Assessing Traceability of User Jobs in Absence of End User Certificates in GlideinWMS Framework

Anand Padmanabhan
CyberGIS Center for Advanced Digital and Spatial Studies
National Center for Supercomputing Applications
Urbana, IL, USA
E-mail: `apadmana@illinois.edu`

Mine Altunay
Fermi National Accelerator Laboratory
Batavia, IL, USA
E-mail: `maltunay@fnal.gov`

Kevin Hill
Fermi National Accelerator Laboratory
Batavia, IL, USA
E-mail:`kevinh@fnal.gov`

An increasing number of Virtual Organizations (VOs) want to shield their users from the details of managing their jobs on the Grid. Various tools such as GlideinWMS allow VOs to achieve this goal. However, even with GlideinWMS, the end user certificate management remains a challenging issue. Ultimately, the VOs want to utilize the GlideinWMS and the Grid in such a way that the need for end user personal certificates can be eliminated or made completely transparent to the end user. With GlideinWMS, running jobs without end user certificates is possible and in this paper, we evaluate whether this operational mode provides necessary traceability information that allows us to associate Grid jobs with end users.

The certificates have long provided a way for identifying users and their jobs. Knowing the owner of a job is important from an operational security perspective; without this information, it would be impossible to enforce user traceability and accountability for the actions taken on the Grid. However, user traceability is not solely achieved through a specific technology such as X.509 certificates; it can be achieved through different technical means. Our hypothesis is that GlideinWMS system collects extensive data about the users and their jobs and that even without end user certificates a Grid job can be traced back to the individual who initiated the job. In this study we systematically assess the capabilities of the GlideinWMS system on top of HTCondor with regards to tracing jobs back to their owners in the absence of end user certificates. The findings and conclusion from this assessment are laid out in this paper. A main goal is to understand whether eliminating end users certificates introduces significant

additional risks to the GlideinWMS system. We understand that there are security risks in GlideinWMS system even in the presence of the certificates. Therefore, to be fair to both operational modes (with and without certificates), we focus on the additional risks that are introduced due to the lack of end user certificates and that are not applicable when the end user certificates are present.

Through a systematic assessment, we conclude that the GlideinWMS system (collectively the Frontend, the Factory and the Grid resource providers) records an extensive amount of traceability data about the user and his/her jobs. As a result, the system has shown to have capabilities to provide sufficient traceability information in the absence of the certificates. This conclusion was reached not only by evaluating the system design, but also conducting security exercises, where each component of the system were challenged to produce traceability data for a specific job. Over the course of this assessment, no significant gaps were identified though some challenges were discovered and already addressed, or mitigation strategies identified. More importantly, when we compare the two modes of running jobs in GlideinWMS, i.e. with and without end users certificates, we find that both operational modes possess equivalent traceability capabilities. The results presented in this paper are primarily informed by drills that were conducted on environment with HTCondor as a local scheduler. However, we believe that the results will likely hold on other batch systems, though this will need to be verified with future experiments. We did not find any additional significant risk introduced due to elimination of the end user certificates. Furthermore, we found that the removal of end user certificate requirement lowers the barrier to access computing resources for users and boasts the opportunistic usage of resources.

The International Symposium on Grids and Clouds (ISGC) 2014
March 23-28, 2014
Academia Sinica, Taipei, Taiwan

1. Introduction

As new user and science communities benefit from the growing number of compute and data resources offered by Grids, there is an increasing need to lower the technological barriers and promote ease-of-use of Grid resources [1]. To this end, a number of Virtual Organizations (VOs) are already submitting jobs on behalf of their users by using pilot job framework, such as GlideinWMS [2] and PANDA [3]. These frameworks manage user jobs and insulate users from having to deal with individual sites on the Grid. A user on whose behalf the job is executed is the owner of the job, and he/she is still responsible for the behavior of the job [4].

From a security perspective, being able to identify an owner for each job remains extremely important. First, it provides user accountability and enables action to be taken when a user job causes problems on the Grid. Second, it allows fine-grain access privileges assigned for each user. When users need access to variety of resources, each user may need a different set of access privileges. In the grid world, especially in pilot job systems, we observe that fine-grain access privileges are rarely utilized. Most users have uniform needs and hence require the same access privileges. For a successful job execution, it is not necessary to know the identity of a user because almost all users require the same type of resources and associated access privileges. As a result, traceability and accountability of end users are the most important reasons for knowing the user identities. This observation was crucial in that it motivated our work and launched our effort to understand whether we can satisfy the traceability requirement without handing out individual access credentials, i.e. certificates, to each Grid user.

X.509 digital user certificates have long provided an easy way of associating users with jobs and made it possible to enforce user traceability and accountability for the actions taken on the Grid. The digital user certificates are traditionally sent as part of the user job payload and are individually authenticated and authorized by the resource providers. Since the certificates are associated with individual users, it has been fairly straightforward for the sites to make connections between a job and a user. However, the use of X.509 user certificates represents a significant technological barrier since they expose users to the complexities of the authentication system. Hence there is a growing trend for VOs to move away from the model of requiring every individual to get a user certificate. This is especially true for small or new VOs in the OSG since this eliminates the learning curve of understanding and getting X.509 certificates. By making this technically feasible, the pilot job frameworks are accelerating this trend.

The hypothesis we investigate in this work is that the pilot job frameworks already collect a large amount of data about users and jobs, and combined with an access control model that does not require certificates they can provide sufficient traceability information and usability. To test this hypothesis, we conducted experiments using the GlideinWMS pilot framework and evaluated whether user traceability can be achieved in the absence of end-user certificates with specific focus on any additional security risks that are due to the lack of certificates. While comparing the system in two different modes, with and without certificates, we were careful to evaluate them fairly: we did not penalize the new system for lacking traceability information that did not exist in the existing system. Specifically, for this paper, the GlideinWMS system (collectively with frontend (with user pool), glidein factory and WMS pool, and Grid sites) was assessed by both evaluating the system design and conducting security exercises (i.e. jobs submitted without end-user certificates) with HTCondor service as local scheduler, where each component of the system was challenged to produce traceability data for a specific job. The exercises were conducted by labeling a random (non-malicious) job as malicious and searched for (1) the job's owner, (2) other jobs and resources used by the same user. We considered both running jobs and completed jobs. Our assessment found that by conflating information recorded in system logs by associated components extensive amount of traceability information can be gleaned about a user and his/her jobs. We concluded that the GlideinWMS system without certificates showed no significant differences in achieving user traceability than the existing system that uses certificates.

From these observations, this paper documents the processes and technical means to establish user traceability and identifies the gaps in the system that would hinder it. For each identified gap, we evaluate the likelihood of running into that specific problem and the consequences we would face in such an event. However, there cannot be a "perfect" system with no possibility of problems; therefore, the likelihood of problems being materialized and the cost of their consequences are also discussed to guide our understanding of whether the GlideinWMS system provides sufficient traceability of end users. Finally, as a case study, we will outline the results from implementation of certificate-free access for some Open Science Grid (OSG) [5] VOs on a selected OSG resource provider.

Allowing users to access the Grid resources without certificates causes a profound shift in the trust paradigm between users and resource providers. In the old trust paradigm the users presented a certificate so the providers knew exactly which user was using their resources and which VO they belonged to. However with the new model the sites do not know the identity of the user and have to trust VO to provide that information. We discuss the effects of changes in trust relationships in Chapter 5 in detail.

The rest of this document is organized as follows. General background information about the GlideinWMS systems is outlined in Section 2. Section 3 presents the traceability process, where a Grid site admin can follow to trace a job back to its owner. Sections 4 and 5 respectively discuss the challenges of conducting traceability and the changing trust relationships. We wrap up with summary and concluding discussions in Section 6.

2. Background

The goal of this paper as we outlined in the introduction is to research if traceability can be achieved through technical means in the absence of personal user certificates. In this section we outline the GlideinWMS architecture and its typical usage within OSG (see Figure 1).

A typical Glidein usage [6] involves the user accessing the Glidein frontend using an authentication mechanism that is logged using syslog. A VO user submits a job to an HTCondor [7] dynamically sized cluster assembled by the frontend. The frontend in turn is configured to request Glideins from one or more factory (currently OSG production factories hosted at GOC and UCSD), which in turn submits Glidein jobs to OSG production sites. The Glidein job ensures the worker node at the site suitable for executing the actual payload and then starts the user job. A Glidein job is a parent process to the actual user job and watches over the user job throughout its lifetime. Hence to conduct traceability analysis we rely on log files from each of these entities:

- GlideinWMS frontend
- GlideinWMS factory
- OSG Sites

The following are important details of the GlideinWMS system

- A worker node can run multiple Glideins and user jobs simultaneously.
- An individual Glidein only starts a single job at a time. However a Glidein might run a sequence of jobs one after the other. These jobs may belong to different users.
- All HTCondor logs associated with the Glidein are written on the worker nodes and are transferred back to the Glidein factory after the Glidein completes execution.
- Glideins typically run for 24-48 hours.

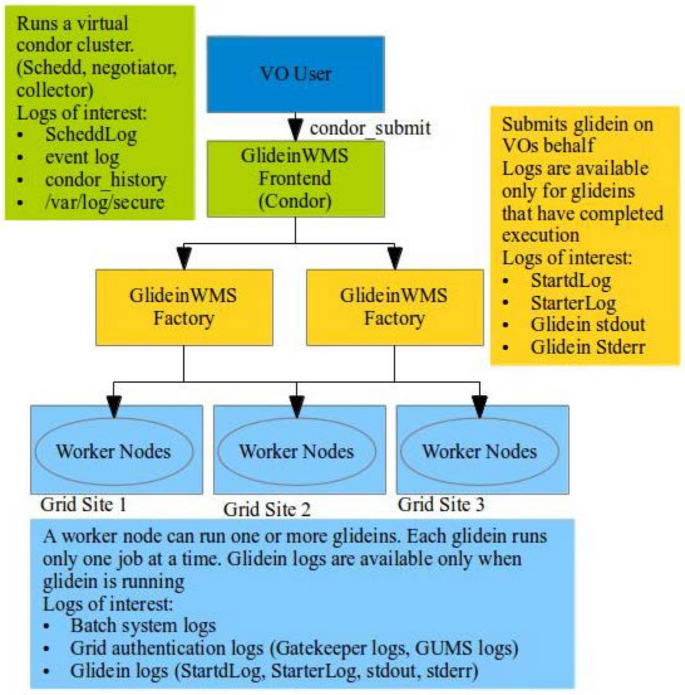
Runs a virtual condor cluster. (Schedd, negotiator, collector)
Logs of interest:
• ScheddLog
• event log
• condor_history
• /var/log/secure
VO User
condor_submit
GlideinWMS Frontend (Condor)
Submits glidein on VOs behalf
Logs are available only for glideins that have completed execution
Logs of interest:
• StartdLog
• StarterLog
• Glidein stdout
• Glidein Stderr
GlideinWMS Factory
GlideinWMS Factory
Worker Nodes
Worker Nodes
Worker Nodes
Grid Site 1
Grid Site 2
Grid Site 3
A worker node can run one or more glideins. Each glidein runs only one job at a time. Glidein logs are available only when glidein is running
Logs of interest:
• Batch system logs
• Grid authentication logs (Gatekeeper logs, GUMS logs)
• Glidein logs (StartdLog, StarterLog, stdout, stderr)

***Figure 1**: Glideinwms Architecture*

In this study, we do not specifically consider the flocking case, which happens when the user job is initially submitted to a HTCondor pool, but later for whatever reason the pool decides to submit the job to another pool. In case of flocked jobs, the flocked HTCondor pool needs to trust the remote HTCondor pool for tracing the users. Although we do not evaluate flocking in this study, technically the process of setting up an evaluation should be identical and all we would need to do is to trace the job back step by step using the same techniques outlined here. The choice of local batch system could affect the steps outlined and traceability achieved. This study was conducted with HTCondor serving as the local scheduler. However, we believe that the results will likely hold on other batch systems, though this needs to be verified with future experiments. In addition to providing traceability, X.509 certificates also provide straightforward ability of isolating user jobs (i.e. by mapping each DN to a unique local account) which might be lost without end user certificates. Though that could impact the overall security we believe its impact on traceability is limited (VOs can still be identified and their logs can be consulted). Furthermore, a number of Grid sites already use shared pool accounts where multiple users from a VO are mapped to the same system user, which in principle is a scenario similar to the Glidein approach evaluated in this paper. Hence the authors felt the issue of isolating user jobs is broader and do not fully explore its impact in this paper.

3. Job Traceability

In this section, we consider two scenarios and outline how to conduct traceability

- Job is currently running on the worker node;
- Job has finished execution.

For the discussion below we assume HTCondor as the local batch system at the site.

3.1 Job is currently running on worker nodes

The process of tracing user will proceed as follows:

1. The site admin identifies a problem process on a worker node at the site. The admin will record the unique process ID of the problem process and its parent processes (using pstree for example), and the IP address or full hostname of the worker node.
2. Since the job is currently running, this implies that the Glidein pilot is also running and its logs are locally available on the worker node instead of the factory. The factory in this case does not have any logs about the problem job yet.
3. Establish when the offending process begins to execute at the site and the user account (uid) under which it executes.
4. Identify the Glidein process that started the problem job. Every Glidein writes standard error and out, Starter and Startd logs, which are named with a Glidein's unique HTCondor job id number. (The Glidein is uniquely identified by the job id, under which the Glidein is queued in the WMSPool at the GlideinWMS factory.) Of primary interest for traceability are the StartdLog and the StarterLog. The StartdLog periodically dumps the entire classad of the user job using the Glidein.
    - Find the HTCondor StarterLogs of the Glidein and search for the process ID of the problem process identified in step 1. When you find the process id in HTCondor StarterLog, compare it against the timeframe of the process (process IDs may be recycled after a time interval). If there is match, this is the Glidein instance that started the problem process.
    - If the worker node has multiple Glideins running, each Glidein will have its own Starter log. Repeat the above step for each Glidein StarterLog until an HTCondor job id for the problematic process is found.
5. Identify the VO that owns the Glidein and the problem job.

- Identify the unix account the problem process is running under. This is the same account under which the Glidein started executing.
- Consult the CE/site authentication logs (e.g. globus gatekeeper and GUMS log) to identify the certificate Distinguished Name (DN) under which the Glidein is submitted.
- Each Glidein has a unique certificate and is assigned to a VO. With the Glidein certificate DN, you may ask the Factory to help you determine the VO.

6. The information in the StartdLogs includes the frontend (the Schedd) that the user job is submitted from (i.e. the HTCondor pool), Cmd (on frontend), Owner (on frontend), Cluster ID (HTCondor job id on frontend), JobPid (the Pid the user job is running as on the WN). Details of exactly when the user job started and completed on the worker node can be seen in the corresponding StarterLog.
7. Provide the information from step 6, to the frontend operator of the VO identified in step 5. The GlideinWMS factory will be able to provide the contact information of the frontend operator.
8. Using the information provided (identified in step 6), the frontend operator will consult the HTCondor log files and HTCondor history recorded at the VO's HTCondor pool to identify the user account submitted this job. The frontend operator is also able to provide information on if and where other jobs from the same user are running on the Grid.
9. Using the user account on the frontend node, the frontend admin will find out which user logged into this account, how he was authenticated, and other pertinent login information. The admin should also be able to provide the real identity and contact information of the end user.

3.2 Job has finished execution

There are two distinct scenarios when the user job has been completed: 1) the Glidein that ran the user job is still active; and 2) the Glidein that ran the user job has also completed.

For scenario 1, the Glidein logs are still available on the worker node where it is being executed. So the for the purposes of job traceability, this scenario is identical with the case of "job is currently running on worker nodes".

For scenario 2, when the Glidein job completes, the logs associated with the Glidein (i.e. standard error and out, Starter, and Startd logs) are transferred back to the factory. Hence the site admin no longer has access to the Glidein logs needed for steps 4 and 6. Instead, the factory operator conducts these steps and the site admin is responsible for providing the information collected in other steps. However, a challenge for doing steps 4 and 6 at the factory is that the factory can have logs from a large number of Glideins. Though given a unique hostname of the worker node and the timeframe it is possible to search through the logs to identify which Glidein ran the user job, this task can be tedious and time consuming in the absence of software tools designed for analyzing the logs.

A special case in this category is when a job is started, but cannot successfully complete due to a bug in the job or problem at the worker node such as insufficient memory. These jobs are treated in the same way as the jobs that are successfully completed. The same information is logged and made available to the factory when the failing job stops executing.

4. Challenges Associated with User Traceability

In this section we outline the challenges we encountered while evaluating the GlideinWMS system. For each challenge identified, we provide a description of the problem, the likelihood of running into this problem, mitigation techniques to eliminate or reduce the impact of the problem, and an overall rating for the problem. The rating indicates how big of a challenge this issue considering the likelihood and the mitigations provided.

Challenge 1: Site unable to provide process id
Details: System-level security best practices (beyond the scope of the GlideinWMS system) recommend employing a system-level logging mechanism, where the running and completed jobs can be logged and later examined when needed. Although GlideinWMS system keeps logs of running jobs, such logs are transported back to the Factory once the Glidein completes execution. If the worker node lacks system-level logging tools, it may be difficult to find the process id (pid assigned by the operating system) of a problem job after the job finishes execution.
Likelihood: Low
Mitigation: In the absence of the process id information, if there is only a single Glidein running at the worker node, the job owner can be uniquely identified. If there are multiple Glideins, there will be just as many job owners as the number of Glideins (remember that each Glidein runs a single job at a time). So even though we are not able to trace the problem job back to an individual, we are still able to narrow it down to potentially a small set of users. Additional information from the site, such as a tight timeframe, regarding the problem process might be able to further refine this list.
Rating: Low. A list of users who run jobs at the given worker node at the given timeframe can be found.

Challenge 2: HTCondor logs on the Frontend get deleted too quickly after being rotated
Details: The HTCondor_history mechanism (a command line tool provided by HTCondor) allows a Frontend admin to query the HTCondor history log. The tool can use HTCondor classad constraints and formatting to get just the data required. The query interface is a big advantage of this method. However, the default HTCondor configuration only keeps a 20MB HTCondor_history log file, which is rotated once. This is small for a busy submit host, and there is a good chance that data might have rolled off the back when needed.
Likelihood: High
Mitigation: The size and the rotations can be increased so that more data is kept:

```
MAX_HISTORY_LOG = 1000000000
ENABLE_HISTORY_ROTATION = True
MAX_HISTORY_ROTATIONS = 5
```

HTCondor team already addressed this issue in their latest release. The new configuration allows for Frontend operator to specify the desired number of days the logs must be kept. Alternatively, the frontend operator can also define the maximum size of the log files.
Rating: Non-issue. This issue is fixed in the latest HTCondor release..

Challenge 3: Tracing jobs that were preempted on site
Details: An issue with HTCondor_history is that the job match information is lost when new matches are made. The new job match occurs when a job comes out of the hold state (meaning job is hung at the worker node and needs to be killed and restarted somewhere else) or job is pre-empted (meaning job was once in the running state, but the worker node got a higher priority job, and the job must be sent back to the VO HTCondor queue and get started on another worker node). When the new job match is made, HTCondor_history doesn't store the previous match records.
Likelihood: Moderate
Mitigation: To enable a more complete picture of where a job has been before the final execution host, HTCondor can be configured to add historical machine attributes to the job classad. For example, to remember the last 10 site names, Glidein names, and StartdIp addresses:

```
SYSTEM_JOB_MACHINE_ATTRS=GLIDEIN_ResourceName, Name, StartdIpAddr
SYSTEM_JOB_MACHINE_ATTRS_HISTORY_LENGTH = 10
```

This helps, but we also need to keep JobStartDate and other attributes to tell us when the job ran on those Glideins. A history length of 10 might also be too small.

A more complete log is the HTCondor event log. This log keeps all the state changes, so preempted/held jobs show up in this log with correct timestamps. The downside of the event log is that we do not have a good query tool for the log, so currently the Frontend administrator has to dig out information by hand. Similar to the history log, the event log size needs to be big enough to match security policies, and it would be nice to configure this with for example daily rotation, for 7 days. We might also need a query/summary tool, specific to the Glidein case.
Rating: Non-issue. At the worst case, the frontend administrator can find the data manually.

Challenge 4: Attacker overwrites the Glidein log files
Details: If a malicious user could hijack the Glidein infrastructure, overwrite the log files, and no useful info returned to the factory (i.e. fake logs).
Likelihood: Low
Mitigation: To address this, we have two potential venues: 1) getting more information back to the Schedd/Frontend; and 2) making sure we always do the UID switch.
Rating: Low. After careful evaluation we concluded with the mitigations in place the risk profile was similar to when X.509 certificates are used.

5. Change in Trust Paradigm

In the previous sections we have demonstrated that it is possible to trace individual users even in the absence of user certificates. However, the move to the certificate-free job submission mode causes profound changes in the trust relationships between the sites, users, and VOs.

With certificates included in the user jobs, a resource provider would know which user has submitted a particular job immediately upon receiving the job on his/her resource. As a result, resource owners do not need to rely on the VO, the GlideinWMS operators, or anybody else to help them find out who is running on their resources because the user certificate attached to the job provides all the information needed. As shown in Figure 2, the trust relationship is directly between the end user and the resource owner.

With jobs that do not include user certificates, resource providers have to rely on the VO and the GlideinWMS operators to find out the identities of the users. As a result, the resources owners has no way of having a direct relationship with the end users, but have to develop a trust relationship with the VO, which then develops a direct trust relationship with the user, as shown in Figure 2.

It is true that sites and VOs have always had a trust relationship. After all a resource owner provides access to users not because the resource owner knows or understands each individual user’s scientific mission, but because the resource owner supports the mission of a particular VO and desires to provide access to all VO members. Therefore, the trust relationship between resource and VO is nothing new. However, with the certificate-free jobs, this relationship became even more crucial and obvious. The sites are more dependent on this relationship to find basic information about the users. In the past, a resource could bypass the VO and still have a direct connection to the user. With the new job submission mode, the resource and VO relationship cannot be bypassed and resource is dependent on the VO to protect itself from malicious users.

The effects of this new trust model, manifests itself most glaringly in the case of a malicious user. A site has no way of identifying and blocking access to a malicious user because there are no access credentials that can distinguish VO members from one another when they access the site. Instead, the VO and GlideinWMS operators have to identify the user and remove his/her credentials to stop her/him from submitting jobs. In the extreme cases that a VO does not perform its responsibilities, a site’s only defense is to block access to the entire VO members. Our observation is that it is in VO’s best interest to have a trustworthy and functional relationship with a site in order to enjoy continued access to site resources. Therefore, the VOs are very motivated to identify their misbehaving users and ban them from the Grid. The changed trust model hence results in new or updated responsibilities for each of the stakeholders in thr system which we will next outline.

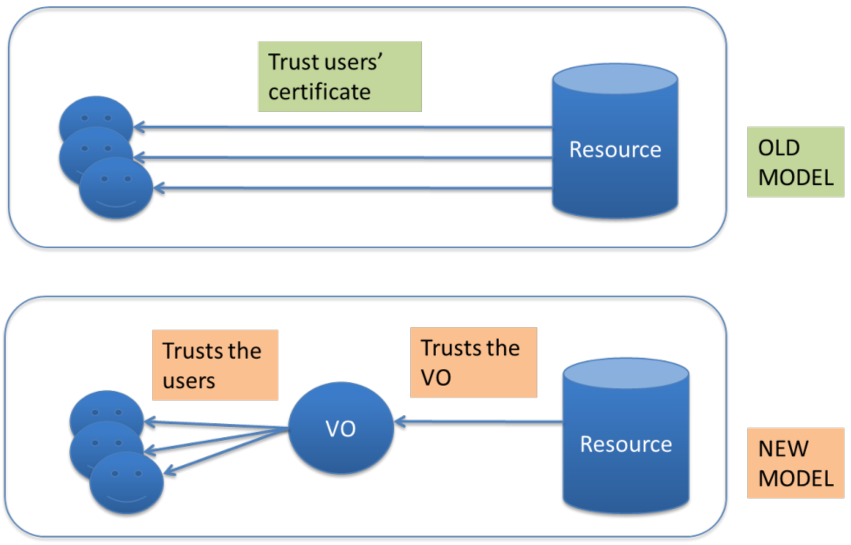

***Figure 2**: Changes to Trust Paradigm*

5.1 Responsibilities of glideinWMS operators

The glideinWMS framework, as we saw in section 2 (see Figure 1), is operated in a distributed fashion with each of the components operated independently by different operators. Hence each of the glidein component operators in the distributed framework has unique and crucial roles to play to ensure traceability of user jobs and in this section we will outline these responsibilities.

5.1.1 Glidein frontend operator

Glidein frontend operator (typically an OSG VO) is responsible for securing access to the frontend machine and setting up authentication mechanism (typically ssh with login and password) to ensure only legitimate and trusted users get login privileges to the frontend. A rigorous process of verifying user identities is required to be in place. The system logs associated with user logins should be maintained for at least 30 days (preferably 90 days). In case user account compromise is either detected or reported the frontend operator will immediately take steps to disable access to the account in question and report the matter to the OSG security promptly. Since a dynamically sized HTCondor virtual cluster is being assembled at the frontend, logs associated with HTCondor need to be kept for at least 7 days (preferably 30 days) by the frontend operator. The job traceability critically depends on ScheddLog, and EventLog HTCondor logs from the frontend and the frontend operator would be responsible to ensuring these are securely stored for the necessary timeframe (these crucial logs should be kept for 30 days). The frontend operator (and the associated VO) is responsible for 1) documenting procedures used to obtain access to the frontend; 2) maintaining a list of its members/staff who have user-level or/and privileged access to the frontend; and 3) documenting process for removing its users from VO membership and terminating access frontend services when needed and will make these documentation and lists available to grid resource owners and/or OSG security team when requested.

It should be noted that the VO is ultimately responsible for doing the heavy-lifting of identifying, authorizing, managing and banning the user when needed. OSG is only playing a facilitator role between sites and VOs.

5.1.2 GlideinWMS factory operator

Glideins factories (managed centrally by OSG) submit Glidein jobs to OSG production sites on behalf of the frontend. The frontend just publishes that it needs some glideins and the factory takes care of the rest by knowing the details of various Grid sites and properly configures the glidein entry points. For glideins that have completed, the HTCondor logs are automatically included in the glidein logs sent back to the factory, and these logs may be available only at the factory. The factory admin are required to keep these glidein logs for atleast 30 days (preferably 90 days). If a glidein is still running, the OSG site can access the glidein and condor logs on the worker node where the glidein is being executed. StartdLog, StarterLog, Glidein stdout, and Glidein stderr, associated with a glidein and available at the factory critically impact job traceability. The factory admin is responsible for ensuring that only legitimate frontends (typically associated with OSG VOs) are authorized to use the factory. The factory operator is responsible for 1) documenting procedures used to obtain access to the factory; 2) maintaining a list of its members/staff who have user-level or/and privileged access to the factory service; and 3) documenting process and condition for terminating access factory services when needed and will make these documentation and lists available to grid resource owners and/or OSG security team when requested. OSG security should be notified promptly if any unauthorized or suspicious use of factory resources is detected or reported.

5.1.3 OSG resource provider

OSG resource providers (i.e. OSG sites) executing the job has number of pieces of information at its disposal to help job traceability. Specifically batch system logs, and Grid authentication logs (Gatekeeper logs, GUMS logs) provide the site with a ability to track the VO and the DN used to submit the job. The site should keep these logs for at least 30 days (preferably 90 days). Additionally when a glidein job is running glidein logs (StartdLog, StarterLog, stdout, stderr) are available on the worker node where the glidein is being executed. For jobs that have completed execution these logs may be available at the factory, for at-least 30 days. However if a sites would prefer to or their policies require them to collect the information locally, they may do so by just having their batch system capture and store the stdout and stderr of the glidein as it is transferred back to the factory. OSG security should be notified promptly if any unauthorized or suspicious activity is detected or reported.

6. Summary and Conclusions

A systematic traceability study was conducted on two frontend/VOs to allow certificate free access to a significant resource provider (Fermilab (FNAL)), which previously required certificates. The specific frontends evaluated are OSG-XSEDE frontend (OSG VO) (June 2013) and CHTC frontends (GLOW VO) (March 2014). Based on the careful study we reached the following conclusions. The GlideinWMS has shown to possess significant tracing capabilities. We are satisfied that the system is capable of finding a finite set of users who run jobs at a given worker node at a given timeframe. Furthermore, the system can identify a unique owner for a Grid job at a worker node for a given timeframe. There are a few corner cases discussed in Section 4 that make tracing an individual user for a given job challenging. However, the likelihood of these cases being materialized is quite low; therefore, we are confident in GlideinWMS system's ability in providing traceability information without using end user certificates. Furthermore, when we compare running jobs in GlideinWMS without certificates against running jobs with certificates, we find that both operational modes have equivalent ability in providing traceability information. Having said that some improvements to the GlideinWMS system as discussed in Section 4 can enhance the system in a positive direction. However, the absence of these improvements should not hold back sites and VOs from adopting GlideinWMS system without the end user certificates.

The conclusions and recommendations were presented to FNAL security team, for evaluation, since they were the resource provider that was making a transition to accepting pilot jobs without end user certificates. The recommendations were accepted and implemented, which successfully opened up the use of opportunistic resources from FNAL to a large number of users. A simple metric we can use to evaluate the effectiveness is increased opportunistic usage of FNAL resources by the VOs that took part in the study. Figure 3 shows a sharp increase in computing usage of FNAL resources by OSG-XSEDE frontend after October 2013 when the recommendations were accepted. Figure 4 shows the spikes in the number of jobs running in the same timeframe clearly showing opportunistic usage.

In summary, through this study we conclude that GlideinWMS system provides equivalent ability in providing traceability with or without certificates. Furthermore, this study shows an approach to lower the barrier to access computing resources on the Grid without much impact on traceability. In our case studies this has resulted in a spike in the number of computing hours used by this VO and a high level of user satisfaction with the Grid usage.

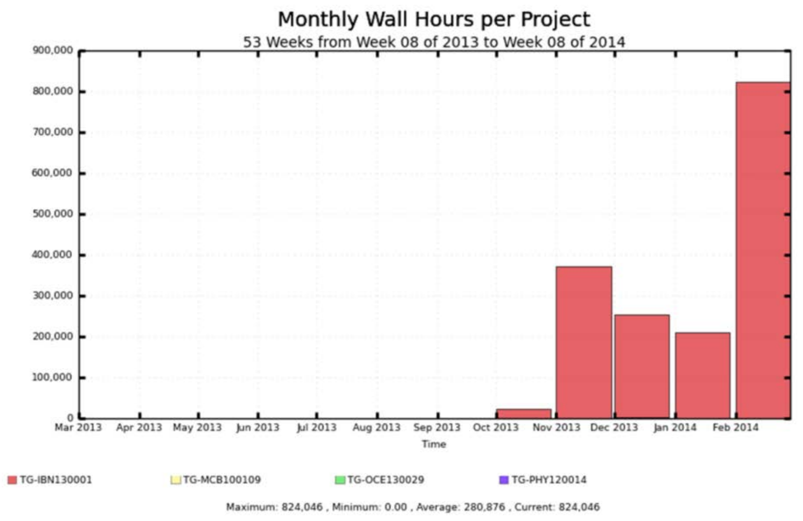

Figure 3*: Usage (Computing Walltime) of FNAL Resources by OSG-XSEDE Frontend*

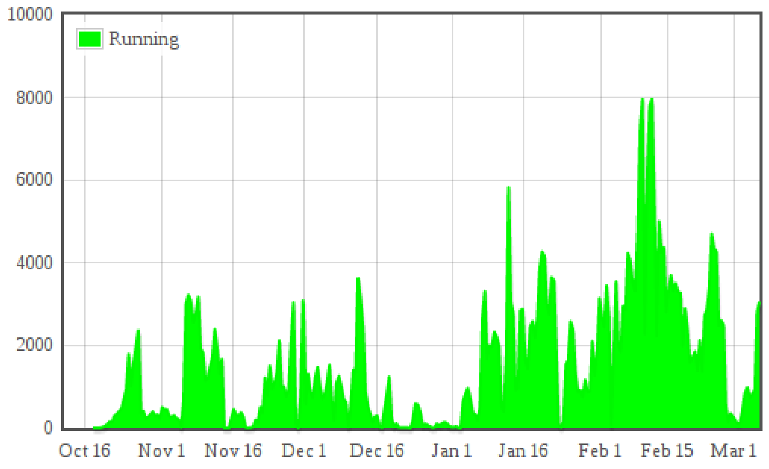

***Figure 4**: Usage (Number of Jobs) of FNAL Resources by OSG VO*

References

[1] Sfiligoi, I. (2007, October). Making science in the Grid world: using glideins to maximize scientific output. In *Nuclear Science Symposium Conference Record, 2007. NSS'07. IEEE* (Vol. 2, pp. 1107-1109). IEEE.

[2] Sfiligoi, I. (2008, July). glideinWMS—a generic pilot-based workload management system. In *Journal of Physics: Conference Series* (Vol. 119, No. 6, p. 062044). IOP Publishing.

[3] Maeno, T. (2008, July). PanDA: distributed production and distributed analysis system for ATLAS. In *Journal of Physics: Conference Series* (Vol. 119, No. 6, p. 062036). IOP Publishing.

[4] Sfiligoi, I., Bradley, D. C., Holzman, B., Mhashilkar, P., Padhi, S., & Wurthwein, F. (2009, March). The pilot way to grid resources using glideinwms. In *Computer Science and Information Engineering, 2009 WRI World Congress on* (Vol. 2, pp. 428-432). IEEE.

[5] Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., ... & Quick, R. (2007, July). The open science grid. In *Journal of Physics: Conference Series* (Vol. 78, No. 1, p. 012057). IOP Publishing.

[6] GlideinWMS Online Documentation. Available at http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/documentation.html. (accessed 27th March 2014)

[7] Litzkow, M. J., Livny, M., & Mutka, M. W. (1988, June). Condor-a hunter of idle workstations. In *Distributed Computing Systems, 1988., 8th International Conference on* (pp. 104-111). IEEE.