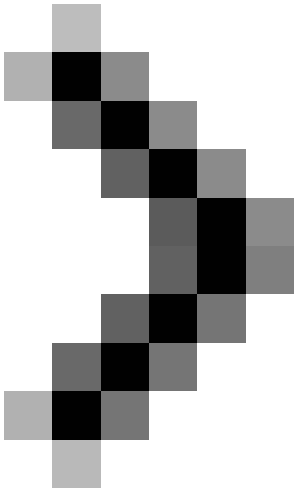
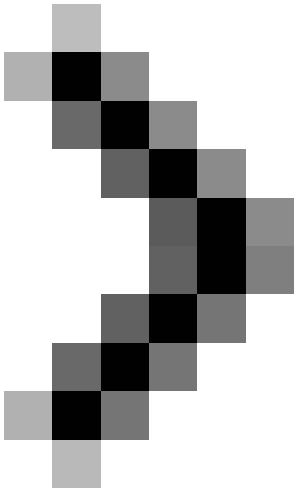
Analysis of an attack (Part 3)

In Part 2 of this series, we have left all the necessary information required for an attack on the victim network. With that note, let's continue with a real attack. This attack follows the transmission of ecommerce



Analysis of an attack (Part 1)



Analysis of an attack (Part 2)

Don Parker

In Part 2 of this series, we have left all the necessary information required for an attack on the victim network. With that note, let's continue with a real attack. This attack follows the transmission on some required programs to be able to go deeper into exploiting an attack. It is really meaningless to simply attack a computer and then withdraw, so we will do a strong attack. Usually the goal of a malicious attacker is not only to increase the presence on the computer network but also to maintain it. That means the attacker also wants to continue hiding his presence and perform other actions.

Interesting issues

Now we will use Metasploit Framework to facilitate a real attack. This mechanism of work is really interesting because it provides you with different types of exploits as well as various options in the choice of load. You may not want a reverse utility, or inject VNC. Loads often depend on your upcoming goals, network architecture and ultimate goals. In this case, we'll do it with a reverse utility. This is often a way of many advantages, especially in case our target is behind the router and not directly accessible. For example, you 'hit' a webserver but the load is still balanced. Not sure if you will be able to connect to it with a forward-facing utility, so you want the computer to create a utility back. We will not discuss how to use the Metasploit Framework because it may have been introduced in another article. So we just focus on things like package levels.

Now, instead of using the method of introducing each attack step with brief images and excerpts, we will give

another attack. What will be done is to recreate the attack with the help of Snort. We will take advantage of the binary record in the attack that we have implemented, then parse it through Snort. Ideally it will see everything like what we have done. In fact, what will be done is a demonstration package. The goal here is to see if it is possible to reassemble exactly what happened. With that in mind, we will use the binary package record that has written everything done and parsed through Snort through some of its default rules.

Output Snort

The syntax used to call Snort is as follows:

C: snortbinsnort.exe –rc: article_binary –dv –c snort.conf –A full

This syntax makes Snort analyze binary packages called article_binary, the results are given below. We truncated Snort's output so we could look at each part in detail.

========== ==========
Snort processed 1345 packets.
========== ==========
Breakdown by protocol:
TCP: 524 (38.959%)
UDP: 810 (60.223%)
ICMP: 11 (0.818%)
ARP: 0 (0.000%)
EAPOL: 0 (0.000%)
IPv6: 0 (0.000%)
ETHLOOP: 0 (0.000%)
IPX: 0 (0.000%)
FRAG: 0 (0.000%)
OTHER: 0 (0.000%)
DISCARD: 0 (0.000%)
========== ==========
Action Stats:
ALERTS: 63
LOGGED: 63
PASSED: 0

This section is very attractive because up to 63 alerts have been triggered by an attack. We will take a look at the alert.ids file, which is a file that can give more details about what happened. Now, if you remember the first thing an attacker did was to use Nmap to perform network scanning, that problem also created the first warning that was triggered by Snort.

[**] [1: 469: 3] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
08 / 09-15: 37: 07.296875 192.168.111.17 -> 192.168.111.23
ICMP TTL: 54 TOS: 0x0 ID: 3562 IpLen: 20 DgmLen: 28
Type: 8 Code: 0 ID: 30208 Seq: 54825 ECHO

[Xref => http://www.whitehats.com/info/IDS162]

In this way, the attacker used netcat to list the webserver to find out what kind of webserver is. This action did not trigger any Snort alerts. We also want to find out what happened, so let's take a closer look at the package record. After observing the usual TCP / IP handshake procedure, we will see the package below.

15: 04: 51.546875 IP (tos 0x0, ttl 128, id 9588, offset 0, flags [DF], proto: TCP (6), length: 51) 192.168.111.17.1347> 192.168.111.23.80: P, cksum 0x5b06 (correct), 3389462932: 3389462943 (11) ack 2975555611 win 64240
0x0000: 4500 0033 2574 4000 8006 75d7 c0a8 6f11 E.3% t @ . u . o.
0x0010: c0a8 6f17 0543 0050 ca07 1994 b15b 601b .o.CP . [`.
0x0020: 5018 faf0 5b06 0000 4745 5420 736c 736c P . [. ***GET.slsl***
0x0030: 736c 0a ***sl*** *.*

There is nothing remarkable in this package other than the fact that it has a GET request with some of the following internal issues such as slslsl. So in fact, there is nothing for Snort to act on. Therefore, it is very difficult to build a signature (or can call an IDS sign) effectively to enable this type of listing attempt. That is why there are no such signatures. The next package is where the victim network's webserver lists itself.

After the enumeration is performed, the attacker immediately sends a code to execute the exploit to the webserver. This code will then give some results with Snort signatures enabled. Especially for the exploit shown below, we can see this Snort signature.

[**] [***1:12*** *: 13]* ***WEB-FRONTPAGE rad fp30reg.dll access*** *[**]*
[Classification: truy c?p ?? có th? dùng web ?ng d?ng] [Priority:
2] 08 / 09-15: 39: 23.000000 192.168.111.17:1454 -> 192.168.111.23:80
TCP TTL: 128 TOS: 0x0 ID: 15851 IpLen: 20 DgmLen: 1500 DF
**** A **** Seq: 0x7779253A Ack: 0xAA1FBC5B Win: 0xFAF0 TcpLen: 20*
[Xref => http://www.microsoft.com/technet/security/bulletin/MS01-035.mspx][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0341][Xref => http:///www.securityfocus.com/bid/2906][Xref => http://www.whitehats.com/info/IDS555]

When an attacker has increased access to the webserver, he will start using the TFTP client to transmit four files: nc.exe, ipeye.exe, fu.exe, msdirectx.exe. After these files have been transferred, the attacker uses netcat to send a utility back to his computer. From there, he can disconnect and other utilities, the utility has resulted from the initial attack and performed all the remaining work in the netcat utility. Very attractive, no action taken by the attacker via the reverse utility was recorded by Snort. However, regardless of the problem, the attacker used the rootkit he transmitted via TFTP to hide the process information for netcat.

Conclude

In the third part of this series, we saw the attack demonstrated when using Snort. We can completely recreate one of the things that have been done except the use of rootkits. Even if the IDS is a fairly useful piece of technology and is part of your network protection system, it is not always perfect. IDSs can only warn you about the traffic it can sense. Notice that we will learn how to build Snort markers in the final part of this series. Along with that, we will learn how to test a digital signature (sign) to verify their effectiveness.

You finished reading the article "**Analysis of an attack (Part 3)**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and

guides. Thank you for reading and for following us regularly.

© 2019 TipsMake.com