Abstract

Volunteer Computing is a low cost and easy deployable solution for scientific projects that require large amounts of computational power. Volunteer computing public access capabilities comes with several security threats to the projects, thus, this projects must be designed and focused towards security in order to be able to stand against this threats. The following paper presents a summary of the security threats identified in volunteer computing environments deploying the Berkeley Open Infrastructure for Network Computing (BOINC) and some solutions implementing Sanboxing Techniques on distributed computing environments.

Keywords: BOINC; security; Volunteer Computing; sandbox; virtual machine

1. Introduction

In scientific computational projects, the volunteer computing work model presents a solution to projects that have low resources to access to large amounts of computational power [1]. In volunteer computing, a server contains a project that can be accessed and many users voluntarily provide computing resources (processing time, memory, disk space) whenever the user’s computer is in idle state. This users are known as volunteers and their computers are known as host to the project to which the user is providing the resources.

2. Security Threats

BOINC Project developers group have identified a series of possible security threats for projects running on volunteer computing networks [2]. This security threats have been taken in consideration along with the development and improvement of the BOINC platform. BOINC hasn’t reported any incidents regarding the following security threats, but that is very important for...
every user and project developer to “keep an eye” on them [3].

We can summarize BOINC’s security issues as follows[2]:

Results and credits falsification: Volunteers might send incorrect information regarding data results or claim more CPU usage than they are actually offering. The way to prevent this, BOINC [4] require that projects running under it must have two routines: a validator, to confirm that the results will be valid and an assimilator to handle validated results. To validate, BOINC supports replication techniques, in which rather than making each task one time only, the same task is assigned to multiple host computers. This way, the results can be validated in an easier way. Depending on the project, developers could use some mathematical libraries to reduce to minimum calculation discrepancies. In some projects where discrepancy could be tolerated, a measured tolerance level allows the developers to have a better reference on the results.

Malicious executable distribution: An attacker could change the project’s content in order to distribute malicious code through the volunteer network. BOINC requires the use of correctly implemented methods of code signing [3]. The digital signatures apply a set of public-private keys. BOINC recommends to follow a series of steps to generate the key set and how to manage it according to code signing policies [3].

Overrun of data servers: An attacker could send large amounts of data overrunning the servers and disabling them. To prevent these sorts of attacks, BOINC provides a tool called “upload certificates” [2]. Each project defines a maximum output file sizes and each project must have a unique identification key. When a set of output files are sent to the server, the file sizes is checked and verified and the authentication key is matched.

Theft of participant account information by network attack: Attackers could get access to the BOINC servers and use network vulnerabilities to steal project or user information. The protection of the users and project information is to be taken care of by the project’s developer. BOINC requires that the servers must be protected according to security standards implementing firewalls and secure access protocols such as SSH.

Theft of project files: An attacker could steal any of the input or output information of any project running on BOINC. Although the generated files are not encrypted by default, BOINC project developers can apply any encryption technique upon their projects.

Intentional or accidental abuse of participants hosts projects: Any routine might intentionally abuse of a volunteer host by stealing information or accidentally modifying configuration files within the host computer. To prevent this kind of abuses, BOINC provide us with two very useful tools. The first uses sandboxing to prevent the project from accessing any files outside of the BOINC environment. The second technique detects the amount of resources being used by the project, if the project is taking too much disk space, memory or processing time the project will be closed.

One way to guarantee the integrity of the information is using a not-intrusive security model that will allow easy deployment independent of the host platform. These working environments are within the reach of the sandbox concept and represent a viable solution to many security threats.

3. Sandbox: a solution to security issues

Sandboxing gives us an isolated environment, denying the project application any access to the host files and external resources [5]. BOINC has identified two different sandbox environments. Account-based sandboxing uses the OS capabilities of adding and assigning different user account levels [3]. The idea is to create a user account with no special privileges and then run the BOINC application within this account. Any routine that tries to access system information or file will not be allowed to.

The second sandboxing technique uses virtual machines completely encapsulating the applications running within [6]. Applications running on the virtual machines are isolated from routines, resources and host files. Virtual machines are the best example where we can see the work of sandbox environments. In security testing areas, virtual machines play the role of host computers to run, test and study malicious applications without causing any harm to the physical computer.

BOINC offers a tool to develop projects on virtual machines called vboxwrapper [7]. This tool allows the developers to mount projects on virtual machine environments in a simpler manner, letting them take advantage of the benefits in virtual machines.

Many of the security threats identified by BOINC can be contained using a virtual machine-based sandbox work model. Virtual Machines allow the project developers to work completely independent of the host system. This takes care of the compatibility issue between a project and
different host operating systems (Mac, Linux, Windows). These isolating capabilities also prevent malicious users from abusing and attacking the projects files [8]. Another great advantage of the isolating capabilities of virtual machines is against attacks such as information theft, malicious code distribution and server overload.

BOINC provides examples of ready-to-use virtual machines based on vboxwrapper and ready for the developers to shape them according to their projects. The problem at this point is that the vboxwrapper is not compatible with any other processor architecture but Intel [7].

In volunteer computing environments it is very important to be able to determine the availability of the volunteer host in order to have some understanding on how much will a task take to be finished. This situation can be solved with virtual machine sandboxing [5]. The virtual machines have different methods to keep record of status and to pause the task, so it can be retaken without any work lost.

Figure 1: BOINC Wrapper tool to communicate the applications with the client [11].

There are some libraries that can be adapted to connect the BOINC middleware with virtual machines [9]. Libboinexec is a library that allows you to monitor several virtual machines.

In [10] it is presented an adaption to the library alongside with some modifications to the BOINC wrapper’s virtual machine.

As shown in fig 1, Wrapper is a BOINC tool that allows communication between the BOINC client and one or more applications that are interpreted as a sub process in sequence [11]. The method presented in [10] represents a valuable step towards the implementation of virtual machines for sandbox environments. It is focused on implementing libboinexec and virtual machines on BOINC working environments. In [5] it is presented an architecture model scalable to any grid project implementing virtual machines for sandboxing. The point is to create an application that could launch a virtual machine and support the architecture on figure 2:

Figure 2: Virtual Machine Model Architecture [5].

1. The Virtual Machine Application: hides the under running architecture and manages the different virtual machines and interacting with the different instances.
2. The Backend: Saves the metadata of each virtual machine instance allowing the sandbox to open and close at any given time.
3. The Base Image: Is the base to all the virtual machine instances. Each instance is created from this base image containing all the necessary information to run the project.
4. Instance Images: Each instance must store its own image from the base image without corrupting the original.
5. Communication Daemon: Handles the communications between the instances and the working hosts.
6. Execution environment: Is where all applications run and it is stored within each virtual machine.

Proposed methods such as the Minimum intrusion grids seek the means to mix grid architecture and virtual machines. This work falls under the virtual machine sandbox model and help developers to improve security services and prevent possible threats.
4. Minimum Intrusion grids

The concept for Minimum intrusion Grids (MiG) is presented in [8]. MiG is a technique independent to any previous grid implemented system. The main idea is to build a grid infrastructure that takes as minimum requirements as possible from the host computers.

The MiG implements virtualization techniques in a way that users will only need to have an X.509 certificate and support HTTP, HTTPS or SSH connection. MiG virtual machines contain a ttylinux image as operating system allowing very low loading times [8]. The MiG using virtual machines and following a sandboxing model provides a security advantage that makes it a multiplatform solution.

MiG also includes some improvements to the sandbox model. MiGs have 2 components: file remote access, uses identification techniques to access to resources libraries and task scheduling setting a time limit for some task to be finished. If the task finishing time is exceeded, said task will be given to a different host.

5. Virtual machine based security framework (VMF)

Code signing and the implementation of validating and assimilating result modules are some of the requirements that BOINC requests from the projects developers. In many cases, the tools and implementation of security measures are not very well understood by the project developers, thus making the deployment of said security measures very difficult to apply [12].

In figure 3, we propose a shell to develop a framework focused towards virtual machine sandboxing environments and including the basic security requirements for any BOINC project.

The proposed framework includes 2 main modules with a sub module aimed to attend the security issues and a result handler module as follows:

1. Project Application Module: This module includes the BOINC framework to create and install the projects. To this existing framework we proposed adding two new distinctive sub-modules: security and result handling.

2. Result Handling Sub-module: This sub-module includes the BOINC result handling requirements. The main idea is to organize the result handling requirements into one standard structural block in a way that developers could apply result handling according to BOINC requirements [4].

3. Security Sub-module: This sub-module includes the necessary tools to fulfill BOINC security requirements [3]. This module will present, to the project developers, the set of tools in a structured and organized way, improving the learning and deploying process of this security measures. The advantage on implementing this sub-module, is to make the deployment of security measures more accessible to the project developers on computer volunteer environments [12].

4. Virtual Machine Module: This module combines the development platform of BOINC with the implementation of virtual machine sandboxing. We propose to apply the architecture describe in section III, in a way that the virtual machine development would be modular [5].

Figure 3: Framework Shell for the development of BOINC projects focused on virtual machine sandbox based security.
Conclusions and further work

From the work we have presented in this paper, the next step will be to make a statistic study of the implementation of the different security options within the BOINC environment presented in section II.

We have presented a robust and scalable platform such as BOINC and its many security options which might turn complex and difficult to implement. This complexity represents the main disadvantage in developing stronger projects in security measures [13].

The poor understanding of the developers regarding the implementation of security measures due the complexity of the implementation is the greatest obstacle in implementing solutions and preventing security threats [12]. The solution is to focus our work in developing friendly environments and more friendly and organized techniques in order to make it more accessible to the projects developers to implement security measures within distributed computing environments [12].

The implementation of sandboxing techniques and modular frameworks, such as the one on section V, together with organized tools lowering the complexity of deployment [12, 13] is the key solution to all security issues presented in this paper and allows to have a working model scalable to any distributed computational environment.

References: