SlapOS and NGINX Architecture for DDOS attacks

Vlad Andrei Poenaru, George Suciu, Cristian George Cernat, Gyorgy Todoran and Traian Lucian Militaru
The Faculty of Electronics Telecommunications and Information Technology
at Politehnica University of Bucharest, Bd. Iuliu Maniu nr.1-3
Bucharest 060042, Romania
{vlad.wing@gmail.com} {george@beia.ro}
{cernatcristi@gmail.com} {todoran.gyorgy@gmail.com} {gelmosro@yahoo.com}

ABSTRACT: *The cloud is a new technology for the delivery of web applications. However, there is little literature on the impact of distributed denial-of-service (DDOS) attacks on clouds and the performance of other types of attacks against a well-designed cloud architecture. So, is cloud resilience enough to be a solution? Or will traditional web architectures with caching and reverse proxy always be better? This paper attempts to answer these questions by testing two architectures: SlapOS vs. architecture with nginx pre-configured and a few web servers behind the architecture against various types of DDOS attacks, including slowloris and RA flood attacks.*

Keywords: Cloud, DDOS, Cloud Architecture, Flood Attacks

Received: 25 April 2023, Revised 14 June 2023, Accepted 10 July 2023

DOI: 10.6025/jisr/2023/14/4/96-100

1. Introduction

In this paper we present a study about distributed denial of service attacks (*DDOS*) in open source cloud platform SlapOS, the first open source operating system for Distributed Cloud Computing. This will include writing security testing scripts, collecting results and automating scripts for improving security of software deployment and configuration on cloud nodes.

We develop a test platform for cloud computing and use it as a case study for testing and monitoring different security threats. We use different types of attacks and monitor important information such as *CPU* load, number of processes and intrusion level from installed sensors. The sensors will transmit intrusion detection data from our cloud platform in real-time, display it in our web-based visualization application and get detailed recommendations when and where security threats did occur - resulting in optimized automating patching.

Also we will introduce in this article SlapOS, the first open source operating system for Distributed Cloud Computing. SlapOS is based on a grid computing daemon called slapgrid which is capable of installing any software on a *PC* and instantiate any number of processes of potentially infinite duration of any installed software. Slapgrid daemon receives requests from a central scheduler the SlapOS Master which collects back accounting information from each process. SlapOS Master follows an Enterprise Resource Planning (*ERP*) model to handle at the same time process allocation optimization and billing. SLAP stands for "Simple Language for Accounting and Provisioning".

This structure has been implemented for cloud-based automation of *ERP* and *CRM* software for small businesses and aspects are under development under the framework of the European research project "Cloud Consulting" [1]. We will use our platform hosted on several servers running Ubuntu *Linux – Apache – MySQL* template with current software release. On our cloud testing environment we provide the platform for processing information from hundreds different sensors, enabling the analysis of security data through a large sample of logs.

We demonstrate that open source cloud platforms are welldeveloped and mature technologies offering a secure environment for deploying a growing number of applications.

2. Problem Formulation

SlapOS is an open source Cloud Operating system which was inspired by recent research in Grid Computing and in particular by BonjourGrid [2] a meta Desktop Grid middleware for the coordination of multiple instances of Desktop Grid middleware. It is based on the motto that "everything is a process". SlapOS is now an *OW*2 project. Figure 1 shows the current architecture.

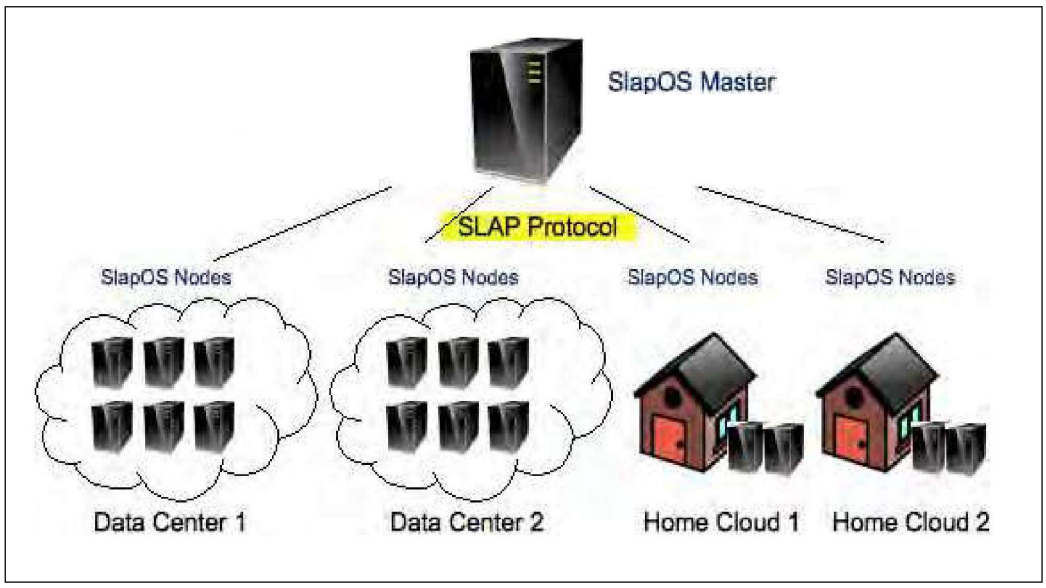

Figure 1. The SlapOS Architecture

SlapOS defines two types of servers: SlapOS Nodes and SlapOS Master. SlapOS Nodes can be installed inside data centers or at home. Their role is to install software and run processes. SlapOS Master acts as a central directory of all SlapOS Nodes, knowing where each SlapOS Node is located and which software can be installed on each node. The role of SlapOS Master is to allocate processes to SlapOS Nodes.

SlapOS Nodes and SlapOS Master exchange are interconnected through the HTTP and XML based SLAP protocol. SlapOS Master sends to each SlapOS Node a description of which software should be installed and executed. Each SlapOS Node sends to SlapOS Master a description of how much resources were used during a given period of time for accounting and billing purpose.

From a user point of view, SlapOS Node looks like an online shop for Cloud Computing resources. The user connects to SlapOS Master through a simplified front end, selects which software he or she needs. SlapOS Master then allocates the software onto a SlapOS Node and provides the connection information to the user. The allocated software can be of any type: virtual machine, database server, application server, web cache front end, etc.

From a developer point of view, as seen in Figure 2, SlapOS is a simple and universal *API* to create instances of any software daemon through a programmatic interface.

Figure 2. An example of SlapOS front-end

A simple code allows a developer to request a new instance of a memcache server by invoking the request method of SlapOS API. Memcache [3] is a widely adopted key-value store protocol which is used to cache values in large scale web infrastructure. It is usually installed and configured by system administrators using packaging systems such *RPM* or *DEB*. In this example, a single method call does in a few seconds what a human system administrator would have done in few minutes at best.

3. Problem Solution

SlapOS is implemented as an extension of widely adopted open source software: *GNU*/*Linux*, Buildout [4] and Supervisord [5] and as depicted on Figure 3. The only new software introduced by SlapOS is Slapgrid, a daemon in charge of implementing the *SLAP* protocol on each SlapOS Node.

Each time slapgrid receives a request from SlapOS master to install a software, it downloads a description of that software in the form of so-called buildout profile. It then runs the buildout bootstrap process to install the software. Buildout is a Python-based build system for creating, assembling and deploying applications from multiple parts, some of which may be non-Python-based. Buildout can be used to build *C*, *C*++, ruby, java, perl, etc. software on Linux, MacOS, Windows, etc.

Buildout can either build applications by downloading their source code from source repositories (subversion, git, mercurial, etc.) or by downloading binaries from package repositories (rpm, deb, eggs, gems, war, etc.). Buildout excels in particular at building applications in a way which is operating system agnostic and to automate application configuration process in a reproducible way.

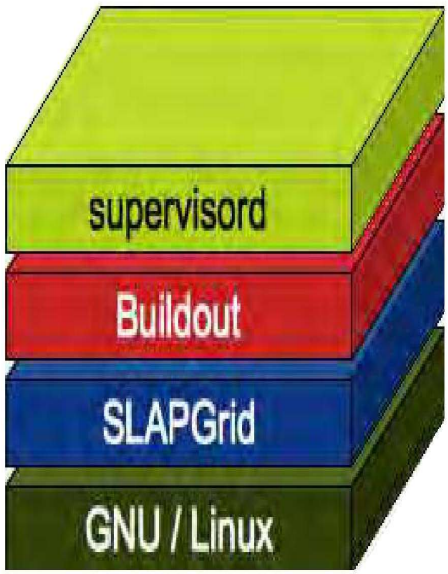

Figure 3. The SlapOS kernel

Each time slapgrid receives a request from SlapOS master to run a software as a new process, it calls first buildout to create all configuration files for that process then delegates to supervisord the execution of the process. Supervisor is a client/server system that allows its users to monitor and control a number of processes on *UNIX*-like operating systems. It provides a higher abstraction and flexibility than traditional sysinit.

After some time, a typical SlapOS Node will include multiple software applications and, for each software application, multiple instances, each of which running in a different process. For example, both Mediawiki and *OS* Commerce could be installed onto the same SlapOS Node, with six instances of each being run as processes. By running software instances as processes, rather than by creating a virtual machine for each software instance as one would do with Amazon *EC*2 [6], SlapOS is able to use hardware resources and *RAM* in particular more efficiently.

SlapOS Master runs *ERP*5 Cloud Engine, a version of ERP5 open source ERP capable of allocating processes in relation with accounting and billing rules. Initial versions of SlapOS Master were installed and configured by human. Newer versions of SlapOS Master are implemented themselves as SlapOS Nodes, in a completely reflexive ways. A SlapOS Master can thus allocate a SlapOS Master which in turn can allocate another SlapOS Master, etc.:

After running security testing scripts and collecting results, as shown in Table 1, we conclude that our open source cloud platform delivers better performance in attacks against it.

Architecture	**SlapOS cloud**	**Nginx Web**
CPU load (1) DDOS (2) slowloris (3) RA flood attacks	 (1) 85% (2) 90% (3) 60%	 (1) 95% (2) 99% (3) 80%
Number of processes (slowloris)	200K	14K
Exploits detected	5238/5545	5211/ 5545

Table 1

4. Conclusion

SlapOS can be described as a cloud operating system in which "everything is a process" unlike Unix in which "everything is a file". If one has to manage thousands of servers with thousands of processes, hundred different applications in multiple different releases or versions, SlapOS can help you a lot by making the whole security management process well specified, automated and under control.

Therefore cloud security is shared with the processes of the applications running on the nodes.

The second result with SlapOS is that the best way to create a reliable and secure cloud computing system is to follow the original principles of the Internet: distribution and simplicity.

Our system can also help keeping track of exploit development, optimize patching for zero-days threats and to produce log auditing to improve security risk management.

As future work we envision an early warning system of cloud attacks that applies intrusion prevention measures based on sensor information from different partitions on the distributed nodes.

Acknowledgement

This paper is presented as part of the project *"Valorificarea capitalului uman din cercetare prin burse doctorale (ValueDoc)"* Project co-financed from the European Social Fund through POSDRU, financing contract POSDRU/107/1.5/S/76909 and part of the project "Cloud Consulting".

References

[1] Suciu, George., Fratu, Octavian., Halunga, Simona., Cernat, Cristian George., Poenaru, Vlad Andrei., Suciu, Victor. (2011). Cloud Consulting: ERP and Communication Application Integration in Open Source Cloud Systems. *In 19th Telecommunications Forum - TELFOR 2011 (pp. 578-581). IEEE Communications Society.*

[2] Abbes, Heithem., C´erin, Christophe., Jemni, Mohamed. (2008). Bonjourgrid as a decentralised job scheduler. In APSCC 08. Proceedings of the 2008 *IEEE Asia-Pacific Services Computing Conference* (pp. 89–94). *IEEE Computer Society*.

[3] Memcached. (n.d.). A free and open-source, high-performance, distributed memory object caching system. http://memcached.org/

[4] Buildout - software build system reloaded. (n.d.). http://www.buildout.org/

[5] Supervisor: A Process Control System. (n.d.). http://supervisord.org/

[6] Ng, Tze., Wang, Guohui. (2010). The impact of virtualization on network performance of Amazon EC2 data center. *IEEE INFOCOM 2010 - 029th IEEE International Conference on Computer Communication*s, 29 (01), March 2010.