Shadow Software Attack

Angelo Rosiello
angelo@rosiello.org

Version: 1.0
Date: 12/07/2004
Copyright © Rosiello Security 2004, Italy

Abstract

In this paper, I'm going to demonstrate the fact that a shadow software attack is still possible. In fact, many users and system admins are not aware of the importance of the protection mechanisms against these kind of attacks. There are many possible solutions to resolve this scenario, but it often requires some engagement from the server and the user’s side and probably this is the very essence of the entity of the problem that we are going to face.

Introduction

During the last years we could see how shadow server[3] attacks were a serious problem for many companies. It’s true that, for a security “expert”, a shadow server attack can be considered obsolete and a “stupid” attack but in a security contest there is no banal problem, mainly if it is still feasible.

The shadow software[1] attack, discussed in this paper, is very similar to the shadow server’s one, if we abstract to its essence.

Usually, the user does not require the authentication of the server and the exchange of information begins trusting the look-and-feel of the server[3]. This is very dangerous since we don’t know if the server we are connected to is the real one.

The shadow software attack is based on the concept that an attacker could simulate the look-and-feel of a software, launched by the victim, to steal his or other people's information.

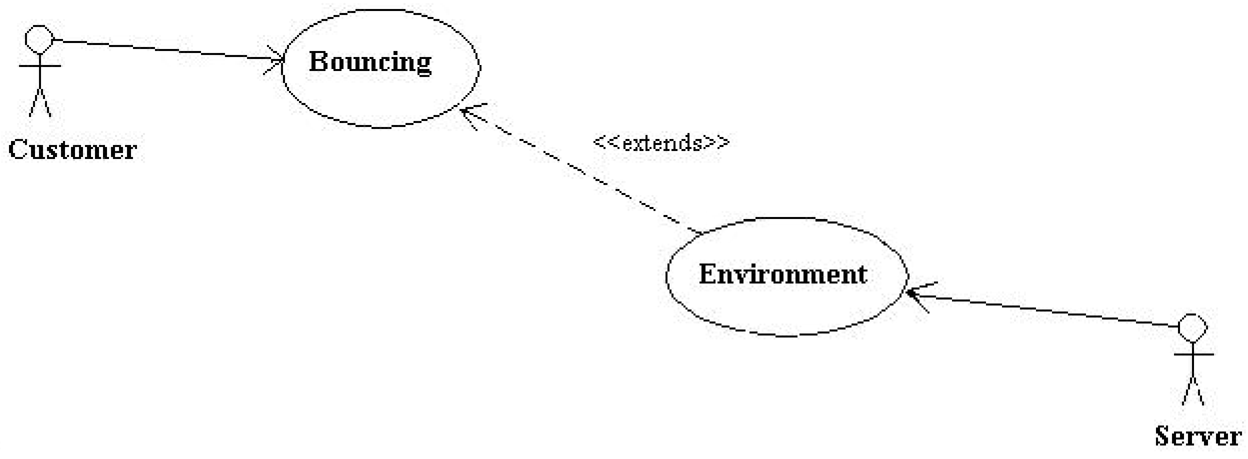
Background

To face the problem it’s important to have a detailed description of the vulnerable situation and I’m going to provide it by an example formalized with use case and sequence diagrams[4].

Example Scenario

The chosen actors of the contest are a shell provider, a customer of the shell provider (victim) and the attacker.

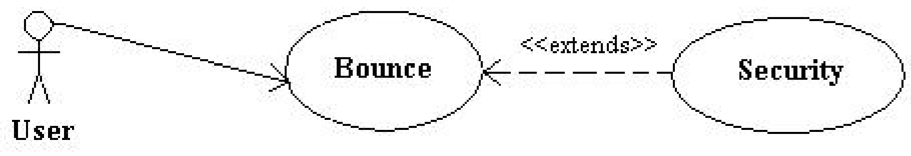
The goal of the customer is to use his account on the provider to launch a bouncer (it could be a proxy for the web, ftp, irc and so on but this attack can be brought to any similar software).



The server must provide the environment for the customer to launch his application and this is just a working shell (I'm describing a *NIX platform but the same reasoning can be done for a Windows' one).

Now, the user will download, compile, configure and launch his favourite software by his shell account.

Let's have a look at the goal of the user.



The software that the user is going to download must provide the bounce and optionally a secure way of doing it. A careful user should require that one of the goals of the bouncer be to use a <<included>> step of the security, but that's not always the paradigm. Usually, users are victims of attacks because they don't apply any preventive mechanism of defense, even if there is support in the software.

Now, that we have a background of the situation, let's try to think about a way that could be applied to steal user's information.

The common way is to compromise the server itself but this has nothing to do with a shadow software attack.

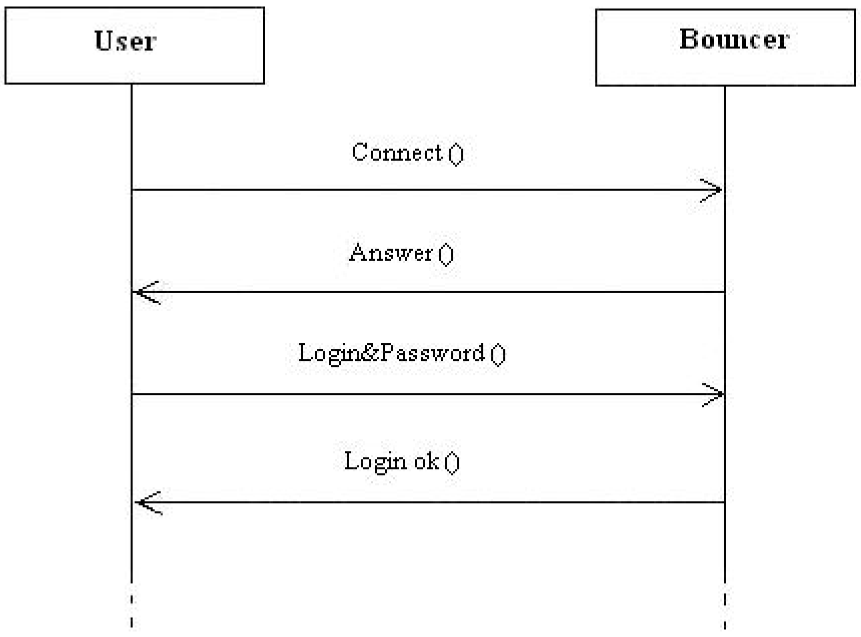
What we're going to do is to simulate the software of the victim, with a shadow software, in order to steal his logging data, then his account and password. This pattern does not need any privilege escalation or other requirements.

Analysis

A bouncer is a program that listens on a port and requires the authentication of the user to offer him an access.

I chose this software as example because it is a good abstraction for many applications. In fact, the described attack will work against any similar contest of business.

The following scenario shows the interactions between the user and his real bouncer.



This diagram is very important because the shadow software must act as the real one[3].

To have success the interaction with the user must be perfect.

In order to complete the attack the attacker must know/own:

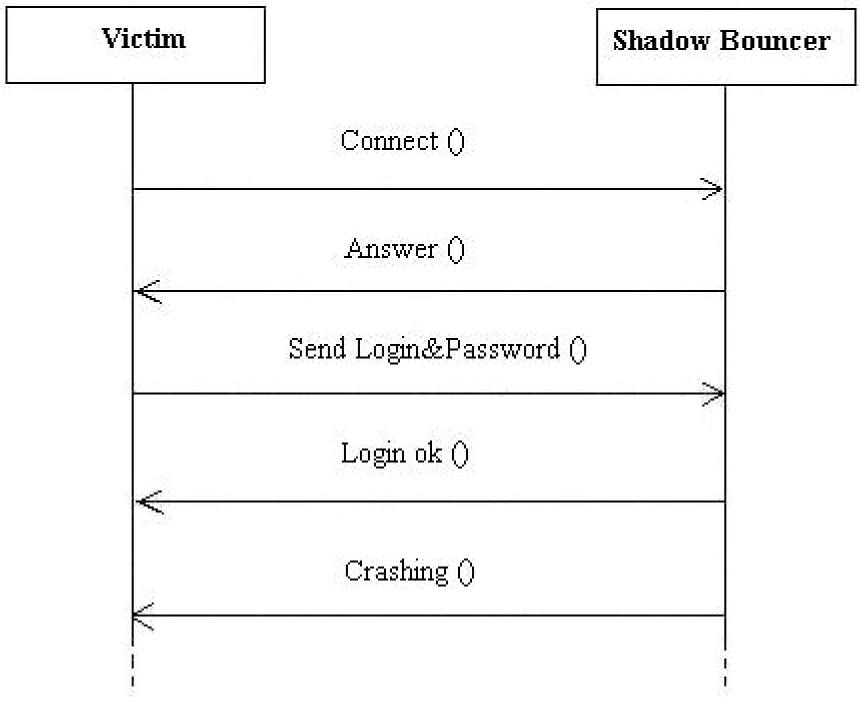
- The software to be simulated;
- The listening port;

- An account on the same machine of the victim;

The above three requirements can be easily gained, in fact, to observe the software it's enough a connection; to know the listening port a port scan will be useful, or a social engineering solution, and some money will give us an account on the same machine of the victim, but usually an attacker will not get an account paying… (then it would be too simple to catch him…).

Let the attacker satisfied all the points discussed above. He can't still launch the fake software because the port is still bound to the victim's bouncer. This "hold up" could be resolved in many ways such as crashing the server or the real bouncer or just waiting for a reboot of the machine. When the victim's software is down the victim will not detect it until he won't be able to use it (why should he gets into the shell account?), in this case he will join the shell account to restart it but this event will not happen because a good attacker will start the shadow bouncer before the user could notice that his bouncer went down.

At last the attacker could launch the shadow software that will simulate perfectly the bouncer of the victim. When the victim will execute the authentication phase, he will loose his private data, then, the shadow software will simulate a crash to evade any suspicious.



Definitely, the user lost his account and password that are in the hands of the attacker.

An example of shadow software that simulates the psyBNC can be downloaded from http://www.rosiello.org/archivio/fakepsy.c

Solutions and Comments

There are many solutions to avoid a shadow software attack. Probably, the better one is to use TLS[2], in this way we get an authentication of the server/software we're going to connect to. Another possible solution is to use a challenge-response authentication.

The above solutions and many others need a "cooperation" among the server admins, softwares and also users of the server.

A quick fix that a shell provider could apply and that is independent from the softwares of the users, is to assign a range of ports for every user, managing the bind() SYS_call (an example is provided by this lkm http://www.rosiello.org/archivio/fixbind.c but it's only a proof of concept!).

I'm not going to discuss about what solution should be implemented, because it depends on the contest, but I just want to evidence that if the scenario of the attack is possible the attack is also possible.

To conclude, the shadow software attack does not exploit a bug on the server or client but rather the unsecure way to interact with applications, thus, it's important to adopt automatic security systems to detect and prevent it.

References

[1] Angelo Rosiello – Shadow software Attack still works. http://www.rosiello.org, April 2004.

[2] The Internet Society - RFC 2246 - The TLS Protocol Version 1.0. The Internet Society, January 1999.

[3] A.Lioy – Volume V Sistemi Distribuiti. Franco Angeli, 2001.

[4] Jim Conallen – Web Applications with UML. Pearson, 2003.