All your cluster-grids *are belong to us*: Monitoring the (in)security of infrastructure monitoring systems

Andrei Costin
EURECOM
Sophia Antipolis, France
andrei.costin@eurecom.fr

***Abstract*—Monitoring of the high-performance computing systems and their components, such as *clusters*, *grids* and *federations of clusters*, is performed using *monitoring systems for servers and networks*, or *Network Monitoring Systems (NMS)*. These monitoring tools assist system administrators in assessing and improving the health of their infrastructure.**

A successful attack on the *infrastructure monitoring tools* grants the attacker elevated power over the monitoring tasks, and eventually over some management functionality of the interface or over hosts running those interfaces. Additionally, detailed and accurate fingerprinting and reconnaissance of a target infrastructure is possible when such interfaces are publicly exposed. A successful reconnaissance allows an attacker to craft an efficient second-stage attacks, such as targeted, mimicry and blended attacks.

We provide in this paper a comprehensive security analysis of some of the most popular infrastructure monitoring tools for grids, clusters and High-Performance Computing (HPC) systems. We also provide insights based on the infrastructure data *openly exposed* over the Internet. The wide use of some of the most popular infrastructure monitoring tools makes this data exposure possible. For example, we found such monitoring interfaces to expose infrastructure details of systems inside many high-profile organizations, including two top national laboratories for nuclear research and one top Internet non-profit foundation. We also present our findings on *a plethora of web vulnerabilities* in the entire version-span of such monitoring tools, and discuss at a high-level the possible attacks. The results of our research allow us to "monitor" an "alarming" mismanagement reality of grid infrastructure. The aim of this work is to raise the awareness about this novel risk to cloud infrastructure.

I. Introduction

Cloud computing is a hot topic at the moment and it is unsurprising that significant research is done in the direction of security [28], [37], [43]–[45], [47] and privacy [29], [40] of cloud computing. However, as Idziorek points out [27]: *"When broken down, cloud computing is a specialized distributed computing model. Building upon the desirable characteristics of* ***cluster, grid, utility****, and service-oriented computing, cloud computing introduces a unique complement of features to create a new computing paradigm"*. This fundamentally means that both the security and the privacy of the Cloud Computing (CC) and High-Performance Computing (HPC) platforms strongly depend on the security and the privacy of the grids, clusters and nodes on which those CC and HPC platforms build upon. As a consequence, it is vital to secure the clusters and the grids as building blocks of these new computing paradigms. This is especially true from the perspective of the web applications used to administer grids, clusters and HPC systems. It is also important to research and understand the ways these interfaces may be compromised or abused, and study how additional breaches can be prevented.

At the same time, it is long and well known that it is not trivial to secure the web applications. Infamous vulnerabilities, such as Cross Site Scripting (XSS) [50], represent an important percentage of the vulnerabilities discovered each year [21], and are used very often in real-world attacks [23]. Additionally, information leakage or fingerprinting vulnerabilities are prevalent as well [52] and cannot be completely ignored.

a) Overview: We research web vulnerabilities and information leakage risks associated with *Ganglia*, *Cacti* and *Observium*, some of the most popular and widely used monitoring platforms for servers and networks. We show that such monitoring web interfaces are highly vulnerable due to numerous flaws we found in their code. We also show that systems and networks managed using these tools (and similar tools as well) are overly exposed to the public (e.g., on the Internet) and leak important details about the infrastructure. Both these threats combined result in a high risk of targeted attacks. Such attacks can lead to successful intrusions that have very high chances of evading the intrusion detection.

For example, as we demonstrate and discuss in Section III, we found Ganglia being used by far many more high-profile organizations, including two top national laboratories for nuclear research and one top Internet non-profit foundation (Section III-B). As we will show, these tools have a wide usage and can be found even inside very important environments. However, the security and privacy risks posed by these tools have not been systematically and well studied in the past.

b) Datasets: Our results and analysis are based on several datasets. First, we collected data [1] regarding online presence of around 364 Ganglia, 5K Cacti and 2K Observium infrastructure

[1]Data was sampled at multiple points in time between 01-Jan-2015 and 20-May-2015.

monitoring web interfaces. Second, we collected entire version-spans of these tools and analyzed them for web interface vulnerabilities. The tools we analyzed are: 25 Ganglia versions, 35 Cacti packages, 4 Job Monarch releases, and 1 Observium version.

c) Contributions: Our contributions are as follows:

- We are the first to systematically analyze at large scale the risks and vulnerabilities posed by the use of web monitoring tools for grids, clusters and HPC systems.
- We collect and analyze the internal details of networks and systems of a large number of grid and cluster environments. We also present the risks of such data being openly available to the large public in general, and to potential attackers in particular.
- We reveal multiple vulnerabilities in *Ganglia*, *Cacti* and *Observium* – some of the most popular and widely used NMS tools for grids, clusters and HPC systems.

d) Paper Outline: This paper is organized as follows. Section II presents an overview of grids, clusters and infrastructure monitoring tools. In Section III and Section IV we discuss the information leakage problem, analyze systems exposed online and present our findings on vulnerabilities, along with the security impact of all these findings. In Section V we discuss possible attacks and countermeasures. Section VI presents ethical aspects of our work. We explore the related work in Section VII and conclude with Section VIII.

II. Overview of Grids, Clusters and Infrastructure Monitoring Tools

Grids, clusters and HPC systems are essentially large-scale distributed infrastructures of servers and networks. Managing and monitoring those servers and networks is not a trivial task. For such tasks, specialized management and monitoring software exists. There are many popular free and open-source tools to monitor servers and networks, such as Monit [8], Munin [9], Nagios [10], Zabbix [16], Zenoss [17], Collectd [4], Argus [2]. Finally, Ganglia [6], [36], Cacti [3] and Observium [11] are some of the most popular and widely used NMS platforms.

A. Ganglia Summary

Ganglia is *"a scalable distributed monitoring system for high-performance computing systems such as clusters and grids"* [6]. It is being used in various ways in many different large-scale clusters, including systems *"for advanced applications in scientific computing, simulation, and modeling"* [36] (Millennium HPC Cluster [42]), as well as in *"the acceleration of cancer research, specifically investigation into the human genome, bioinformatics, protein structure prediction, and large-scale computer simulations"* [36] (SUNY Buffalo HPC Cluster [14]). Ganglia is officially used by more than several dozens of high-profile enterprises and organizations [2].

[2]http://ganglia.info/#text-9, "Who uses Ganglia?" section

a) Ganglia Monitor Daemon (`gmond`): The Ganglia Monitoring Daemon (`gmond`) is running on every cluster node that is configured for Ganglia monitoring. Its main function is to monitor the state changes in the monitored host and subsequently to announce the relevant changes. Finally, it listens to all other Ganglia nodes for their states and responds the queries with an XML-encoded information about the cluster state. By default it listens on port 8649 and the wide accessibility of this port exposes the nodes to new information leak risks.

b) Ganglia Meta Daemon (`gmetad`): The Ganglia Meta Daemon (`gmetad`) gathers the monitoring information from connected data sources, such as other `gmond` and `gmetad` instances. It then saves this information to local databases and exports the concatenation of all data sources as XML. The Ganglia Meta Daemon works as a backend for the Ganglia Web Frontend. It aggregates the historical information and can export XML-encoded summaries. These summaries can then be used by the web interface to display useful snapshots and present evolutions for all the nodes that Ganglia monitors. By default it listens on port 8652 and the wide accessibility of this port exposes the entire cluster to new information leak risks.

c) Ganglia Web Frontend (`ganglia-web`): The Ganglia Web Frontend presents a real-time snapshot of the information collected by the `gmetad` daemon on which it depends. It displays in a meaningful way the monitoring data to the administrators of the infrastructure. For example, it can graph and display the resource utilization (e.g., CPU, memory, storage, network) over a past period of time (e.g., year, month, week, day, hour).

d) Ganglia Remote Execution Daemon (`gexecd`): Ganglia's `gexecd` is a scalable cluster remote execution system. Its purpose is to provide remote execution of distributed jobs in a fast and cryptographically-authenticated way. It is designed to scale on infrastructures containing thousands of nodes, and it does so in a highly robust manner. By default `gexecd` runs on port 2875 and the wide accessibility of this port exposes the entire cluster to new risks of intrusion (e.g., buffer overflows, unauthorized remote command execution).

e) Job Monarch (`jobmonarch`): Job Monarch, which stands for "Job Monitoring and Archiving", provides batch job monitoring and archiving, as well as a graphical overview of the clusters and assorted batch systems. It is an add-on for Ganglia and is usually accessible within Ganglia's URL (e.g., http://host/ganglia/addons/job_monarch).

B. Cacti Summary

Cacti is *"a complete network graphing solution"* [3]. It is also widely used in clusters and HPC systems similar to the ones managed by Ganglia. For example, Cacti is used in seismic monitoring networks [35] and in traffic monitoring of campus networks [26].

C. Observium Summary

Observium is *"an autodiscovering network monitoring platform supporting a wide range of hardware platforms and operating systems including Cisco, Windows, Linux, HP, Juniper,* *Dell, FreeBSD, Brocade, Netscaler, NetApp and many more. Observium seeks to provide a powerful yet simple and intuitive interface to the health and status of your network."* [11].

III. Infrastructure Information Leaks and Exposed Online Systems

A. Ganglia Info Leaks

In this section we argue and show that information leakage from Ganglia systems present a serious threat. Our hypothesis is also indirectly confirmed by the fact that various vulnerability scanners already integrate modules to detect and alert on such information leaks [3]. There are several advantages for an attacker to (ab)use information leakage from infrastructure monitoring systems. First, collecting this information is minimally intrusive compared to other approaches. For example, crawling of the exposed monitoring interfaces can be throttled and disguised as web search engine spiders. Second, contrary to other side-channel techniques to fingerprint the computing architecture and the OS/kernel version (e.g., NMAP's stack-based probing), the information provided by Ganglia and its `gmond` daemon is extremely accurate [4]. This makes the reconnaissance based on Ganglia's reports to be very precise and thus highly valuable from an attacker viewpoint. A successful and very accurate reconnaissance allows an attacker to craft efficient second-stage attacks (e.g., mimicry, blended, targeted [39], [46], [55]), and can help a successful intrusion to evade detection.

a) Overview of exposed online systems: Using Shodan and Google Search, we were able to find at least 364 Ganglia web interfaces which are openly accessible to the public. These interfaces overall manage around 43K hosts, configured into 1370 clusters, which are in turn partitioned into 490 grids. In Table I we summarized the geographical distribution of those web interfaces and the clusters they manage.

b) Distribution of OS kernel versions: We analyzed OS kernel details of the 43K hosts under the management of the 364 Ganglia web interface we found. These hosts run at least 411 kernel sub-version based on 120 main kernel versions. The most popular main kernel version is `2.6.32`, containing at least 1600 vulnerabilities [5] and running on 38% of the reported host. Further analysis of OS kernel versions revealed that only 9 hosts are running `grsecurity` [7] enhanced kernels and only 6 hosts have kernels built from `hardened-sources` repository. Additionally, we found 45 hosts running kernels with `amzn` tag, which prompts us to conclude that such hosts are part of some Amazon Web Services (AWS) cloud computing setup. This is not a surprise, since in some cases top IT&C companies deployed some of theirs *production* Ganglia monitoring nodes onto AWS and made them publicly available. On the same note, we discovered that a top national nuclear research laboratory has its own kernel distribution which runs on at least 1696 hosts. Finally, to our surprise, we found 16 hosts running on ARM platforms and 4 hosts running on PowerPC processors.

c) Distribution of HTTP and HTTPS usage: We also analyzed how the HTTP and HTTPS usage is distributed among those 364 Ganglia web interfaces discovered online. Unsurprisingly, 322 of those interfaces run over plain HTTP, while the rest of 42 run over various HTTPS setups. However, to our surprise around 26 interfaces (i.e., 62%) are running *untrusted HTTPS* setups (e.g., self-signed, wrong domain, expired) and only 16 are running trusted HTTPS configurations [5]. Finally, it is surprising that while simple countermeasures to these insecure configurations exist (Section V-C), they are not implemented. This confirms that the monitoring of grids and clusters is not considered to pose a serious threat to organizations. It also confirms that infrastructure monitoring systems suffer from very poor secure management oversight.

Below we discuss several other Ganglia's components that can expose the infrastructure to additional risks.

d) Ganglia Monitor Daemon (`gmond`): We ran a query on Shodan [12] for the `gmond` nodes and present the summary of results in Table I. In fact, we found around 40K public IP addresses that expose this port and serve the XML descriptions of their clusters. For an attacker, the advantage of using the XML from the `gmond` is that it can be parsed and processed easier and more reliably than the HTML pages crawled from the Ganglia Web Frontend.

TABLE I
Distribution and counts of unique hosts, split by Ganglia's module and country of hosts' IP.

Country (iso2 code)	**Ganglia Gmond**	**Ganglia Web Frontend**
US	51%	32%
CN	10%	4%
KR	8%	8%
ES	6%	3%
FR	4%	3%
TW	3%	7%
DE	3%	3%
IT	$\approx 1\%$	3%
CH	$\ll 1\%$	5%
Others	14%	32%
Total (count)	39553	364

e) Job Monarch (`jobmonarch`): Job Monarch can reveal usernames and command lines of distributed programs being run along with their parameters. While the former can pose a privacy risk or expose particular accounts to targeted attacks, the latter can reveal types of programs usually being run and potentially sensitive details passed as arguments, such as credentials, URLs, paths. Using Google Search we were able to confirm there are around 50K search results [6] likely providing data from `jobmonarch` interfaces.

[3]Ganglia Cluster Report Information Disclosure Plugin ID 2492 – http://www.tenable.com/pvs-plugins/2492

[4]`gmond` is a trusted daemon running on the Ganglia monitored nodes.

[5]We did not check if those suffer from other SSL or TLS vulnerabilities.

[6]This is not a count of unique hosts. Triage and deduplication is required, and is left as future work.

B. Ganglia Info Leak – Case Study

We found an openly accessible Ganglia web interface monitoring the grid of a large non-profit Internet organization. Many online projects and some software packages depend on resources and APIs provided by this organization. Surprisingly, Ganglia monitoring interface of this organization is largely open to the public. We discovered that this organization's infrastructure consists of 54 different clusters having specific function (e.g., application API, load balancers, multimedia processing), totaling 1K hosts and summing up around 25K CPUs. Additionally, the hosts within this particular grid run 51 different kernel sub-versions based on 5 different kernel main versions [7]. The kernel sub-version `3.13.0-24-generic` is deployed on nearly 50% hosts within this grid.

C. Cacti and Observium Info Leaks

Shodan and Google Search queries returned around 5K Cacti and 2K Observium online web interfaces. Random sampling showed that more than 80% of them are password protected.

IV. Vulnerability Analysis

A. General Overview

a) Static Analysis: Static analysis is the process of testing an application by examining its source code, byte code or application binaries for conditions leading to a security vulnerability, without actually running it. This approach to testing has many practical benefits since the tools are often automated and do not require complex setups. Usually static analysis tools require only the source code or the binaries to be provided, and subsequently generate analysis reports. At the same time, static analysis techniques have well known limitations. They cannot find all the vulnerabilities, resulting in a number of missed vulnerabilities, i.e., *False Negatives (FN)*. They also tend to alert on non-vulnerabilities, i.e., *False Positives (FP)*.

Up to date PHP is still the most popular server-side programming language for the web [13], [15]. Due to this fact a number of state of the art static analysis tools for PHP exist [22], [31]. Fortunately, from the code analysis point of view, many of the infrastructure monitoring tools, such as Ganglia, Cacti and Observium, are (partially) written in PHP. We take advantage of this and perform a comprehensive static analysis using RIPS [22], which is a state of the art and open source static analysis tool for PHP.

b) Dynamic Analysis: Dynamic analysis is the process of testing the application by running it, and it comes with many benefits. Firstly, the dynamic analysis of the web applications is in general independent from the server-side language of the web application. Secondly, the dynamic analysis can be applied to confirm (part of) the vulnerabilities detected during the static analysis. Finally, dynamic analysis tools provide more accurate analysis results and generally perform a more qualitative validation of the reported vulnerabilities, as opposed to many static analysis tools.

[7]Versions `2.6.32, 3.13.0, 3.16.0, 3.19.0, 3.2.0.`

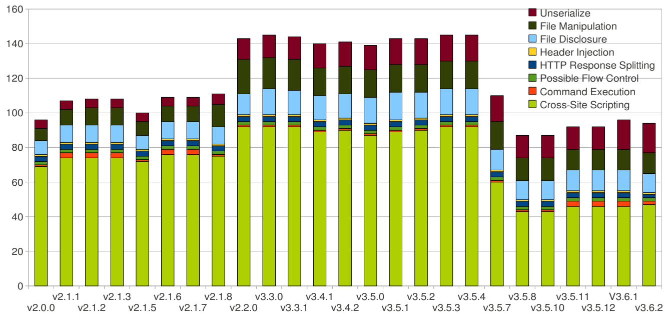

Fig. 1. Vulnerabilities in Ganglia Web Frontend found statically with RIPS. Distribution by Ganglia's version and vulnerability type.

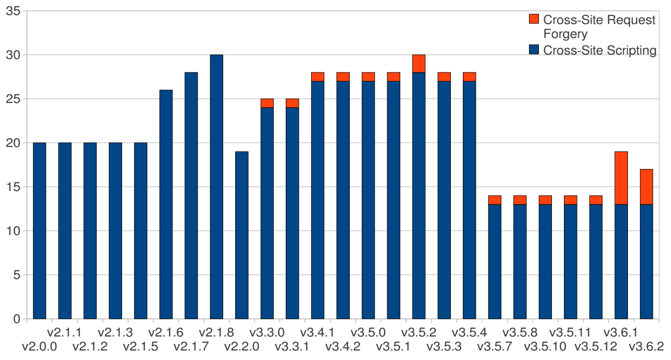

Fig. 2. Vulnerabilities in Ganglia Web Frontend found dynamically with Arachni. Distribution by Ganglia's version and vulnerability type.

There are many dynamic analysis tools to test the security of web applications [18]. For our experiments we used Arachni [1], which is an open source framework written in Ruby. It is designed for penetration testing of web applications and in our experiments we found it to perform well and fast.

c) General Observations: Even though Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) vulnerabilities are usually not considered to be critical, they can have a high impact. Johns [30] discussed in-depth the risks and impact of XSS and CSRF attacks. XSS is an important attack vector in many attack types, as it is an actively used "vehicle" which delivers and deploys the attack payload with no user intervention. This fact is particularly relevant in blended attacks, because those attacks require multiple steps to trigger the exploit and deliver the payload.

B. Vulnerabilities in Ganglia Web Frontend

The Ganglia Web Frontend is written in PHP, therefore we performed static analysis on it using RIPS. The results are summarized in Figure 1. For a more detailed distribution of vulnerabilities for each Ganglia Web Frontend version see Table II.

We also deployed Ganglia on a local testing host and performed dynamic analysis using Arachni. The results are summarized in Figure 2.

a) General observations: An important observation is the strong correlation of vulnerability graphs resulting from both static (Figure 1) and dynamic analysis (Figure 2). This also confirms the intuitive level of (in)security for each version.

The correlation clearly shows that versions between `v3.3.0` and `3.5.4` (both inclusive) have more potential (static analysis) and actual (dynamic analysis) vulnerabilities than other versions. A possible explanation could be that version `v3.3.0` was a rewrite of the previous baseline version (i.e., `v2.2.0`) and potentially introduced new flaws. It also clearly indicates that grids and clusters running versions between `v3.5.7` and `v3.6.2` (both inclusive) are exposed to less security vulnerabilities compared to those running any other previous version. It is safe to assume that migrating to one of these version makes it both a desirable and a reasonable choice. At the same time, the graphs show that the latest versions do not necessarily mean "the most secure". We can see that both latest versions, i.e., `v3.6.1` and `v3.6.2`, have more potential *command executions* (static analysis) and actual CSRFs (dynamic analysis).

Finally, to date of this submission Ganglia has very few (only four to this date) CVEs assigned [8]. This is in very high contrast to our static and dynamic analysis charts presented in Figure 1 and Figure 2 respectively. While many of the static analysis findings have to go through additional scrutiny, we believe that Ganglia's application security is not enough evaluated, as we have shown.

b) CVE-2012-3448: Ganglia's web frontends prior to version `v3.5.1` contain an unspecified *Remote Code Execution (RCE)* vulnerability. It allows remote attackers to execute arbitrary PHP code via unknown attack vectors. No other details were published about this vulnerability, except that ASAP actions are urged, such as an upgrade to version `v3.5.1` or a password protection mitigation at least [9]. This particular type of "incomplete" security advisory is a sword with two edges. On the one hand, it prevents script-kiddies from easily exploiting this vulnerability on critical systems by not sharing too many details on the vulnerability. On the other hand, it prevents the system administrators to properly configure their IDS and IPS systems to defend against this particular RCE vulnerability.

We performed a `diff` analysis between version `v3.5.1` and `v3.5.0` and isolated the root cause. It is a call to PHP's `eval()` function that could take potentially unsanitized input from the user via the some HTTP GET parameters, e.g., `graph`. The exploitable vulnerability relates to the following fragment of code [10]:

```
...
$graph_arguments = NULL;
$pos = strpos($graph, ",");
if ($pos !== FALSE) {
  $graph_report = substr($graph, 0, $pos);
  $graph_arguments =
    substr($graph, $pos + 1);
  $graph = $graph_report;
}
...
eval('$graph_function($rrdtool_graph,' .
    $graph_arguments . ');');
...
```

This analysis allowed us to create a proof-of-concept exploit for a simplified attack scenario. Surprisingly, our data shows that there are around 193 unique hosts on the Internet (i.e., 53% from those we crawled), that are running Ganglia Web Frontend interface with a version number prior to `v3.5.1` (i.e., *may* be vulnerable to CVE-2012-3448).

C. Vulnerabilities in Job Monarch

Job Monarch web interface is written in PHP and partially uses PHP APIs exposed by the Ganglia core that hosts the add-on. We used RIPS to statically analyze 4 versions of Job Monarch web interface. Table II presents the detailed distribution of vulnerabilities for each Job Monarch version. As can be seen, there are not many vulnerabilities reported for any of the Job Monarch version. The only issues that look interesting are XSS and *command execution*. After a quick manual check, it seems that the only exploitable vulnerabilities are from XSS category. We leave dynamic analysis and the confirmation of the issues' exploitability as future work.

D. Vulnerabilities in Cacti

Cacti's web interface is written in PHP and we used RIPS to statically analyze 35 versions of it. Table II presents the detailed distribution of vulnerabilities for each available version of Cacti. In contrast with Ganglia, Cacti has a larger documented list of vulnerabilities (31 to this date) [11]. It clearly shows that XSS vulnerabilities are a problem for any Cacti version. In fact, nearly more than 100 XSS vulnerabilities are statically reported for each version. At the same time, the total number of XSS static alerts for all Cacti versions approaches an impressive count of 6K. Another interesting observation is that starting from version `cacti-0.8.7h` the number of statically reported *code execution* vulnerabilities spiked. This might be due to new functionality being added, such as one based on `eval()` function or similar. A version `diff` between `cacti-0.8.7g` and `cacti-0.8.7h` could help find the root cause of the alerts, and we leave this for future work.

Finally, although each version's total number of *code execution* and *command execution* on average is not very high, it could be still beneficial to perform dynamic analysis of Cacti instances. Even though we did not perform dynamic analysis of Cacti web interfaces, we plan this as future work. In addition to the simple dynamic analysis, it would be also interesting to see how many of the Cacti's known CVEs can be actually found using automated tools such as Arachni.

[8]http://cvedetails.com/product/22764/Ganglia-Ganglia-web.html?vendor_id=1473

[9]http://ganglia.info/?p=549

[10]Simplified and shortened for clarity.

[11]http://www.cvedetails.com/vendor/7458/Cacti.html

E. Vulnerabilities in Observium

Observium's web interface is written in PHP and we used RIPS to statically analyze it. We tested only the latest Community Free version that was available for download. While we did not find any CVE entries publicly reported for Observium, our static analysis tests reported around 80 potential vulnerabilities. Among these, XSS prevails with around 50 vulnerabilities, while *command injection* scores only 2 vulnerabilities. Table II presents the detailed distribution of static analysis vulnerabilities for the latest available version of Observium. We plan to address dynamic analysis of Observium in future work.

V. Attack Scenarios and Countermeasures

Below we describe some attack scenarios which are possible when details from the infrastructure monitoring tools are publicly exposed. These scenarios can be used by highly motivated attackers for the purpose of data theft and covert intelligence gathering in public infrastructures (e.g., clouds, clusters, grids).

A. Mimicry and Blending Attacks

Mimicry [34], [51] and blending [24] attacks are well established directions in offensive and defensive [25] security research. Researchers have shown it is possible to use these types of attack to evade Intrusion Detection Systems (IDS). One way to achieve this is for an attack to blend with normal network traffic [33]. Alternatively, the attacks can be crafted to blend with normal program flows and instructions [32], [48].

Therefore, in order to increase their chances of mimicry or blending attacks, the attackers can follow these steps to abuse the information leaks from infrastructure monitoring systems:

1) The attackers start by silently monitoring and collecting the data of resources activities. This provides the attackers additional side-channel information about the allocation and activity of computing resources, such as network traffic, CPU usage, memory and storage (Section III). For example, it was shown that a particular pattern of resources usage (e.g., CPU, memory, storage, network) can be associated with particular processes or activities running on a computing platform [43], [56].
2) The attackers then use the data learned from the monitoring tools to build that knowledge into advanced automated mimicry attacks.
3) Subsequently, the attackers perform malicious activities within the attacked infrastructure (e.g., penetrate further systems and resources). They can also perform attacks towards outside resources from within penetrated infrastructure (e.g., DDoS from penetrated infrastructure to a DDoS victim).
4) Finally, by using blending and mimicry attacks, the malicious activity "blends" into the general activity pattern. This in turn provides the attack with higher chances to escape Intrusion Detection Systems (IDS) based on abnormal activity patterns.

B. Targeted Attacks

Similarly to mimicry and blending attacks (Section V-A), an attacker can (ab)use the infrastructure details gathered from openly accessible infrastructure monitoring systems (Section III) in order to target particular systems. For example, this can be done to increase the chances of a successful infrastructure penetration [12], or to target the infrastructure of a particular organization.

The attackers can perform any combination of the below activities in order to increase their penetration success rate:

1) The attackers identify of kernel versions that are most likely to be exploitable in a particular infrastructure. For example, Ganglia's `Operating System Release` metric can be used for this purpose. Also, vulnerable-looking kernel versions that have security hardening or custom patches (e.g., `grsecurity`) can be filtered out.
2) The attackers then identify the reachable IP addresses [13] of the hosts running vulnerable kernel versions (above).
3) The attackers try to penetrate the targeted infrastructure. For example, the attackers can send a phishing email to the administrator linking to an XSS-vulnerable page of the monitoring interface (Section IV). Alternatively, the attackers can use other ways to deliver the exploits to the hosts running exploitable kernels.
4) Finally, potential vulnerabilities in Internet facing daemons (e.g., `gmond`, `gmetad`, `gexecd`) can be used to penetrate the targeted infrastructure or to achieve privilege escalation on those hosts.

C. Countermeasures

There are several methods to perform monitoring while increasing the resilience of monitoring systems against integrity and confidentiality attacks. For example, monitoring tasks can be performed by a separate network of distributed non-dedicated nodes which use zero-knowledge protocols [38].

Complementary to such methods, there are also simple steps that are easy to implement and that can immediately increase the security and privacy of existing deployments. First, it is important to enable password authentication on any part of the monitoring interfaces. For the web frontend, a tool-independent solution is to use HTTP based authentication, e.g., Digest Access Authentication (DAA). An additional level of protection can be achieved by implementing authentication mechanisms based on HTML forms which can be additionally enforced with CAPTCHA. Second, it is advised to reconfigure the monitoring web interfaces from HTTP to HTTPS-only. Of course, the use of untrusted or self-signed HTTPS certificates is discouraged [14]. Also, the use of proper CA-signed certificates and non-vulnerable TLS/SSL implementations is necessary for this countermeasure to be effective. Third, whenever possible, it is a good practice to reduce the exposure to the Internet

[12] Target infrastructures having hosts that run exploitable old kernels.

[13] Public or private IP addresses, depends on the attack entry point.

[14] MITM HTTPS attack tools exist, e.g., https://mitmproxy.org/doc/howmitmproxy.html

of these monitoring web interfaces and daemons. The exposure of devices and web interfaces to the Internet is often a misconfiguration, where exposing those interfaces only to certain LAN segments is often sufficient. If exposure to the Internet is necessary, an IP-based Access Control List (ACL) could help increase the resistance of these interfaces to attack tentatives. Finally, keeping the software up-to-date is of utmost importance. This includes both the infrastructure monitoring tools as well as kernel and operating systems distributions. In addition, security hardening of software is advised whenever possible, such as using `grsecurity` security enhancement to the Linux kernel.

VI. Ethical Considerations

For this research we were careful to work within ethical and legal boundaries. We follow the *responsible disclosure* policy and do our best to notify the vendors for the vulnerabilities we discovered and confirmed during our experiments. We perform the static and dynamic analysis on private hosts within our lab and no attempt was made to try the exploits on live online systems. At the same time, we perform our data gathering experiments under the assumption that a service providing public access is an implicit authorization to do so. In other words, we access the same amount of publicly available infrastructure data that is accessible to services such as Shodan and Google Search. In fact, most of the data we analyzed comes from services like Shodan and Google Search. Finally, it is important to emphasize that we made no attempt to hack or disrupt the networks and systems within the infrastructures we analyzed. Also, we made no attempt to acquire or exploit non-public information within any of those exposed infrastructures.

VII. Related Work

a) Security and Privacy for Grids, Clusters and HPC: Pourzandi et al. [41] investigated and presented an overview of the security challenges as they apply to clusters. The authors proposed a systematic approach to address the security of clustered systems and networks. Yurcik et al. [54] try to address main challenges faced by system administrators of large HPC deployments. The authors propose *NVisionCC*, a visualization framework for the security of HPC systems. Yurcik and Liu [53] seek to address the user masquerade attacks in HPC clusters, where attacks are enabled by stolen credentials. For this, the authors build SVM classifiers that could classify various categories of users according to their command behavior. Vieira et al. [49] argue that attacks on grids, clusters and clouds can be accomplished stealthy. They also argue that in grid and cloud environments, the classical IDS systems are unable to detect abnormal activities in an effective and efficient manner. The authors developed the Grid and Cloud Computing Intrusion Detection System (GCCIDS). The GCCIDS contains an audit system able to detect grid and cloud environment attacks.

b) Security and Privacy for Clouds and CC: Somorovsky et al. [47] performed a security analysis related to Amazon's EC2 public cloud platform and Eucalyptus private cloud framework. In particular, they focused on the analysis of the control interfaces of those frameworks. The authors revealed several highly critical vulnerabilities which include gaining root access to arbitrary virtual machines, as well as the possibility to collect and exfiltrate arbitrary data and files from those cloud platforms. Ristenpart et al. [43] used the Amazon EC2 service as an example and showed it is possible to perform a mapping of the internal cloud infrastructure. Once the mapping is accomplished, it is enough to perform side-channel attacks. Such attacks could be further used to extract data from the targeted virtual machine. Molnar and Schechter [37] demonstrate from a security point of view the advantages and disadvantages of processing data by public cloud providers. They show that cloud usage can result in new types of threats (e.g., jurisdictional, organizational, technological). The authors also present and discuss a set of countermeasures. Idziorek and Tannian [28] show that cloud services that are accessible over the Internet expose various resources that are subject to fraudulent resource consumption. The authors argue that transactions which are specially-crafted, but only differ by the intent and not by the content, are hard to recognize. Therefore, this type of attacks may be difficult to detect or prevent. Pearson et al. [40] assess how issues of privacy, security and trust occur in the context of cloud and grid computing. The authors discuss ways in which those issues could be effectively and efficiently addressed. Bhadauria and Sanyal [19] performed a survey on numerous privacy and security challenges threatening the cloud environments. Finally, Janse and Grance [29] propose *NIST SP 800-144*, a NIST guideline on security and privacy in public cloud computing.

c) Infrastructure Management Interfaces: Related to large infrastructure monitoring and management systems, Bruno et al. [20] studied the existing open-source code implementing "looking-glass" web interfaces for BGP routing. They also studied the Internet facing deployments of such web interfaces. The authors argued that similar flaws inside the networks operators can be exploited by attackers with limited resources. For example, this way the attackers can get access to the core Internet infrastructure.

VIII. Conclusion

In this paper we presented a comprehensive security analysis of some of most popular infrastructure monitoring tools for grids, clusters and HPC systems. We analyzed the infrastructure internal data that many of these interfaces provide online to general public. This allowed to analyze various host details, such as OS kernel version, internal IP addresses and network structure of at least 43K hosts that are monitored using these openly accessible interfaces. In addition, we used both static and dynamic analysis on multiple versions of those tools and discovered a plethora of vulnerabilities in their web interfaces. On average, we estimate that around more than a half of the currently deployed Ganglia systems may be vulnerable to *Remote Code Execution (RCE)* under CVE-2012-3448. Finally, we discussed the possible attack scenarios and recommended countermeasures. We conclude that the privacy and security state of monitoring the grids, clusters and HPC systems, as cloud underlying paradigms, is not sufficient and must be improved at the earliest.

Acknowledgments

We thank John Matherly of Shodan search engine for providing generous access to Shodan's data and resources. We also thank the anonymous reviewers and close friends for their comments and suggestions that helped us improve the paper.

References

[1] Arachni scanner. http://arachni-scanner.com.

[2] Argus. http://argus.tcp4me.com/.

[3] Cacti. http://www.cacti.net/.

[4] Collectd. http://collectd.org/.

[5] CVEs for Linux Kernel 2.6.32. http://cvedetails.com/version-search.php?vendor=linux&product=&version=2.6.32.

[6] Ganglia. http://ganglia.info/.

[7] Grsecurity. https://grsecurity.net/.

[8] Monit. http://mmonit.com/monit/.

[9] Munin. http://munin-monitoring.org/.

[10] Nagios. http://www.nagios.org/.

[11] Observium. http://www.observium.org.

[12] Shodan search engine. http://shodan.io.

[13] Statistics for websites using framework technologies. http://trends.builtwith.com/framework.

[14] SUNY Buffalo HPC. http://casc.org/meetings/10sep/Furlani.ppt.

[15] Usage of server-side programming languages for websites. http://w3techs.com/technologies/overview/programming_language/all.

[16] Zabbix. http://www.zabbix.com/.

[17] Zenoss. http://www.zenoss.com/.

[18] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell. State of the Art: Automated Black-Box Web Application Vulnerability Testing. In *IEEE Symposium on Security and Privacy*, 2010.

[19] R. Bhadauria and S. Sanyal. Survey on security issues in cloud computing and associated mitigation techniques. *arXiv:1204.0764*, 2012.

[20] L. Bruno, M. Graziano, D. Balzarotti, and A. Francillon. Through the looking-glass, and what eve found there. In *USENIX Workshop on Offensive Technologies (WOOT)*. USENIX Association, 2014.

[21] S. Christey and R. A. Martin. Vulnerability type distributions in CVE. *Mitre Report*, 2007.

[22] J. Dahse and T. Holz. Simulation of Built-in PHP Features for Precise Static Code Analysis. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2014.

[23] Firehost. The Superfecta Report Special Edition. 2013.

[24] P. Fogla, M. I. Sharif, R. Perdisci, O. M. Kolesnikov, and W. Lee. Polymorphic Blending Attacks. In *USENIX Security Symposium*, 2006.

[25] J. T. Giffin, S. Jha, and B. P. Miller. Automated discovery of mimicry attacks. In *International Symposium on Recent Advances in Intrusion Detection (RAID)*. Springer, 2006.

[26] L. Y. L. J. Z. Haiyan. The application of cacti in the campus network traffic monitoring [j]. *Computer & Telecommunication*, 2008.

[27] J. Idziorek. Exploiting cloud utility models for profit and ruin. 2012.

[28] J. Idziorek and M. Tannian. Exploiting cloud utility models for profit and ruin. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011.

[29] W. Jansen and T. Grance. Nist sp 800-144. guidelines on security and privacy in public cloud computing. 2011.

[30] M. Johns. Code-injection Vulnerabilities in Web ApplicationsExemplified at Cross-site Scripting. *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 2011.

[31] N. Jovanovic, C. Kruegel, and E. Kirda. Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper). In *IEEE Symposium on Security and Privacy*, 2006.

[32] H. G. Kayacik and A. N. Zincir-Heywood. Mimicry attacks demystified: What can attackers do to evade detection? In *Privacy, Security and Trust, 2008. PST'08. Sixth Annual Conference on*. IEEE, 2008.

[33] O. Kolesnikov and W. Lee. Advanced polymorphic worms: Evading IDS by blending in with normal traffic. 2005.

[34] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Automating mimicry attacks using static binary analysis. In *USENIX Security Symposium*. USENIX Association, 2005.

[35] G. LI, L.-x. ZHOU, X.-l. WANG, S.-c. QI, H.-q. YAO, and J.-y. SUN. Application of open source systems in the seismic monitoring network [j]. *Northwestern Seismological Journal*, 2011.

[36] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 2004.

[37] D. Molnar and S. E. Schechter. Self Hosting vs. Cloud Hosting: Accounting for the Security Impact of Hosting in the Cloud. In *Workshop on Economics of Information Security (WEIS)*, 2010.

[38] M. Montanari and R. H. Campbell. Attack-resilient compliance monitoring for large distributed infrastructure systems. In *Network and System Security (NSS), 2011 5th International Conference on*. IEEE, 2011.

[39] M. Moses. Network and service reconnaissance. 2013.

[40] S. Pearson and A. Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), Second International Conference on*. IEEE, 2010.

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

[42] U. B. M. Project. Millennium project web page. http://www.millennium.berkeley.edu, 1999.

[43] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2009.

[44] S. Sengupta, V. Kaulgud, and V. S. Sharma. Cloud computing security–trends and research directions. In *Services (SERVICES), 2011 IEEE World Congress on*. IEEE, 2011.

[45] P. Sharma, S. K. Sood, and S. Kaur. Security issues in cloud computing. In *High Performance Architecture and Grid Computing*. Springer, 2011.

[46] E. Skoudis and T. Liston. *Counter hack reloaded: a step-by-step guide to computer attacks and effective defenses*. Prentice Hall Press, 2005.

[47] J. Somorovsky, M. Heiderich, M. Jensen, J. Schwenk, N. Gruschka, and L. Lo Iacono. All Your Clouds Are Belong to Us: Security Analysis of Cloud Management Interfaces. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, 2011.

[48] K. Tan, J. McHugh, and K. Killourhy. Hiding intrusions: From the abnormal to the normal and beyond. In *Information Hiding*. Springer, 2003.

[49] K. Vieira, A. Schulter, C. Westphall, and C. Westphall. Intrusion detection for grid and cloud computing. *It Professional*, 2009.

[50] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna. Cross Site Scripting Prevention with Dynamic Data Tainting and Static Analysis. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2007.

[51] D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2002.

[52] WhiteHat. Website Security Statistics Report. *WhiteHat Report*, 2013.

[53] W. Yurcik and C. Liu. A first step toward detecting SSH identity theft in HPC cluster environments: discriminating masqueraders based on command behavior. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*. IEEE, 2005.

[54] W. Yurcik, X. Meng, and N. Kiyanclar. Nvisioncc: a visualization framework for high performance cluster security. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*. ACM, 2004.

[55] M. Zalewski. *Silence on the wire: a field guide to passive reconnaissance and indirect attacks*. No Starch Press, 2005.

[56] Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *IEEE Symposium on Security and Privacy*. IEEE, 2011.

TABLE II
Summary of all vulnerabilities found statically with RIPS. Distribution by vulnerability type. Legend: CE=Code Execution; HTRS=HTTP Response Splitting; CI=Command Injection; HI=Header Injection; PFC=Possible Flow Control; UNS=Unserialize; SQLI=SQL Injection; LDAPI=LDAP Injection; FI=File Inclusion; FM=File Manipulation; FD=File Disclosure; XSS=Cross-Site Scripting.

Version / Vulnerability Type	CE	HTRS	CI	HI	PFC	UNS	SQLI	LDAPI	FI	FM	FD	XSS	TOTAL/version
cacti-0.8	0	68	2	0	21	5	0	0	0	1	0	92	189
cacti-0.8.1	0	68	2	0	21	5	0	0	0	1	0	95	192
cacti-0.8.2	0	68	2	0	21	5	0	0	0	1	0	95	192
cacti-0.8.2a	0	68	2	0	21	5	0	0	0	1	0	95	192
cacti-0.8.3	0	70	2	0	21	5	0	0	0	1	0	102	201
cacti-0.8.3a	0	70	2	0	21	5	0	0	0	1	0	102	201
cacti-0.8.4	0	75	4	0	23	5	0	0	0	1	2	98	208
cacti-0.8.5	0	78	4	0	26	5	0	0	0	1	2	113	229
cacti-0.8.5a	0	78	4	0	23	5	0	0	0	1	2	116	229
cacti-0.8.6	2	78	5	1	25	7	0	0	1	7	5	160	291
cacti-0.8.6a	2	78	5	1	25	7	0	0	1	7	6	161	293
cacti-0.8.6b	2	78	5	1	25	7	0	0	1	7	4	161	291
cacti-0.8.6c	2	78	5	1	25	7	0	0	1	7	5	161	292
cacti-0.8.6d	2	78	5	1	25	7	0	0	1	7	5	160	291
cacti-0.8.6e	2	78	5	1	25	7	0	0	1	7	5	179	310
cacti-0.8.6f	2	78	6	1	25	7	0	0	1	7	5	180	312
cacti-0.8.6g	2	78	6	1	27	7	0	0	1	8	5	179	314
cacti-0.8.6h	2	78	6	1	29	7	0	0	1	8	5	190	327
cacti-0.8.6i	2	78	5	1	29	7	0	0	1	7	5	261	396
cacti-0.8.6j	2	78	5	1	29	7	0	0	1	7	5	265	400
cacti-0.8.6k	2	78	5	1	29	7	0	0	1	7	5	234	369
cacti-0.8.7	2	78	3	1	30	10	0	1	1	7	5	205	343
cacti-0.8.7a	2	78	3	1	30	10	0	1	1	7	5	205	343
cacti-0.8.7b	2	78	3	1	30	10	0	1	1	7	5	154	292
cacti-0.8.7c	2	78	3	1	31	11	0	1	1	10	3	202	343
cacti-0.8.7d	2	78	3	1	31	11	0	1	1	10	3	206	347
cacti-0.8.7e	2	78	3	1	31	11	0	1	1	10	3	208	349
cacti-0.8.7g	2	77	2	1	29	11	0	0	1	8	3	141	275
cacti-0.8.7h	33	75	2	1	1	11	0	0	1	8	3	159	294
cacti-0.8.7i	33	75	2	1	1	11	0	0	1	8	3	159	294
cacti-0.8.7i-PIA-3.1	38	76	2	1	1	11	0	0	2	8	3	169	311
cacti-0.8.8	36	76	2	1	1	11	0	0	2	8	3	169	309
cacti-0.8.8a	36	76	2	1	1	11	0	0	2	8	3	169	309
cacti-0.8.8b	36	76	2	1	1	11	0	0	2	8	3	169	309
cacti-0.8.8c	36	77	2	1	1	11	0	0	2	8	3	204	345
ganglia_jobmonarch-1.0	0	0	1	0	3	2	0	0	0	0	0	1	7
ganglia_jobmonarch-1.1	0	0	1	0	3	2	0	0	0	0	0	1	7
ganglia_jobmonarch-1.1.1	0	0	1	0	3	2	0	0	0	0	0	1	7
ganglia_jobmonarch-1.1.2	0	0	1	0	3	2	0	0	0	0	0	1	7
ganglia-web-3.4.1	0	3	1	1	2	14	0	0	0	16	14	89	140
ganglia-web-3.4.2	0	3	1	1	2	14	0	0	0	16	14	90	141
ganglia-web-3.5.0	0	3	1	1	2	14	0	0	0	16	15	87	139
ganglia-web-3.5.10	0	3	1	1	2	13	0	0	0	13	11	43	87
ganglia-web-3.5.1	0	3	1	1	2	15	0	0	0	16	16	89	143
ganglia-web-3.5.11	0	3	3	1	2	13	0	0	0	12	12	46	92
ganglia-web-3.5.12	0	3	3	1	2	13	0	0	0	12	12	46	92
ganglia-web-3.5.2	0	3	1	1	2	15	0	0	0	16	15	90	143
ganglia-web-3.5.3	0	3	1	1	2	15	0	0	0	16	15	92	145
ganglia-web-3.5.4	0	3	1	1	2	15	0	0	0	16	15	92	145
ganglia-web-3.5.7	0	3	1	1	2	15	0	0	0	16	12	60	110
ganglia-web-3.5.8	0	3	1	1	2	13	0	0	0	13	11	43	87
ganglia-web-3.6.2	0	2	2	1	2	17	0	0	0	12	11	47	94
gweb-2.0.0	0	3	1	1	2	5	0	0	0	7	8	69	96
gweb-2.1.1	0	3	3	1	2	5	0	0	0	9	10	74	107
gweb-2.1.2	0	3	3	1	2	5	0	0	0	10	10	74	108
gweb-2.1.3	0	3	3	1	2	5	0	0	0	10	10	74	108
gweb-2.1.5	0	3	1	1	2	5	0	0	0	8	8	72	100
gweb-2.1.6	0	3	3	1	2	5	0	0	0	9	10	76	109
gweb-2.1.7	0	3	3	1	2	5	0	0	0	9	10	76	109
gweb-2.1.8	0	3	1	1	2	6	0	0	0	13	10	75	111
gweb-2.2.0	0	3	1	1	2	12	0	0	0	20	12	92	143
gweb-3.3.0	0	3	1	1	2	13	0	0	0	18	15	92	145
gweb-3.3.1	0	3	1	1	2	13	0	0	0	18	14	92	144
observium	0	1	2	0	3	3	4	0	9	5	4	51	82
TOTAL/vulnerability	286	2727	166	50	798	556	4	6	40	536	408	7553	-