A Novel Lightweight Authentication Protocol for YML Framework

Xin Lv
College of Water Conservancy and Hydropower Engineering
Hohai University, HHU
Nanjing, CHINA
e-mail: lvxin.gs@163.com

Hao Chen
Huaneng Lancang River Hydropower CO., LTD
Kunming, CHINA
e-mail: chenhao.hnlcj@foxmail.com

Feng Xu, Yingchi Mao
College of Computer and Information
Hohai University, HHU
Nanjing, CHINA
e-mail: xufeng@hhu.edu.cn

***Abstract*—**YML framework is a well-adapted advanced tool to support designing and executing portable parallel applications over large scale peer to peer and grid middleware. It is necessary to introduce some security mechanisms for the improvement and extension of the framework. To secure the authentication process, a novel lightweight authentication protocol was proposed, enabling the legitimate user to log in YML efficiently, with 2 XOR and 1 HASH operations. The protocol can resist denial of service attack, replay attack, and even all the knowledge of the user was stolen, the system still can recovery the permission of the user which has been attacked effectively by updating the value of the corresponding parameter.

Keywords-YML framework; front-end; lightweight authentication; replay attack

I. INTRODUCTION

Many Grid and Peer to Peer computing middleware solutions have been developed and are becoming stable. They harness available computing and storage resources in order to build large scale platforms. Then, they are used to solve huge applications or store large amount of data. Although computing platforms have become popular tools, the current challenging issues are better scalability and availability. Indeed, a large number of independent platforms are currently managed by different global computing middleware products without any runtime interoperability.

YML [1] is one workflow solution which has been developed at PRiSM laboratories [2] in collaboration with Inria-Futurs/LIFL [3, 4]. This framework is dedicated to the development and the execution of parallel applications over large scale middleware. YML includes a workflow language named *YvetteML* used in the description of applications and their executions. YML furnishes a compiler and a just-in-time scheduler for *YvetteML*. It allows the user to manage the execution of the application over the underlying parallel architecture which can be a peer to peer or a grid middleware. The specificity of each middleware is hidden to the user through YML, making the user can easily develop a complex parallel application which may transparently execute on multiple middleware during one application execution. The framework provides workflow engine capabilities on top of a global computing platform, and it is designed to act transparently for complex applications using numerous communications, code coupling, etc. on dynamic platforms.

On the YML point of view, an application is divided into different computing sections, each of them containing some tasks executed sequentially or concurrently. A task, called a component, is a piece of work that can be mapped to one node in a parallel environment. It has some input and output parameters and is generally reusable in different parts of the application as well as in different applications. YML provides a special type of components, called graph component, which consists in the description of subgraph. This kind of components will be exploited for the distribution of the application.

YML framework has been developed since 2000. It acts as a well-adapted advanced tool to support designing and executing portable parallel applications over large scale peer to peer and grid middleware [5, 6], and its extension version is able to manage at the run-time several middleware back-ends, achieving a dynamic federation of computing middleware [7]. We also extend the framework to be middleware for cloud computing platform [8, 9]. With the development of global computing model over the Internet infrastructure, the collaboration, agility and scale of computing are significantly enhanced. However, in this environment, security and privacy problems have become more and more severe. As a well-designed front-end, YML connects different computing resources to complete the tasks, so it is indispensable to integrate some appropriate security mechanisms into the framework. Through our work we aim at ensuring the computing resources are utilized legitimately and properly. We provide a lightweight authentication protocol instead of currently-used password-based mechanism which is easy to crack. The user needs to register in YML, and then he can create a new execution or monitor current and past executions as usual.

The rest of the paper is organized as follows. An overview of YML is given in Section II, and we also point out the security problem exist in the framework. A novel lightweight authentication protocol for YML framework is specified in Section III with corresponding security and efficiency analysis. Finally we conclude the paper in Section IV.

II. Overview of YML

YML is a framework dedicated to the creation and the execution of parallel and distributed applications on various middleware. It proposes an intuitive representation of a distributed and parallel application by means of a workflow. A workflow consists of a graph whose vertices are independent and communication-less computing tasks, while the edges represent the precedence relationships between the tasks. The graph description language is called *YvetteML*. A graph represents a control flow, and not a data flow, since a precedence does not necessary concern a data dependency. The main structures of YvetteML are: the services execution, the parallel sections, the sequential loops, the parallelized loops, the conditional branch and the event notification/reception.

YML represents the notion of computing task by components, called services. The reusability is the main motivation. Besides, this representation helps to clearly separate computational blocks of his application and communication expressing dependences. Each computational task is described by an *abstract service* and is implemented in an *implementation service*. Services information is contained into two catalogs. A Development Catalog stores information used at the time of the application development. An Execution Catalog stores information used during the execution of the application.

Figure 1 gives a simplified view of an YML framework.

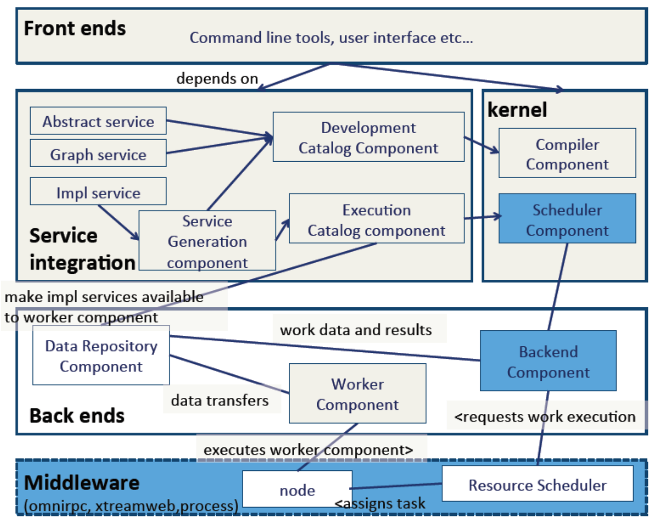

Figure 1. Overview of YML Framework

The front-end provides the user a way to control and administer the execution of YML application. Once a workflow application has been registered into the portal, users can schedule and execute applications from this web-based environment. But it just uses a password-based authentication which has an inherited limitation and poses significant risk. In global computing environment, we must ensure that the computing resources are properly and effectively utilized by legitimate users. So it is necessary to introduce a more secure authentication mechanism to prevent the computing resources from misusing.

III. A Novel Lightweight Authentication Protocol for YML Framework

In order to ensure the computing resources being utilized properly and legitimately, we introduce an authentication mechanism into the YML framework. The user and the front-end just need to interact on a little registered information with a few lightweight operations to complete the validation of identity, which almost have no negative influence on the efficiency.

In the following description, let pw denote user's password, $\oplus$ denote XOR operation, □ denote concatenation operation, $h(\ )$ is a collision resistant hash function, ID is the unique identifier of the user, maybe extract from the user's ID card number or its email address and so on.

A. *User Registration*

The user submits its registration information $\{ID, h(b \oplus pw)\}$ to the YML front-end through secure channel, with b is a random number generated by user itself, which is stored on its own device.

When the front-end receives the information, it computes $T = h((ID \square i) \oplus x) \oplus h(b \oplus pw)$ with x is a secret key which is long enough to ensure the security of the registration, and the front end keeps it for all the registered users. The initialization value of i is 1.

Then the front end returns $\{ID, T\}$ to the user, and store the information $\{ID, N, i\}$ for each registered user. The user and front-end both maintain a $counter$ synchronously after registration, and its initialization value is 1.

B. *Authentication*

Before executing YML application, the user needs to interact with front end to complete authentication.

1) User: computes $y = h(b \oplus pw) \oplus T$ and $c = h(y \oplus counter)$, then sends $ID, c, counter$ to the front-end;

2) Front-End: firstly checks whether the received $counter$ is same as the local one, if so, computes $y = h\left(\left(ID \| i\right) \oplus x\right)$ and $h\left(y \oplus counter\right)$. If the result is equal to the received c, then the authentication passes, otherwise, rejects the user.

3) If the authentication is passed, the user and the front-end update $counter$ to $counter + 1$ synchronously.

C. Change the password

The user can change its password in an efficient way.

1) User: firstly completes the authentication process in Section *B*.

2) User: chooses its new password pw_{new}, and computes $h\left(b \oplus pw_{new}\right)$, then sends it to the front end.

3) Front-End: computes $T_{new} = T \oplus h\left(b \oplus pw_{old}\right) \oplus h\left(b \oplus pw_{new}\right)$ to updates the value of T, and return T_{new} to the user.

During the process of updating, other parameters are fixed.

D. Security Analysis

We consider the security of the system in the following attack.

1) Leak of password

We don't need to worry about the leak of password. The password is always combined with b in the calculating process, which is only stored on the user's device. So the attacker cannot launch any attack by utilizing the password. Furthermore, the user can change its password efficiently following the steps in Section *C*.

2) The user's device suffers attack

If the device has been breached, the attacker can obtain $\left\{ID, T, b, counter\right\}$, if it finds the user's password pw at the same time, the key proof of user's identity $y = h\left(\left(ID \| i\right) \oplus x\right)$ will be exposed. Now, the attacker is able to impersonate a legitimate user to access to YML service. Once the user discovers the situation above-mentioned, it can report to the front-end, and request for stopping to response its service application and re-initializing the system. From the authentication process, we know the value of y must be re-generated in the re-initialization process (otherwise, even the user changes the password and the value of b, the attacker still can pass the authentication, just need to eavesdrop on the value of $counter$). Through replacing i by $i+1$, the value of y is altering while the user's ID and the front-end's secret key x keep fixed, then the user re-registers in the front-end to complete the re-initialization.

3) Denial of service (DoS)

This is a common and destructive attacking fashion. By introducing lightweight but effective authentication, all the calculations with high complexity in YML workflow will be carried out only the authentication is passed, so the protocol can resist DoS in a great extent.

4) Replay attack

In the protocol, the user and front-end maintain a $counter$ synchronously, and the value of $counter$ is checked before verifying the correctness of $c = h\left(y \oplus counter\right)$ during the authentication process, so replay attack is resisted. Meanwhile, the attacker cannot exploit replay attack to launch flood Dos attack induced by executing high complexity calculations after passing authentication. In addition, considering packet loss happens in network channel, the difference between both sides' $counter$ is permissible within a certain range. Once the difference is overranged, then the system probably suffers from replay attack.

E. Efficiency Analysis

Let X, H, and CON respectively denote XOR operation, Hash operation, and Concatenation operation. Table I lists amounts of calculation needed in each process of our protocol. User registration (UR), Authentication (AU), Change the Password (CP). User (U), Front End (FE).

TABLE I. AMOUNTS OF CALCULATION NEEDED IN EACH PROCESS OF OUR PROTOCOL

Calculation	**UR**		**AU**		**CP**	
	U	***FE***	***U***	***FE***	***U***	***FE***
X	1	2	2	2	1	2
H	1	1	1	2	1	0
CON	0	1	0	1	0	0

Through the analysis in TABLE I, only a few lightweight calculations, including XOR, Hash, and Concatenation operation, need to operate in each process. So our protocol is feasible in computational efficiency, and can be easily integrated into the existed workflow.

IV. CONCLUSIONS AND PERSPECTIVES

Based on the analysis of the security concerns of YML framework, a lightweight authentication protocol was advanced to allow the legitimate users logging in the framework efficiently, and the protocol can resist denial of service attack, replay attack, meanwhile, the system possesses the capacity of self-healing, thus the utilization of computing resources can be controlled.

Our aim is to improve the proposed protocol by deploying and testing it in the YML framework. The testing phase will bring new perspectives and will show needed adaptation of the current method.

ACKNOWLEDGMENT

This paper is supported by National Natural Science Foundation of China: "Research on Trusted Technologies for The Terminals in The Distributed Network Environment"

(Grant No. 60903018), “Research on the Security Technologies for Cloud Computing Platform” (Grant No. 61272543), and “The National Twelfth Five-Year Key Technology Research and Development Program of the Ministry of Science and Technology of China” (Grant No. 2013BAB06B04).

REFERENCES

[1] YML Project Page, http://yml.prism.uvsq.fr

[2] PRiSM Laboratory, http://www.prism.uvsq.fr/

[3] Inria-Futurs, http://www.inria.fr/

[4] LIFL, http://www.lifl.fr/

[5] O. Delannoy, S. Petiton. A Peer to Peer Computing Framework: Design and Performance Evaluation of YML. In: Proceedings of Third International Workshop on Parallel and Distributed Computing, 2004. IEEE Computer Society, Los Alamitos, 2004. 362-369.

[6] O. Delannoy, N. Emad, S. Petiton. Workflow Global Computing with YML. In: Proceedings of the 7th IEEE/ACM International Conference on Grid Computing, GRID’06. ACM Press, New York, 2006. 25-32.

[7] L. Choy, O. Delannoy, N. Emad, S. Petiton. Federation and Abstraction of Heterogeneous Global Computing Platforms with the YML Framework. In: Proceedings of International Conference on Complex, Intelligent and Software Intensive Systems, CISIS’09. IEEE Computer Society, Los Alamitos, 2009. 451-456.

[8] L. Shang, S. Petiton, N. Emad, X. L. Yang, ZH. J. Wang. Extending YML to Be a Middleware for Scientific Cloud Computing. In: Proceedings of First International Conference on Cloud Computing, CloudCom 2009. Berlin: Springer-Verlag, 2009. LNCS 5931: 662-667.

[9] L. Shang, S. Petiton, N. Emad, X. L. Yang. YML-PC: A Reference Architecture Based on Workflow for Building Scientific Private Clouds. Cloud Computing Principles, Systems and Applications. Berlin: Springer-Verlag, 2010. Volume 0, Part 2, 145-162.