

Analiza w³amania od A do Z (czê¶æ 3)

Autor: Marek
26.05.2008.
Zmieniony 30.06.2008.

W czê¶ci 2 zakończyliśmy zbieranie wymaganych informacji z sieci docelowej. W czê¶ci 3 dokonamy w³amania i prześlemy niektóre narzędzia do zaatakowanego serwera sieciowego.

Teraz przejdziemy do samego w³amania. Po nim nastąpi przekazanie niektórych programów wymaganych, aby haker mógł ponownie uzyskać dostęp do komputera ofiary. Nie byłoby sensu w³amać się do komputera, a potem się wycofać. Zazwyczaj celem jakiegokolwiek z³o¶liwego hakera jest nie tylko wej¶cie do sieci komputera, ale także utrzymanie się w niej. Oznacza to próbê ukrycia swojej obecno¶ci.

Czas na zabawê

Teraz skorzystam z Metasploit Framework, aby u³atwiæ samo w³amanie. To narzędzie jest naprawdę piêkn± spraw±, oferuj±c± wielk± ró¿norodno¶æ exploitów i – co jest tak samo wa¿ne – wielu ró¿nych opcji „payload” do wyboru. Pewnie nie zawsze bêdziesz chcia³ mieæ zwrotn± pow³okê albo wprowadzaæ VNC. Payload bêdzie czêsto zale¿eæ od najbli¿szego celu, architektury sieci i Twojego celu koñcowego. W naszym przypadku pos³ujemy się pow³ok± zwrotn±. Jest to zawsze korzystne, a szczególnie w przypadku, gdy Twój cel jest za routerem i nie jest bezpo¶rednio dostêpny. Na przyk³ad: uderzasz w serwer sieciowy, ale jest on jednym z kilku, które mają wyrównane obci±¿enie. Nie ma gwarancji, ¿e bêdziesz w stanie się z nim po³±czyæ za pomoc± pow³oki typu „forward” i dlatego chcesz, aby komputer przerzuci³ pow³okê z powrotem do Ciebie. Nie bêdê omawia³ rzeczywistego u¿ycia Metasploit Framework do tego celu, poniewa¿ jest to co¶, co omawia³em ju¿ kilka razy w innych artyku³ach. Skupimy się na tym, jak to wszystko wygl±da na poziomie pakietu.

Teraz zamiast omawiania ka¿dego kroku w³amania za pomoc± zrzutów ekranu i fragmentów kodu, przyjmiemy inn± taktykê. To co zrobimy, to odtworzenie w³amania za pomoc± systemu Snort. Przyjmiemy binarny rejestr w³amania, którego dokona³em, a nastêpnie sparsujê go do systemu Snort, aby zobaczyæ to, co on zobaczy³. Idealnie: zobaczy³ to wszystko, co i ja zobaczy³em. W rzeczywisto¶ci to, co zrobimy bêdzie analiz± pakietu. Celem bêdzie zobaczenie tego, jak precyzyjnie możemy przej¶æ do z³o¿enia dok³adnie tego, co się sta³o. Przyjmê tutaj binarny rejestr pakietu, który nagra³ wszystko co zrobi³em i sparsujê to do systemu Snort ze swoim domy¶lnym zestawem regu³ na w³a¶ciwym miejscu.

Dane wyj¶ciowe systemu Snort

Składnia użyta do wywołania snortu jest następująca:

```
C:\snort\bin\snort.exe -r c:\article_binary -dv -c snort.conf -A full
```

To z kolei spowodowało, że Snort sparsował pakiet binarny nazwany article_binary, czego wynikiem były poniższe dane wyjściowe. Skróciłem dane wyjściowe systemu Snort do istotnych części, aby się nie rozpisywał.

```
=====
```

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

Interesuje nas przede wszystkim fakt, że 63 alerty były spowodowane czynnościami w³amania. Teraz przyjrzymy siê plikowi alert.ids, który da nam bardziej szczegó³owe informacje na temat tego, co siê wydarzy³o. Teraz, je¶li pamiętasz, pierwsz± rzeczą, jak± zrobi³ haker by³o u¿ycie Nmap do skanowania sieci. Tak siê sk³ada, że jest to pierwszy alert wywo³any przez 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>]

Nastêpnie haker u¿y³ narzêdzia netcat do okre¶lenia serwera sieciowego, aby dowiedzieæ siê jakiego by³ typu. Ta próba okre¶lenia nie wywo³a³a ¿adnych alertów Snort. Z ciekawo¶ci d³aczego tak siê sta³o przygl±dnie my siê rejestrowi pakietu. Zaraz po zobaczeniu normalnego porozumienia TCP/IP typu "three way handshake", widoczny by³ nastêpuj±cy pakiet:

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, csum 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..C.P.....[.

0x0020: 5018 faf0 5b06 0000 4745 5420 736c 736c P...[...GET.sls]

0x0030: 736c 0a

sl.

Nie ma nic zadziwiającego w tym pakiecie ponad fakt, że wystąpiła próba GET z paroma bezużytecznymi danymi jak potwierdza slslsl. Tak więc w rzeczywistości Snort nie miał czego wywołać. Raczej trudno byłoby zbudować skuteczny podpis IDS do wywołania tej próby określenia systemu. I prawdopodobnie dlatego nie ma takich podpisów. Kolejnym pakietem po tym jest ten, gdzie serwer sieciowy ofiary określił sam siebie.

Po wyciszeniu haker natychmiast wysłał kod "exploit" do serwera sieciowego. Kod ten powoduje, że wywołanych zostaje kilka podpisów Snort. Zobaczysz taki podpis Snort dla konkretnego exploitu pokazanego poniżej.

```
[**] [1:1248:13] WEB-FRONTPAGE rad fp30reg.dll access [**]
[Classification: access to a potentially vulnerable web application] [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.s
ecurityfocus.com/bid/2906][Xref => http://www.whitehats.com/info/IDS555]
```

Następnie po tej serii alertów, Snort uruchamia serię alertów TFTP. W momencie gdy haker zdobył dostęp do serwera sieciowego, rozpoczął stosowanie wbudowanego klienta TFTP do przesłania czterech plików: nc.exe, ipeyes.exe, fu.exe, msdirectx.exe. Po przetransportowaniu tych plików, haker zastosował netcat, aby wysłać powłokę z powrotem do swojego komputera. Od tego momentu rozpoczęła poprzednia powłoka, której użył do zapoczątkowania ataku, a cała pozostała praca wykonana w powłoce netcat. Interesującym jest fakt, że żadna z postępujących aktywności dokonanych przez hakera poprzez powłokę zwrotną nie została odnotowana przez Snort. Nie została odnotowana, chociaż haker zastosował rootkit, przetransportowany przy użyciu TFTP, w celu ukrycia informacji procesowej dla netcat. Może to być rozczarowaniem dla kogoś, kto oczekiwania, że wystarczy użyć rootkita fu.exe do uruchomienia podpisu IDS.

Podsumowanie

W trzeciej części tej serii artykułów zobaczyliśmy wspaniałe, które zostały dokonane przez hakera jak to widział Snort.

Byliśmy w stanie odtworzyć prawie wszystko, co zostało zrobione z wyjątkiem użycia rootkita. I chociaż IDS jest całkiem pomocną technologią i powinna być częścią zabezpieczenia Twoich sieci, nie jest ona idealna. Będzie Cię powiadamiać tylko o ruchu w sieci, dla których posiada podpisy. Mając to na uwadze, powinniśmy się nauczyć, jak budować podpisy Snort i to będzie tematem ostatniej części tej serii artykułów. Ponadto zobaczymy także jak sprawdzić te podpisy, aby zweryfikować ich efektywność. Do zobaczenia.

Źródło: www.windowsecurity.com