Pragmatic Security Audits

Fortifying HPC Environments at a Consumable Pace

Dennis Walker
HPC Solutions Engineering
Hewlett Packard Enterprise
Las Vegas, NV, United States
dennis.walker@hpe.com

Monica Dessouky
HPC Deployment Program
Hewlett Packard Enterprise
Los Angeles, CA United States
monica.dessouky@hpe.com

Alden Stradling
HPC Systems Engineer
Los Alamos National Laboratories
Los Alamos, New Mexico, United States
stradling@lanl.gov

ABSTRACT

Securing high performance computing (HPC) environments requires more than ad hoc patching—it calls for a structured, scalable, and focused approach that can keep pace with evolving threats. Pragmatic Security Audits outlines a practical methodology developed during the deployment of a complex, hybrid-cloud, multi-zoned supercomputing system. This paper introduces an efficient, repeatable framework to identify and assess vulnerabilities using industry-standard scanning tools and then narrow the focus of immediate remediation on high-threat targets using a standardized risk-based prioritization model.

INTRODUCTION

The critical importance of robust security cannot be overstated. A single deep exploit can imperil data privacy, compromise the integrity of ongoing computational jobs, and destabilize the security foundation of an entire supercomputer, potentially necessitating a complete system reinstall or even a rebuild of the network. Security is only as strong as its weakest link. Recent public exploits have demonstrated the severe impacts of security breaches, such as the SolarWinds incident. Attackers infiltrated numerous systems globally, leading to data compromise and operational disruptions. System administrators face the daunting task of staying current on the threats in play and keeping their systems protected, in addition to keeping everything working. It's therefore important to establish a consistent, recurring schedule for audits and remediation to ensure ongoing protection against emerging threats and prevent issues from piling up.This pileup can be a serious issue with large systems. A large number of interacting node types, user packages, and containerized workloads can overwhelm administrators and conceal priority threats.. This document outlines a practical, scalable approach to HPC security implemented at a recent customer site to safeguard their many new environments effectively.

Recently, a customer nearing a production launch of a hybrid-cloud, multi-zoned supercomputing infrastructure designed to handle workloads with varying security classifications was required to demonstrate adherence to stringent, industry-standard security protocols. All services, software, hardware, and user accounts were scanned using the tooling recommendations. A remediation backlog was built to guarantee the environment protection against vulnerabilities and potential exploits. The supercomputing system was measured against threats following the Center for Internet Security (CIS) guidelines, processes, and standards from the National Cyber Security Centre (NCSC).

The customer's contractual language required a routine analysis to ensure ongoing protection against vulnerabilities and formally demonstrate that the system is protected. The end user accesses supercomputing through public cloud computing services. The system software stack includes Cray OS (COS) for the compute nodes, Cray System Management (CSM), multiple Cray Programming Environments (CPE), HPE Performance Cluster Manager (HPCM), Slingshot and other shared infrastructure software such as ClusterStor Data Services (CDS). This is an excellent example of the need for multiple trust zones to allow for research collaboration on an open user ingress, while hardening the management plane.

Security hardening needs to be a continuous process to protect against new vulnerabilities as they are discovered. To that end, the tools, techniques and processes for ongoing stewardship of security hardening need to be well defined and demonstrated. Like the broader IT infrastructure, supercomputers require umbrella mechanisms for proactive prevention of vulnerabilities that are tuned to the nuances of specific implementations. Considering that supercomputing often delivers services where life or national security depend on it, the fiduciary and ethical responsibility to impose protections is paramount.

CCS CONCEPTS

• Operating systems security
• Vulnerability management
• File system security
• Firewalls
• Distributed systems security
• Network security
• Software and application security
• Software security engineering

KEYWORDS

DevSecOps, Continuous Integration (CI), Continuous Security, Vulnerability Management, Network Port Scanning, Security Benchmarks, Threat Modeling

1 Gather Data

1.1 Inventory Node Configurations

Vulnerabilities and exploits are introduced with any software. The rate of discovery is generally escalating. In order to profile risk, taking inventory of node configurations is an important first step.

Since configurations across node types are generally consistent, an example node for each configuration type is added to the test footprint, including servers, switches, and network appliances. These can include (but are not limited to) Compute, Login, Management, Tiered Boot Nodes, IO, Fabric Manager,, Scheduler, Visualization, Scratch Storage, Home/Project Storage, Management Switches, and High-Speed Network (fabric) Switches. Scans are orchestrated against these nodes, leveraging industry-standard tools to identify exploits, threats, and vulnerabilities.

The following sections describe how to gather data through open source scanning tools, aggregating results to later identify and prioritize remediations to the findings.

1.2 Common Vulnerabilities and Exposures

Software vendors provide an Open Vulnerability and Assessment Language (OVAL) file for most Linux Distros, Windows, and Mac operating systems. This file describes security vulnerability and system configuration in a machine-parseable format for OpenSCAP(3) tooling. Many vendors also provide Security Technical Implementation Guide or STIG file. This file contains a long list of machine-orchestrated checks to ensure the node is configured according to the Department of Defense (DoD) security standards.

For this approach, we'll use the OVAL file and cover the security benchmarks using the separate NIST tooling. Below is an example of installing the tool, downloading the OVAL file, and scanning the machine on a SUSE-based Linux distro. Findings are converted into CSV and imported into a shared spreadsheet for threat modeling and prioritization.

Example 1.2.1 - Installing openscap on SuSE and downloading the list of vulnerabilities.

```
zypper install openscap-utils

wget
https://ftp.suse.com/pub/projects/security/oval/suse.linux.enterprise.server.15.xml

oscap oval eval --report ./[NODE TYPE].html
./suse.linux.enterprise.server.15.xml
```

A custom python script is used to translate the results into a simplified CSV view. This script also fetches the remediation verbiage and severity score from internet sources. [link to be provided]

Example 1.2.2 - Invoking script to enrich data and export to CSV.

```
scrape_to_csv.py [NODE TYPE].html
```

Next, results are aggregated into a shared excel or similar spreadsheet, and values for threat and LoE (level of effort) are calculated for stack-ranking priority.

1.3 Security Benchmarks

Scanning for security benchmarks is a critical part of maintaining a secure and compliant IT infrastructure. These benchmarks are essentially best-practice configurations developed by trusted organizations like the Center for Internet Security (CIS) and the National Institute of Standards and Technology (NIST). They help ensure that systems are hardened against known vulnerabilities and misconfigurations that could be exploited by malicious actors. Regularly scanning for compliance with these benchmarks allows organizations to detect drift from secure

configurations, reduce attack surfaces, and demonstrate due diligence for regulatory requirements like FISMA, HIPAA, and PCI-DSS.

Examples of security benchmark findings might contain the following:

- Check that only the minimum level of file permissions is set on all system files for the system to function correctly.
- All interfaces to external services (such as LDAP and NTP) are properly protected via encryption keys for authentication.
- Tune the lifetime of Keycloak JWT authentication tokens and make sure proper token renewal policies are implemented.
- Users only have access to files they need to have access to via Unix file permissions and ACLs.
- Privileged operators and administrators are restricted via role-based access control lists, Sudo permissions and standard Unix file permissions and ACLs.
- Disable all unnecessary daemons on the nodes.
- Constrain externally routed network interfaces on compute nodes to the required level to minimize them being used for denial-of-service (DoS) attacks.
- Control the ability of users to run Cron jobs.
- Enable all user/admin login activity logging and selected auditing and implement automated screening of logs for suspicious events on a regular basis.

One of the most effective tools for performing these scans is CIS-CAT Pro(2), a configuration assessment tool provided by the Center for Internet Security. This tool is particularly valuable because it supports scanning against a wide array of CIS Benchmarks, covering everything from operating systems and network devices to cloud services. It automates the process of comparing system configurations against benchmark standards and generates detailed reports highlighting compliance levels and remediation guidance.

Using CIS-CAT Pro involves a few key steps. First, download the tool from the [CIS WorkBench](https://workbench.cisecurity.org/) (membership required for Pro access). Once installed, configure it with the appropriate benchmark profiles and credentials for target systems. You can run scans either locally or remotely, depending on your environment. After scanning, CIS-CAT generates HTML or CSV reports showing pass/fail results for each control, along with recommendations for bringing non-compliant settings into alignment. These reports can be used to prioritize security efforts, guide system administrators, and inform broader risk management strategies.

Integrating CIS-CAT Pro into a continuous monitoring or DevSecOps workflow can significantly improve an organization's security posture by ensuring that systems remain compliant with industry-accepted security configurations over time.

Here's an example of how to invoke **CIS-CAT Pro** from the command line to scan a local system against a specific benchmark:

Example 1.3.1 - Invoking CIS-cat security benchmark scan
```
java -jar CIS-CAT-Assessor.jar -b
benchmarks/CIS_SUSE_Linux_Enterprise_15_Benc
hmark_v1.1.1-xccdf.xml -p Level_2 -html -r
reports/
```

Explanation of the flags:

- `-jar CIS-CAT-Assessor.jar` – Launches the CIS-CAT Pro Assessor.
- `-b` – Specifies the path to the XCCDF benchmark file (you can download these from the CIS WorkBench).
- `-p Level_1` – Chooses the profile to use (typically Level 1 or Level 2, with Level 1 being less restrictive and focused on essential security).
- `-a` – Runs an assessment on the local system.
- `-r reports/` – Outputs the assessment report to the `reports/` directory.

Optional and useful flags:

- `-u` – If running a remote assessment, use this for the username.
- `-pw` – Password for remote user.

- `-t` – Target IP or hostname for remote assessment.
- `-f csv|html|json` – Specifies output format(s).
- `--help` – Displays all available options.

This example assumes you're running the scan from a directory where the `CIS-CAT-Assessor.jar` and `benchmarks/` folder are located. Make sure Java is installed on the system, as the Assessor is a Java-based tool.

Next, a custom Python script parses the results in CSV and assigns a threat score.

Example 1.3.2 - Invoking script to convert results from HTML to CSV.

```
./convert_to_csv.py [path to html file]
```

This produces a report consistent with the vulnerabilities report in the previous section which is useful for rapidly making sense of priorities.

1.4 Port Scans

For port scans, nmap(4) is a free and widely available tool for fingerprinting and scanning open ports from each node configuration type. It provides an option to save the results in machine-parseable XML.

Example 1.4.1 - Port scanning, exporting results to XML.

```
# Show OS, service type, and vulnerabilities
nmap -A 10.92.100.0/16 -oX output.xml

# Detect firewall rules
nmap -sA 10.94.100.0/16 -oX
firewall_rules.xml
```

As before, these results are converted to CSV file for the purposes of aggregating results into a composite view.

Example 1.4.2 - Invoking script to convert results from XML to CSV

```
./convert_to_csv.py [path to XML file]
```

2 Analyze Data

2.1 Identify Trust Zones

Identifying trust zones refers to the process of segmenting a digital environment into areas based on differing levels of trust, risk, and required security controls. Each zone is defined by who or what has access to it, the sensitivity of the data it handles, and the potential consequences if that zone is compromised. This concept is foundational in modern network architecture and security frameworks such as zero trust, where no implicit trust is granted based on location or credentials alone.

For example, a typical agency might define separate trust zones for public-facing login nodes, internal machine learning (ML) applications, administrative services, and sensitive data stores. If nodes in one zone are compromised, the attacker's next goal is generally to infiltrate the next zone.

Identifying trust zones allows organizations to apply defense-in-depth strategies, enforcing granular security policies at each boundary and reducing the overall attack surface. It also helps in incident response, enabling security teams to quickly assess the scope of a breach by understanding which zone was affected and what level of access was granted within that zone. As HPC environments and remote work increasingly blur traditional perimeters, clearly defining and enforcing trust zones becomes even more critical to maintaining security and compliance. An example of attempts to standardize this trust zone model can be found in NIST Special Publication 800.223 [1]

2.2 Establish Threat Model

Establishing a threat model in cybersecurity involves systematically identifying potential threats, vulnerabilities, and attack vectors that could impact a system, application, or organization. The goal is to understand how an attacker might compromise assets, what they might target, and what methods they might use. This process typically includes mapping out the architecture of a system, identifying valuable assets, recognizing potential adversaries (e.g., insiders, cybercriminals, foreign intelligence), and analyzing how those adversaries could exploit weaknesses.

A strong threat model answers questions such as: What needs protection? Who are the potential attackers? What are the most likely attack paths? It often includes the classification of threats using frameworks such as STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) or DREAD

(Damage potential, Reproducibility, Exploitability, Affected users, Discoverability), which help categorize and score the severity of threats. By documenting and visualizing these risks, organizations can gain a clearer understanding of their security posture.

Once established, the threat model becomes a vital tool for prioritizing remediation efforts. Instead of addressing vulnerabilities in a random or purely severity-based order, organizations can focus on fixing the issues that pose the greatest real-world risk—those most likely to be exploited and most damaging if successful. For example, a medium-severity vulnerability in a publicly exposed admin portal might take priority over a high-severity issue in a well-isolated internal system. This risk-based approach ensures that security resources are used efficiently, helping organizations reduce their exposure in a way that's aligned with their specific threat landscape.

Example Supercomputer Threat Model

In this example, a 3-factor risk model that includes impact, user adjacency and exploitability, a total weight can be determined for each finding by adding individual assigned weights. For example: 6 (impact 3) + 8 (1 degree of separation) + 2 (exploitability) = 16 Threat. After assigning a weight to each finding, the weights are ranked from highest weight to lowest weight for a prioritized list of vulnerabilities. The higher the weight, the more critical and higher the priority for implementation. Using this approach, any vulnerabilities identified during a proactive audit may be remediated according to priority, with the highest criticality (rank) being addressed first. A formal written report documenting the risk model used, prioritized rank assignments of vulnerabilities, and remediation actions is created.

Table 2.1.1 - Example threat scores by vulnerability node type

Vulner -ability	Location	Impact (1-10)	Exploit -ability (1-10)	Expo -sure (1-10)	**Threat Score**
A	Login Node	3	8	9	**6.67**
B	System Manageme nt API	9	4	2	**5**
C	Compute Node	4	7	5	**5.33**

How Risk Score Is Calculated

This example uses a basic formula for risk score:
Threat Score = (Impact + Exploitability + Exposure) / 3

This is a simplified threat model—many organizations might weight these factors differently or use more complex formulas, but this approach illustrates the concept. Whatever you choose, the threat score should roughly align to plain English terms describing the threat.

Table 2.1.2 - What the threat score should approximately mean in plain English

Threat	Description
10	The critical workload is unable to run on any part of the supercomputer
8	The critical workload is running on its designated supercomputer partition, but is unable to run on any other partition; or the research workload and/or total integrated solution availability is significantly impacted
6	A single supercomputing partition is unavailable for use, but the critical workload can run on at least two partitions; or the research workload and/or total integrated solution availability is impacted
4	The total integrated solution is not available for use by a small subset of users
2	Aspects or total integrated solution are not performing as expected, but critical workload and research workload are not impacted

How Impact Is Calculated

Contributing to the above formula, "impact" indicates the amount of potential damage to a supercomputer an attacker might be able to inflict.

This is typically supplied by the scan tool, such as a CVSS score in OSCAP or the severity score from CIS-cat. These tools will usually supply both a numerical score and a qualitative score, e.g. "critical" or "medium".

Table 2.1.3 - CVSS severity description to numerical score

Weight	CVSS Scores
Critical	CVE Score 9-10

High	CVE Score 7-8
Medium	CVE Score 5-6
Low	CVE Score 0-4

How Exposure is Calculated

Likewise, exposure or user adjacency is given a score on the basis of how accessible the exploit is to a user. Below is an example scoring matrix.

Table 2.1.4 - Exposure calculation by node adjacency to user accessible node

Weight	User Adjacency to Exploit
10	User has direct access; found on a login node and does not require elevated permissions
8	User is separated by 1 degree; found on a login node, but not accessible unless another exploit is compromised
6	User is separated by 2 degrees; found on a Compute node and requires elevated permissions
4	User is separated by 3 degrees; found on a management node but requires local privilege escalation and service exploit
2	User is separated by more than 3 degrees

The resulting list of prioritized remediation actions from the internal audit is implemented in accordance with the entity's security policy and change governance process.

How Exploitability is Calculated

Exploitability is a measurement of the efficiency to access the attack. For example, if the attack requires physical access or execution during a specific timewindow, the score reduces. Exploitability accumulates a score up to ten on the basis of yes or no questions.

Access Vector (+2)

- Can the attack be performed over the network? Does it require physical access?
- Remote attacks (e.g. over the internet) are generally more exploitable.

Authentication Not Required (+2)

- Does the attacker need to be logged in, or have special privileges?
- Fewer required privileges = higher exploitability.

Complexity (+2)

- Is the exploit straightforward (e.g., submitting a crafted form), or does it require perfect timing or specific conditions?
- Lower complexity = higher exploitability.

Available Exploits (+2)

- Are there known, public exploits or tools already available?
- If yes, that raises the exploitability significantly.

User Interaction Not Required (+2)

- Does the attack require tricking a user (e.g., phishing or clicking a malicious link)?
- If user interaction is required, exploitability usually decreases.

3 Automate

3.1 DevSecOps Pipelines

In section one, three types of security scans are executed against each node configuration type. For the purposes of this paper, let's assume twelve different node configuration types replicated across three HPC environments. At three scans per node type per environment, that's 108 security scans to execute across all three environments.

$|E| \times |N| \times |S| = 3 \times 12 \times 3 = 108$

Where:

- E: set of environments, $|E| = 3$
- N: set of node types, $|N| = 12$
- S: set of security scans, $|S| = 3$

This would represent the **total number of scans**.

These scans can be fully automated. Jenkins is a ubiquitously chosen CI/CD tool for managing software deployments and infrastructure. It is also great for building out shift-left DevSecOps pipelines.

In this example, we'll build out a pipeline for each node configuration type, where connection details are parameterized in the Jenkins job, supplied by configuration secret. We'll also build a Jenkins job for each environment, running all 12 node scans in parallel. We'll finish with a Job that aggregates all of the results into a single reports

A pipeline to scan a single node takes approximately 15 minutes. Thankfully, all node types can be scanned in parallel.

For each environment scan, hundreds of findings are collected and assessed for threat score on the basis of user adjacency and severity provided by the tooling.

4 Remediate

In high performance computing (HPC) environments, managing security vulnerabilities can feel overwhelming—especially when faced with a long list of findings. However, it's essential to keep in mind that meaningful progress does not require solving every issue all at once. Even addressing a small number of vulnerabilities can yield a significant reduction in overall risk, particularly when those vulnerabilities exist on the system's most exposed components.

Login nodes and compute nodes are the primary surfaces that users interact with. These nodes, due to their exposure, are often the most critical to secure quickly. The good news is that applying patches to these components is typically more straightforward compared to deeper layers of the infrastructure, such as the management or provisioning systems. In many cases, addressing these vulnerabilities involves updating the package repository of the underlying Linux distribution and applying available security patches via standard package management tools like rpm, zypper, yum, dnf, or apt. This familiar process avoids the complexities of custom patching or kernel-level rebuilds and can usually be completed with minimal system downtime.

To realistically plan remediation efforts, teams can use a simple capacity model. A commonly used estimate is that one full-time engineer can address approximately two to four security findings per week. By multiplying the number of full-time equivalents (FTEs) available by three, and then by the number of weeks in the sprint or remediation cycle, organizations can estimate the number of findings they can reasonably expect to resolve. It's also wise to account for operational overhead—typically one week is subtracted from each cycle to accommodate deployment, testing, and verification.

Let:

- E = Number of full-time engineers (FTEs)
- C = Average number of changes/remediations per engineer per week (typically 2–4, so we use an average like 3)
- W = Number of weeks in the sprint or remediation cycle
- O = Weeks allocated for operational overhead (e.g., 1 week)

Then the total remediation capacity R is:

$R = E \times C \times (W - O)$

Applied Example:

Given:

- E = 2 engineers
- C = 3 changes per week
- W = 4 weeks
- O = 1 week

$R = 2 \times 3 \times (4-1) = 2 \times 3 \times 3 = 18$

So, the team can **realistically handle 18 remediations per month**.

Using this model enables an organization to limit its focus to the 10-20 most exposed and impactful changes per month.

This pragmatic approach allows organizations to chip away at a larger vulnerability backlog in a manageable, predictable way—starting with the most exposed and potentially impactful areas of their systems. By doing so, they can maintain progress and reduce risk without requiring an all-at-once overhaul of the entire infrastructure.

An organization can fully automate the application of updates, configuration changes, and submit them to

functional benchmark testing. This is the topic of several other CUG whitepapers, including one by the author(5). By implementing and maturing DevOps pipelines to manage change, the velocity of remediation multiplies 3x - 10x or even more.

5 SUMMARY

In summary, keeping clear visibility into the current state of your HPC environmental security can be maintained efficiently with common tools and the right automation. An actionable and achievable backlog can be created by using a threat model and capacity plan to translate the list of vulnerabilities into a remediation plan addressing real threats.

ACKNOWLEDGMENTS

ChatGPT was used to scaffold some of the basic content structure. Almost all of it was rewritten and restructured to provide the detail, but some phrasing may remain. Grammarly was used for grammar checking and simplification.

Thank you to Larry Kaplan and Harold Longley at HPE for providing helpful feedback on the content structure of this paper.

REFERENCES

[1] Guo Y, Chandramouli R, Wofford L, Gregg R, Key G, Clark A, Hinton C, Prout A, Reuther A, Adamson R, Warren A, Bangalore P, Deumens E, Farkas C (2024) High-Performance Computing Security: Architecture, Threat Analysis, and Security Posture. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-223. https://doi.org/10.6028/NIST.SP.800-223)

[2] CIS-cat Pro - https://www.cisecurity.org/cybersecurity-tools/cis-cat-pro

[3] OpenSCAP Vulnerability Assessment - https://www.open-scap.org/features/vulnerability-assessment/

[4] nmap port scanning - https://nmap.org/book/port-scanning-tutorial.html

[5] A Lovell-Troy, D Walker, SV Khalsa (2025) From Weeks To Hours: Harnessing Configuration Management and Deployment PIpelines. (Cray User Group), CUG 2025