Bites-PenTesting

Penetration Test Report

Client:

Date of test:

D I S C L A I M E R

This report is intended only for the use of the individual or entity
to which it is addressed and may contain information that is privileged, confidential and exempt from disclosure under applicable law. If the reader of this disclaimer is not the intended recipient, you are hereby notified that any dissemination, distribution or copying of this document is strictly prohibited. If you received this document in error, please notify us immediately by telephone and return the original document to us at the post address below.

Thank you

Contents

1 Introduction to the penetration test 3
1.1 Some Definitions ……..……3
1.2 Motivation of an Attacker ……4

2 The Penetration Test ……5
2.1 IP Authentication Vulnerability ……..…..…..5
2.2 Reverse DNS Vulnerability ……6
2.3 Standard Universe Job Vulnerability ……..6

3 Conclusions ……..……7

Chapter 1

Introduction to the penetration test

The aim of this penetration test is to help the administrator of the com- pany to secure the network. Although this report contains technical terms, it has been written so that a non-initiated reader with a basic knowledge of computing would understand it. However, references to more technical content, to be found in the appendices, is given along the test report for the administrator and security consultant of Bites-PenTesting to review them and possibly reproduce the test. Should the reader meet difficulties at un- derstanding the penetration test report, going directly to the "Conclusions and Recommendations" section will give him the executive information. For further help, we remain open to answer any of your questions.

In order to increase the understanding of the reader, some definitions and clarifications are given in the following sections.

1.1 Some definitions

- Hacker: word given by the masse media to define what we will more accurately call attacker or intruder in this report.

- Vulnerability: a bug in computer program that may be abused to gain privileges on a computer.

- Exploit: a program or strategy to exploit a vulnerability. Depending on the vulnerability, an exploit may be either local, in which a previous "local" access to the target computer is required prior gain
  higher privileges, or remote where the exploit can be run without this prerequisite.

- Rootkit: a set of programs replacing the tools, that an administrator would generally use to detect the presence of an intruder, by modified versions detecting everything but the presence and activities of the intruder, thus making the administrator confident that the system is free of any intrusions.

1.2 Motivation of an attacker

There are mainly three reasons why someone might want to penetrate your network.

- Information theft: to steal valuable information of your business such as contracts, documents or e-mail communication. In other words, information that, for example, competitors may like to know.

- Identity theft: by using your network as relay to attack other networks, an attacker can mask his identity.

- Challenge to overcome: to most attackers, your network represents a challenge that must be conquered or a way to prove their superior intelligence and technical skills.

Understanding the psychology of an attacker helps considering why your network is at risk whenever it is connected to the Internet and how to protect it. Indeed, whatever the final motivation really is, gaining access to a network always remains a challenge for an attacker. Though intruding a network is rewarding for his ego, failing to gain the access brings a high level of frustration. An attacker, usually, doesn't give up easily and will try, again and again, by any means, to get all kind of information that might be useful to detect weaknesses and mount attacks.

Therefore, while performing the penetration test, we have been through the same stages as an attacker would have, even though our strategy or tools might be slightly differ.

Chapter 2

The penetration test

The penetration test was carried out from one Linux PC inspecting your Cloud IaaS (Infrastructure as a service) Virtual machine cluster software, more specifically, Condor. The test is limited to service models you are responsible for as agreed. These services are, API/GUI, Application, Solution Stack and the Virtual machine.

This is not a blackbox test as we have discussed in some length your service and systems therefor I can skip over any host discovery, network mapping, port mapping and most information gathering techniques.

The vulnerabilities found were not in direct response to any command or request from out attacking computer or indeed from the host server therefor to prevent this report becoming an ineligible mess of computer code I have left out most of commands and packets that were used in their discovery.

2.1 IP Authentication Vulnerability

The server is using IP based authentication, the set of IP addresses that are allowed can be more permissive than expected when denying IP addresses. This can allow an attacker to perform actions against the Condor daemon that should not be allowed.

If multiple IP address ranges are specified and the netmask is not the same in each all the IP address ranges will be ignored. In the case of HOSTALLOW options this will make the list of hosts that can perform the operation more restrictive than expected, but in the case of HOSTDENY options it will make the list of IP addresses that can perform the operation less restrictive.
In the example below, machines with IP address in the range 192.168.0.0 to 192.168.0.255 and 10.0.0.0 to 10.0.255.255 should not be allowed to perform adminstrative commands, but this will not be the case:

```
HOSTDENY_ADMINISTRATOR = 192.168.0.* 10.0.*
```

Instead no hosts will be denied, since the netmask portion of the 192.168.0.* is 24 bits while the network portion of 10.0.* is 16 bits.

Cause:

logic error
Coding error.

Fix:

The code was fixed to treat each specified network/netmask pair individually and to not ignore them if the number of bits in the netmask varied.

2.2 Reverse DNS Vulnerability

When looking up information, Condor does not validate that the DNS name returned actually points to the IP address claimed. An attacker who controls an IP, its reverse-DNS entry and has knowledge of a target site's security configuration. With this control and knowledge, the attacker can bypass the target site's host-based authentication and be authorised to perform privileged actions (i.e. actions requiring ALLOW_ADMINISTRATOR or ALLOW_WRITE). Condor deployments using host-based authentication that contain no hostnames (IPs or IP globs only) or use authentication stronger than host-based are not vulnerable.

If an attacker is able to modify their own reverse DNS records, and can connect to the Condor daemon, they may be authorised to perform privileged actions. To exploit this, an attacker just needs to be have configuration knowledge of the target Condor daemons. If an attacker is successfully able to circumvent the authorisation, they may perform actions as the Condor administrator (such as turning off Condor) or potentially as other users of the system (such as running a job).

Fix:
Check the DNS mapping.

2.3 Standard Universe Job Vulnerability

Your Condor service supports Standard Universe jobs and run the daemons on the submit machine as root are vulnerable to local privilege escalation. If a user submits a job into the standard universe, the user job may then execute code on the submit machine as the root user. If your Condor installation does not contain the condor_shadow.std executable, then you are not affected by this vulnerability.

Any person who can submit standard universe jobs to the condor_schedd can exploit this. Submissions are authenticated and are typically done locally. However, if Condor is configured to allow remote submits, jobs submitted remotely into the standard universe can also exploit this. To exploit this, an attacker just needs to know the correct sequence of communications with the condor_shadow.std. If an attacker is successfully able to communicate correctly with the condor_shadow.std, they may instruct the shadow to run arbitrary code as the root user, including spawning additional processes. Condor should never spawn user processes as root, and makes explicit checks in most places to ensure this never happens. In the standard universe shadow, an unrelated change opened a new code path where privilege checking did not exist.

Fix:
Remove the code, as it should never be used.

Conclusion

The security issues found on the server are very severe and for the most part do not require a great deal of effort to exploit however the fixes are simple and easy to implement. Due to the nature of this program I would recommend if at all possible to make these servers private only and sit them behind a tightly controlled SSH server to stop remote attacks to the condor service directly. You may see a rise in SSH attacks however they can be managed and mitigated much more easily.