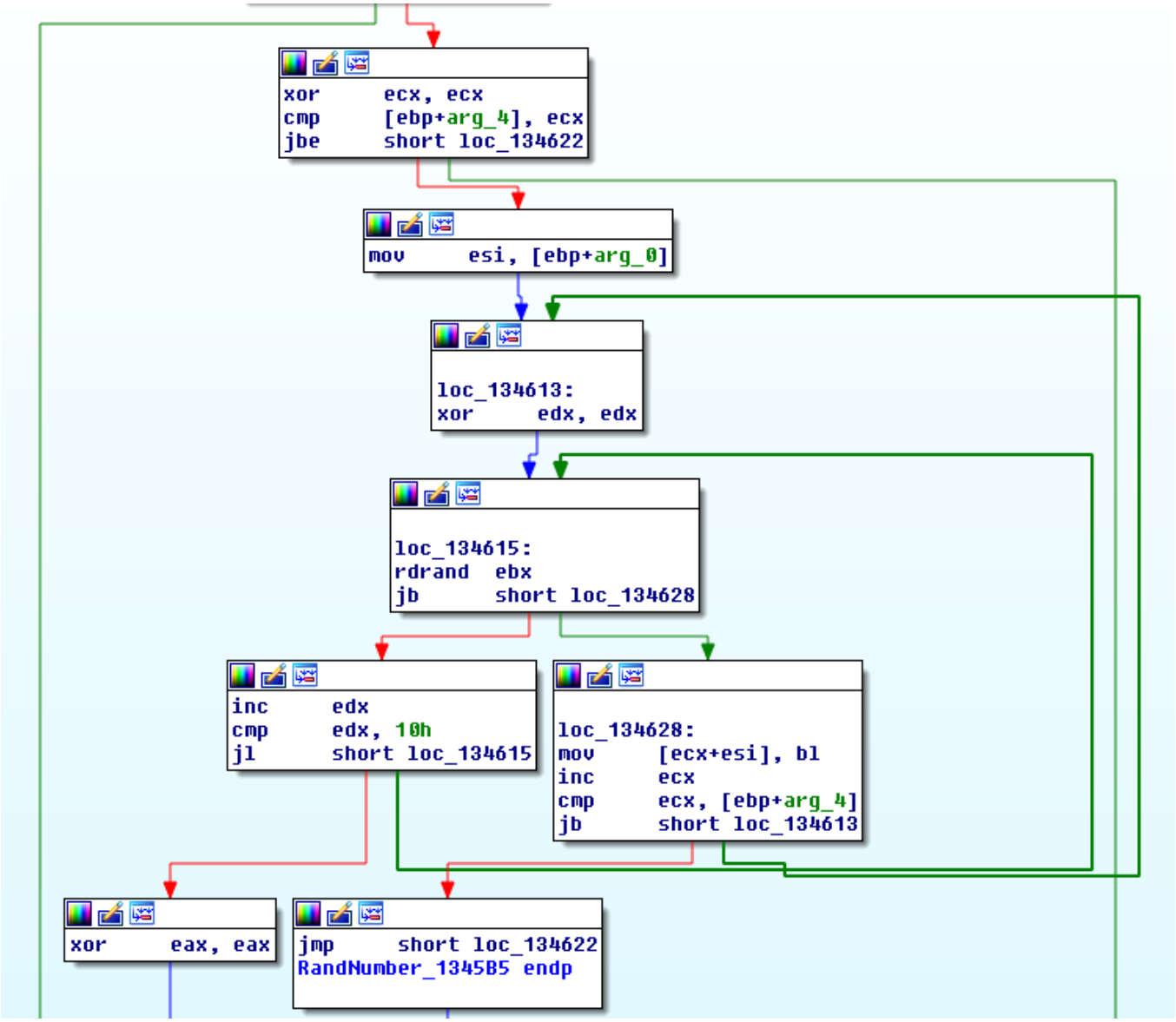


```
; Calcola_Key_Privata_Pubblica (pKeyPrivata, pKeyPubblica)
; Attributes: bp-based frame

_Calcola_Key_Privata_Pubblica_1355B8 proc near

arg_0= dword ptr  8
arg_4= dword ptr  0Ch

push    ebp
mov     ebp, esp
push    [ebp+arg_0]     ; key privata
call    _Calcola_NumeroRandom_20h_13560B
pop     ecx
test    eax, eax
jnz     short loc_1355CA ; key pubblica
```

```
pop     ebp
retn
```

```
loc_1355CA:
push    [ebp+arg_4]     ; key pubblica
push    [ebp+arg_0]     ; key privata
call    sub_1355DC      ; (pKeyPrivata, pKeyPubblica)
pop     ecx
xor     eax, eax
pop     ecx
inc     eax
pop     ebp
retn
_Calcola_Key_Privata_Pubblica_1355B8 endp
```

La chiave privata viene generata partendo da un numero random di 256 bit, come possiamo vedere in figura dalla sua subroutine di generazione del PRNG (PseudoRandom Number Generators):



La funzione per la generazione del PRNG utilizza l'hardware Intel Ivy Bridge, basato sulle linee guide del NIST's SP 800-90, attraverso la chiamata dell'istruzione assembly **rdrand.**

Il numero random generato, prima che diventi la chiave privata, viene elaborato in questo modo:

```
mov     al, [esi+1Fh]
and     byte ptr [esi], 0F8h
and     al, 3Fh
or      al, 40h
mov     [esi+1Fh], al
xor     eax, eax
inc     eax
```

A questo punto viene generata la chiave pubblica a partire dalla chiave privata. La coppia di chiavi (privata e pubblica) è generata attraverso il metodo delle curve crittografiche ellittiche ECC (Elliptic Curve Cryptography).

Sia la chiave privata sia la chiave pubblica sono 2 numeri a 256 bit che individuano 2 punti nella curva ellittica.

Lo scambio delle chiavi avviene con il metodo chiamato "E*lliptic Curve Diffie-Hellman*" (ECDH), dove:

$$d_A P_B = d_B P_A$$

Sia G un punto fisso della curva, dove:

- d_A = key privata di A (numero segreto casuale)
- $P_A = G * d_A$ = key pubblica di A (il punto G è moltiplicato da d_A)
- d_B = key privata di B (numero segreto casuale)
- $P_B = G * d_B$ = key pubblica di B

Sodinokibi usa la curva ellittica "Curve25519" con G={9}, sviluppata da Dan Bernestein, come supposto nel tweet di Eric Klonowski (@noblebarstool).

Ora che Sodinokibi ha generato in memoria la coppia ECC di chiavi privata e pubblica che chiameremo rispettivamente dk_key e pk_key, la chiave pubblica viene memorizzata nel chiave di registro `recfg` all'interno di `pk_key`:

`HKEY_LOCAL_MACHINE\SOFTWARE\recfg`

`[pk_key]` = Chiave pubblica

Struttura dati sk_key

A questo punto viene generata la struttura dati sk_key, attraverso la chiamata alla subroutine Sub_13597B:

```
pBuff_sk_key = Sub_13597B (key_pubblica_json, key_privata, size IN, size out)
```

Lo scopo di Sub_13597B è quello di cifrare la chiave privata generata all'interno della struttura dati sk_key.

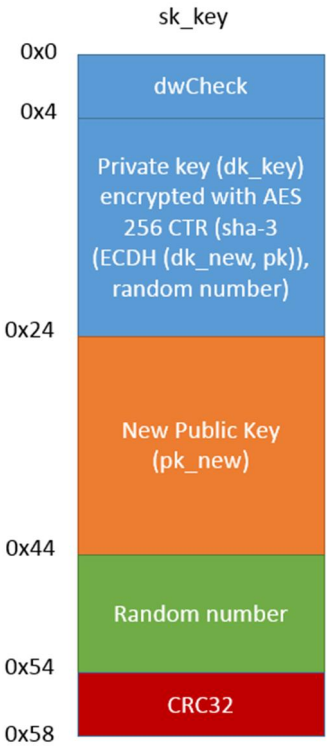
La Sub_13597B prende in input 4 parametri:

- key_pubblica_json: chiave pubblica "pk" all'interno della sezione di configurazione in json
- key_privata: chiave privata generata "dk"
- size IN: dimensione della chiave privata "dk"
- size out: dimensione della struttura sk_key

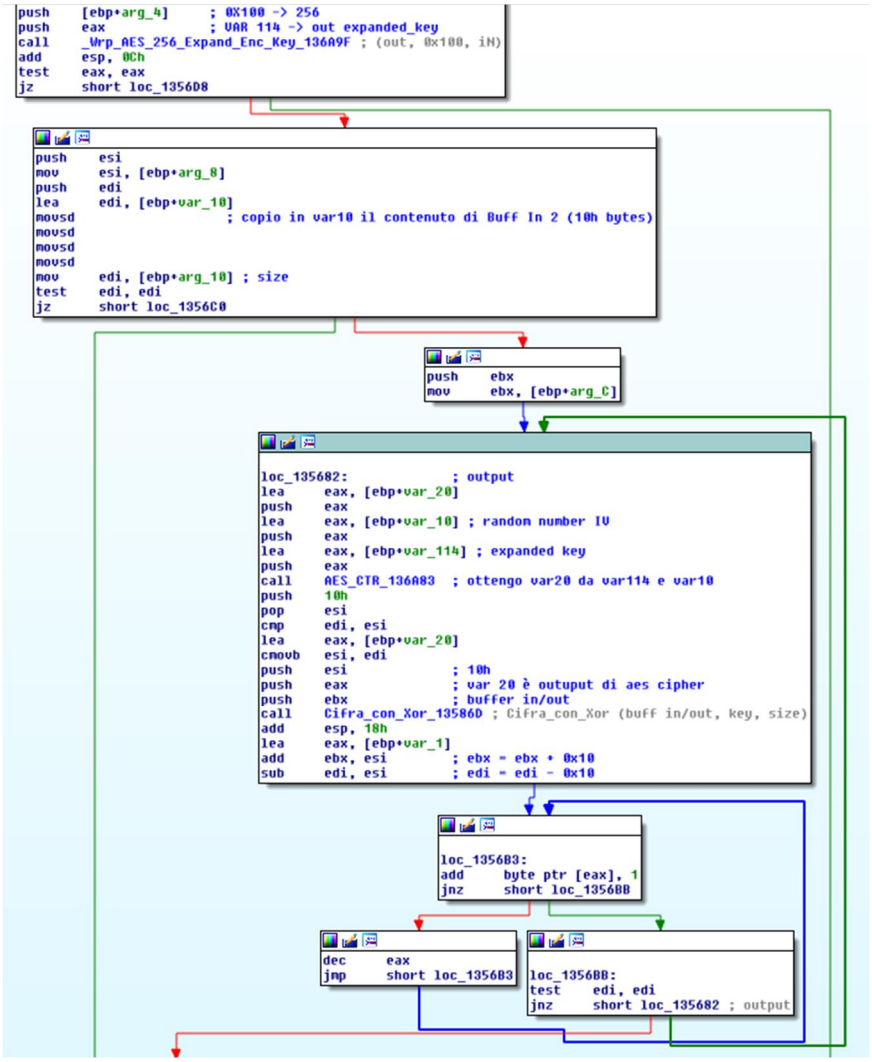
La subroutine Sub_13597B esegue i seguenti passi:

1. Alloca un buffer da 0x58 byte e copia dall'offset 0x4 la chiave privata (*dk_key*) "key_privata"
2. Calcola una nuova coppia di chiavi ECC privata (*dk_new*) e pubblica (*pk_new*)
3. Calcola *dk_new*pk -> shared_key_new* (dove *pk* è la chiave pubblica della sezione di configurazione in json) e il risultato viene "hashato" con SHA-3.
4. Calcola un numero casuale di 16 byte -> *random_16* che verrà utilizzato come IV (vettore di inizializzazione per AES)
5. Cifra il buffer allocato da 0 a 0x24 con AES-256 CTR attraverso il vettore di inizializzazione IV e SHA-3 (*shared_key_new*)
6. Copia nel buffer allocato all'offset 0x24 la chiave pubblica *pk_new*
7. Copia nel buffer allocato all'offset 0x44 il numero casuale *random_16*
8. Calcola il CRC32 del buffer allocato da 0 a 0x24 e salva il risultato all'offset 0x54

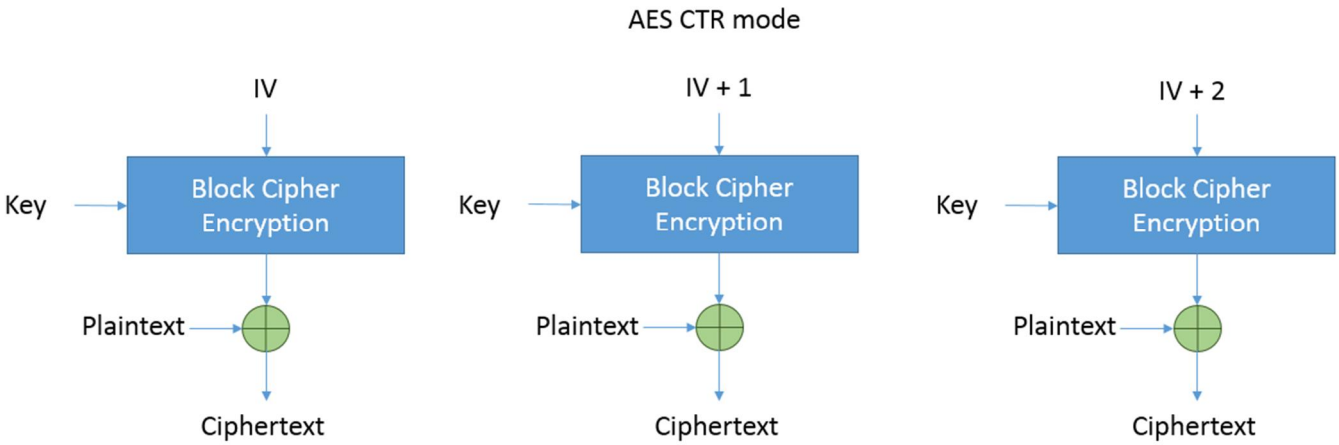
La subroutine Sub_13597B restituisce il puntatore al buffer allocato di 0x58 byte della struttura dati sk_key.
La struttura dati sk_key, che vediamo nella figura a destra, verrà memorizzata nel registro sotto l'omonima voce.



Nella figura sottostante possiamo vedere la chiamata ad AES-256 in modalità CTR:



AES CTR segue il seguente schema:



Struttura dati 0_key

Ora viene generata la struttura dati 0_key, sempre attraverso la chiamata alla subroutine Sub_13597B:

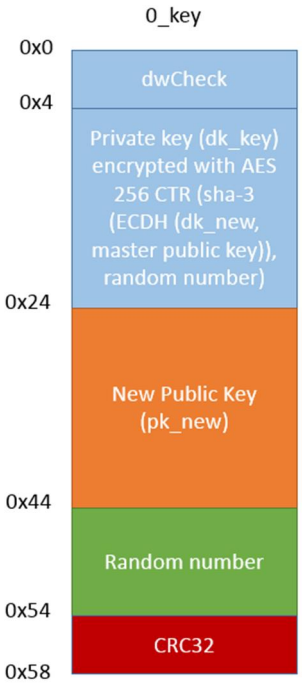
```
pBuff_0_key = Sub_13597B (master_key_pubblica, key_privata, size IN, size out)
```

La procedura per la generazione della struttura dati di 0_key è analoga a quella di sk_key, in questo caso viene utilizzata una "master key pubblica" memorizzata all'interno del file eseguibile al posto della chiave pubblica pk (quella del json).

La master key pubblica "embedded" è:

```
79 CD 20 FC E7 3E E1 B8 1A 43 38 12 C1 56 28 1A
04 C9 22 55 E0 D7 08 BB 9F 0B 1F 1C B9 13 06 35
```

All'interno di 0_key abbiamo la chiave privata dk cifrata attraverso la "master public key".
Anche la struttura dati 0_key, che vediamo nella figura sottostante, verrà memorizzata nel registro sotto l'omonima voce.



Chiave di registro "rnd_ext"

All'interno della chiave di registro **REcfg** viene memorizzato il valore "rnd_ext", che contiene l'estensione dei file cifrati calcolata in modo casuale.

Chiave di registro "stat"

All'interno della chiave di registro **REcfg** viene memorizzato il valore "stat", che contiene la seguente stringa formattata:

```
{"ver":%d,"pid":"%s","sub":"%s","pk":"%s","uid":"%s","sk":"%s","unm":"%s","net":"%s","grp":"%s","lng":"%s","bro":%s,"os":"%s","bit":%d,"dsk":"%s","ext":"%s"}
```

Questa viene memorizzata all'interno di "stat" in forma cifrata e codificata in base64.

Nome	Descrizione
ver	Versione di Sodinokibi
pid	PID del json
sub	SUB del json
pk	PK del json
uid	CRC32 di "processor brand string" e Volume Serial Number (8 bytes)
sk	sk_key in BASE64
unm	Nome utente
net	Nome del computer
Grp	Nome del gruppo di lavoro o dominio
lng	Codice lingua
bro	True / False in base al codice della lingua
Os	Sistema Operativo
Bit	Valore 86 o 64
Dsk	Informazioni dei dischi in base 64 (drive e spazio libero)
Ext	Estensione dei file cifrati

Paesi considerati "amici" sulla base del valore di "bro":

- Romania
- Russia
- Ucraina
- Bielorussia
- Estonia
- Lettonia
- Lituania
- Tajikistan
- Iran
- Armenia
- Azerbaijan
- Georgia
- Kazakistan
- Kyrgyzstan
- Turkmenistan
- Uzbekistan

Il ransomware Sodinokibi termina il processo corrente se la lingua della tastiera risulta essere appartenente alla lista dei paesi considerati "amici".

La stringa formattata “stat” viene cifrata con una chiave master pubblica memorizzata all’interno del file eseguibile.

La master key pubblica “embedded” è:

```
36 7D 49 30 85 35 C2 C3 68 60 4B 4B 7A BE 83 53
AB E6 8E 42 F9 C6 62 A5 D0 6A AD C6 F1 7D F6 1D
```

Istruzioni del riscatto

Successivamente vengono preparate le istruzioni del riscatto partendo dal body che viene estratto dal campo “nbody” della configurazione json.

Il body viene formattato con i seguenti valori:

- uid
- rnd_ext
- stat in base 64

L’”uid” è l’user ID calcolato dal CRC di “processor brand string” e Volume Serial Number che viene utilizzato per comporre l’URL dove sarà possibile effettuare il pagamento del riscatto:

- http://aplebzu47wgazapdqks6vrcv6zcnjppkbxbr6wketf56nf6aq2nmyoyd.onion/<uid>
- http://decryptor.top/<uid>

Termina Processi e Cancella Shadow Copy

In questa fase vengono terminati i processi elencati nel JSON di configurazione alla voce “prc” e cancellate le Copie Shadow di Windows con il seguente comando:

```
cmd.exe /c vssadmin.exe Delete Shadows /All /Quiet & bcdedit /set {default}
recoveryenabled No & bcdedit /set {default} bootstatuspolicy
ignoreallfailures
```

Wipe

In seguito il malware verifica il valore di “wipe” nel JSON di configurazione e se posto a true cancella tutti i file contenuti nelle cartelle che corrispondono al valore “wfld” del JSON di configurazione.

Cifratura dei file

Viene creato un nuovo Thread che rimane in attesa sulla funzione “GetQueuedCompletionStatus”.

Vengono enumerati tutti i file nei dischi locali e nelle cartelle di rete (solo se il parametro “net” del JSON di configurazione è a valore “*true*”), per procedere quindi alla cifratura dei file.

In ogni cartella viene creato un file .lock e le istruzioni del riscatto con nome *{estensione random}-readme.txt*.

Vengono esclusi dalla cifratura i file e le cartelle che corrispondono al campo JSON “wht” che contiene i sottocampi “fld”,”fls” e ”ext”, che stanno rispettivamente per “folder”, “files” e “estensione”.

Ne riportiamo un esempio:

```
"wht": {

        "fld": ["google", "mozilla", "$windows.~bt", "programdata",
"$recycle.bin", "program files (x86)", "appdata", "msocache", "program
files", "windows.old", "$windows.~ws", "application data", "perflogs",
"windows", "boot", "intel", "system volume information", "tor browser"],

        "fls": ["bootsect.bak", "autorun.inf", "ntldr", "ntuser.dat.log",
"ntuser.ini", "boot.ini", "ntuser.dat", "bootfont.bin", "desktop.ini",
"thumbs.db", "iconcache.db"],

        "ext": ["exe"]

    }
```

Per ogni file candidato alla cifratura viene generata una chiave di Salsa20 nel seguente modo:

```
push    eax             ; var_20
call    _Calcola_Key_Privata_Pubblica_1355B8 ; Calcola_Key_Privata_Pubblica (pKeyPrivata, pKeyPubblica)
lea     eax, [ebp+var_40]
push    eax
lea     eax, [ebp+var_20]
push    offset pk_key_14D5A0 ; pk_key del registro
push    eax             ; var 20
call    _Calcola_SHA3_ECDH_135822 ; (Buffer IN, Key, Buffer OUT)
lea     eax, [ebp+var_20]
push    20h
push    eax
call    _Wrp_ZeroMemory_135966
mov     esi, [ebp+arg_0] ; struttura dati
lea     eax, [ebp+var_40] ; key di cifratura che viene copiata nella tabella master di Salsa20
push    40h
push    100h
push    eax
lea     edi, [esi+108h]
push    edi
call    __Set_Salsa_Tabella_136EA3
lea     eax, [ebp+var_40]
push    20h
push    eax
call    _Wrp_ZeroMemory_135966
add     esi, 0F8h
push    8               ; size vettore
push    esi             ;  Buffer Vettore Inizializzazione
call    _Calcola_RandomNumber_13578B ; _Calcola_RandomNumber (PBuffer, dwSize)
push    esi             ;  puntatore al Vettore di Inizializzazione IV
push    edi             ; edi punta alla struttura Dati offset 0x108 Tbl Master Salsa
call    _Set_IV_Tabella_Salsa_136E85
add     esp, 44h
push    20h             ;  size
push    ebx             ;  buffer
push    0
call    _CRC32_1356DC   ; calcola in eax il CRC32 (val, buffer, size)
mov     ecx, [ebp+arg_0] ; struttura dati
add     esp, 0Ch
pop     edi
mov     [ecx+100h], eax ; crc32 del buffer D8
mov     eax, dword_14D714
pop     esi
mov     [ecx+104h], eax
pop     ebx
mov     esp, ebp
pop     ebp
retn
_CalcolaKeySalsa20_13289C endp
```

L'algoritmo di cifratura utilizzato da Sodinokibi è Salsa20.

La chiave di cifratura per Salsa20 è ottenuta in questo modo:

1. Calcola una nuova coppia di chiavi Private/Pubbliche ECC (dk_new_file, pk_new_file)
2. Calcola SHA-3 (*dk_new_file*pk_key) -> shared_key_salsa* (dove *pk_key* è la chiave pubblica memorizzata nel registro alla voce pk_key). In *shared_key_salsa* avremo la chiave che viene inserita nella tabella master di Salsa20.
3. Calcola un numero casuale di 8 byte per il Vettore di inizializzazione della tabella master di Salsa20.
4. Compone la tabella master di Salsa20.

Viene creata in memoria una struttura dati che contiene:

- Handle del file da cifrare
- Sk_key
- 0_key
- pk_new_file
- Il vettore di inizializzazione di Salsa20
- Il CRC32 di pk_new_file
- Tabella master di Salsa20

Questa struttura dati viene passata al Thread creato in precedenza attraverso le funzioni API:

- CreateIoCompletationPort
- PostQueuedCompletionStatus

Il thread è in attesa sulla funzione API GetQueuedCompletionStatus, quando riceve una nuova chiamata inizia la fase di cifratura del file attraverso l'algoritmo Salsa20 e successivamente appende una parte della struttura dati che contiene i seguenti campi:

- Sk_key
- 0_key
- pk_new_file
- Il vettore di inizializzazione di Salsa20
- Il CRC32 di pk_new_file

La dimensione della parte appesa varia in base alla versione del malware Sodinokibi. Nelle versioni 1.0 e 1.1 la lunghezza è di 0xE0 byte mentre nella versione 1.2 è di 0xE4 byte.

In figura possiamo vedere lo schema di cifratura di Sodinokibi versione 1.1:

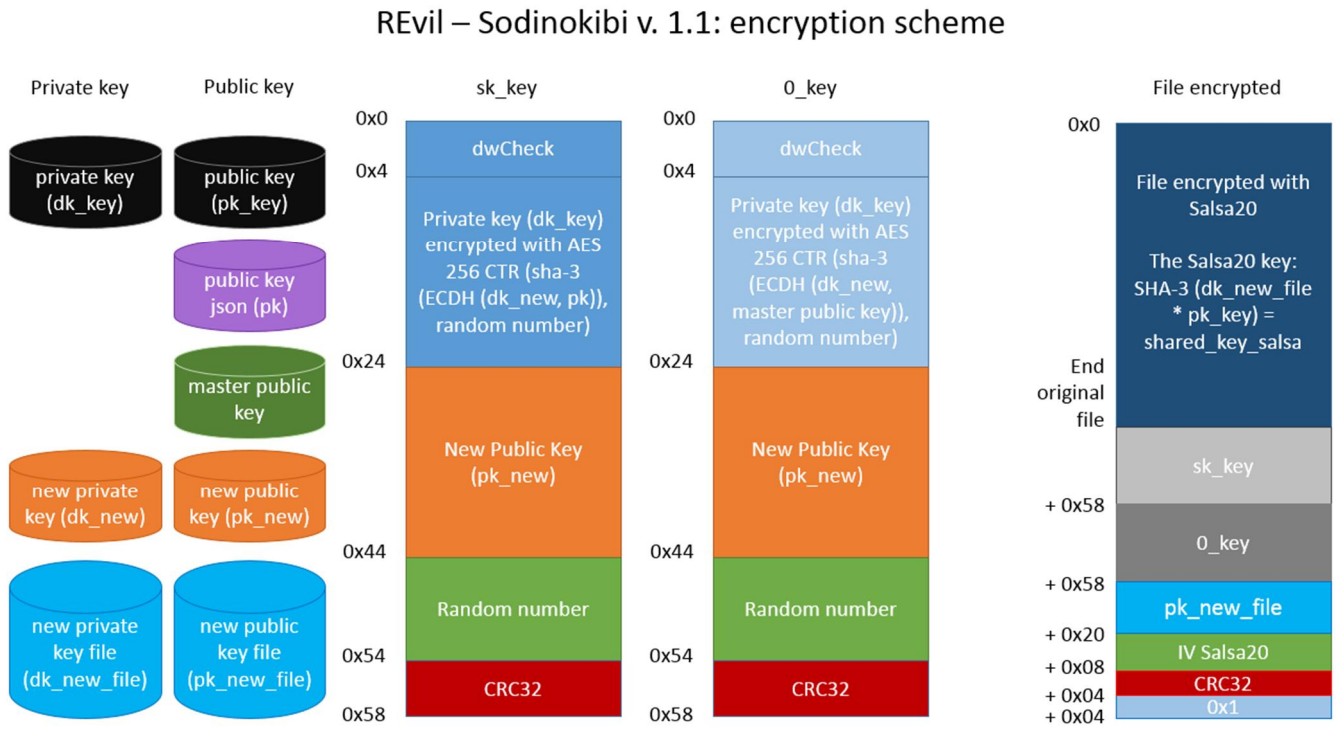


Immagine del desktop

Terminata la cifratura dei file, il passo successivo è la modifica dell'immagine del desktop, che possiamo vedere nella figura sottostante:

L'immagine viene generata utilizzando le funzioni API per la grafica BitMap e attraverso la funzione "DrawText" viene inserito il testo che viene caricato dal JSON di configurazione al campo "img".

Server C2

All'interno del JSON di configurazione vi troviamo una lista di 1079 domini. Sodinokibi contatta ogni dominio di questa lista generando un URL attraverso un algoritmo DGA utilizzando i seguenti termini:

Termine	Estensione
• wp-content • pictures • news • pics • admin • data • temp • graphic • game • static • assets • tmp • uploads • images • include • image • content	• jpg • gif • png

https://<host>/<termine 1>/<termine 2>/<random chars>.<estensione>

Alcuni esempi:

- https://stagefxinc.com/wp-content/pictures/pmkapi.jpg
- https://birthplacemag.com/admin/pictures/hpxxqbak.gif
- https://clemenfoto.dk/news/pics/ohxkyt.gif
- https://wineandgo.hu/admin/pics/ahlpbrzo.jpg
- https://lexced.com/data/temp/hpttgdyg.png

Sodinokibi attraverso un "POST" invia ad ogni dominio della lista la struttura dati "stat" in forma cifrata.

Dalla nostra analisi solo i seguenti domini hanno risposto con "HTTP/1.1 200 OK":

www.zuerich-umzug.ch	geitoniatonaggelon.gr
belofloripa.be	insane.agency
www.soundseeing.net	acb-gruppe.ch
utilisacteur.fr	www.cardsandloyalty.com
www.airserviceunlimited.com	www.sbit.ag
www.mediahub.co.nz	yourhappyevents.fr
www.irizar.com	tieronechic.com
www.cleanroomequipment.ie	mariajosediazdemera.com
www.pinkxgayvideoawards.com	www.skyscanner.ro
www.rhino-turf.com	11.in.ua
mike.matthies.de	funworx.de
drbenveniste.com	www.omnicademy.com
scotlandsroute66.co.uk	www.bratek-immobilien.de
m2graph.fr	metroton.ru

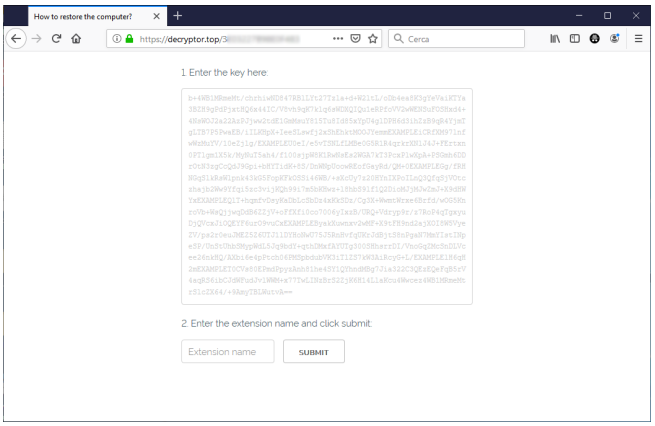
Ma questo non significa che uno di questi domini sia quello del server C2 di Sodinokibi.

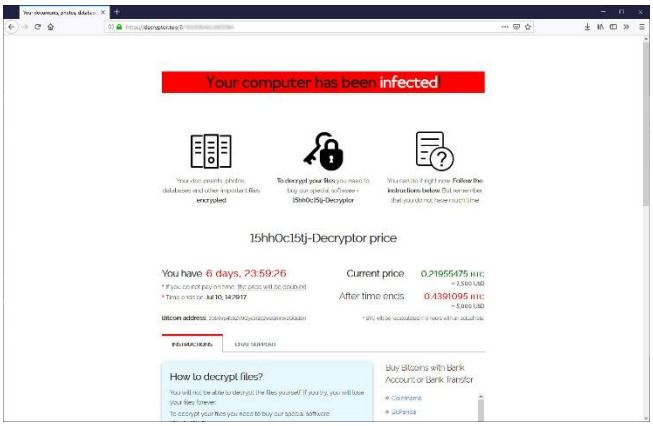
Pagamento riscatto

In base alle istruzioni del riscatto la vittima deve collegarsi ai seguenti domini per le modalità di pagamento:

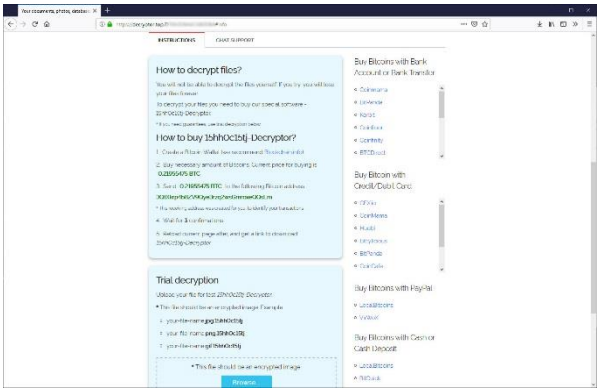
- http://aplebzu47wgazapdqks6vrcv6zcnjppkbxbr6wketf56nf6aq2nmyoyd.onion/<uid>
- http://decryptor.top/<uid>

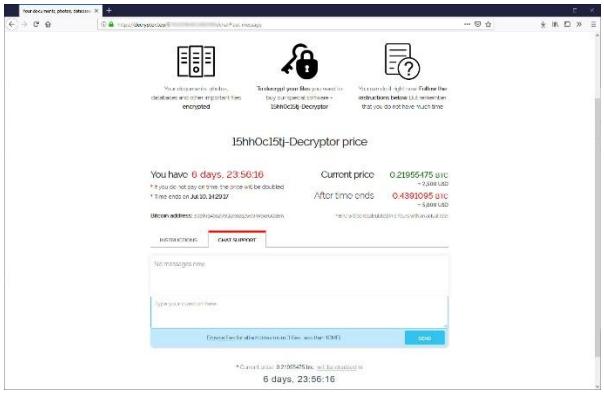
Come primo step viene richiesto di inserire l'estensione random e il valore "Key" contenuto nelle istruzioni del riscatto (si tratta della versione di "stat" cifrata in base 64).





Una volta inseriti i dati viene generato l'importo del pagamento e vengono fornite le istruzioni su come reperire i BitCoin ed in aggiunta una Chat di supporto come è possibile vedere nelle successive immagini:





Il wallet per il pagamento è generato in automatico per ogni vittima, il prezzo del riscatto è di 2500$ che raddoppia a 5000$ se non viene effettuato il pagamento entro 7 giorni.

Come avviene la decifratura

Per recuperare i file cifrati da Sodinokibi è necessario conoscere la private key "dk_key". Questa chiave viene cifrata all'interno di "sk_key" e di "0_key".

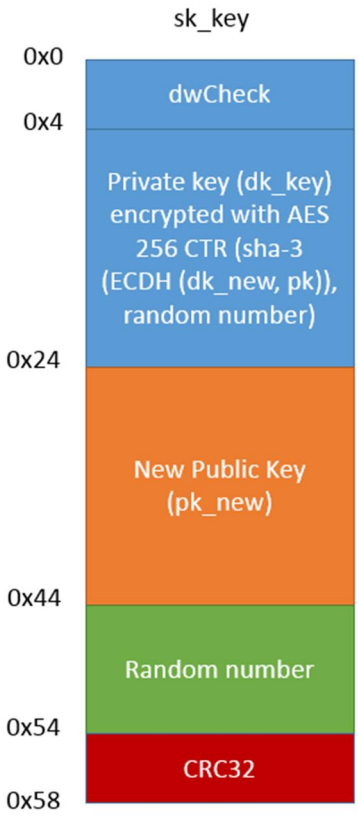
Gli autori di Sodinokibi recuperano la chiave "dk_key" nei seguenti modi:

1. Decifrando sk_key
2. Decifrando 0_key

Per decifrare “sk_key” gli autori di Sodinokibi utilizzano la chiave privata “dk” che solo loro conoscono. La chiave privata “dk” è la controparte della chiave pubblica “pk” memorizzata nel json di configurazione.
All’interno della struttura “sk_key” vi è in chiaro la chiave pubblica “pk_new”.
Viene calcolato il valore: dk * pk_new = *shared_key_new*
La “*shared_key_new*” è uguale anche a: dk_new*pk.

La private key (dk_key) è cifrata con AES-256 CTR attraverso la “SHA-3 (shared_key_new)” e il random number (IV) che si trova all’offset 0x44.
Decifrando con AES-256 il buffer da 0x4 a 0x24 con “SHA-3 (shared_key_new)” e il random number si ottiene “dk_key”.

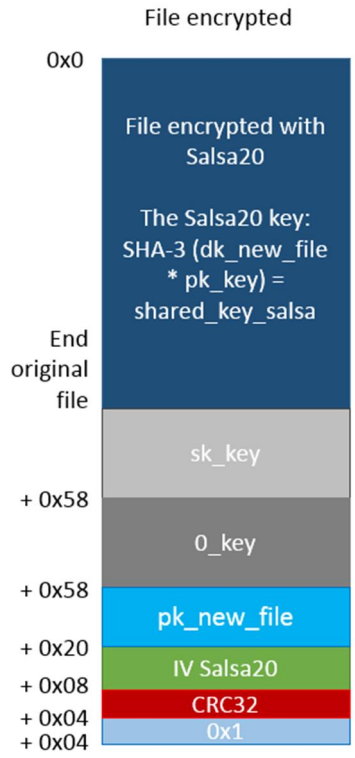
La stessa operazione può essere ripetuta partendo da “0_key”, in questo caso per ottenere la “dk_key” viene utilizzata la master private key che solo gli autori di Sodinokibi conoscono.

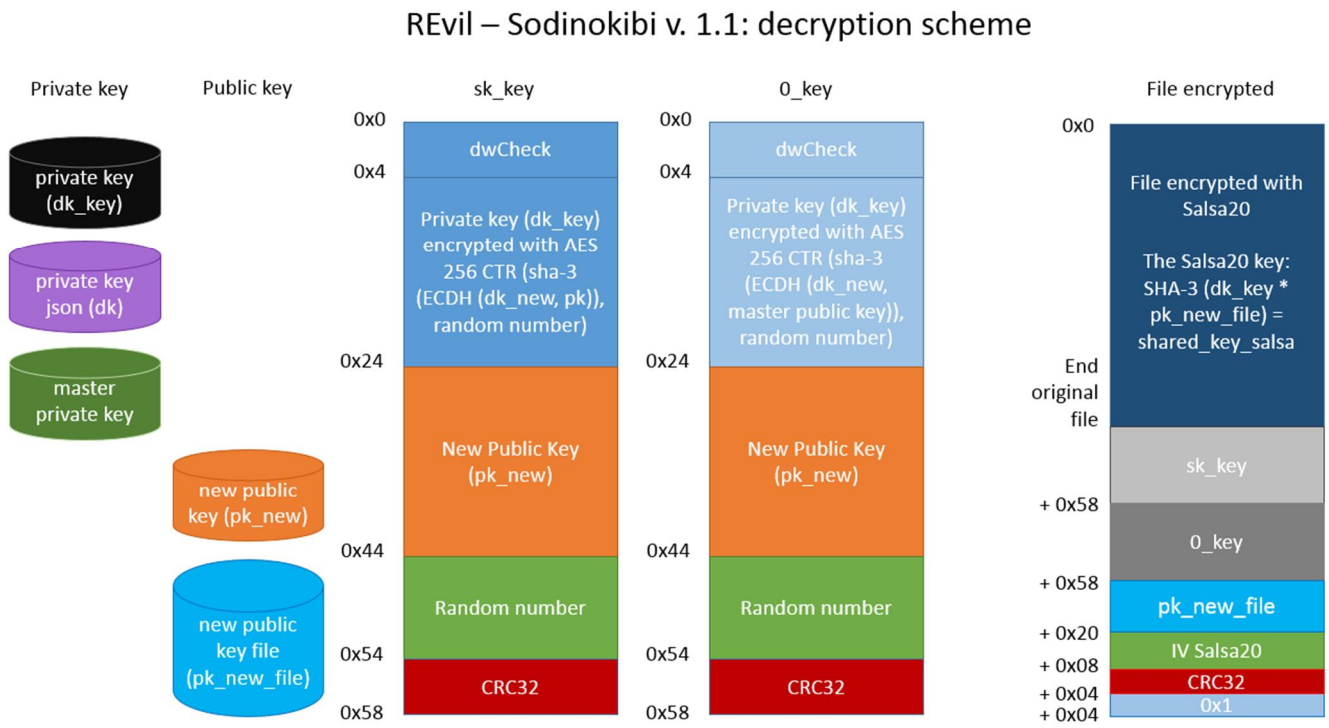


Nota ora la dk_key per determinare la chiave di cifratura utilizzata in Salsa20 si esegue la seguente operazione:
SHA-3 (dk_key *pk_new_file) = shared_key_salsa
Dove pk_new_file è la chiave pubblica appesa in chiaro alla fine del file cifrato.

shared_key_salsa è anche uguale a SHA-3 (*dk_new_file*pk_key)*

In shared_key_salsa avremo la chiave che viene inserita nella tabella master di Salsa20.
A questo punto è possibile decifrare i file attraverso la shared_key_salsa.





Versioni

Gli autori di Sodinokibi hanno sviluppato le seguenti versioni:

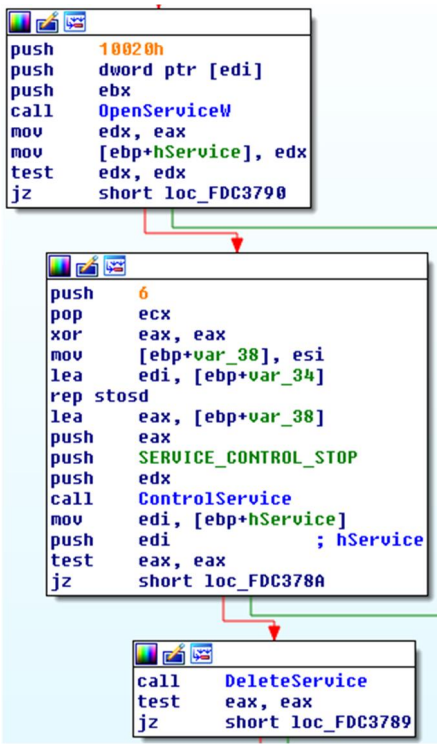
Versione	Data	Dimensione Dati appesi
1.0a	2019-04-23	0xe0
1.0b	2019-04-27	0xe0
1.0c	2019-04-29	0xe0
1.1	2019-05-05	0xe0
1.2	2019-06-10	0xe4
1.3	2019-07-08	0xe4

Versione 1.2

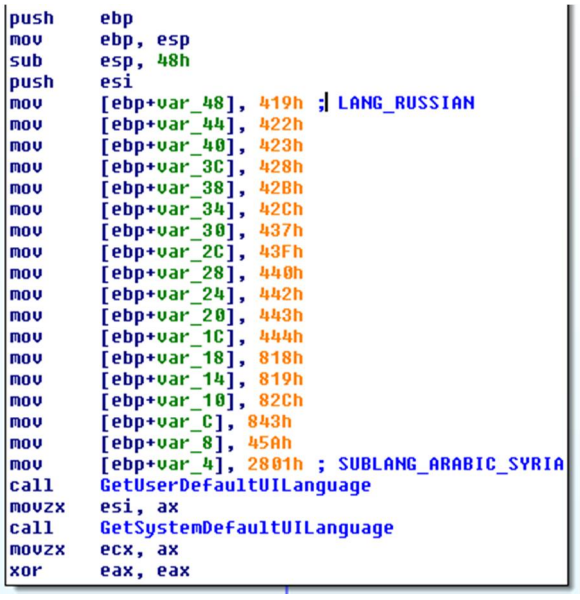
Nella versione 1.2 è stata aggiunta la chiave di registro “sub_key” che contiene la chiave pubblica del json di configurazione (pk) e la dimensione dei datti appesi nei file cifrati è di 0xe4 bytes, dove viene aggiunta un ulteriore dword di controllo con valore 0.

Versione 1.3

Nella versione 1.3 è stato aggiunto il campo “svc” nel json di configurazione, in questo campo è contenuto una lista di servizi da cancellare, come possiamo vedere in figura.



Inoltre per verificare se la vittima appartiene ad uno stato “amico”, oltre al controllo della tastiera sono stati aggiunti i controlli sulla lingua default e di sistema, come possiamo vedere in figura.



Usa WQL per la determinazione della creazione dei processi:

```
SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA
'Win32_Process'
```

Inoltre utilizza una nuova chiave di registro al posto della “**REcfg**”:

- HKEY_LOCAL_MACHINE\SOFTWARE\QtProject\OrganizationDefaults

All’interno di QtProject\OrganizationDefaults vengono creati i seguenti valori:

- pvg
- sxsP
- BDDC8
- f7gVD7
- Xu7Nnkd
- sMMnxpgk

Tabella di confronto tra la versione 1.2 e 1.3:

Vers. 1.2: REcfg	**Vers. 1.3: QtProject\OrganizationDefaults**
sub_key	pvg
pk_key	sxsP
sk_key	BDDC8
0_key	f7gVD7
rnd_ext	Xu7Nnkd
stat	sMMnxpgk

Telemetria

E' stato monitorato l'andamento delle campagne malware di Sodinokibi nell'arco dei mesi di Aprile 2019/Luglio 2019.

Nella tabella sottostante possiamo vedere le campagne monitorate:

Data Campagna	Campagna	PK	PID	SUB	Versione	Data compilazione
25/04/2019	Oracle Weblogic	nAjfiPcolyelwwCkM1hLhXo5HUQMtrAB+7m8eHzerho=	7	3	1.0a	2019-04-23 18:21:53
25/04/2019	Oracle Weblogic	nAjfiPcolyelwwCkM1hLhXo5HUQMtrAB+7m8eHzerho=	7	3	1.0a	2019-04-23 18:21:53
25/04/2019	Oracle Weblogic	nAjfiPcolyelwwCkM1hLhXo5HUQMtrAB+7m8eHzerho=	7	3	1.0a	2019-04-23 18:21:53
25/04/2019	Oracle Weblogic	nAjfiPcolyelwwCkM1hLhXo5HUQMtrAB+7m8eHzerho=	7	3	1.0a	2019-04-23 18:21:53
25/04/2019	Oracle Weblogic	nAjfiPcolyelwwCkM1hLhXo5HUQMtrAB+7m8eHzerho=	7	3	1.0a	2019-04-23 18:21:53
		a54FxmOM4c90SBAgCVw4ykJv62ImcbOvaHKwO8OKegI=	19	29	1.0b	2019-04-27 18:11:51
		a54FxmOM4c90SBAgCVw4ykJv62ImcbOvaHKwO8OKegI=	19	29	1.0c	2019-04-29 19:06:06
		a54FxmOM4c90SBAgCVw4ykJv62ImcbOvaHKwO8OKegI=	19	29	1.0c	2019-04-29 19:06:06
		N3IqbCUZr/g/XgALTUaGw7K8E5UvA+CcRa5zto0xg0A=	20	45	1.0c	2019-04-29 19:06:06
		TmrkEVU29HHz1nfhwl0C6p4U5syGzUCmcyAJQnZSHyY=	8	10	1.0c	2019-04-29 19:06:06
		4hKQrOidB69uTPA/7uaOuTipRsh2y956X1K+jyyLUjA=	17	11	1.1	2019-05-05 17:38:48
		eYI9jfld2wfrBiZk/ABspJesaySH6q+XbmHRQ55NBkE=	19	100	1.1	2019-05-19 18:08:46
		w0qhPcoO83YCbvmGl4ySs7ZiTUaT5YAk0DXIM/hOnjQ=	20	44	1.1	2019-05-22 18:42:29
		Xew60HCSStmaZwEnoW4XuhBiy5I3SyKugEH5PM4P7RA=	15	19	1.1	2019-05-22 18:42:29
		io3clxJXtLLzcA1anNSmn//tKeld5pGV/mVuqwvms3g=	20	46	1.1	2019-05-22 18:42:29
		eYI9jfld2wfrBiZk/ABspJesaySH6q+XbmHRQ55NBkE=	19	100	1.1	2019-05-22 18:42:29
		ClwOJSOhyaamJ5eplhJrLN5UJdwH29Ky8t+Yn3WeLzg=	30	128	1.1	2019-05-24 14:41:21
		duPwGxBEa19yzAI27JhOVXw155oZWe3CWVbWJ7uwhBU=	16	165	1.1	2019-05-24 14:41:21
24/05/19	RDP	2Dj6WyDEOKff6CVJadXjX+ogDuXN/XnIdrVWffa6/B0=	19	36	1.1	2019-05-24 14:41:21
03/06/19	Malpam	pzprC6xbhNFhM/+qJI6gCrd2pnCgyRdai+B89OUhWAw=	30	97	1.1	2019-05-24 14:41:21
		m7cFgORjlUsRFy4odzcrLk+3iOTw9TNGLdSy6RjQImQ=	19	96	1.1	2019-06-03 18:09:45
		pzprC6xbhNFhM/+qJI6gCrd2pnCgyRdai+B89OUhWAw=	30	97	1.1	2019-06-03 18:09:51
		U5gGGTWKYrgvh5QFI+53Jc7aj8ntwjj0C4ai0/2A+jg=	34	298	1.1	2019-06-03 18:09:51
		N9tiPqA45L8cXACRHIBdJFayV8M5MEF4JjppDRO+oHU=	30	113	1.1	2019-06-03 18:09:51
		pzprC6xbhNFhM/+qJI6gCrd2pnCgyRdai+B89OUhWAw=	30	97	1.1	2019-06-03 18:09:51
		p+iVJIiHGF12r1Q7fPSAF3Y36m0DmS4bbOtZMLKszAI=	16	288	1.1	2019-06-03 18:09:51
		1LSb3+cEvUYZYvzU06n8wFiQCczYZ0MrZwUCy0HN7TY=	34	295	1.1	2019-06-03 18:09:51
		fXWQXz0Or53eh4p5JZnqYIiIQ+tPjrrni5z6Y+Ocvw0=	16	267	1.1	2019-06-03 18:09:51
		F5YmiEk1fBN5E7SkF7sRqBE5+QRpLLYtk0ONclTtzWM=	16	250	1.1	2019-06-03 18:09:51
		KtKn8udbrebS5jbzcimIkGAbGMIwX9Ks85rOWrmJ23Q=	28	285	1.2	2019-06-10 15:29:32
		jD6pLfwUHIEoWBKadIZ4A78CLm8I0UKIzdzW7XautWE=	33	357	1.2	2019-06-10 15:29:32
		w2TWFCLDTFMuBv5VN6eA5NHyUM7SRRLt+hIuKWXk8mE=	40	450	1.2	2019-06-10 15:29:32
		PdQtqjCAKZmlJn1Fbw1ZGic+XVzOOTwt4Gm1gdXGsXg=	16	314	1.2	2019-06-10 15:29:32
		X5KVRMdkoLhmeigRMY9Ve4j+/3uVeOOjDgMAM4V22mA=	12	313	1.2	2019-06-10 15:29:32
		Xew60HCSStmaZwEnoW4XuhBiy5I3SyKugEH5PM4P7RA=	15	19	1.2	2019-06-10 15:29:32
		/SvNLPYVd04yhjQWFntNHZ0bsHYz2DzRIF+HjkQuTmE=	33	331	1.2	2019-06-10 15:29:32
18/06/19	Malspam – Booking	Js9mSQ5X8GfxGiHDyNSEBzRCDIONrR0tet7eKc6ptCk=	27	439	1.2	2019-06-10 15:29:32
19/06/19	Malspam – DHL	ClwOJSOhyaamJ5eplhJrLN5UJdwH29Ky8t+Yn3WeLzg=	30	128	1.2	2019-06-10 15:29:32
		Js9mSQ5X8GfxGiHDyNSEBzRCDIONrR0tet7eKc6ptCk=	27	439	1.2	2019-06-18 19:36:45
		w6mw66IFMUJDfNK5Y4RQDLCGX6MPgfNXIaY42EhURkM=	17	538	1.2	2019-06-18 19:36:45
		vYOXI2Z84mknj8GgTaOG/tyi9eAg0Kv8cTvqCPE3Jkg=	7	474	1.2	2019-06-18 19:36:45
19/06/19	Winrar	vYOXI2Z84mknj8GgTaOG/tyi9eAg0Kv8cTvqCPE3Jkg=	7	474	1.2	2019-06-18 19:36:45
19/06/19	Winrar	vYOXI2Z84mknj8GgTaOG/tyi9eAg0Kv8cTvqCPE3Jkg=	7	474	1.2	2019-06-18 19:36:45
24/06/19	RigEK	qmLSnN9s+6ZosKo1tV0sbdd6RjBKuJ4pkq66+7tRWHY=	35	531	1.2	2019-06-18 19:36:45
26/06/19	Targeting South Korea	w6mw66IFMUJDfNK5Y4RQDLCGX6MPgfNXIaY42EhURkM=	17	538	1.2	2019-06-18 19:36:45
25/06/19	Malspam – Booking	RJLY2jLnGa3qAJx5s3slwl0flZjJFSxHjZgDYwHKaBI=	27	564	1.2	2019-06-24 15:53:35
01/07/19	RigEK	Zrui05IT0bzVjJv7WuNIq6PZyXjBMEStA2eSxQT8TjY=	22	607	1.2	2019-06-24 15:53:35

Nella tabella possiamo vedere i seguenti campi:

1. Data Campagna
2. Tipologia Campagna
3. PK (chiave pubblica presente nel JSON di configurazione)
4. PID presente nel JSON di configurazione
5. SUB presente nel JSON di configurazione
6. Versione di Sodinokibi
7. Data di compilazione del file master di Sodinokibi

Il campo PID va ad identificare il gruppo che ha acquistato il servizio Ransomware Sodinokibi (RAAS). Il campo SUB sembra invece identificare la "SUBSCRIPTION" ovvero il periodo di validità del servizio.

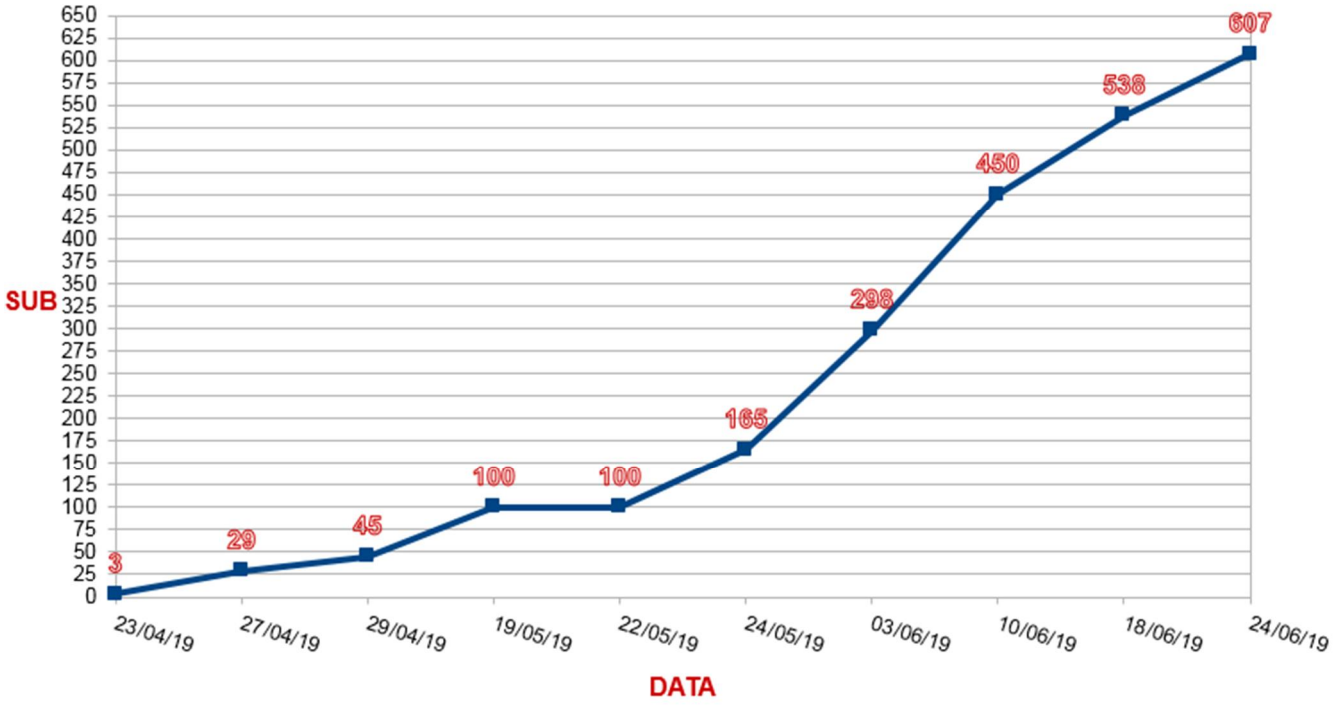
Le coppie PID & SUB uguali hanno sempre la stessa chiave pubblica PK come si può vedere nei casi PID: 7 e SUB: 3.

Si può notare che il gruppo con PID 7 è stato il primo ad usare Sodinokibi attraverso la campagna che sfruttava la vulnerabilità di Oracle Weblogic il 25 Aprile 2019 (SUB: 3), lo stesso gruppo sembra essere associato alla campagna dell'attacco di Watering Hole al distributore italiano di Winrar il 19 Giugno 2019 con una nuova SUB: 474.

Come si può vedere il gruppo con PID: 7 ha acquistato più periodi di sottoscrizione. Utilizzando il trio PID-SUB-PK si possono individuare le campagne associate allo stesso attore.

Fino ad inizio Luglio 2019 il PID con valore più alto è stato 40 facendo quindi ipotizzare che vi siano almeno 40 gruppi differenti. Il valore massimo di SUB è stato 607 che potrebbe indicare siano stati acquistati almeno 607 periodi di sottoscrizione.

Nel grafico visibile qui sotto mettiamo in relazione la data di compilazione del malware con il valore SUB contenuto nel JSON di configurazione. E' subito possibile notare come la crescita sia in forte aumento facendo supporre che il CryptoMalware Sodinokibi sia appunto distribuito con il metodo "as-a-service"



Conclusioni

Il Ransomware Sodinokibi è stato realizzato da soggetti con un certo livello di conoscenza tecnica ed è attivamente sviluppato, molto probabilmente non sono alla loro prima creazione di Ransomware.

Il progetto è chiaramente strutturato per essere distribuito come servizio (raas).

Il Ransomware Sodinokibi utilizza lo scambio di chiavi ECDH e la cifratura dei dati avviene attraverso l'algoritmo Salsa20.

La distribuzione nell'ultimo mese ha avuto una forte impennata, attraverso varie tipologie di diffusione come Malspam, RigEK, attacchi RDP, ecc. Molto probabilmente la recente decisone dei creatori di GandCrab di chiudere ufficialmente il progetto ha aperto la strada a nuovi attori e Sodinokibi sembra aver sfruttato subito l'occasione.

IOC

MD5:

```
DB42F17991A7BA10218649B978D78674
E713658B666FF04C9863EBECB458F174
16863F6727BC5DD44891678EBCA492D2
FD3F3AF76D31D8F134E2E02463D89D29
6E543C13594F987A6051BC3D9456499F
CCFDE149220E87E97198C23FB8115D5A
FB68A02333431394A9A0CDBFF3717B24
692870E1445E372DDD82AEDD2D43F9B8
DB6D3A460DEDE97CA7E8C5FBFAEF3A72
48A673157DA3940244CE0DFB3ECB58E9
79F2341510D9FB5291AEFC3E69D18253
3DF42FA9732864A9755F5C8FB7ED456A
```

URL:

```
aplebzu47wgazapdqks6vrcv6zcnjppkbxbr6wketf56nf6aq2nmyoyd.onion
decryptor.top
```