19 Shades of LockBit5.0, Inside the Latest Cross-Platform Ransomware's Newest Leaked Samples: Part 1

By Mark Tsipershtein, Evgeny Ananin, Nikita Kazymirskyi

Published: 2026-01-30 · Archived: 2026-04-05 21:33:56 UTC

January 30, 2026 8 Minute Read by Mark Tsipershtein, Evgeny Ananin, Nikita Kazymirskyi

This three-part blog series presents an analysis of 19 samples of a cross-platform LockBit 5.0 ransomware payload affecting Windows, Linux (LINUX Locker v1.06/v1.08), and ESXi (LINUX ESXi Locker v1.07) environments, highlighting how the ransomware operates, encrypts data, and interacts with targeted systems.

Please check out Part 2 and Part 3 in the series.

By reverse engineering multiple samples, we identified shared components across platforms as well as operating system–specific behaviors that allow the malware to function efficiently in different environments.

At the core of the threat is the use of a fast encryption method (ChaCha20) to encrypt files and data, replacing the AES-based encryption used in previous versions. The malware is designed for performance and scalability, enabling it to process large volumes of data quickly while avoiding reliance on standard cryptographic libraries, which can make detection more difficult.

This report examines how the malware accesses files, manages system processes, and executes tasks in parallel, giving defenders insight into its operational workflow. While the underlying encryption approach is consistent, the techniques used to interact with Windows and Linux systems differ, reflecting deliberate adaptation to each platform.

At the time of writing, the sample had a detection score of 1/65 on VirusTotal.

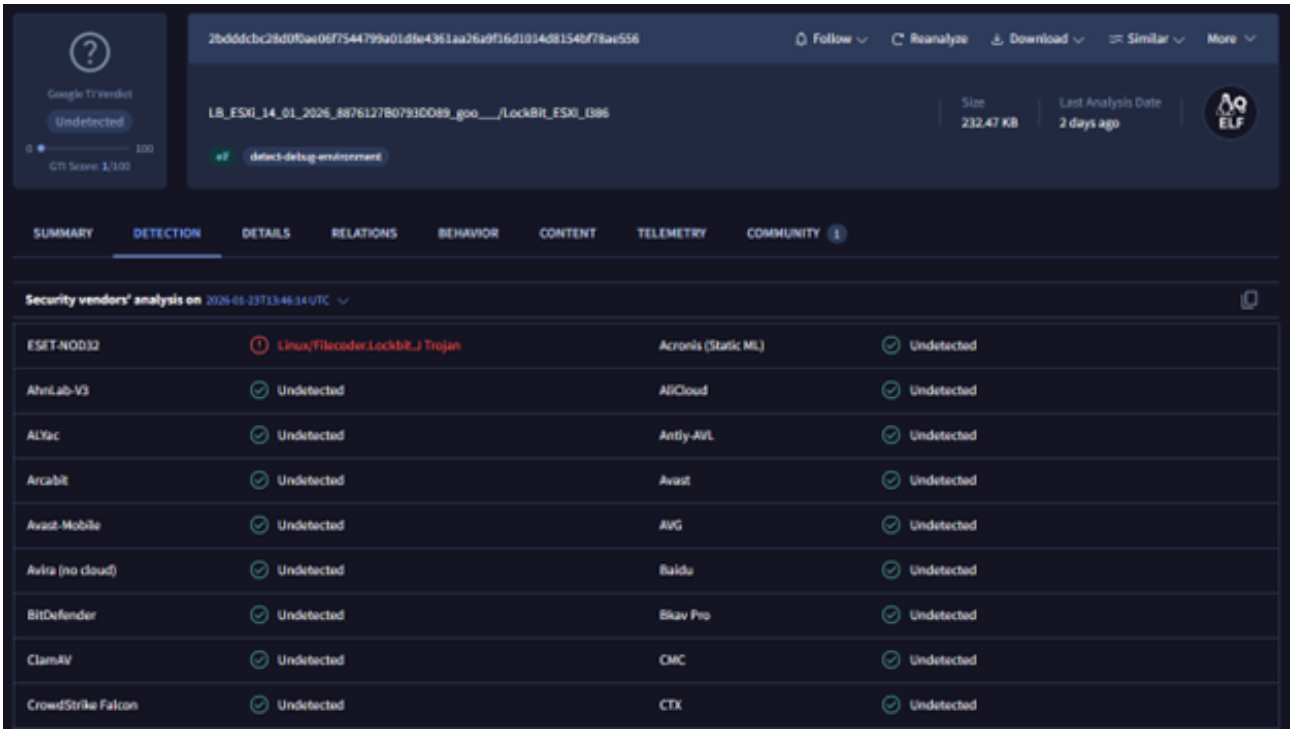

Figure 1. VirusTotal detection results for the LockBit 5.0 sample we analyzed as of writing

Threat Landscape

LockBit is one of the clearest examples of ransomware evolving into an industrial business model. It operates as ransomware-as-a-service (RaaS): a core team maintains the malware, negotiation/payment infrastructure, and leak site, while affiliates carry out intrusions and deployments for a revenue share. That division of labor is what lets the operation scale, and why LockBit has repeatedly been described in public reporting as a high-volume, high-impact ecosystem.

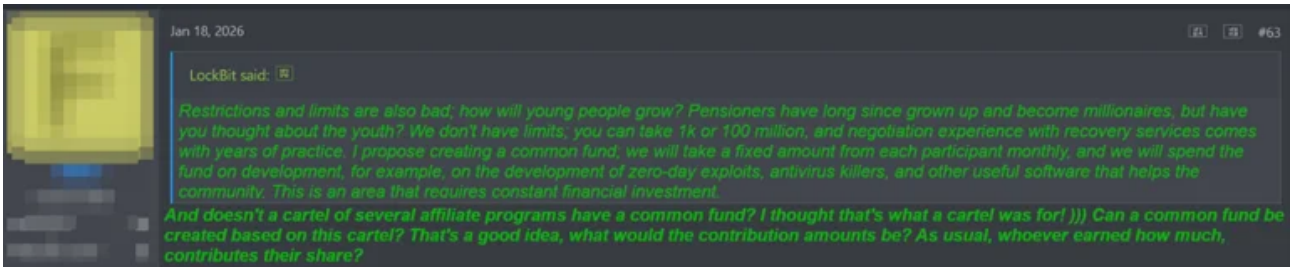

Figure 2. Discussion of development plans and a shared fund on a dark web forum

More recently, LockBit's posture points to a potentially more serious market shift: consolidation. Reporting in October 2025 described LockBit aligning with other ransomware brands (including DragonForce and Qilin) in a cartel-style approach that emphasizes shared resources and invites additional crews to join. If that umbrella model gains traction, defenders may face not just a single prolific RaaS brand, but an aggregation layer that pools affiliates, infrastructure, and playbooks, raising both the baseline capability and the potential scale of ransomware operations.

In this report, we analyze the latest observed samples of the LockBit 5.0 ransomware family, comparing builds across platforms and architectures to understand what stays consistent, what changes, and what it reveals about the

group’s current development direction and operational tradecraft.

Part1: ESXi Variant Technical Analysis

Overview

VMware ESXi is a widely adopted bare-metal hypervisor used to host and manage large numbers of virtual machines in enterprise environments. By consolidating critical workloads, such as databases, application servers, and domain controllers, into a single physical host, ESXi enables efficiency and scalability, but it also creates a high-value target. A successful compromise of an ESXi host can result in the simultaneous disruption of dozens or even hundreds of systems.

In recent years, ransomware operators have increasingly shifted their focus from individual endpoints to hypervisors like ESXi. Rather than encrypting files within a single operating system, ESXi-targeting ransomware attacks the underlying virtual disk files and configuration data that power entire virtual infrastructures. This approach maximizes operational impact while reducing the attacker’s effort.

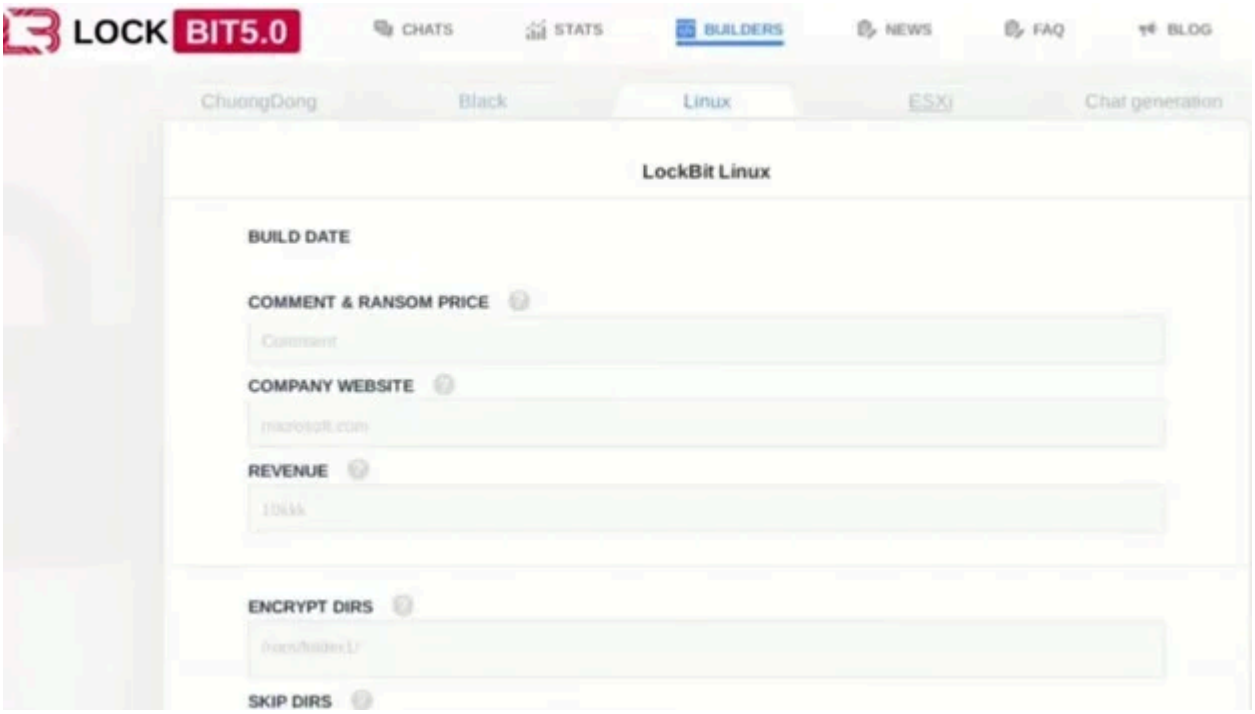

Figure 3. LockBit Builder Linux version view

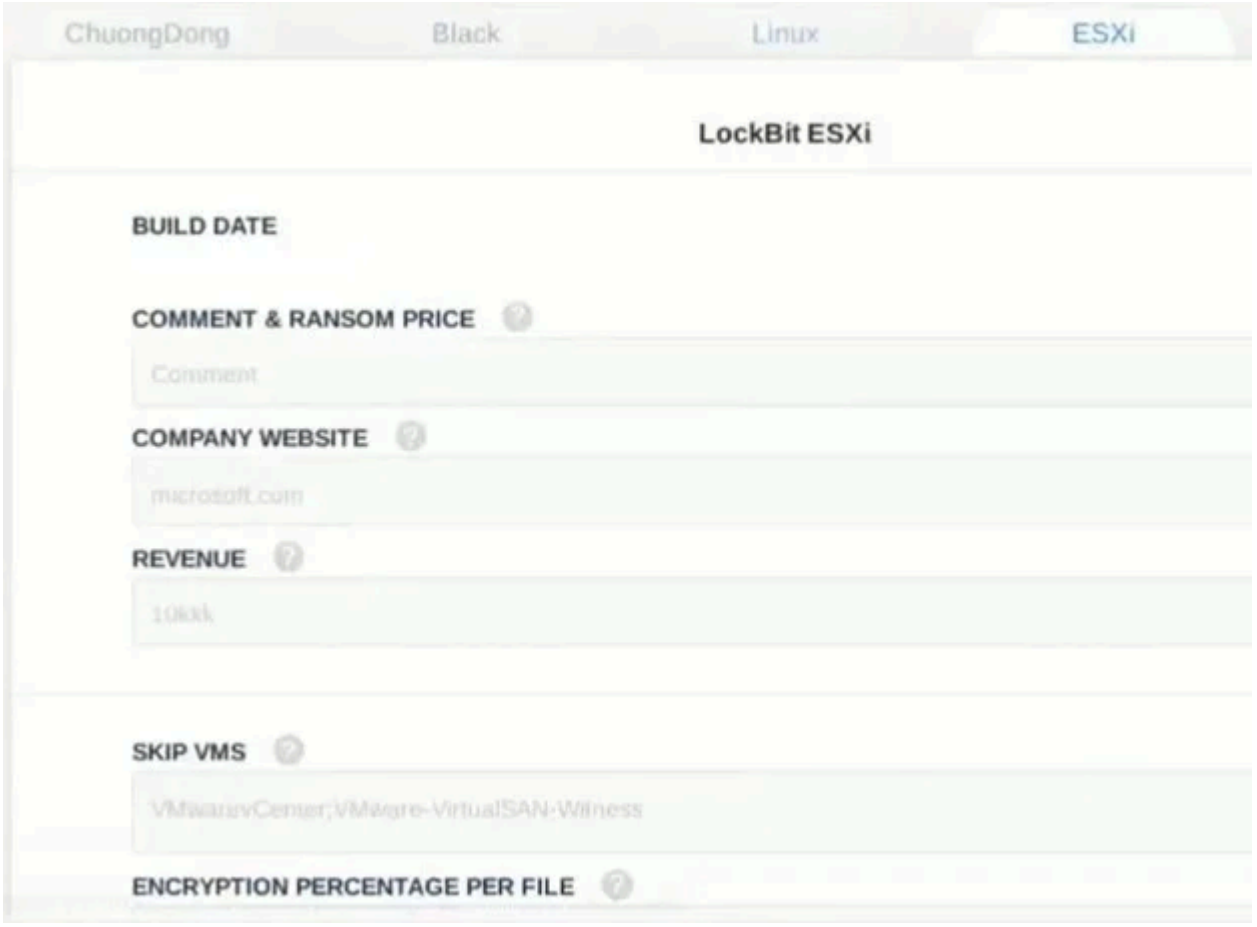

Figure 4. LockBit Builder ESXi version view

LockBit 5.0 ESXi Version

Figure 4 illustrates the execution flow of the LockBit 5.0 ESXi ransomware based on static analysis. The malware first validates that it is running in a VMware ESXi environment, then enumerates and forcibly shuts down active virtual machines to release file locks. It proceeds with a multi-threaded encryption routine using a fast pass followed by a full pass on targeted VM files. Instead of dropping a traditional ransom note, victim notification is handled via console output or log files, reflecting the headless nature of ESXi systems.

Figure 5. LockBit ESXi execution flow

This is a 64-bit Linux ELF executable compiled for the x86-64 architecture. The binary is dynamically linked. The file lacks ELF section headers, a common anti-analysis technique used by malware.

```
file lockbit_AMD64
```

```
lockbit_AMD64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2, no section header
```

This ldd command output shows the shared libraries required by the LockBit ESXi binary at runtime. libpthread.so.0 indicates the use of threads, supporting multi-threaded execution consistent with parallel encryption activity.

```
ldd lockbit_AMD64
```

```
linux-vdso.so.1 (0x00007f341514f000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f341512b000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f3414f35000)
/lib64/ld-linux-x86-64.so.2 (0x00007f3415151000)
```

A Binwalk scan of the LockBit ESXi sample highlights embedded file paths within the binary.

```
binwalk lockbit_AMD64
```

```
 DECIMAL HEXADECIMAL DESCRIPTION
--------------------------------------------------------------------
0 0x0 ELF, 64-bit LSB executable, AMD x86-64, version 1 (SYSV)
362953 0x589C9 Unix path: /var/log/encrypt.log
367904 0x59D20 Unix path: /var/tmp/.guestfs-0/appliance.d/root
```

The path /var/log/encrypt.log suggests the malware may record encryption activity or status messages to a log file during execution.

The second path, /var/tmp/.guestfs-0/appliance.d/root, appears to reference a temporary directory structure associated with virtual appliance. Its presence indicates the malware may interact with temporary VM-related files

Analysis of embedded strings within the sample provides valuable insight into the ransomware's functionality, execution flow, and intended interaction with the ESXi environment.

Embedded versioning within the binary:

```
 LINUX Locker v1.07
LOCKBIT5.0
```

The extracted strings clearly show that this ransomware is specifically designed to target VMware ESXi environments. References to /vmfs and /vmfs/volumes/ indicate an intent to access VMFS datastores where virtual machine disk files reside, while embedded vim-cmd commands such as vmsvc/getallvms, vmsvc/power.off, and vmsvc/power.getstate demonstrate built-in functionality to enumerate, check the status of, and forcibly shut down virtual machines before encryption. Additional environment-validation strings like ESXi 4, ESXi x64, and vmware -v suggest the malware verifies it is executing on a legitimate ESXi host, helping avoid accidental deployment on non-target systems or analysis environments.

Status messages such as "[OK] VM %s (ID: %d) powered off." indicate that the ransomware tracks and confirms successful shutdown of individual virtual machines.

```
 # ESXi Specific indicators:
/vmfs
/vmfs/volumes/
/bin/vim-cmd vmsvc/getallvms - enumeration
/bin/vim-cmd vmsvc/power.off %d - poweroff
/bin/vim-cmd vmsvc/power.getstate %d
ESXi 4
ESXi x64
vmware -v
```

Meanwhile, "[INFO] VM %s (ID: %d) in bypass list, skipping" suggests the presence of an internal exclusion list that prevents certain VMs from being targeted. The command fragment ps | grep vmx | grep "%s" further supports this workflow, showing that the malware searches running processes for VMware VMX instances associated with specific virtual machines.

```
 VM Logic:
[OK] VM %s (ID: %d) powered off.
[INFO] VM %s (ID: %d) in bypass list, skipping
ps | grep vmx | grep "%s"
```

Options such as -f (fast mode), -d (target specific directories), -l (enable logging), and -b (background execution) provide operational flexibility, while -m (note storage mode) and -k (disable self-destruction) indicate adjustable post-encryption behaviors.

Additional flags directly influence the encryption process: -r controls the percentage of each file to encrypt, -w enables wiping of free disk space, -t sets a delayed start, -o disables automatic VM termination, and -n allows specific virtual machines to be excluded by ID. Supporting status messages such as “Fast mode enabled (1% encryption)” and “Fast pass completed. Starting full encryption pass”, along with the marker .vmdk.fastpass, confirm a two-stage encryption strategy in which a rapid partial pass is followed by more thorough encryption.

```
 Parameters:
-f Fast mode (1% encryption)
-d Specific directories to encrypt
-l Enable logging
-b Background mode
-m Note storage mode (0,1,2)
-k Do not self-destruct

 Encryption:
-r Encryption percentage (10-90)
-w Wipe free disk space
-t Timeout before starting
-o Don't try to automatically kill VMs
-n Skip VMs by ID

 Fast mode enabled (1% encryption)
Fast pass completed. Starting full encryption pass
.vmdk.fastpass
```

The list of targeted file extensions clearly shows that the ransomware is engineered to disrupt virtual machine infrastructure at the hypervisor level. Files such as .vmdk (virtual disk files) and .vmx (VM configuration files) are essential for virtual machine operation, while supporting files like vswp (swap files), vmem (memory snapshots), and vmsn (snapshot state files) are also included to ensure full impact. Additional metadata and state files: vmsd, vmtx, vmss, vmxf, and nvram, further indicate an intent to corrupt both runtime and configuration data.

```
 # Targeted Files:
.vmdk
.vmx
vswp
```

```
vmem
vmsn
vmsd
vmtx
vmss
vmxf
nvram
```

The reference to /var/log/encrypt.log along with the message “[INFO] Logging enabled” suggests the malware can record its activity to a local log file, likely to provide operators with execution feedback in ESXi’s command-line driven environment. Additionally, messages such as “[INFO] Binary removed successfully.” and “[ERROR] Failed to remove binary.” reveal a built-in self-deletion routine intended to remove the executable after deployment.

```
 Logging:
/var/log/encrypt.log
[INFO] Logging enabled
Self delete:
[INFO] Binary removed successfully.
[ERROR] Failed to remove binary.
```

References to /proc/self/maps and /proc/self/status, along with the field TracerPid:, indicate that the malware inspects its own process status to determine whether it is being debugged or traced. Mentions of tools such as valgrind, frida, strace, and ltrace show that it actively looks for common dynamic analysis and instrumentation frameworks. Additionally, the presence of asan, tsan, msan, and ubsan, components of compilers used in debugging and research environments, suggests further checks designed to avoid execution under memory analysis.

```
 Anti Analysis and Anti Sandbox:
/proc/self/maps
TracerPid:
/proc/self/status
valgrind
frida
strace
ltrace
asan
tsan
msan
ubsan
```

Directories such as /proc, /dev, and /sys contain virtual- and device-related files essential for kernel and hardware interaction, while /bootbank and /altbootbank store ESXi boot images and recovery partitions. Core system locations like /etc, /lib, and /bin are also excluded, preventing corruption of configuration files and essential

binaries. By excluding these paths, the malware ensures the host remains bootable and reachable, allowing victims to access the system and potentially pay the ransom.

```
Exclutions (avoid breaking the hypervisor):
/proc
/dev
/sys
/bootbank
/altbootbank
/etc
/lib
/bin
```

String analysis revealed extensive ESXi-specific functionality within the sample, including native interaction with VMware management utilities (vim-cmd), VM enumeration and shutdown logic, and targeted encryption of VM disk and configuration artifacts. The binary exposes a comprehensive command-line interface allowing operators to control encryption scope, percentage, execution mode, logging behavior, and VM exclusions, consistent with LockBit's affiliate-driven ransomware-as-a-service model. Additional features include fast-pass partial encryption, optional free-space wiping, self-deletion, and multiple anti-analysis checks targeting debugging and instrumentation frameworks.

Control Flow And Core Routines

To make sure the ransomware is running on the ESXi host, it first performs an OS check via execution the following command:

```
01036("vmware -v 2> /dev/null"
```

The command **vmware -v 2> /dev/null** is used to quietly check whether the system is running VMware by querying the version information while suppressing any error messages, allowing the malware to detect an ESXi environment without producing visible output.

```
_00437350(&lStack_15538,0x200,
          "ps | grep vmx | grep \"%s\" | grep -v grep | awk \'{print $2}\' | sort -u"
```

The command **ps | grep vmx | grep "%s" | grep -v grep | awk '{print $2}' | sort -u** is used to locate running VMware virtual machine processes that match a specific identifier, extract their process IDs, and produce a unique list, likely so the malware can terminate active VMs before encryption.

```
  FUN_004172a0("[OK] VM %s (ID: %d) is already powered off.\n",pbVar16,iVar39);
}
else {
  iVar11 = 5;
  FUN_00437350(&bStack_15738,0x100,"/bin/vim-cmd vmsvc/power.off %d",iVar39);
  do {
    thunk_FUN_00400f06(&bStack_15738);
    iVar12 = FUN_00449e40(iVar39);
    if (iVar12 == 0) {
      FUN_004172a0("[OK] VM %s (ID: %d) powered off.\n",pbVar16,iVar39);
      goto LAB_00448aef;
    }
    syscall();
```

Virtual machine shutdown routine, where the malware checks a VM's power state and, if still running, repeatedly issues a **vim-cmd vmsvc/power.off** command while verifying the result, confirming an automated loop designed to ensure VMs are fully powered off before encryption begins.

```
if (DAT_0065d2b0 != 0) {
  FUN_004172a0("[INFO] Fast mode enabled (1%% encryption)\n");
}
if (local_1328 != 0) {
  FUN_004172a0("[INFO] Background mode enabled\n");
}
if (DAT_0065d340 != 0) {
  FUN_004172a0("[INFO] Quiet mode enabled (no extension changes)\n");
}
if (local_1318 != -1) {
  FUN_004172a0("[INFO] Note storage mode: %d (0=none, 1=all dirs, 2=root only)\n");
}
if (local_1314 != 0) {
  FUN_004172a0("[INFO] Self-destruction disabled by command line\n");
}
if (DAT_0065d400 != 0) {
  FUN_004172a0("[INFO] Free space wiping enabled\n");
}
if (local_131c != -1) {
  FUN_004172a0("[INFO] Manual encryption percentage: %d%%\n");
```

Command-line options such as fast encryption mode, background execution, quiet mode, note storage behavior, self-destruction control, free space wiping, and custom encryption percentage, indicating a highly configurable execution flow.

```
a0("\n[INFO] Statistics:\n");
a0("Files found: %llu\n",(long)DAT_0065d2f4);
a0("Files encrypted: %llu\n",(long)DAT_0065d2f0);
a0("Files skipped: %llu\n",(long)DAT_0065d440);
a0("Time elapsed: %ld seconds\n",DAT_0065d448 - _
```

The ransomware maintains real-time operational metrics, likely to provide operators with progress visibility and to verify that the encryption process completed successfully across the targeted environment.

Encryption

ChaCha20 stream cipher implementation:

```
if (DAT_00660860 == 0) {
  DAT_00660860 = 1;
  _DAT_00660870 = 0x3320646e61707865;
  uRam0000000000660878 = 0x6b20657479622d32;
}
```

Hex Values Decode to ASCII

Input

```
if (DAT_00660860 == 0) {
  DAT_00660860 = 1;
  _DAT_00660870 = 0x3320646e61707865;
  uRam0000000000660878 = 0x6b20657479622d32;
}
```

133 5

Output

```
Ú f ` Ú f ` Ú f p3 dnapxe
f xk etyb-2
```

Reordered in memory (little-endian), this value spells “**expand 32-byte k**”, which is the ChaCha20 constant string that is used to initialize the state.

ESXi I386 Variant Analysis

The analyzed sample is a 32-bit Linux ELF (i386) associated with VMware ESXi targeting and has been observed labeled as “LINUX Locker v1.06”, that may indicate an unchanged build from a prior release or a version counter oversight by the operator. While it can execute on a generic Linux host that provides the required 32-bit runtime or loader, its orchestration is clearly optimized for ESXi: the operational flow is equivalent to that observed in previous x64 ESXi sample. It assumes datastore discovery and traversal under **/vmfs/volumes/…** and integrates

VMware management tooling (e.g., **vim-cmd**) for VM enumeration and power-state handling, meaning ESXi-dependent stages may fail or be bypassed on non-ESXi Linux unless paths are supplied manually.

```
uVar29 = (int)((ulonglong)uVar16 * (ulonglong)uVar4 >> 0x20) + uVar4 * uVar6 +
         (int)((ulonglong)uVar18 * (ulonglong)uVar2 >> 0x20) + uVar2 * uVar7 +
         (uint)CARRY4(uVar11,uVar13) + (uint)CARRY4(uVar21,uVar24 * local_94) +
         (int)((ulonglong)uVar17 * (ulonglong)uVar25 >> 0x20) + uVar25 * uVar9 +
         (uint)CARRY4(uVar22,uVar14) +
         (int)((ulonglong)uVar26 * (ulonglong)uVar3 >> 0x20) + uVar3 * uVar8 +
         (uint)CARRY4(uVar27,uVar15);
```

The sample of decompiled code illustrating Poly1305 operations

From an analysis perspective, the binary is stripped and notably omits the section header table, reducing the effectiveness of common automated triage and section-driven reversing workflows. The implementation also indicates concurrent processing via threading and TLS state, and it encodes auxiliary strings and operational parameters behind a runtime string obfuscation layer where high-value text (paths, command templates, and messages) is decoded on demand using a rotate-based transform. Functionally, the codebase includes a modern cryptographic stack ChaCha-family present.

In the next parts of this series, we will cover the Linux and Windows variants.
