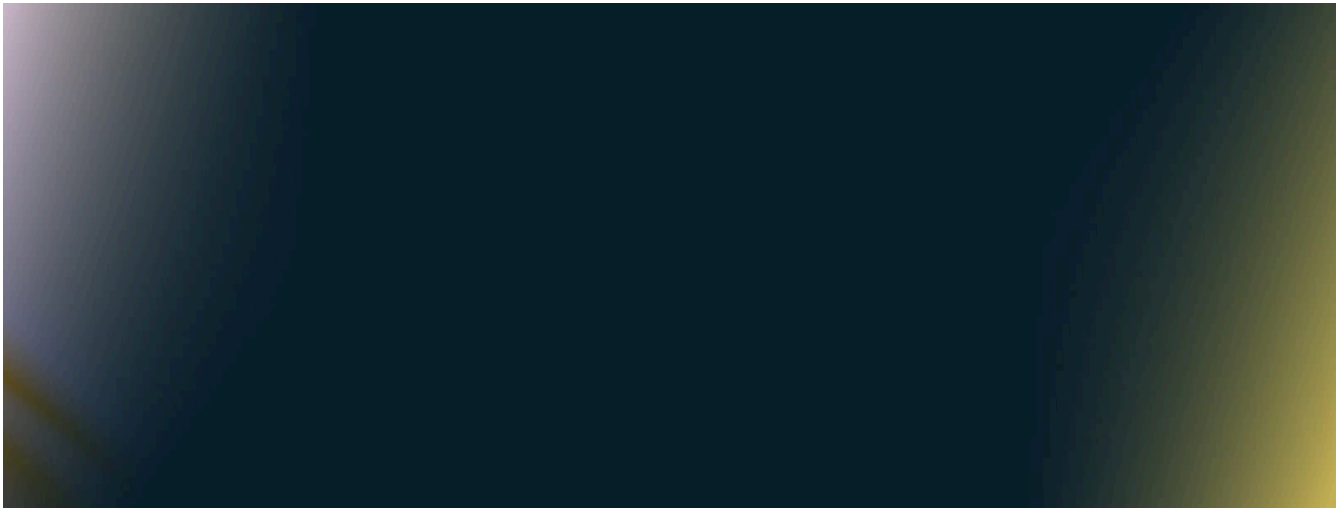


DeadLock ransomware: Breaking down a Rust-based encryptor with decentralized recovery infrastructure

 microsoft.com/en-us/security/blog/2026/08/10/deadlock-ransomware-breaking-down-a-rust-based-encryptor-with-decentralized-recovery-infrastructure

Microsoft Threat Intelligence

August 10, 2026



Microsoft Threat Intelligence tracks DeadLock ransomware as an emerging financially motivated operation distinguished by its use of decentralized infrastructure to support victim communications and data leak operations. Its recovery ecosystem combines the Session messaging network with blockchain-backed services that store and deliver resources used throughout the extortion process. This architecture likely increases the resilience of portions of its communication, leak-hosting, and negotiation infrastructure, allowing DeadLock operators to recover from some disruption efforts while maintaining continuity for victims. Microsoft has observed DeadLock ransomware being deployed by multiple groups including an affiliate of the Lynx and INC ransomware ecosystems.

First observed in July 2025, DeadLock operators employ double extortion tactics, encrypting victim environments while threatening to publicly release exfiltrated data. As of July 2026, the operators have published more than 80 compromised organizations on their data leak site, called the DeadLock blog, with more than half of the claimed victims in Europe. Microsoft identified DeadLock ransomware impacting organizations across information technology (IT), mining, transportation and logistics, manufacturing, hospitality, consumer goods, and other sectors in Europe, Asia, North America, South America, and Africa.

The DeadLock encryptor includes a resource-aware throttling mechanism designed to maintain system responsiveness during encryption. In addition to its encryption capabilities, the ransomware also appears to implement language or country-based geofencing designed to avoid running in environments associated with former Soviet and Commonwealth of

Independent States (CIS)-linked countries as well as select Middle Eastern countries, a pattern commonly observed among ransomware operators believed to operate from those regions. Together, these capabilities demonstrate how DeadLock combines established [ransomware tradecraft](#) with decentralized infrastructure designed to improve operational resilience.

In this blog, we present a technical analysis of the DeadLock ransomware encryptor, covering its execution flow, defense evasion techniques, encryption design, and post-encryption behaviors, including a decentralized recovery chat system. We also provide indicators of compromise (IOCs), [Microsoft Defender](#) detections, and mitigation guidance to help organizations defend against this threat and similar ransomware activity.

Pre-encryption

Configuration parsing

Before performing any malicious activity, the DeadLock encryptor decrypts an embedded configuration blob using XOR decoding with an 8-byte key.

Below are the malware's configuration fields and their values.

Field	Value
<i>Victim UID</i>	<redacted>
<i>Malware public key</i>	03bf50bbf97c4e951e66ff12b689a37a3ce675b4921e254eae76da77573843e4a9
<i>Encryption rule</i>	1000,05052429880,025124288000,010524288000,F991114288000
<i>Language exclude list</i>	Geofencing language IDs (see)
<i>Process stop list</i>	Processes to terminate (see)
<i>Service stop list</i>	Services to stop and delete (see)
<i>File exclude list</i>	Extensions and file names to avoid encrypting (see)
<i>Directory exclude list</i>	Pre-traversal filter with directories to avoid encrypting (see)
<i>Sub-path Exclude List</i>	Sub-paths to avoid encrypting during traversal (see)
<i>Text ransom note</i>	Full text ransom note content (see)
<i>HTML recovery chat</i>	Full HTML/JS interactive chat page (see)

Language geofencing

As an early exit check, the malware queries the system's default and user interface (UI) languages. If either language matches the exclude list in the configuration, the malware self-deletes immediately without performing any encryption.

The following languages trigger this exit behavior:

LANGID	Language	Country
1049	Russian	Russia
1058	Ukrainian	Ukraine
1059	Belarusian	Belarus
1064	Tajik (Cyrillic)	Tajikistan
1065	Persian	Iran
1067	Armenian	Armenia
1068	Azeri (Latin)	Azerbaijan
1079	Georgian	Georgia
1087	Kazakh	Kazakhstan
1088	Kyrgyz	Kyrgyzstan
1090	Turkmen	Turkmenistan
1114	Syriac	Syria
2072	Romanian (Moldova)	Moldova
2092	Azeri (Cyrillic)	Azerbaijan
2115	Uzbek (Cyrillic)	Uzbekistan
8193	Arabic	Oman
9217	Arabic (Yemen)	Yemen

Command-line processing and privilege elevation

The encryptor's behavior branches based on command-line arguments and the current privilege level. If a target directory path is provided as the command-line argument, the malware skips all preparation steps and jumps directly to encryption. This feature allows the operator to invoke the encryptor with specific targets for focused encryption. If no sub-commands are provided and the process is already elevated, the malware proceeds normally through all execution phases.

The more interesting case occurs when no command-line argument is provided while the process is not elevated. In this scenario, the malware attempts to gain administrator privileges through a batch-script-based elevation technique. It generates a randomly named *.cmd* file (8 uppercase characters, such as *ESYEKQSY.cmd*) and executes it using *ShellExecuteW* with the *RunAs* verb, which triggers the Windows User Account Control (UAC) consent dialog. If the user denies the prompt, the malware retries up to 10 times before giving up and exiting.

During dynamic analysis, the sample did not successfully relaunch itself with elevated privileges. As a result, full pre-encryption preparation appears to require execution from an already elevated context. When invoked with a target path, the malware bypasses preparation and proceeds directly to encrypt accessible files. This behavior is specific to the analyzed sample and may change in later variants.

Token privilege escalation

When running with administrator privileges, the malware further expands its access by enabling *SeDebugPrivilege*, *SeRestorePrivilege*, *SeBackupPrivilege*, *SeTakeOwnershipPrivilege*, *SeAuditPrivilege*, and *SeSecurityPrivilege*. These privileges increase the malware's ability to interact with system processes, protected files, and security-related settings, helping it overcome common access restrictions and maximize the scope of files and resources it can target during the encryption phase.

Recycle bin emptying

The malware silently empties the recycle bin on all drives without any UI or confirmation dialog, eliminating a potential source of file recovery for victims.

Custom icon registration

To visually brand encrypted files, the malware writes an embedded *.ico* file to *C:\ProgramData\<UID>.ico* and registers it as the default icon for files with the extension *.dlock*.

To associate the custom icon with encrypted files, the ransomware creates the `HKLM\SOFTWARE\Classes\.dlock\DefaultIcon` registry key and sets its (*Default*) value to the path of the dropped icon file.

Below is the malware's embedded `.ico` file.



Figure 1. DeadLock icon for encrypted files

Process and service termination

Before starting encryption, the malware terminates processes and disables services that could interfere with file access or provide defensive capabilities. This approach ensures that locked files become accessible for encryption while simultaneously disrupting the environment's ability to detect, respond to, or recover from the attack.

For services, the malware enumerates all active Win32 services and compares them against the stop list in the configuration. For each matching service, DeadLock sets its start type to *DISABLED* and sends a stop command to terminate that service. Notable targets include *windefend* (Windows Defender), *vss/swprv/wbengine* (Volume Shadow Copy and Backup services), *mssearch*, Hyper-V services (*vmcompute*, *vmms*), and Active Directory services (*adws*, *ntds*, *kdc*). Below is the full service stop list in the malware configuration:

```

udkusersvc*,devicesflowusersvc*,aarsvc*,wpnuserservice*,cdpusersvc*,cbdhsvc*,userdatasvc*,
pimindexmaintenancesvc*,unistoresvc*,vmicguestinterface,vmicheartbeat,vmickvpexchange,vmicrdv,
vmicshutdown,vmictimesync,vmicvss,licenseservice,mssearch,smphost,srmreports,srmvc,ddpvssvc,
msiserver,ktmrm,deviceinstall,onesyncsvc*,wsusservice,ngcsvc,clipsvc,alg,filesynchelper,lxpsvc,
dmwappushservice,peerdistsvc,eaphost,wlansvc,wwansvc,dot3svc,server,workstation,windefend,wsrm,wins,
silsvc,waasmedicsvc,ngcctnrsvc,rmsvc,adws,adfssrv,aelookupsvc,anydesk,apphostsvc,appinfo,apppgmt,
appxsvc,bfe,bits,brokerinfrastructure,browser,bthavctpsvc,camsvc,cdpsvc,certproptsvc,cscservice,
certsvc,smbwitness,clussvc,comsysapp,coremessagingregistrar,cryptsvc,mmcsc,dbxsvc,dbupdate,dbupdatem,
dcomlaunch,deviceassociationsservice,dfs,nfscInt,nfsservice,dfsr,dhcp,dhcpserver,diagtrack,
tabletinputservice,dispbrokerdesktopsvc,dns,dnscache,dnsproxy,dosvc,dps,dmsvc,dusmsvc,dssvc,efs,
eventlog,ntlmssp,eventsystem,fdphost,fdrespub,fontcache*,gpsvc,hidserv,hvhost,ias,ikeext,iphlpvc,
ismserv,kdc,kdssvc,keyiso,kpssvc,lanmanserver,lanmanworkstation,lfsvc,licenseanalyzer,lmhosts,lsm,lsm,
mpssvc,msdtc,wintarget,msiscsi,msmq,ncbservice,netlogon,netman,netprofm,netsetupsvc,wzcsvc,nlasvc,nla,
sharedaccess,nsi,ntds,ntfrs,uplugplay,plugplay,policyagent,power,profsvc,psexesvc,ramgmtsvc,rasauto,
rasman,remoteaccess,remoteregistry,rpceptrmapper,rpchttplbs,rcss,samss,scdeviceenum,schedule,seclogon,
securityhealthservice,sens,sessionenv,shellhwdetection,ssdpsrv,sstpsvc,staterepository,sgmrbroker,
stisvc,storsvc,swprv,sysmain,protectedstorage,systemeventsbroker,rdms,rcapd,tapisrv,termsservice,
tssdis,tscpubrpc,termsservlicensing,themes,typedatamodelsvc,timebroker,timebrokersvc,tokenbroker,
trkwks,trustedinstaller,tsgateway,ualsvc,uhssvc,umrpservice,upnphost,usermanager,uxsms,
onedrive updater service,lldtsvc,vaultsvc,vds,vmicvmsession,vmcompute,vmms,nvspwmi,vhdsvc,vss,w32time,
was,wcmsvc,wcncsvc,wscsvc,wdiservicehost,webclient,winhttpautoproxysvc,winmgmt,winrm,wlidsvc,wmiaprv,
wpdbusenum,wpnsservice,wsearch,dmservice,wuauaserv,wudfsvc

```

Figure 2. Service stop list

For processes, the malware enumerates all running processes and terminates any matching its stop list while skipping its own process ID. Targeted processes include security tools (*mssmpeng*, *securityhealthservice*, *smartscreen*), backup and cloud sync applications (*onedrive*, *dropbox*, *googledrivefs*, *owncloud*), remote access tools (*anydesk*, *putty*, *mstsc*, *rustdesk*), shell and system processes (*explorer*, *powershell*, *taskmgr*, *cmd*), and search/indexing services. Below is the full process stop list in the malware configuration:

```

aggregatorhost,anydesk,applicationframehost,certsrv,clussvc,cmd,conhost,csrss,ctfmon,dashost,dfsr,
dfssvc,dllhost,dns,dropbox,dwm,explorer,filecoauth,filesynchelper,fontdrvhost,googledrivefs,idle,
ismserv,kihost,logonui,lsass,lsass,lsm,microsoft.activeDirectory.webservices,msdtc,mssmpeng,mstsc,
netscan,nissrv,onedrive,onedrive.sync.service,openconsole,outlook,powershell,putty,rclone,rdpclip,
registry,rhs,rundll32,runtimebroker,rustdesk,searchapp,searchfilterhost,searchhost,searchindexer,
searchprotocolhost,searchui,securityhealthservice,services,shelllexperiencehost,shellhost,sihost,
smartscreen,smss,spoolsv,svchost,svhost,system,taskhostw,taskmgr,trustedinstaller,vds,vmcompute,vmms,
vmwp,vssvc>windowsterminal,wininit,winlogon,wlanext,wmiaprv,wudfhost,wwahost,owncloud

```

Figure 3. Process stop list

Event log clearing

To eliminate forensic evidence, the malware employs three complementary methods that collectively ensure every event log channel on the system is cleared of existing entries, disabled from recording future events, and has its access permissions locked down:

- **Direct clearing:** Clears the following log channels via the classic Event Log API: *Application*, *Security*, *Setup*, *Servicing*, *Eventlog*, *Forwarded Events*, *Windows PowerShell*, and *System*.

- **Registry-based disabling:** Enumerates every sub-key under `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels`. For each channel, sets `Enabled` to 0 (disabling all future logging) and overwrites `ChannelAccess` with a restrictive Security Descriptor Definition Language (SDDL) string that limits access to SYSTEM, built-in administrators, and local admin.
- **Modern API enumeration:** Uses `wevtapi.dll` to enumerate all registered event log channel paths (including custom application channels not in the hardcoded list) before clearing each one.

By combining API-based clearing, registry manipulation, and full channel enumeration, the malware covers multiple log sources, including third-party application logs and custom diagnostic channels, to minimize existing forensic evidence on the infected device.

Directory traversal

To maintain system stability and ensure the victim can access ransom instructions, the malware excludes specific directories, file extensions, and file names from encryption. This selective encryption model is a common ransomware design pattern where the system must remain operational enough for the victim to receive instructions and facilitate payment.

Extensions and file names from the configuration's file exclude list are skipped during encryption:

```
.386, .adv, .ani, .bat, .bin, .cab, .cmd, .com, .cpl, .cur, .deskthemepack,
.diagcab, .diagcfg, .diagpkg, .dll, .drv, .exe, .hlp, .hta, .icl, .icns,
.ico, .ics, .idx, .key, .ldf, .lnk, .lock, .mod, .mpa, .msc, .msi, .msp,
.msstyles, .msu, .nls, .nomedia, .ocx, .pdb, .prf, .ps1, .rom, .rtp, .scr,
.search-ms, .shs, .spl, .sys, .theme, .themepack, .wpx,
```

```
GDIPFONTCACHEV1.DAT, autorun.inf, boot.ini, bootfont.bin, bootmgr, bootnxt,
bootsect.bak, d3d9caps.dat, desktop.ini, iconcache.db, ntldr, ntuser.dat,
ntuser.dat.log, ntuser.ini, thumbs.db
```

Figure 4. List of skipped extensions and file names

For directory processing, the malware uses a two-tier directory exclusion system applied at different stages of the encryption pipeline. Tier 1 provides rough filtering that saves significant time by avoiding traversal overhead, while tier 2 provides granular path-specific exclusions within directories that are traversed. Both prevent encryption, but they operate at different stages of the traversal pipeline.

In its pre-traversal phase (tier 1), the malware checked at the drive batch level before threads are spawned for traversal. If a top-level directory matches against the configured directory

exclude list (`\users\*\appdata`, `program files (x86)\`, `program files\`, and `programdata\`), the entire tree is skipped without being walked.

In its during-traversal phase (tier 2), the malware checked the file name during recursive directory enumeration and applied to both subdirectories and files as they are encountered. In this tier, the directory and file names are checked against the configured sub-path exclude list below.

```
$windows.\~bt,$windows.\~ws,Users\All Users\application data,config.msi,program files (x86)\anydesk,
program files (x86)\common files,program files (x86)\google,program files (x86)\internet explorer,
program files (x86)\microsoft,program files (x86)\windows defender,program files (x86)\windows kits,
program files (x86)\windows mail,program files (x86)\windows nt,
program files (x86)\windows security,program files (x86)\windows sidebar,
program files (x86)\windowspowershell,program files\7-zip,program files\common files,
program files\google,program files\internet explorer,program files\modifiablewindowsapps,
program files\uninstall information,program files\windows defender,program files\windows mail,
program files\windows media player,program files\windows nt,program files\windows photo viewer,
program files\windows portable devices,program files\windows security,program files\windows sidebar,
program files\windowsapps,program files\windowspowershell,programdata\anydesk,
programdata\application data,programdata\packages,programdata\regid,programdata\ssh,
programdata\usoprivate,programdata\usoshared,system volume information,users\all users\microsoft,
windows\,Program Files\Fortinet\
```

Figure 5. Sub-path exclude list

Encryption

Resource-aware throttling

One of the more distinctive aspects of the DeadLock encryptor is its resource-aware throttling mechanism, designed to keep the infected system responsive during encryption. The malware spawns a dedicated monitoring/dispatch thread per drive batch that acts as a gatekeeper for file encryption dispatch. Before dispatching each new file to be encrypted, this thread polls system resource utilization and checks against hardcoded thresholds:

1. Polls memory and CPU idle before each file dispatch
2. Calculates memory usage percentage and CPU idle percentage
3. If memory usage exceeds 29% or CPU load exceeds 70% (idle < 30%), the dispatch thread pauses via a waitable timer and retries until resources return below thresholds
4. Once thresholds are within limits, atomically sets a dispatch flag on the work queue and signals waiting encrypting worker threads

With this mechanism, worker threads already encrypting files are not interrupted, and only the dispatch of new files is gated. This means partially encrypted files are expected to complete, and the throttling manifests as reduced parallelism rather than stop/start behavior. This

approach can prevent system hangs that would alert the user and reduce the likelihood of behavioral detection by maintaining normal-looking resource consumption patterns.

Thread architecture

For the encryption work itself, the malware spawns directory processing threads, with the thread count being 2 times the CPU core number. Each thread recursively traverses directories, dropping ransom notes and dispatching files for encryption. Individual file encryption threads are tasked with handling the actual cryptographic operations.

Cryptographic scheme

The DeadLock ransomware implements a hybrid cryptographic design that combines Curve25519 elliptic-curve cryptography with the XChaCha20 stream cipher for file encryption. Key encapsulation uses the Networking and Cryptography Library (NaCl) *crypto_box* construction, which pairs an asymmetric key exchange with authenticated encryption to securely wrap each file's symmetric key.

Layer	Algorithm	Purpose
File content encryption	XChaCha20	Symmetric stream cipher
Key encapsulation	Curve25519 Elliptic Curve Diffie-Hellman (ECDH) + XSalsa20-Poly1305	Asymmetric key wrapping (NaCl <i>crypto_box</i>)
Random generation	Windows CryptoAPI	All key material random generation

The configuration's operator public key

`03bf50bbf97c4e951e66ff12b689a37a3ce675b4921e254eae76da77573843e4a9` is 33 bytes.

The leading `03` byte is a SEC1 compressed point format prefix borrowed from

Bitcoin/secp256k1. The malware validates this prefix byte against a lookup table that accepts `00`, `02`, `03`, `04`, and `05`, mapping each to an expected key length.

After format validation, only the remaining 32 bytes are used in the actual Curve25519 ECDH scalar multiplication. This SEC1 prefix is non-standard for Curve25519, which natively uses bare 32-byte keys, and the malware author has likely adopted it for format versioning across their builder and decryptor tooling.

Per-file encryption process

For each target file, the malware performs the following sequence of operations:

1. Rename the target file from `<filename>` to `<filename>.<UID>.dlock`
2. Open the renamed file and retrieve file size/attributes
3. Clear the system attribute if `FILE_ATTRIBUTE_SYSTEM` is set
4. Determine the encryption strategy based on file size (see [File size-based encryption strategy](#))
5. Generate cryptographic material:
6. 32-byte random XChaCha20 key
7. 24-byte random XChaCha20 nonce (first 16 bytes for HChaCha20 subkey derivation, last 8 bytes as stream nonce)
8. 32-byte random ephemeral Curve25519 private key
9. 12-byte random file tag (only the first byte is functionally referenced by the encryptor to derive padding length; the remaining 11 bytes serve as a random file identifier written to the cleartext footer, likely used by the decryptor for file correlation/tracking)
10. 1–10 bytes random padding ($length = file_tag[0] \% 10 + 1$)
11. Perform Curve25519 ECDH: Multiply the ephemeral private key by the attacker's embedded public key to derive a shared secret
12. Build metadata plaintext: XChaCha20 key + 24-byte XChaCha20 nonce + random padding + *dDIK* magic + optional *FA* flag + chunk parameters
13. Encrypt metadata using *crypto_box* (XSalsa20-Poly1305) with the ECDH shared secret and a zero nonce
14. Encrypt file content using XChaCha20 with the generated key and 24-byte nonce
15. Append the encrypted footer/metadata to the end of the file

The use of a zero *crypto_box* nonce is worth noting. This is cryptographically safe because each file generates a unique ephemeral Curve25519 keypair, which produces a unique ECDH shared secret per file. With this, a constant zero nonce never repeats with the same key.

The entire design ensures that each file is encrypted with a distinct key derived from a per-file ephemeral key exchange, eliminating any possibility of key reuse across files. Overall, the cryptographic construction is sound and does not present a practical path to decryption without the attacker's private key.

File size-based encryption strategy

To balance encryption thoroughness with speed, the malware implements a tiered encryption policy based on file size. The encryption rule in the configuration

`1000,05052429880,025124288000,010524288000,F991114288000` encodes this policy. Each

comma-separated entry is parsed by splitting at position 3: the first 3 characters represent the encryption percentage (decimal), and the remaining characters represent the file size threshold (decimal bytes). The special prefix *F* replaces the percentage field with a chunked-full mode.

Rule	Encryption percent	File size threshold	Behavior
1000	100%	≥ 0 bytes	Default: encrypt entire file
05052429880	50%	$\geq \sim 50$ MB	Encrypt 50% of file in distributed chunks
025124288000	25%	$\geq \sim 118$ MB	Encrypt 25% in distributed chunks
010524288000	10%	$\geq \sim 500$ MB	Encrypt 10% in distributed chunks
F991114288000	Chunked	$\geq \sim 1$ GB	Special full-chunk mode with calculated intervals

Rules are evaluated in order, and the last matching rule wins. For example, when the malware processes a 2 GB file, all rules match, but the final *F99...* entry will determine the encryption behavior.

For partial encryption, the malware calculates:

- Total bytes to encrypt = $\text{ceil}(\text{file_size} \times (\text{percentage} / 100))$
- Encrypted block count = $\text{ceil}(\text{total_bytes_to_encrypt} / 512)$
- Skip interval = $\text{floor}((\text{file_size} - \text{total_bytes_to_encrypt}) / \text{encrypted_block_count})$

This creates an intermittent encryption pattern where 512-byte blocks are encrypted at regular intervals throughout the file. The result is a file that is rendered unusable while requiring only a fraction of the time needed for full encryption. This is a crucial optimization for the ransomware when targeting large files such as databases, virtual machine images, and backups.

File footer

After encryption, the malware appends a structured metadata blob to the end of each file. This footer contains all the information the decryptor needs to reverse the encryption, along with markers for format validation:

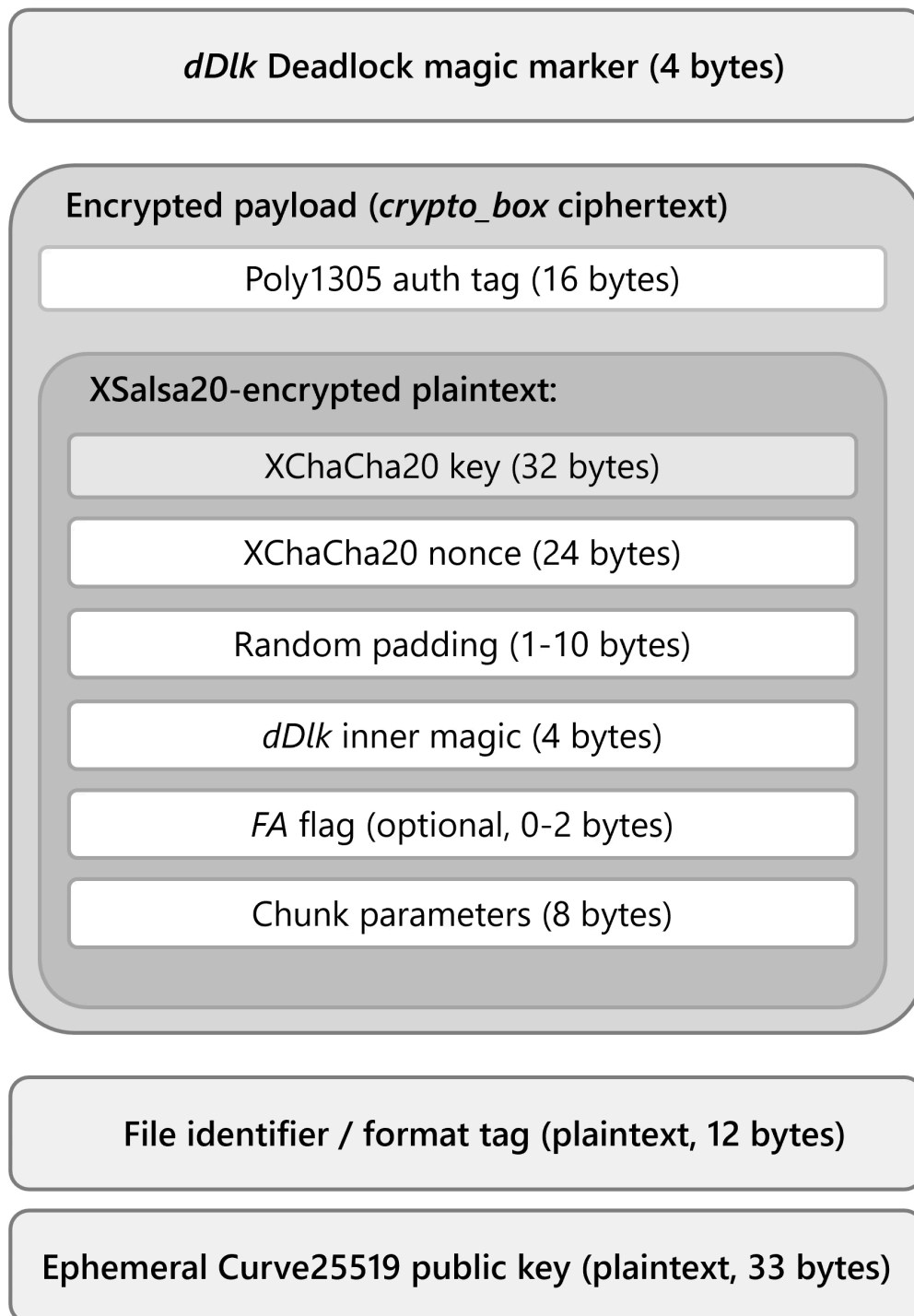


Figure 6. DeadLock file footer

The footer serves several important functions:

Key and nonce reconstruction: The cleartext ephemeral Curve25519 public key (33 bytes) at the end of the footer allows the decryptor to recompute the ECDH shared secret and open the *crypto_box* to recover the XChaCha20 key and nonce used for file content encryption.

Inner *dDIK* magic (decryption validation): After the decryptor opens the *crypto_box*, it checks for the *dDIK* marker at the expected offset ($32 + 24 + padding_length$ bytes into the plaintext) to confirm the correct private key was used and that decryption succeeded. While the Poly1305 Message Authentication Code (MAC) already provides cryptographic integrity verification, this marker offers a fast format-level sanity check.

***FA* flag (decryption mode indicator):** This flag is used by the decryptor to determine which read strategy to use when reversing the encryption. It is present when the file was encrypted using sequential/contiguous block encryption, and absent when intermittent/skip encryption was used. Specifically, *FA* is appended in two cases:

1. **F-prefix rule matched:** When the file size triggers the *F991114288000* config entry (the special chunked-full mode), the *FA* flag is always set.
2. **Percentage rule with zero skip interval:** When a percentage-based rule matches but the calculated skip interval between encrypted chunks works out to zero (meaning the percentage effectively covers the entire file), *FA* is also set.

Without this flag, the 8-byte chunk parameters in the footer would be ambiguous as they could represent either a block count or a skip interval. The *FA* flag resolves this ambiguity and enables the decryptor to correctly reconstruct the original file.

File identifier/format tag: The 12-byte random value in the cleartext footer serves as a file identifier (with the first byte used to derive the padding length inside the encrypted payload).

Post-encryption

Wallpaper

As an immediate visual indicator of compromise, the malware generates a custom BMP wallpaper file at runtime using the victim's screen resolution. Below is an example of the generated BMP wallpaper:

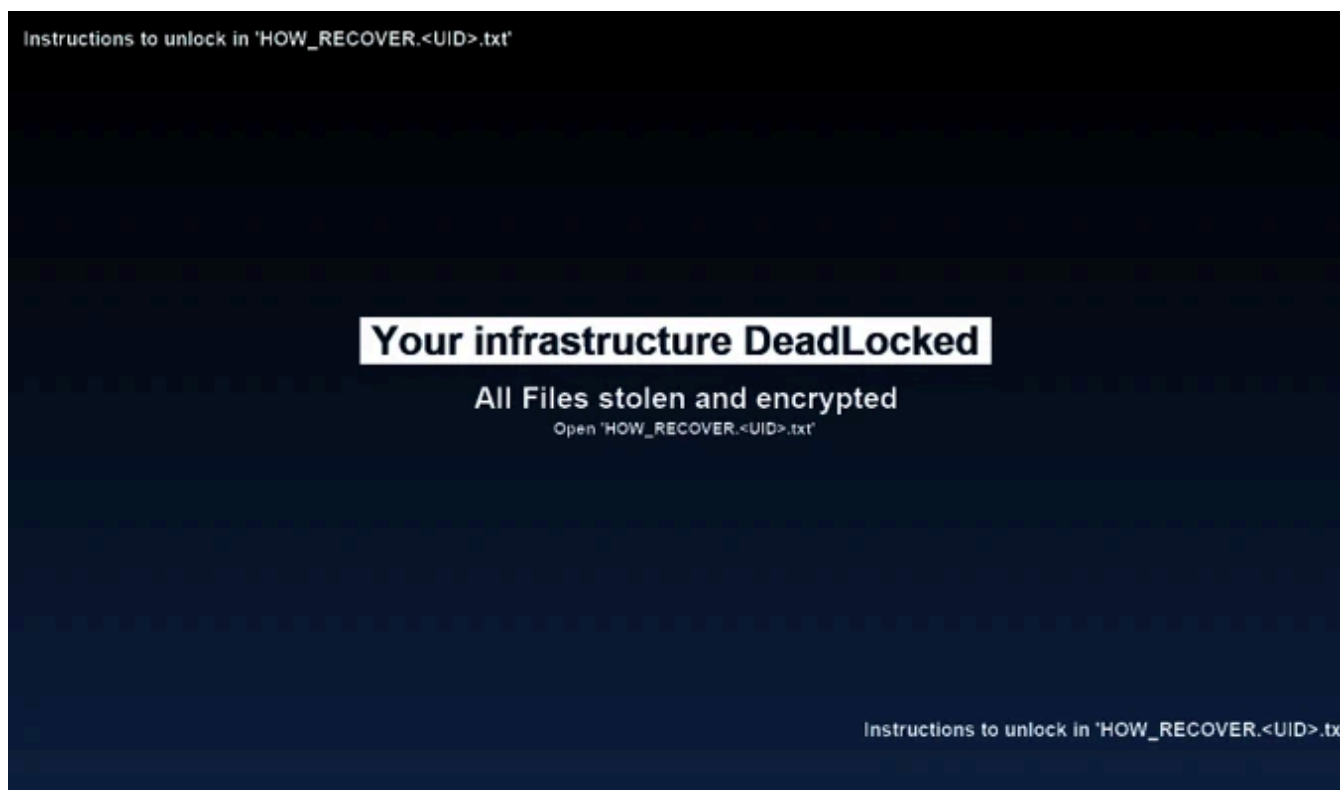


Figure 7. DeadLock wallpaper

The wallpaper is written to *C:\ProgramData\<UID>.bmp* (on Vista and later) or *C:\Documents and Settings\All Users\Application Data\<UID>.bmp* (on XP), set as the desktop background, and persisted in the registry at

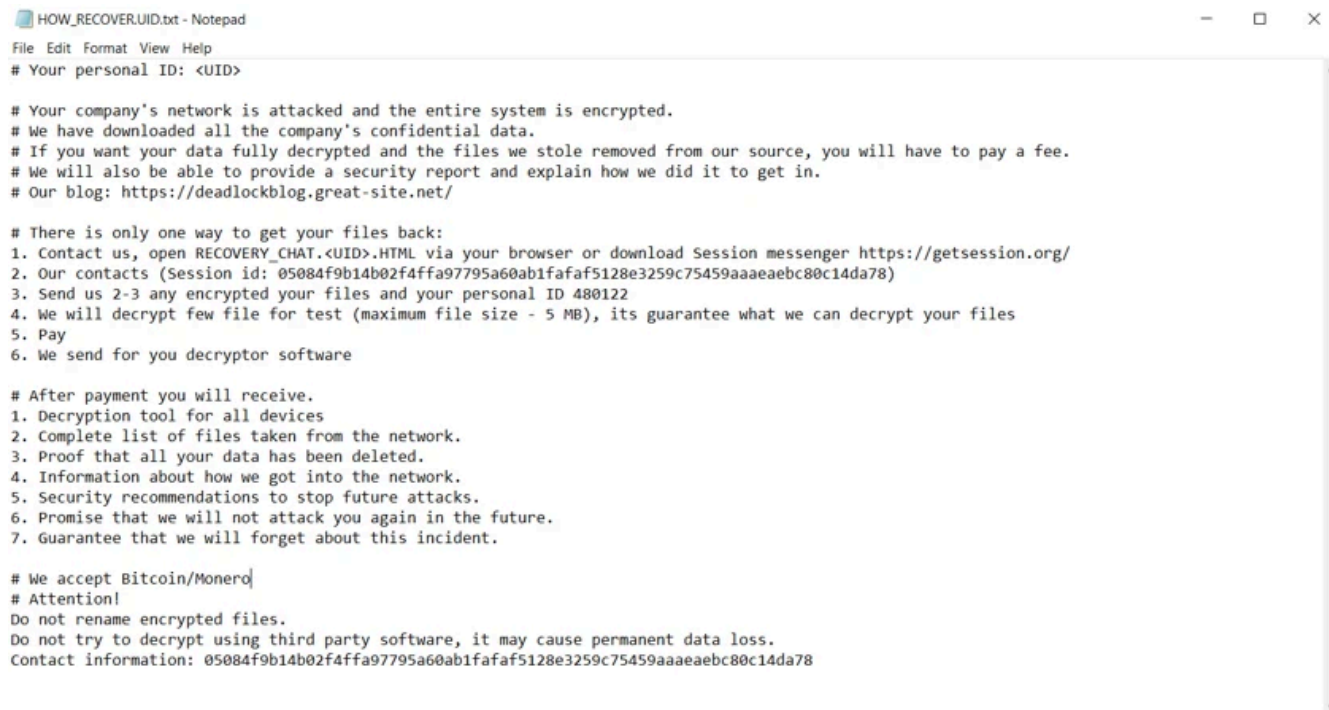
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Wallpaper.

Ransom notes deployment

After encrypting files, the malware deploys two types of ransom notes, each with distinct deployment logic and purpose:

Text note (*HOW_RECOVER.<UID>.txt*): The text note is dropped into every encrypted directory, but with a notable timing behavior: it is only deployed during the second pass of the directory processing loop. The malware iterates over drive batches multiple times, and the text note drop is gated by an iteration counter. On the first pass, the text note is suppressed, likely to prioritize encryption speed before littering the file system with ransom note files. For defenders and analysts, this has a practical implication: if testing with a minimal drive configuration that only triggers a single iteration, the text note will never appear.

Below is the text note content from the malware's configuration.

A screenshot of a Notepad window titled 'HOW_RECOVER.UID.txt - Notepad'. The window contains a ransom note in plain text. The text is as follows:

```
File Edit Format View Help
# Your personal ID: <UID>

# Your company's network is attacked and the entire system is encrypted.
# We have downloaded all the company's confidential data.
# If you want your data fully decrypted and the files we stole removed from our source, you will have to pay a fee.
# We will also be able to provide a security report and explain how we did it to get in.
# Our blog: https://deadlockblog.great-site.net/

# There is only one way to get your files back:
1. Contact us, open RECOVERY_CHAT.<UID>.HTML via your browser or download Session messenger https://getsession.org/
2. Our contacts (Session id: 05084f9b14b02f4ffa97795a60ab1fafaf5128e3259c75459aaaaeabc80c14da78)
3. Send us 2-3 any encrypted your files and your personal ID 480122
4. We will decrypt few file for test (maximum file size - 5 MB), its guarantee what we can decrypt your files
5. Pay
6. We send for you decryptor software

# After payment you will receive.
1. Decryption tool for all devices
2. Complete list of files taken from the network.
3. Proof that all your data has been deleted.
4. Information about how we got into the network.
5. Security recommendations to stop future attacks.
6. Promise that we will not attack you again in the future.
7. Guarantee that we will forget about this incident.

# We accept Bitcoin/Monero
# Attention!
Do not rename encrypted files.
Do not try to decrypt using third party software, it may cause permanent data loss.
Contact information: 05084f9b14b02f4ffa97795a60ab1fafaf5128e3259c75459aaaaeabc80c14da78
```

Figure 8. DeadLock text ransom note

HTML note (*RECOVERY_CHAT.<UID>.html*): This file is dropped to all drive root directories and all *Desktop* folders. Unlike the text note, the HTML note is a full interactive web application with a self-contained single-page application that implements end-to-end encrypted chat, a paginated data leak blog, and a file browser, all without requiring a traditional backend server. The technical architecture of this recovery chat system is detailed in .

Recovery chat: Technical architecture

The most distinctive feature of the DeadLock ransomware is its recovery chat system. The *RECOVERY_CHAT.<UID>.html* file is a self-contained HTML application that implements a full end-to-end encrypted chat system, a paginated data leak blog, and a file browser, all without requiring a traditional backend server.

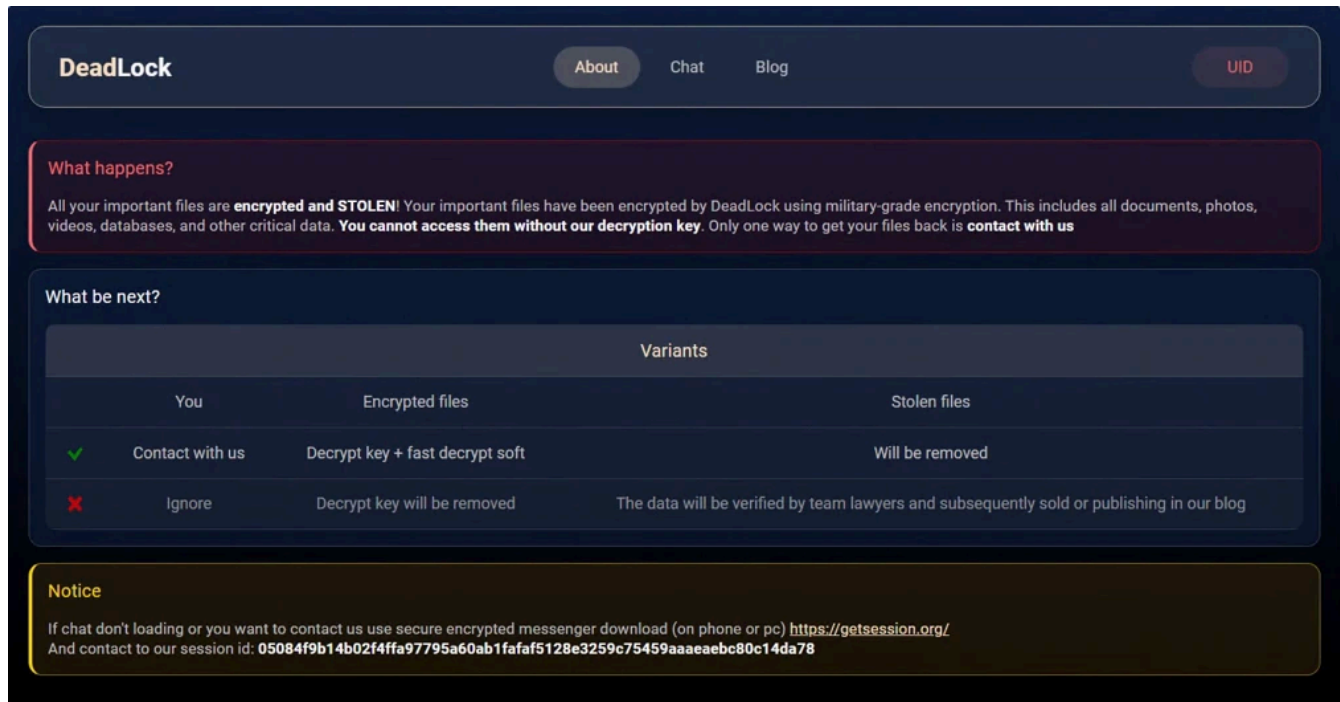


Figure 9. HTML application “About” page UI

The architecture is designed with three decentralized components.

Polygon blockchain as configuration store

Rather than relying on traditional domain-based infrastructure that can be seized or taken offline, the DeadLock operators store configuration data on the Polygon blockchain. Two smart contracts serve as censorship-resistant infrastructure:

Contract	Address	Function selector	Purpose
Chat proxy	0x8EF7c3e531d871D3B9D559722DE77EB1dEc19dAe	0x933a9ce8	Stores the proxy server URL
Blog	0x757984507c82c8dA1d3969c535dB5706eEE6426C	0xd4070542	Stores actor's blog posts

The HTML page issues *eth_call* requests to public Polygon Remote Procedure Call (RPC)

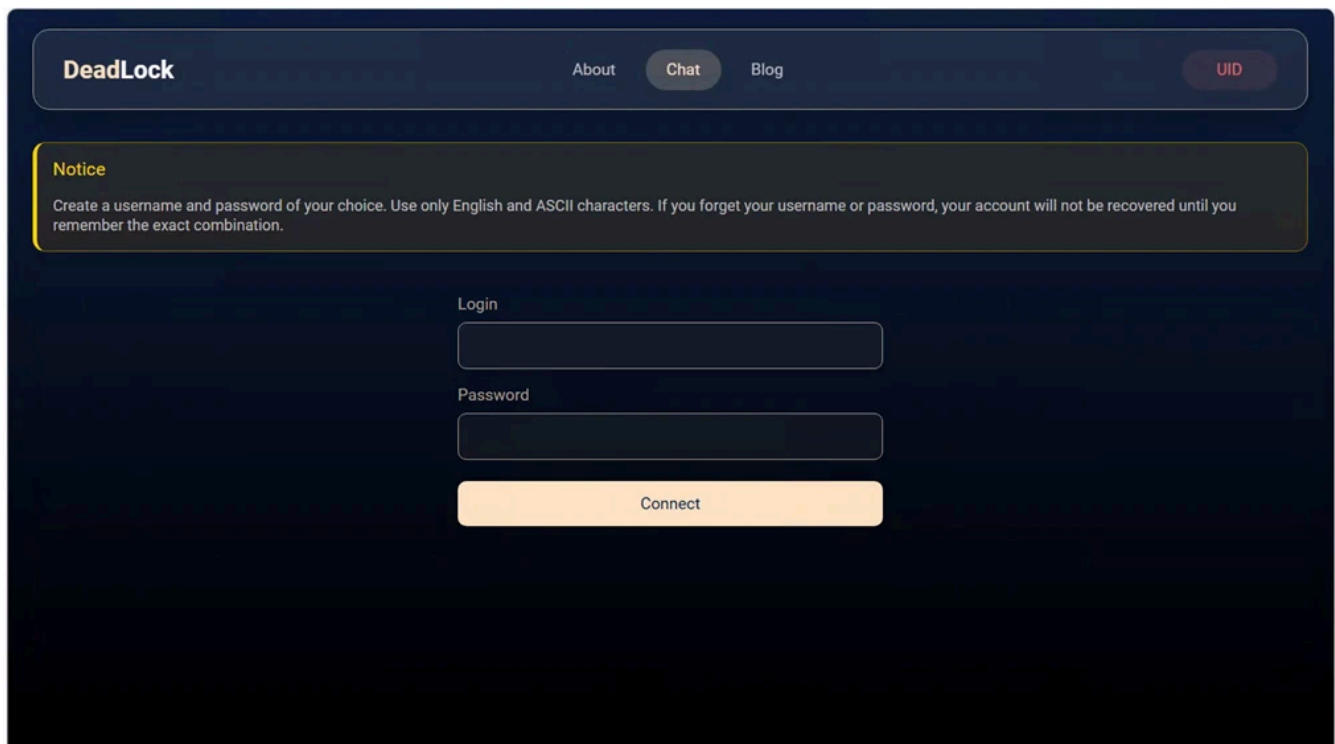
endpoints (no wallet required with read-only calls) to obtain the proxy server address. The blog contract takes offset and limit parameters (for pagination) and returns structured data including post titles, bodies, timestamps, image URLs, and file attachment links.

On-chain storage provides several strategic advantages for the threat actor: the proxy URL can be updated by modifying the smart contract without changing any victim-facing infrastructure, and no domain registration or DNS infrastructure is required. This represents a notable evolution in ransomware infrastructure design.

The HTML recovery chat cycles through six public RPC endpoints for redundancy: *polygon-bor-rpc.publicnode[.]com*, *polygon.drpc[.]org*, *polygon-pokt.nodies[.]app*, *polygon-rpc[.]com*, *1rpc[.]jio/matic*, and *polygon.meowrpc[.]com*.

Session network for end-to-end encrypted chat

For victim-operator communication, chat messages are routed through the Session decentralized messenger network, which is an onion-routed, swarm-based messaging protocol that provides anonymity for both parties. The proxy server (whose URL is retrieved from the blockchain) acts as a relay between the victim's browser and Session swarm nodes.



The image shows a web application interface for 'DeadLock'. The top navigation bar includes the 'DeadLock' logo, links for 'About', 'Chat' (which is the active tab), and 'Blog', and a 'UID' button. A yellow-bordered notice box contains instructions for creating a username and password. Below this is a login section with labels for 'Login' and 'Password', corresponding input fields, and a 'Connect' button.

Figure 10. HTML application "Chat" page UI

Key generation: DeadLock's design choice is that the victim's Session identity is derived deterministically from their sign-in credentials. When the victim enters their credentials on the HTML page, the following derivation occurs:

```

seed_input = login + "_" + UID + "_" + password
hash = SHA-512(seed_input)
seed = hash[0:16] (first 16 bytes, zero-padded to 32)
ed25519_keypair = crypto_sign_seed_keypair(seed)
curve25519_pk = ed2curve_convertPublicKey(ed25519_keypair.signPk)
curve25519_sk = ed2curve_convertSecretKey(ed25519_keypair.signSk[0:32])
session_address = "05" + hex(curve25519_pk)

```

Figure 11. Derivation after victim entered credentials

This deterministic derivation means the same credentials always produce the same keypair, and no account registration is needed as the victim's Session identity exists only when they enter the correct credentials. If the victim forgets their credentials, the identity is unrecoverable (as stated by the actor in the chat UI). The 05 prefix is Session's standard network identifier for user accounts.

Sending a message: The following sequence occurs when a message is sent:

1. Encode the body and timestamp as protobuf
2. Create an actor message and a self-sync copy
3. Pad plaintext to 160-byte boundary
4. Sign the padded content and key context with Ed25519
5. Append the sender public key and signature
6. Seal each payload with the recipient's Curve25519 key
7. Wrap in Session's onion request protobuf format (verb: *PUT*, path: */api/v1/message*)
8. Ask the proxy to submit both copies to their respective swarms

Receiving a message: The following sequence occurs when a message is received:

1. Sign "*retrieve*" + *timestamp* with the victim's Ed25519 key
2. Select a node associated with the victim's own swarm
3. Ask *the proxy to poll for messages addressed to that identity*
4. Open each sealed box with the victim's Curve25519 keypair
5. Remove the appended public key and signature
6. Strip padding, decode protobuf, and extract the message body

Data leak blog and Wasabi file hosting

The recovery chat page also provides access to a data leak blog whose content is stored on the Polygon blockchain.

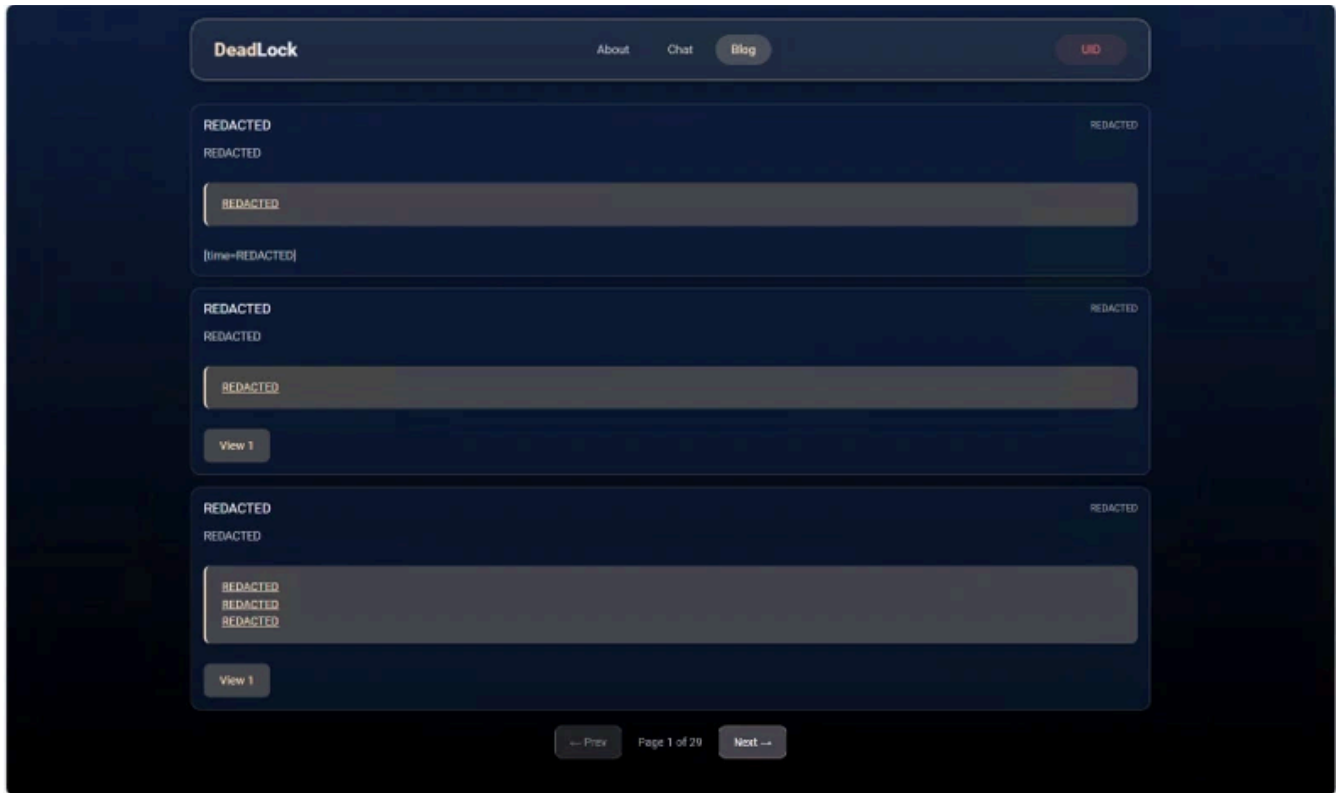


Figure 12. Redacted HTML app “Blog” page UI

Blog posts retrieved from the smart contract support BBCode formatting, image galleries, and file attachments using either direct URLs or Wasabi protocol links that open an in-browser file explorer. The HTML application contains a full Amazon Web Services (AWS) S3-compatible file browser that parses the Wasabi credentials from the URI, generates AWS4-HMAC-SHA256 signed requests, lists bucket contents with folder navigation, and generates pre-signed download URLs for individual files. This allows the attacker to host stolen data on Wasabi and provide victims or the public with browsable access to the leaked files without running a web server.

Infrastructure resilience summary

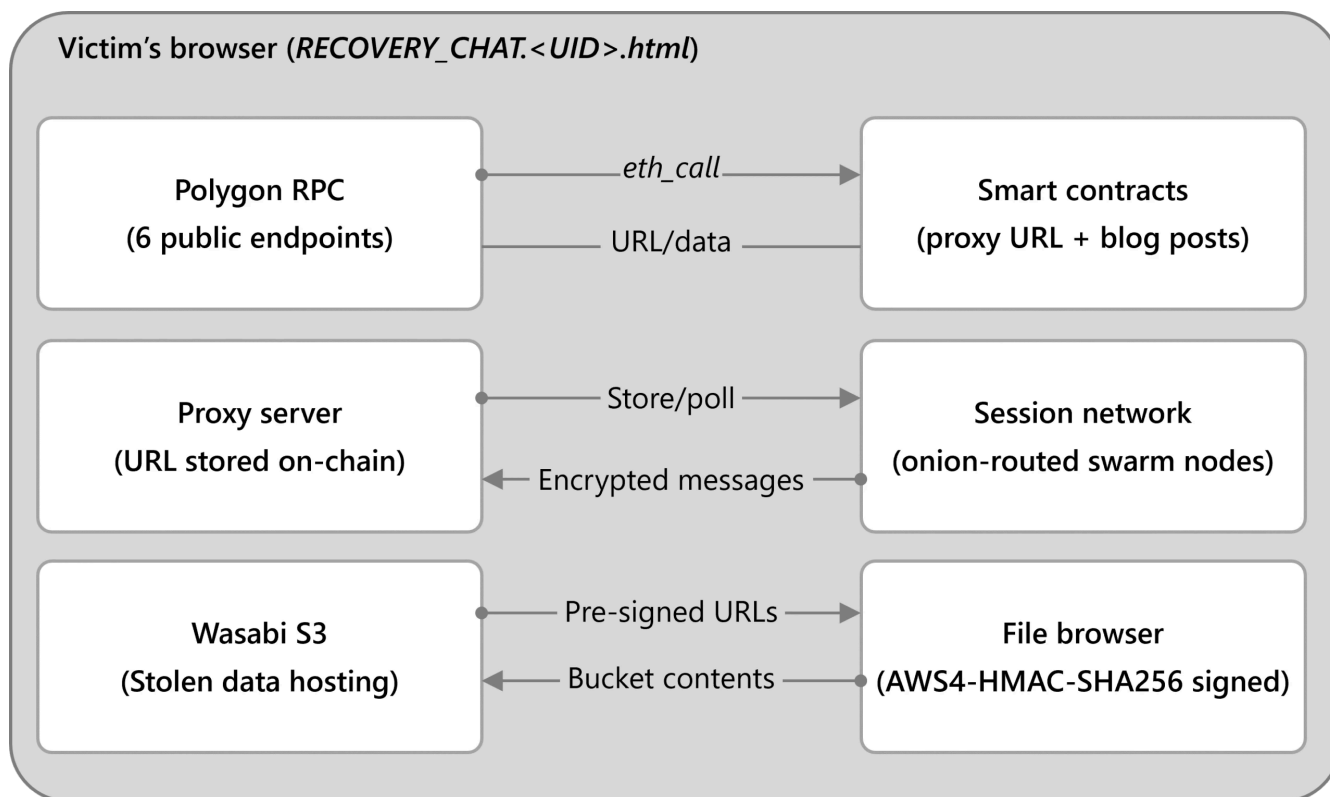


Figure 13. HTML recovery chat infrastructure summary

The architecture is significantly more resilient to takedown and censorship efforts, but it is not independent of off-chain infrastructure:

- **Proxy replacement:** The actor can update the on-chain proxy URL without changing the HTML
- **On-chain persistence:** Contract-stored blog data is resistant to conventional hosting takedowns
- **RPC dependency:** The page still requires access to at least one public Polygon RPC endpoint
- **Proxy dependency:** Chat access depends on the current custom proxy remaining reachable
- **Storage dependency:** Images and leaked files can be removed from CDN or Wasabi hosting
- **Session resilience:** Distributed swarm storage reduces reliance on a single messaging server

This infrastructure model represents a meaningful evolution from traditional ransomware communication channels and poses new challenges for takedown efforts.

Self-deletion

As a final cleanup step after encryption completes, the malware creates a batch to delete its own binary from disk. The cleanup batch loops until it successfully deletes the malware binary, then removes itself:

```
@echo off
chcp 65001 > NUL
:1
if exist "<malware path>" (del "<malware path>") else (goto e)
goto 1
:e
del %0
```

Figure 14. Self-deleting batch loop

Defending against DeadLock ransomware

Microsoft recommends the following mitigations to reduce the impact of this threat.

- Read the [human-operated ransomware threat overview](#) for advice on developing a holistic security posture to prevent ransomware, including credential hygiene and hardening recommendations.
- Turn on [cloud-delivered protection](#) in Microsoft Defender Antivirus or the equivalent for your antivirus product to cover rapidly evolving attacker tools and techniques. Cloud-based machine learning protections block a huge majority of new and unknown variants.
- Run [endpoint detection and response \(EDR\) in block mode](#) so that Microsoft Defender for Endpoint can block malicious artifacts, even when your non-Microsoft antivirus does not detect the threat or when Microsoft Defender Antivirus is running in passive mode. EDR in block mode works behind the scenes to remediate malicious artifacts that are detected post-breach.
- Turn on [tamper protection](#) features to prevent attackers from stopping security services. In addition to tamper protection, you can also [enable and configure Microsoft Defender Antivirus always-on protection in Group Policy](#).
- Configure [investigation and remediation](#) in full automated mode to let Microsoft Defender for Endpoint take immediate action on alerts to resolve breaches, significantly reducing alert volume.
- Configure [automatic attack disruption](#) in Microsoft Defender XDR. Automatic attack disruption is designed to contain attacks in progress, limit the impact on an organization's assets, and provide more time for security teams to remediate the attack fully.

- To help preserve existing systems in the event of a ransomware attack, [configure a Controlled Folder Access \(CFA\) policy](#) to be as strict as possible. CFA protects valuable data from threats like ransomware by preventing write access to common system folders; more folders can also be added. Establishing this policy ahead of a ransomware event can enable organizations to respond quickly to ransomware signals, deploying the CFA policy to limit the destructive impact of an active attack. In certain instances, a CFA policy can also be leveraged proactively on specific sensitive assets that will not be negatively impacted by restrictive protections. Use [audit mode](#) to evaluate the impact to your organization in these cases.
- Microsoft Defender XDR customers can turn on [attack surface reduction rules](#) to prevent several of the infection vectors of this threat. These rules, which can be configured by any user, offer significant hardening against targeted attacks. In observed attacks, Microsoft customers who had the following rules turned on could mitigate the attack in the initial stages and prevent hands-on-keyboard activity:

[Block executable files from running unless they meet a prevalence, age, or trusted list criterion](#)

[Block process creations originating from PSExec and WMI commands](#) (Some organizations might experience compatibility issues with this rule on certain server systems but should deploy it to other systems to prevent lateral movement originating from PsExec and WMI)

[Use advanced protection against ransomware](#)

You can assess how an attack surface reduction rule might impact your network by opening the [security recommendation](#) for that rule in Vulnerability management. In the Recommendation details pane, check the user impact to determine what percentage of your devices can accept a new policy enabling the rule in blocking mode without adverse impact to user productivity.

Microsoft Defender detections

[Microsoft Defender](#) customers can refer to the list of applicable detections below. Microsoft Defender coordinates detection, prevention, investigation, and response across endpoints, identities, email, apps to provide integrated protection against attacks like the threat discussed in this blog.

Microsoft Defender Antivirus

Microsoft Defender Antivirus detects threat components as the following malware:

[Ransom:Win32/Deadlock.*](#)

Microsoft Defender for Endpoint

The following alerts might indicate threat activity associated with this threat. These alerts, however, can be triggered by unrelated threat activity and are not monitored in the status cards provided with this report.

- Ransomware-linked threat actor detected
- Ransomware behavior detected in the file system
- Possible ransomware activity
- File backups were deleted
- Potential human-operated malicious activity
- Possible data exfiltration
- Suspicious wallpaper change

The following alerts might indicate threat activity associated with DeadLock ransomware if Defender for Endpoint is set to block mode.

- 'DeadLock' ransomware was detected
- 'DeadLock' ransomware was prevented

Microsoft Defender for Cloud Apps

The following alert might indicate threat activity associated with this threat. This alert, however, can be triggered by unrelated threat activity and are not monitored in the status cards provided with this report.

Ransomware activity

Microsoft Security Copilot

[Microsoft Security Copilot](#) is [embedded in Microsoft Defender](#) and provides security teams with AI-powered capabilities to summarize incidents, analyze files and scripts, summarize identities, use guided responses, and generate device summaries, hunting queries, and incident reports.

Customers can also [deploy AI agents](#), including the following [Microsoft Security Copilot agents](#), to perform security tasks efficiently:

- [Threat Intelligence Briefing agent](#)
- [Phishing Triage agent](#)
- [Threat Hunting agent](#)
- [Dynamic Threat Detection agent](#)

Security Copilot is also available as a [standalone experience](#) where customers can perform specific security-related tasks, such as incident investigation, user analysis, and vulnerability

impact assessment. In addition, Security Copilot offers [developer scenarios](#) that allow customers to build, test, publish, and integrate AI agents and plugins to meet unique security needs.

Threat intelligence reports

Microsoft Defender XDR customers can use the following [threat analytics](#) reports in the Defender portal (requires license for at least one Defender XDR product) to get the most up-to-date information about the threat actor, malicious activity, and techniques discussed in this blog. These reports provide the intelligence, protection information, and recommended actions to prevent, mitigate, or respond to associated threats found in customer environments.

- [Tool profile: Deadlock ransomware](#)
- [Tool profile: Lynx ransomware](#)
- [Tool profile: INC ransomware](#)
- [Threat overview profile: Human-operated ransomware](#)

Microsoft Security Copilot customers can also use the [Microsoft Security Copilot integration](#) in Microsoft Defender Threat Intelligence, either in the Security Copilot standalone portal or in the [embedded experience](#) in the Microsoft Defender portal to get more information about this threat actor.

Indicators of compromise

Indicator	Type	Description
a1fdf65020ce4a0f0940c793c6425baf8a0b994ec48b9baaf72788661a9d29f4	SHA-256	DeadLock ransomware encryptor
<i>deadlock.liveblog365[.]com</i>	URL	Leak site domain
<i>dlock.liveblog365[.]com</i>	URL	Leak site domain
<i>deadblogdbdu5wprek7wa2o4ce7rnt6u6ntqeud3hzjjcveosgpsqqd[.]onion</i>	URL	Leak site domain
<i>deadlockblog.great-site[.]net</i>	URL	Leak site domain
<i>deadlockblog.medianewsonline[.]com</i>	URL	Leak site domain

Learn more

For the latest security research from the Microsoft Threat Intelligence community, check out the .

To hear stories and insights from the Microsoft Threat Intelligence community about the ever-evolving threat landscape, listen to the .