Security of HPC Systems: From a Log-analyzing Perspective

Zhengping Luo[1,*], Zhe Qu[1], Tung Thanh Nguyen[2], Hui Zeng[2], Zhuo Lu[1]

[1]Department of Electrical Engineering/Florida Center for Cybersecurity, University of South Florida, Tampa FL 33620, USA.

[2]Intelligent Automation Inc., Rockville MD 20855, USA.

Abstract

High Performance Computing (HPC) systems mainly focused on how to improve performances of the computing. It has competitive processing capacity both in terms of calculation speed and available memory. HPC infrastructures are valuable computing resources that need to be carefully guarded and avoid being maliciously used. Thus, vulnerabilities are quintessential issues in HPC systems due to most of jobs and resources run or stored usually are sensitive and high-profit information. In this survey, we comprehensively review securities of HPC systems from a log-analyzing perspective, including well-known attacks and widely used defenses, especially intruder detection methods. We found that log files are used for the security purposes much less than what we expected. How to use all the available log files comprehensively and employ state-of-the-art intrusion techniques to improve the robustness of HPC systems still lies for future research.

Received on 07 July 2019; accepted on 29 July 2019; published on 01 August 2019

Keywords: Security, high performance computing, attacks and defenses, intrusion detection, log file analysis

doi:10.4108/eai.19-8-2019.163134

1. Introduction

High performance computing, a widely used technology in fields such as aerospace, automotive, semiconductor design, energy explore, financial computing, weather forecast and nuclear simulation etc., mainly focused on how to improve the performance of the computing from software development, parallel algorithms and computer architecture etc. It has become an important way to do scientific research. Clusters and grids are the two dominant and distinct methods of deploying HPC parallelism in both industrial and academia nowadays [1].

As performances and capabilities of HPC infrastructures have continued to evolve, applications and software that are running on HPC infrastructures have also increased, especially with software applications of open source frameworks. However, they also become increasingly desirable targets to attackers [2]. HPC security, which is one of the most concerned problems for both users and HPC system maintainers, is not a significant problem in the past. Because most of HPC systems were built for private use by some dedicated users at the beginning. Now it has been the main worries and obstacles for the spreading of HPC systems as now they are often shared by multiple users, especially with the emerging of the idea of HPC-as-a-service.

The security of HPC systems has long been a research focus to build reliable and accountable systems [3–8], especially as log files being introduced to detect malicious intruders and behaviors. Computing power, as a resource, is required to be used only by those approved users, or further, to prevent approved users from performing unapproved behaviors. On the other hand, legal users of computing resources want to protect communications between users and resource owners to be safe and keep their data away from malicious attackers due to most of jobs and resources run or stored in HPC systems usually are sensitive and high-profit information. It might have serious consequences when hacked. Other security problems like denial-of-service are also a concern for HPC systems that has a lot of access requests from users.

Traditional security vulnerabilities still exist in HPC systems. For example, almost all HPC systems are multiuser systems, which requires using password to authenticate users. However, traditional security

*Corresponding author. Email: Zhengpingluo@mail.usf.edu solutions are often not effective enough provided characteristics of HPC systems [9]. Some HPC systems tend to have very distinctive modes of operation and be used for special purposes, notably mathematical computations or economic modeling. Some HPC systems run highly exotic hardware and software stacks and are extremely "open" to users. These distinctiveness of HPC infrastructure presents its vulnerabilities and security challenges for both users and system managers. Furthermore, the cooperation between users and "security policy" (i.e., the balance between convenience and security) is needed due to that "network security" is highly relied on "host security" [10].

The specific vulnerabilities of HPC systems can be traced back to the following three aspects [11]: *(i) High bandwidth connections* As the practice of HPC-as-a-service in industry, high bandwidth connecting the HPC infrastructure and the users, allowing steady interaction between users and systems have been a necessary requirement. These high bandwidth connections can be manipulated by hackers to launch denial-of-service attacks. *(ii) Extensive computational power.* The massive computing power, on one hand, can be used by innocuous users to solve scientific problems or other computing-extensive challenges. On the other hand, it also might be manipulated by malicious attackers to carry out brute-force attacks and all kinds of illegal intentions. *(iii) Massive storage capacity.* Almost all HPC systems include a large storage capacity. But the large-capacity disk storage is an attractive target for malicious attackers to store illegal files, such as pornographic multimedia files.

In this survey, we research the problem of detecting malicious users or behaviors in HPC systems based on logs. The contributions can be listed as follows:

- We conduct a comprehensive review on the security of HPC systems, including vulnerabilities, consequences, and potential solution strategies.
- Well-known attacks and defense methods, especially intrusion/anomaly detection methods, in HPC systems are identified.
- The process of log-based intrusion detection methods is articulated and surveyed, which is an important aspect of solutions for HPC security problems.
- We review defense mechanisms of HPC systems based on logs, and found that as the widely application of machine learning techniques, log files can be utilized in a more complete way to improve the security of HPC systems,

The remainder of this survey is organized as follows: we review vulnerabilities, threats and common attack examples in Section 2. The well-known defense mechanisms and intrusion detection methods are articulated in Section 3. In Section 4, we comprehensively review the log-based security solutions especially intrusion detection methods. Machine learning techniques used in solving anomaly detection problems in HPC systems and the potential future work are reviewed in Section 5. Conclusion is made in Section 6.

2. Attacks in HPC Systems

Attacks in HPC systems can lead to unauthorized accesses to the high-performance system infrastructure with malicious intents to steal, compromise or vandalize HPC resources, e.g., hackers might vandalize the system on purpose if they failed to extract any useful information or exploit the system to launch a denial-of-service attack [12].

2.1. Vulnerabilities and threats

The security in HPC facilities is similar with those in typical IT context. They are connected to networks just like other computer systems, and often run on the Linux-based systems. Thus, many of the attack styles in typical IT context can also be applied in HPC facilities [11]. However, HPC systems also have their own vulnerabilities, because they usually run exotic hardware and software systems, and have highly different purposes and modes of use compared with typical IT systems.

Probes, scans, brute-force login attempts, and buffer overflow vulnerabilities are the common issues that trouble HPC facilities. Common vulnerabilities of HPC facilities can be listed as follows[2]:

- Intrusion attacks, which can lead to data leakage or other consequences. It is because HPC facilities are extremely "open" to users around the world, thus make them an easy target for attackers.
- Alteration of code or data.
- Misuse of computing cycles, i.e., using the HPC facilities to perform activities other than granted behaviors such as crypto-currency mining.
- Availability related threats, including disruption or denial-of-service attacks against HPC facilities.

Threats confronted by HPC systems can be classified into three well-known types: confidentiality (e.g., data leakages), integrity (e.g., alteration of data or code) and availability (e.g., disruption/denial-of-service attacks against HPC systems or networks that connect them) [13]. The cause of these attacks can be insiders or outsiders of the HPC system. There are two kinds of attacks that are most favored by the outsiders: the brute-force attack against the password system and man-in-the-middle attacks [14].

"Insiders" often have special advantages in terms of HPC security because they can access HPC infrastructure. Thus, it is much easier for insiders to launch attacks than others [11, 15], e.g., malicious users can change or modify the data stored in the system, or it can manipulate the system to do some illegal things such as spreading the rumors or viruses on the Internet. Even more, they can manipulate the system to run harmful programs to consuming the system resources.

HPC systems usually can be decoupled into 4 layers [16, 17]: application layer, middleware layer (including MPI,PVM, etc.), operating system layer and network layer. Each layer is well-instrumented for the sake of health monitoring. HPC log files have been widely used as the paramount dataset sources to analyze the performance and security of HPC systems. Extensive studies have shown the hidden structured patterns exhibited in HPC log files. Through monitoring log files of HPC systems, we can identify applications run on HPC systems and the resource allocation among nodes cluster. Thus, system administrators can verify whether the jobs are approved, or the system are used aggressively or maliciously by running jobs [18].

2.2. Commonly known attacks

Given the vulnerabilities and characteristics of HPC facilities, we list four representative attacks in Linux cluster-based HPC facilities, which can be launched against an HPC cluster architecture:

Daemon process-based attack [12] A daemon process is an application that does not need interaction with a terminal and running in background. It can be launched by any other process, and it can be applied to take over system resources even when the parent process has terminated execution. This sort of attack can be very powerful in MPI environment since when a daemon process is created, the MPI scheduler will lose track of it [12]. The daemon process can take over any kind of CPU usage or other computing resources.

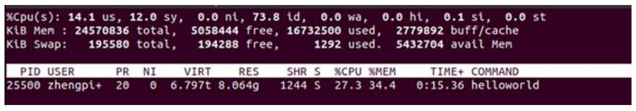
To emulate this type of attacks, one need to insert a Trojan into a trusted application appropriately to avoid being found by the HPC administrator. Example of Daemon process-based attacks include daemon memory allocation attack (running a daemon process to take over the memory resources) and daemon file attack (running a daemon process to take over the storage, memory, and computational resources of HPC facilities). An example of the daemon process-based attack template is shown as in Listing 1:

```
void main(){
int pid;
void *pointerMemory;

pid = fork();
if(pid < 0)
exit(EXIT_FAILURE);
if(pid > 0)
exit(EXIT_SUCCESS);
setsid();
signal(SIGHUP,SIG_IGN);
umask(0);
chdir("/");
pid = fork();
if(pid < 0)
exit(EXIT_FAILURE);
if(pid > 0)
exit(EXIT_SUCCESS);
signal(SIGPIPE,SIG_IGN);
openlog("helloworld daemon",LOG_PID,LOG_DAEMON);
int i = 1;
while(i < 100000000){
/*the attack method is put here*/
pointerMemory = (char*)malloc(SIZE);
syslog(LOG_NOTICE,"memory is allocated!");
i++;
sleep(DURATION_SEC);
}
syslog(LOG_NOTICE, "Terminated!");
closelog();
}
```

Listing 1: Daemon process-based attack example [12].

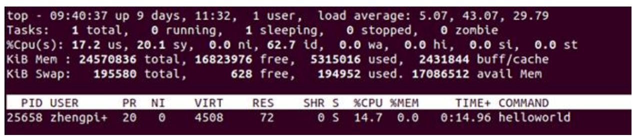
We run the code on Linux systems, the comparison of CPU and memory utilization between running the attack method in the example and without running the attack is shown in Fig. 1, from which we can observe that when the system keeps allocating memory pointer, the CPU time increased from 14.1 μs to 17.2 μs, while CPU time proportion for the user increased from 14.7% to 27.3% and memory usage increased from 0.0% to 34.4%. It demonstrates the attack power of a daemon process-based attack.



```
%Cpu(s): 14.1 us, 12.0 sy,  0.0 ni, 73.8 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
KiB Mem : 24570836 total,  5058444 free, 16732500 used,  2779892 buff/cache
KiB Swap:   195580 total,   194288 free,     1292 used.  5432704 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
25500 zhengpi+  20   0  6.797t 8.064g   1244 S  27.3 34.4   0:15.36 helloworld
```

(a) Daemon process based attack



```
top - 09:40:37 up 9 days, 11:32,  1 user,  load average: 5.07, 43.07, 29.79
Tasks:   1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s): 17.2 us, 20.1 sy,  0.0 ni, 62.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 24570836 total, 16823976 free,  5315016 used,  2431844 buff/cache
KiB Swap:   195580 total,      628 free,   194952 used. 17086512 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
25658 zhengpi+  20   0    4508     72      0 S  14.7  0.0   0:14.96 helloworld
```

(b) Normal daemon process

Figure 1. Comparison of CPU and memory usage (a) when running a daemon-process attack and (b) when running normal daemon process.

Interposition library attack[12] It is an attack by intercepting function calls that an application makes to the stack of share libraries, and then modify the called function to add malicious functionality to it. In Linux systems, interposition is "the process of placing a new or different library function between the application and its reference to a library function" [19]. Example interposition library attacks include attacking the libc library and attacking the MPI library. An example of the interposition library attack is shown in listing 2.

```
static int DoProfile = TRUE;
FILE *fopen(const char *filename,const char *mode)
    {
typedef FILE*(*function_type)(const char *filename
    , const char *mode);
static function_type function = NULL;
static char *function_name = "fopen";
FILE *retval;
if(!function){
function = (function_type)dlsym(RTLD_NEXT,
    function_name);
}
if(DoProfile){
DoProfile = FALSE;
retval = (*function)(filename,mode);
DoProfile = TRUE;
}
else
retval = ((*function)(filename,mode));
sys_call_table[SYS_open] = orig_open;
}
```

Listing 2: Interposition library-based attack example [12].

Probe-based login attack [20, 21] It is similar with brute-force login attack, when the username of the HPC facilities are known to the attacker, they can create a dictionary of all potential passwords to launch the password guessing campaign as HPC facilities usually are "open" to users all around the network.

Intrusion attack[8, 22–25] Given all the security issues in HPC systems, we found that intrusion attack is probably one of the most discussed attacks. Because for any malicious user to launch an effective attack, the primary obstacle is how to access the targeted system. Thus, many problems of security for HPC systems are centered around intrusion attacks and detection.

Some commonly known intrusion attacks include attempts to copy the password file frequently, a lot of unreliable remote procedure call requests in a very short time and attempts to connect to non-exist "bait" hosts frequently [24].

Besides attack examples listed above, there are many other attack methods in HPC systems [2, 11, 25]. Based on all these various attacks, how to guard the HPC system against these attacks becomes a challenging task. To the best of our knowledge, different defenses have been proposed to defend against a set of attacks, while there are no universal solutions.

When a server or cluster comes under attacks or once the attacker lands on the system, owners of user accounts often have no idea of the reality that they have been hacked. Most of the time, it is the activities of attackers expose or alert security officers of the HPC infrastructure. Behaviors of hackers tend to fall into some modes that are obviously different from the true owner of the account and can be easily identified. For example, the sudden burst of network activities, high CPU utilization, longer network latency or unauthorized jobs bypassing the job scheduler etc. Hence monitoring is an important part of the HPC system management, which is a main task for many defense mechanisms.

3. Defense Mechanisms

In this section, we give the background and defense strategies in HPC systems. We especially reviewed intrusion detection methods due to the importance in HPC security.

3.1. Background

Early studies on the security of HPC systems focused on the programming level. A software infrastructure is designed in [3] to ensure the integrity and confidentiality of communications and to authenticate approved users and resource owners. The developed security-enhanced communication library, named as Nexus, can be used to provide secure versions of popular communication libraries, such as Message Passing Interface (MPI), and offered a fine degree of control over what, when and where security mechanisms can be employed in HPC systems.

Traditionally, many physical measures and human-oriented rules have been suggested to ensure the security of HPC systems. For example, Korambath et al. in [11] listed strategies of protecting passwords and how to safeguard computing resources in an HPC environment. Common methods of preventing passwords from hacking include encrypting information exchange between users and resource owners, urging users to employ complicated passwords and changing the passwords routinely, monitoring activities of each user to identify possible attackers and limiting the access rights of each user according to their priority and so on.

Protocols like telnet and ftp are widely used in the 1990s in which clear text format information is transferred between remote computers. Anybody with reasonable expertise of the domain knowledge can launch such an attack, intercepting and reading the content. Now none of HPC sites will run these kinds of protocols. Besides that, a lot of solutions to malicious user detection and security risk reduction have been proposed. The approach used by most of nowadays HPC systems can be summarized as divide-and-conquer. i.e., the security techniques are deployed separately and independently to address the HPC security vulnerabilities [11].

3.2. Defense strategies

What to look for and where to identify the malicious attacker are a complicated cybersecurity problem. Not a single HPC suit, especially within the field of open science, has any solution to identifying and reducing all risks associated with attacks against the system infrastructure. Given the sophistication of attackers and the rapidly changing pace in computational systems and networking, how to develop defense mechanisms becomes a difficult obstacle to tackle with.

When HPC systems are under attacks, the attack activities often have two parts: the initial attack and the follow-up behaviors taken if local access is obtained. The initial attack can come from almost anytime and anyplace. Although we have firewalls in place, there is nothing the firewall can do once the attacker has already landed on the system. However, as of the attack property, the attacker's behavior tends to be somehow well-defined and can be identified easily. From the system administrators' perspective, what to look for and where to identify the malicious attackers are probably one of the most complex cybersecurity problems in nowadays HPC environment [8].

One the other hand, security issues in HPC systems differ from those in traditional platforms due to their distinctive program structures, computing environments and performance requirements. Unlike widely used client-server structure in traditional distributed systems, the communications in HPC systems mainly depend on two-sided message passing, streaming protocols, multicast and so on [3].

Existing technologies used to secure HPC infrastructures can be broadly categorized into four classes, i.e., they try to defend HPC systems from the following four aspects[10]: (i) basic OS hardening, (ii) authentication, (iii) network and host security and (iv) patching and auditing, in which OS hardening techniques can go deeper to be classified into different sub-classes. The detailed category information is shown in Fig.2. From which, password complexity enforcement can be done through enforcing strict well-defined security policies, for example, requiring users to change password periodically; Monitoring HPC facilities for abnormal status [26] can be one example of defenses under authentication category; As HPC facilities are usually used for distinctive purposes such as mathematical computations, they tend to have a much more regular and predictable mode of operations, which can be utilized to detect irregular and potential malicious attacks [25].

3.3. Intrusion detection

Intrusion detection is a well-known defense in HPC systems to counter malicious attacks [22]. To address the problem of where, when, what and how to identify the suspicious intruders, National Energy Research Scientific Computing Center (NERSC) [8] follows a methodology of data gathering and measurements, repeatable testing and careful analysis in designing the intrusion detection mechanism. The detailed steps of the method can be summarized as follows:

- Collecting as much raw data as possible, especially those data in high yield areas, and designing patterns that can be based on to evaluate the suspiciousness of user behaviors. The raw data can be accessed through accounting data or SSHD data from each host, batch scheduler logs from inter-systems or network data and DNS logs from cross-site.

- Cleaning, organizing and normalizing the collected data for the following analysis, which aim to process and reduce the volume of the collected data. The abstracted data can be further canonicalized to transform it into a normal form such that it can be suitable for machine processing use techniques such as machine learning algorithms.

- Employing appropriate tools or methods to analyze data processed in step 2 and comparing with expected or normal data. In this step, local site security policies can be used to evaluate the standardized data.

Based on the above methodology, NERSC introduced the Bro intrusion detection system [8]. To address the invisibility problem of activities happened on the multi-user HPC infrastructure, NERSC introduced an instrumentation layer into the OpenSSH application and then connected the resulted dataset into a real time analysis using Bro IDS.

To address various security issues at different layers of the cluster architecture, from threats of network to vulnerabilities of applications, another intrusion detection mechanism is proposed in [17]. In the mechanism, there exists an instrumented node to help gathering raw data and understanding the communications between clustering nodes in HPC infrastructure at various layers (i.e., applications layer, middleware layer, operating system layer and network layer). The collected data will be audited and analyzed to identify suspicious communications or behaviors.

Unlike the intrusion detection techniques listed above, which focus on detecting anomalies from behaviors or running statistics after the intruder has succeed partially in compromising the targeted system. User authentication is another effective method to prevent intruders or malicious attackers from landing the system at the beginning, which is an intuitive and straightforward method to stop intruders. However, this method usually has limited capabilities in keeping safe of HPC systems as many attackers can find paths to walk around the authentication gate [27, 28]. More existing literature of user authentication techniques and mechanisms used in distributed HPC systems can be found in [5, 15, 29].

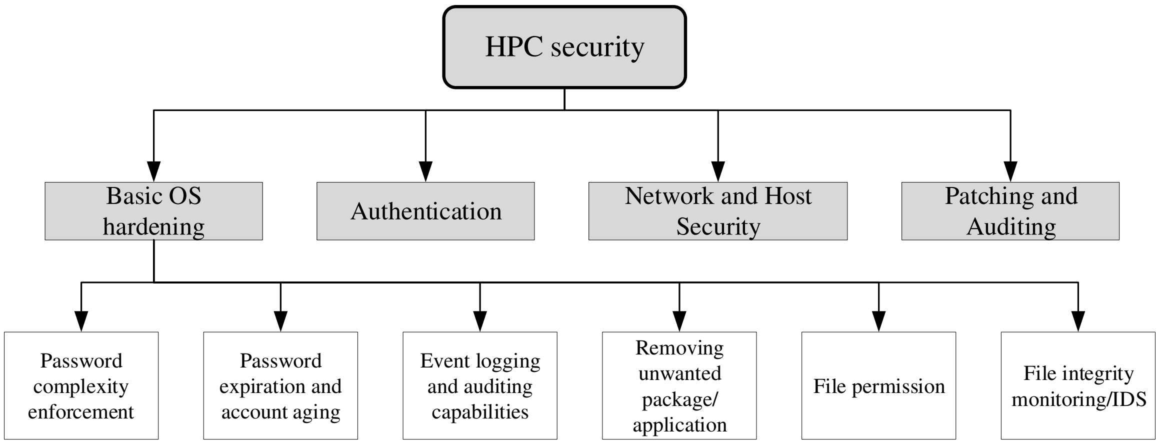

HPC security
Basic OS hardening
Authentication
Network and Host Security
Patching and Auditing
Password complexity enforcement
Password expiration and account aging
Event logging and auditing capabilities
Removing unwanted package/ application
File permission
File integrity monitoring/IDS

Figure 2. Strategies to secure HPC systems [10].

As machine learning has been widely spread into different domains, a more direct and effective strategy has come into people's mind to defend HPC systems, log file-based defense [25, 30, 31], which is a behavior detection method based on log files generated from running jobs. Using log files has many advantages than previous methods, as log files are a detailed record of the running process of the applications. Log files generated at different layers of the HPC systems often contains the fully operation and resource balance process. We give more information on log file-based intrusion detection in the following section.

4. Log-based Security Analysis in HPC Systems

In this section, we give the detailed information about the log file-based security analysis in HPC systems.

4.1. Background and related works

Log files can be found on almost all computer systems universally, which are text files recording behaviors, system status and other running statistics of the jobs active in HPC systems such that administrators can look back to debug system problems or track the running process to find vulnerabilities [32, 33]. Therefore, it can be used as the first and direct information source to monitor and detect intruders in HPC systems [25, 34].

Based on the collected information, the intrusiveness can be downgraded at each level of HPC systems and can even stop monitoring for the sake of efficiency and performance. For example, we can employ two phases in the instrumentation [25, 34]: fully monitoring and adaptive monitoring. Fully monitoring is usually employed at the beginning, in which everything is monitored to build the profile of an application or a user. It is costly in terms of computing resources; thus, it usually only runs a short of time. While adaptive monitoring can adaptively monitor the resources or variables according to the load of the instrumentation node. It can focus on some critical areas while have little impact on the overall system performance.

HPC systems usually can be decoupled into 4 layers [16, 17]: application layer, middleware layer (MPI and PVM, etc.), operating system layer and network layer. The high performance is achieved through the cooperation of each layer. Many of HPC systems are based on Linux-like systems. In Linux systems, user behaviors and operating system's activities are logged such that it can be processed and analyzed by administrators to monitor the system.

When the related raw data are collected by the dedicated instrumented node, all layers of HPC systems are instrumented heavily to evaluate the possibility of existence of intruders[18]:

- *Application layer:* Collect general variables of monitored processes, e.g., percentage of CPU time, memory used, I/O time and so on, to analyze statistics of applications run on HPC systems.
- *Middleware layer:* Audit the calls of messaging passing information, such as MPI calling statements and other middleware related communications.
- *Operating system and network layer:* Audit and monitor access to the file system and related network interfaces, such that the global communications or the local resource access information can be utilized and evaluated.

The general workflow of using log files to detect intruders in existing literature can be summarized and shown in Fig.3. Usually we have four steps: log collection, log parsing, filtering/feature extraction and anomaly/intrusion detection, to conduct log file-based analysis in defending HPC systems against intrusions or other malicious behaviors [18, 23, 25, 26, 35].

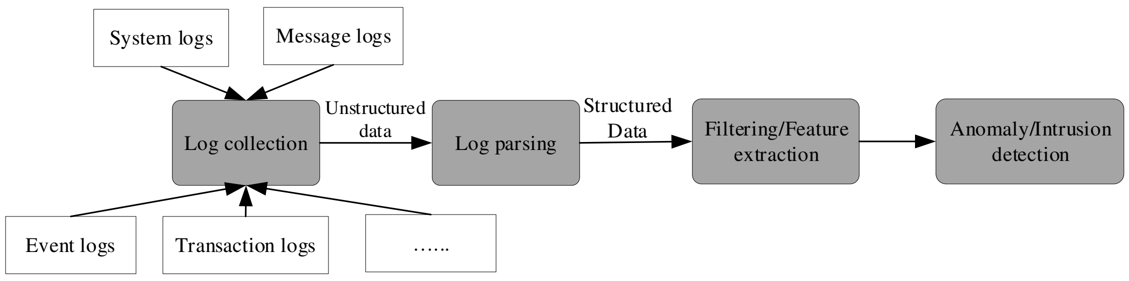


Figure 3. General workflow of using logs to detect intruders [18, 23, 25, 26, 35].

4.2. Log collection

Logs record the history of everything happened in HPC systems. In Linux systems, log files are stored in text form and usually under the directory of /var/log and its subdirectories. They include various information such as system, kernel, access control, package managers and many others. These log files can be roughly classified into four categories: application logs, event logs, service logs and system logs.

Large scale HPC systems generate various types of log files during running procedure of the jobs. For instance, the history of application run on the platform, resources allocated for them, sizes of each job, user information of each application and exit statuses are all logged in log files. Reliability, availability and serviceability (RAS) system logs are capable of extracting and logging data from various sensors both hardware and software, such as processor utilization, temperature sensors and memory errors [36]. Other log files like network system logs, input/output and storage system procedure logs can collect and record network bandwidth, congestion, and resource consuming information of jobs. The performance and running information inspection can rely on different log file (s) with respect to requirements of monitoring.

These log files provide a direct way for system managers to evaluate the system status. In HPC systems, log files can also be collected and analyzed to detect malicious jobs or users. For example, message log files in Linux systems show general messages and information regarding the system. It logs all activities throughout the global system. Secure logs keep authentication information of both successful and failed logins, and the authentication processes. Other logs can also be used to detect attacks and evaluate the system vulnerability.

Log collection is the first step in log analysis-based intrusion or anomaly detection of HPC systems. Distributed systems can continuously generate all kinds of log files to record system states and runtime statistics. This information is vital in examining system condition or debug especially when failures are encountered [37]. In log collection stage, it is important to collect right logs from the system as much as possible to pave way for the following analysis. When machine learning techniques are used to analyze the logs, the volume of the data is critical in learning the patterns of both normal behaviors and abnormal behaviors.

4.3. Log parsing

Log files are usually unstructured data, and their format and semantics might be different from each other, Thus, it is a difficult and complicated problem to design a mechanism that can diagnose abnormal or malicious intruders, especially when log files generated is in huge amount [38–41]. On the other hand, each log file might contain a lot of information that is not related to the security of HPC systems, thus how to filter valuable information from raw log files is also a challenging problem. It usually requires a lot of domain knowledge to design rule-based detectors [42–44]. For example, using the CPU time as a resource utilization measure, using IP address to parse a log into different entities and so on.

Log parsing is the second step of log analysis, which aims to get the unstructured, free-format and semantics text file into a structured representation. There have a substantial research and literature on the parsing of log files, in which representative methods can be listed as follows:

- Du et al. [45] provided an **online streaming method** to parse log files, which utilizes the longest common subsequence method, i.e., the longest common subsequence in logs are used as the sign to parse log files into structured files. The proposed method achieves linear time complexity for each log entry.

- Beschastnikh et al. [6] gave a method using **regular expressions** to determine which log lines will be parsed and which log lines will be ignored. It is a set of channel definitions that corresponds each line of the log files into a vector timestamp, or other channels.

- Xu et al. [46] offered a method leveraging the **source code** to parse log files, in which we need to first get all possible log message template strings from the source code and then match it to log files to parse them into structured files.

- Methods in [47, 48] provided solutions to parse log files purely based on **log characteristics** using data mining approaches. They usually do not rely on other information except log files, which is usually more straightforward but might not be as precise as those rule-based log parse methods, and they usually require machine learning algorithms to learn patterns.

After parsing log files obtained from HPC systems, intrusion/anomaly detection methods can be applied to distinguish out suspicious activities or users.

4.4. Log file-based intrusion/anomaly detection

There are various methods and mechanisms have been proposed to address the intrusion/anomaly detection problem utilizing log files form HPC systems. In this subsection, we first overview two representative types of log file-based defenses used in HPC systems. They both offered an experiment-proved mechanism to detect intrusions or anomalies:

The first perspective is to take log files as a **language model**, and then build relationships between log files and normal/abnormal behaviors. Log files, in essence, are existed in text format. Therefore, it can intuitively be modeled as a natural language sequence model, further it can employ techniques from natural language processing to analyze log files.

Inspired by the observation that log file entries are a sequence of events extracted from the structed source code execution process, Du et al. [25] proposed a deep neural network model employing Long Short-Term Memory (LSTM) to detect anomaly behaviors and potential malicious users through taking log files as structured language sequence. Based on the proposed model, which is named as DeepLog, log patterns of normal execution can be learned continuously from the historical data. Abnormal patterns resulted from intrusions, which are often deviates from patterns of log files from normal execution, can be distinguished by the learned model in monitoring process. DeepLog was designed in an incremental learning style, thus it can adapt to different patterns in the running process.

From a mathematical perspective, DeepLog builds a non-linear and high dimensional relationship between log entries and normal/abnormal execution applications. It categorizes log files into various sequences, thus a workflow model can be constructed for each separate task, and also offers a feedback mechanism, such that a wrongly classified log file can be used to adjust weights of the trained model to make it adaptively fit into its dynamic changing environment.

The second perspective is to build **relationships between log files and source codes**. Many works proposed to address security problems based on log files cannot identify code's behaviors. Inspired by the puzzle of whether the source code of an application or job are unique enough, such that it can be identified from the performance logs generated by the system, DeMasi et al. [18] employed and modified the rule ensemble method to predict what source code was running based on the generated performance log files. The Integrated Performance Monitoring (IPM) logs used in the paper are collected from a broad set of applications at the NERSC facilities.

Extensive works have shown various structured patterns in performance logs of HPC systems. Orianna et al. [18] proposed a method using the Rule Ensemble method to identify a code by its performance logs based on supervised machine learning. Through interpreting the resulting rule model, it can tell users which components of a code are the most distinctive and useful for identification. The proposed method can monitor the performance of applications running on HPC resources. Thus, the unapproved code or jobs or unintended usages of HPC systems can be detected.

5. Machine Learning in HPC and Future Work

In this section, we review the application of machine learning techniques used in HPC systems and potential future work is detailed.

5.1. Machine learning and HPC security

Besides the aforementioned intrusion detection methods, there proposed various advanced frameworks in recent years to mitigate security risks, especially as machine learning techniques are widely used in HPC systems.

For data centers like NERSC, there are hundreds, or thousands of jobs running concurrently on each of their four computing systems, i.e., Cori, Edison, PDSF and Genepool every day. Inspecting and analyzing each job manually is practically infeasible given the vast amount of log files generated simultaneously by each of the systems and the corresponding file systems. Given the huge volumes of log files, a mechanism named as Priolog is designed in [49] to narrow down the volume to comparatively small volumes of most related logs, thus increases the process efficiency.

Besides reducing the volumes of logs, another idea is to design scalable HPC system data analytics framework. [36] designed a scalable mechanism to analyze system logs, further, to distinguish unwanted or malicious jobs running on HPC systems. Based on the framework, users are able to navigate spatial-temporal event space that overlaps with specific system resource allocation, events, errors and identify persistent user behavior patterns etc. Further, users can distinguish performance anomalies and gain valuable insights about the impact brought up by various system jobs. Ultimately all above methods lead to machine learning techniques. Given the often huge amount of log files generated during the running stage (we say it "huge" because log files is usually generated continuously from various layers of the systems) [50], traditional data analysis tools and techniques often fall short of the capability of processing them efficiently and effectively. Machine learning techniques, such as deep neural networks, can offer exactly what traditional data analysis techniques fall short of. It can process large dataset of log files in a batch-processing style and can extract valuable information from them.

In log analysis, machine learning techniques have their unique advantages in dealing with large volumes of streaming data. Processing streaming data generated from HPC systems is challenging as a result of the large volumes and generating speed. An online supervised learning method is proposed in [51] to operate with live streamed data; Another online anomaly detection method using autoencoder is provided in [52]. To better support streaming logs analysis, a visual analytic framework is proposed in [53], which consists of data management, analysis and interactive visualization. It can automatically identify pattern changes and provide a coherent view of the changes and patterns of the performance data. [54] also builds a scalable visualization tool named as MELA to study event log data.

Other applications of machine learning in log file analysis can be found in [30, 55–58]. The application of machine learning techniques in log file analysis brings revolutionary to the intrusion/anomaly detection in HPC systems. Anomaly user detection is an important application of deep learning techniques for user behavior analysis. A comprehensive review of how the machine learning techniques are used in detecting anomaly users is given in [26].

5.2. Future work

The security of HPC systems will experience more challenges in the future especially as the widely applications of machine learning techniques, both in terms of attacks and defenses. Here we give several potential research topics as future work:

- There are various methodologies proposed to defend intrusion attacks in HPC systems based on log files. Through reviewing many of these methods, we found that most of the methods are focused on a small part of the log files, i.e., there still lacks a strategy that can utilize all the available logs that employing state-of-the-art machine learning techniques to study the security problem. We believe this would be a promising research frontier.
- Deep learning techniques are widely used in log file analytics to classify different types of users by their application behaviors. It is worth noting that different platforms will have different user behaviors and data types, thus lead to different feature extraction methods. The efficient and effective feature vector extraction methods are at the core of applying deep learning techniques in future work.
- If we look beyond security problems in HPC systems, we observed that the problem faced by other distributed systems, such as cloud computing, cluster computing and grid computing, are similar with the problems we faced in HPC systems. This is because they all have similar architectures and utilities. Security solutions in cloud computing, cluster computing, grid computing and many other distributed computing models [13, 28] can be transferred to the HPC systems due to their inner similarities and this will be a promising future research topic.

The security of HPC systems has drawn emerging attentions from both academia and industry. With the rapid growing of HPC systems in terms of both scale and complexity, machine learning based security inspection, evaluation and malicious behavior detection will play a more practical role in improving the usability and security of HPC systems.

6. Conclusion

We comprehensively reviewed the security problem in HPC systems from a log-analyzing perspective in this survey, including both the attacks and defense methods, especially the log file-based intrusion detection methods. We found that existing detection methods have focused only on a very small number of logs to detect malicious attacks. To improve the detection performance, we believe a more advanced machine learning, especially natural language-based processing techniques should be used in analyzing various logs of HPC systems. We also give the future research directions. How to build a framework that utilizes all the available log files from each level of HPC systems to evaluate the security will be one of the main future works.

References

[1] Pellerin, D., Ballantyne, D. and Boeglin, A. (2015) An introduction to high performance computing on aws. *Amazon Whitepaper* .

[2] Peisert, S. (2017) Security in high-performance computing environments. *Communications of the ACM* **60**(9): 72–80.

[3] Foster, I., Karonis, N.T., Kesselman, C. and Tuecke, S. (1998) Managing security in high-performance distributed computations. *Cluster Computing* **1**: 95–107.

[4] Ballew, J.D., Davidson, S.V. and Richoux, A.N. (2015), System and method for cluster management based on hpc architecture. US Patent 9,178,784.

[5] Barnell, M., Raymond, C., Capraro, C., Isereau, D., Cicotta, C. and Stokes, N. (2018) High-performance computing (hpc) and machine learning demonstrated in flight using agile condor®. In *2018 IEEE High Performance extreme Computing Conference (HPEC)* (IEEE): 1–4.

[6] Beschastnikh, I., Brun, Y., Ernst, M.D. and Krishnamurthy, A. (2014) Inferring models of concurrent systems from logs of their behavior with csight. In *Proceedings of the 36th International Conference on Software Engineering* (ACM): 468–479.

[7] Blanc, M., Briffaut, J., Gros, D. and Toinard, C. (2011) Piga-hips: Protection of a shared hpc cluster. *International Journal on Advances in Security Volume 4, Number 1 & 2, 2011* .

[8] Campbell, S. and Mellander, J. (2011) Experiences with intrusion detection in high performance computing. *Proceedings of the Cray User Group (CUG)* .

[9] Apostal, D., Foerster, K., Chatterjee, A. and Desell, T. (2012) Password recovery using mpi and cuda. In *2012 19th International Conference on High Performance Computing* (IEEE): 1–9.

[10] Bulusu, R., Jain, P., Pawar, P., Afzal, M. and Wandhekar, S. (2018) Addressing security aspects for hpc infrastructure. In *2018 International Conference on Information and Computer Technologies (ICICT)* (IEEE): 27–30.

[11] Korambath, P. (2014) Cyber security in high-performance computing environment. *Institute for Digital Research and Education* .

[12] Torres, M., Vaughn, R., Bridges, S., Florez, G. and Liu, Z. (2003) Attacking a high performance computer cluster. In *Proceedings of the 15th Annual Canadian Information Technology Security Symposium, Ottawa, Canada*.

[13] Sabahi, F. (2011) Cloud computing security threats and responses. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on* (IEEE): 245–249.

[14] Markowsky, G. and Markowsky, L. (2007) Survey of supercomputer cluster security issues. In *Security and Management*: 474–480.

[15] Schroeder, B. and Gibson, G. (2009) A large-scale study of failures in high-performance computing systems. *IEEE transactions on Dependable and Secure Computing* **7**(4): 337–350.

[16] Braby, R.L., Garlick, J.E. and Goldstone, R.J. (2003) *Achieving order through chaos: the llnl hpc linux cluster experience*. Tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

[17] Gadaud, F., Blanc, M. and Combeau, F. (2005) An adaptive instrumented node for efficient anomalies and misuse detections in hpc environment. In *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.* (IEEE), **1**: 140–145.

[18] DeMasi, O., Samak, T. and Bailey, D.H. (2013) Identifying hpc codes via performance logs and machine learning. In *Proceedings of the first workshop on Changing landscapes in HPC security* (ACM): 23–30.

[19] Curry, T.W. *et al.* (1994) Profiling and tracing dynamic library usage via interposition. In *USENIX Summer*: 267–278.

[20] Lee, J.K., Kim, S.J. and Hong, T. (2016) Brute-force attacks analysis against ssh in hpc multi-user service environment. *Indian Journal of Science and Technology* **9**(24).

[21] Cantu, M., Kim, J. and Zhang, X. (2017) Finding hash collisions using mpi on hpc clusters. In *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)* (IEEE): 1–6.

[22] Kumar, M. and Hanumanthappa, M. (2013) Scalable intrusion detection systems log analysis using cloud computing infrastructure. In *2013 IEEE International Conference on Computational Intelligence and Computing Research* (IEEE): 1–4.

[23] Ma, P. (2003) Log analysis-based intrusion detection via unsupervised learning. *Master of Science, School of Informatics, University of Edinburgh* .

[24] Mishra, B.K., Sahu, M. and Das, S.N. (2014) Intrusion detection systems for high performance computing environment. In *2014 International Conference on High Performance Computing and Applications (ICHPCA)* (IEEE): 1–6.

[25] Du, M., Li, F., Zheng, G. and Srikumar, V. (2017) Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (ACM): 1285–1298.

[26] He, S., Zhu, J., He, P. and Lyu, M.R. (2016) Experience report: system log analysis for anomaly detection. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (IEEE): 207–218.

[27] Prout, A., Arcand, W., Bestor, D., Byun, C., Bergeron, B., Hubbell, M., Kepner, J. *et al.* (2012) Scalable cryptographic authentication for high performance computing. In *2012 IEEE Conference on High Performance Extreme Computing* (IEEE): 1–2.

[28] Pourzandi, M., Gordon, D., Yurcik, W. and Koenig, G.A. (2005) Clusters and security: distributed security for distributed systems. In *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.* (IEEE), **1**: 96–104.

[29] Prabhakar, R., Patrick, C. and Kandemir, M. (2009) Mpisec i/o: Providing data confidentiality in mpi-i/o. In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (IEEE): 388–395.

[30] Gainaru, A., Cappello, F., Trausan-Matu, S. and Kramer, B. (2011) Event log mining tool for large scale hpc systems. In *European Conference on Parallel Processing* (Springer): 52–64.

[31] Park, B.H., Hui, Y., Boehm, S., Ashraf, R.A., Layton, C. and Engelmann, C. (2018) A big data analytics framework for hpc log data: Three case studies using the titan supercomputer log. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)* (IEEE): 571–579.

[32] Oliner, A. and Stearley, J. (2007) What supercomputers say: A study of five system logs. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)* (IEEE): 575–584.

[33] Oliner, A., Ganapathi, A. and Xu, W. (2012) Advances and challenges in log analysis. *Communications of the ACM* **55**(2): 55–61.

[34] Chuah, E., Kuo, S.h., Hiew, P., Tjhi, W.C., Lee, G., Hammond, J., Michalewicz, M.T. *et al.* (2010) Diagnosing the root-causes of failures from cluster log files. In *2010 International Conference on High Performance Computing* (IEEE): 1–10.

[35] He, P., Zhu, J., He, S., Li, J. and Lyu, M.R. (2017) Towards automated log parsing for large-scale log data analysis. *IEEE Transactions on Dependable and Secure Computing* **15**(6): 931–944.

[36] Park, B.H., Hukerikar, S., Adamson, R. and Engelmann, C. (2017) Big data meets hpc log analytics: Scalable approach to understanding systems at extreme scale. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)* (IEEE): 758–765.

[37] Tang, L., Li, T. and Perng, C.S. (2011) Logsig: Generating system events from raw textual logs. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (ACM): 785–794.

[38] Tan, J., Kavulya, S., Gandhi, R. and Narasimhan, P. (2010) Visual, log-based causal tracing for performance debugging of mapreduce systems. In *2010 IEEE 30th International Conference on Distributed Computing Systems* (IEEE): 795–806.

[39] Iosup, A., Ostermann, S., Yigitbasi, M.N., Prodan, R., Fahringer, T. and Epema, D. (2011) Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed systems* **22**(6): 931–945.

[40] Cinque, M., Cotroneo, D., Natella, R. and Pecchia, A. (2010) Assessing and improving the effectiveness of logs for the analysis of software faults. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)* (IEEE): 457–466.

[41] Ghiasvand, S., Ciorba, F.M., Tschüter, R. and Nagel, W.E. (2015) Analysis of node failures in high performance computers based on system logs. In *28th ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2015)* (University_of_Basel).

[42] Nakka, N., Agrawal, A. and Choudhary, A. (2011) Predicting node failure in high performance computing systems from failure and usage logs. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum* (IEEE): 1557–1566.

[43] Lin, X., Wang, P. and Wu, B. (2013) Log analysis in cloud computing environment with hadoop and spark. In *2013 5th IEEE International Conference on Broadband Network & Multimedia Technology* (IEEE): 273–276.

[44] Taerat, N., Naksinehaboon, N., Chandler, C., Elliott, J., Leangsuksun, C., Ostrouchov, G., Scott, S.L. *et al.* (2009) Blue gene/l log analysis and time to interrupt estimation. In *2009 International Conference on Availability, Reliability and Security* (IEEE): 173–180.

[45] Du, M. and Li, F. (2016) Spell: Streaming parsing of system event logs. In *2016 IEEE 16th International Conference on Data Mining (ICDM)* (IEEE): 859–864.

[46] Xu, W., Huang, L., Fox, A., Patterson, D. and Jordan, M.I. (2009) Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles* (ACM): 117–132.

[47] Fu, Q., Lou, J.G., Wang, Y. and Li, J. (2009) Execution anomaly detection in distributed systems through unstructured log analysis. In *2009 ninth IEEE international conference on data mining* (IEEE): 149–158.

[48] Makanju, A.A., Zincir-Heywood, A.N. and Milios, E.E. (2009) Clustering event logs using iterative partitioning. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM): 1255–1264.

[49] Tak, B., Park, S. and Kudva, P. (2019) Priolog: Mining important logs via temporal analysis and prioritization. *Sustainability* **11**(22): 6306.

[50] Sîrbu, A. and Babaoglu, O. (2015) A holistic approach to log data analysis in high-performance computing systems: The case of ibm blue gene/q. In *European Conference on Parallel Processing* (Springer): 631–643.

[51] Netti, A., Kiziltan, Z., Babaoglu, O., Sîrbu, A., Bartolini, A. and Borghesi, A. (2019) Online fault classification in hpc systems through machine learning. In *European Conference on Parallel Processing* (Springer): 3–16.

[52] Borghesi, A., Libri, A., Benini, L. and Bartolini, A. (2019) Online anomaly detection in hpc systems. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (IEEE): 229–233.

[53] Kesavan, S.P., Fujiwara, T., Li, J.K., Ross, C., Mubarak, M., Carothers, C.D., Ross, R.B. *et al.* (2020) A visual analytics framework for reviewing streaming performance data. *arXiv preprint arXiv:2001.09399* .

[54] Shilpika, F., Lusch, B., Emani, M., Vishwanath, V., Papka, M.E. and Ma, K.L. (2019) Mela: A visual analytics tool for studying multifidelity hpc system logs. In *2019 IEEE/ACM Industry/University Joint International Workshop on Data-center Automation, Analytics, and Control (DAAC)* (IEEE): 13–18.

[55] Chen, Y.T.W.Y., Kuo, W.C. and Wang, Y.T. (2009) Building ids log analysis system on novel grid computing architecture. *National Center for High-Performance Computing* .

[56] Ghoshal, D. and Plale, B. (2013) Provenance from log files: a bigdata problem. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops* (ACM): 290–297.

[57] He, P., Zhu, J., He, S., Li, J. and Lyu, M.R. (2016) An evaluation study on log parsing and its use in log mining. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (IEEE): 654–661.

[58] Hu, W., Li, Y., Srihari, V. and Yemeni, R. (2013), High-performance log-based processing. US Patent 8,566,326.