DNS-Adblocker Skript für IPFire

1. Werbung und Tracker waren gestern IPFire Adblock

Ein Adblocker zählt heute zur **Standardausrüstung** der digitalen Selbstverteidigung, um sich gegen die Auslieferung von schadhafter Online-Werbung bzw. Malvertising zu schützen. Mit dem Pi-hole hatte ich euch bereits eine Lösung vorgestellt, mit dem sich schadhafte Werbung und Tracker auf DNS-Ebene filtern lassen. Der Vorteil einer solchen Lösung liegt auf der Hand: Wir können damit Werbung und Tracker auch abseits von Browsern blockieren und damit Geräte schützen, auf die sich kein Adblocker oder ähnliches installieren lässt.

Im zweiten Teil der IPFire-Artikelserie möchte ich euch ein Skript vorstellen, mit dem sich, ähnlich wie auf dem Pi-hole, bereits auf DNS-Ebene Werbung und Tracker filtern lassen.

DIESER BEITRAG IST TEIL EINER ARTIKELSERIE:

- Hardware und Netzwerkaufbau – IPFire Teil1
- DNS-Adblocker Skript für IPFire – IPFire Teil2
- ASN-Skript: Datensammler haben ausgeschnüffelt – IPFire Teil3
- Windows unter Kontrolle – IPFire Teil4

2. dns_blocker.sh

Das Skript mit dem Namen dns_blocker.sh wird von sfeakes auf GitHub zum Download angeboten und verwandelt die IPFire in einen DNS-Adblocker. Auf Basis von Filterlisten wird die Auslieferung von Werbung, Trackern oder schadhaften Seiten bereits auf DNS-Ebene unterdrückt. Das Skript ist sowohl mit dnsmasq, als auch mit unbound kompatibel, das seit Version 2.19 Core 106 dnsmasq als DNS-Resolver abgelöst hat.

Du kannst den Blog aktiv unterstützen!
Mitmachen ➡

2.1 Technischer Hintergrund

Am Beispiel von In-App Werbung möchte ich kurz erläutern, wie das DNS-Adblocking **technisch** funktioniert. Angenommen ein App-Entwickler hat in seiner App Werbung integriert. Bei jedem Start der App oder auch während der Laufzeit wird daher die Adresse »werbung.server1.de« aufgerufen. Dieser Domainname muss allerdings zunächst in eine IP-Adresse übersetzt werden, damit die Werbung anschließend von dort geladen werden kann. Dieser Service wird vom Domain Name System (DNS) erledigt – einer der **wichtigsten** Dienste im Internet, der Domainnamen in die zugehörige IP-Adresse umwandelt. Das kennt jeder: Ihr gebt im Browser eine URL ein (also den Domainnamen), dieser wird dann von einem DNS-Server in die zugehörige IP-Adresse übersetzt. Namen lassen sich eben leichter merken, als IP-Adressen. In eurem Router / der IPFire sind daher üblicherweise DNS-Server von eurem Provider hinterlegt oder ihr habt **manuell** eigene eingetragen, die dann anschließend die Adresse »werbung.server1.de« in eine IP-Adresse umwandeln.

Das DNS-Prinzip macht sich das dns_blocker.sh Skript zunutze. Intern verwaltet die IPFire nach der Anwendung des Skripts eine Liste mit Domainnamen, die Werbung ausliefern oder den Nutzer tracken. Zurück zum Beispiel: Will das Smartphone die Adresse »werbung.server1.de« aufrufen, wird die Anfrage von unbound (auf der IPFire) mit einer Liste abgeglichen. Kommt es zu einem **Treffer**, wird die Übersetzung in die korrekte IP-Adresse unterdrückt und stattdessen bspw. eine »127.0.0.1« (localhost) zurückgeliefert. Die Folge: Die Werbung kann nicht von der eigentlichen Quelle bzw. IP-Adresse **nachgeladen** werden. Anstatt der Werbung sieht der Nutzer einen Platzhalter bzw. einfach nichts. Ein einfaches Prinzip, dass die Werbung noch vor der Auslieferung -ja sogar noch vor der Übersetzung in die IP-Adresse- blockiert.

2.2 Parameter

Über diverse Parameter könnt ihr das Verhalten des Skripts steuern und an eure Bedürfnisse anpassen. So ist es bspw. möglich weitere Filterlisten zu laden, die anschließend vom Skript in unbound eingebunden werden. Hier eine Übersicht:

```
./dns_blocklist.sh
Parameters are the following, only use one of the formats, -p OR --
parameter, do not use both

-h --help This message
-l --listsources list sources available with index number
-w --whitelist Use a white list file
-b --blacklist Use a blacklist file
-r --dns Set the dns return value
-u --force_unbind Force script to use unbind
-d --force_dnsmasq Force script to use dnsmasq
-o --outfile output to filename, do not restart any services
-s --sourcelist list sources to retreive blacklist from (must be comma
seperated) use index number from -l value or URL

Beispiel:- ./dns_blocklist.sh -s 1,2,http://mylist.com/host.txt -r 0.0.0.0
</file>
===== 3. Installation & Konfiguration =====

Die Installation bzw. die Nutzung des Skripts auf der IPFire ist denkbar
einfach und binnen wenigen Minuten erledigt. Loggt euch zunächst mittels
[[https://wiki.ipfire.org/configuration/system/ssh|SSH]] auf eure IPFire ein
und führt anschließend folgende Befehle aus:<code>cd ~
curl -O
https://raw.githubusercontent.com/sfeakes/ipfire-scripts/master/dns_blocklis
t.sh
chmod 755 dns_blocklist.sh
```

Ihr könnt das Skript anschließend editieren / anpassen, also bspw. die Standard-Quellen für die Filterlisten ändern. Das ist aber nicht notwendig, da sich im Prinzip alles über die entsprechenden Parameter steuern lässt.

3.1 Custom Blacklist / Whitelist

Wenn ihr selbst Domainadressen hinzufügen wollt, die anschließend von unbound blockiert werden sollen, könnt ihr dies über die Custom Blacklist tun. Hier ein Beispiel mit Adressen von Facebook:

```
## Facebook
(https://developers.facebook.com/docs/workplace/additional-configuration/dom
ains)
facebook.com
akamaihd.net
fbcdn.net
fb.me
fbsbx.com
```

Mit dem Parameter –blacklist lässt sich diese Liste anschließend laden:

```
dns_blocklist.sh —b /pfad/blacklist.hosts
```

Wenn Seiten hingegen nicht mehr funktionieren wie gewohnt, weil sie bspw. Teil einer Filterliste sind, dann könnt ihr eine Whitelist anlegen. Das Vorgehen ist vergleichbar mit der Blacklist, nur der Aufruf ist dann etwas anders:

```
dns_blocklist.sh —w /pfad/whitelist.hosts
```

Nach der Ausführung des Skripts befinden sich alle blockierten Domains unter folgendem Pfad:

```
/etc/unbound/local.d/blocklist.conf
```

Wollt ihr bspw. herausfinden, ob Twitter blockiert wird, dann könnt ihr dies folgendermaßen tun:

```
cat /etc/unbound/local.d/blocklist.conf | grep twitter.com
```

3.2 Skript ausführen

Über die Parameter könnt ihr das Skript an eure Bedürfnisse anpassen. Anbei alle Parameter, die ich dem Skript übergebe:

Zunächst die Filterlisten, die ich um einige Quellen erweitert habe. Die mitgelieferten Filterlisten lassen sich über die Zahlen (hier: 1,2,3,5,7,8,12,13,14) ansteuern. Weitere Filterlisten könnt ihr von Hochkommatas umschlossen hinzufügen. Sollte es zu doppelten Einträgen bzw. Überschneidungen kommen, werden diese vom Skript erkannt und entfernt. Nach Ausführen des Skripts werden ca. 100.000 Domains vor der Auslieferung blockiert.

```
-s
1,2,3,5,7,8,12,13,14,"https://easylist.to/easylistgermany/easylistgermany.tx
t","https://easylist.to/easylist/easyprivacy.txt","https://easylist.to/easyl
ist/fanboy-
social.txt","https://raw.githubusercontent.com/crazy-max/WindowsSpyBlocker/m
```

```
aster/data/hosts/extra.txt","https://raw.githubusercontent.com/crazy-max/Win
dowsSpyBlocker/master/data/hosts/spy.txt","https://s3.amazonaws.com/lists.di
sconnect.me/simple_tracking.txt","https://s3.amazonaws.com/lists.disconnect.
me/simple_ad.txt","https://raw.githubusercontent.com/AdguardTeam/AdguardFilt
ers/master/MobileFilter/sections/adservers.txt","https://raw.githubuserconte
nt.com/AdguardTeam/AdguardFilters/master/MobileFilter/sections/spyware.txt"
```

Der Pfad zur Blacklist:

```
-b /root/blacklist.hosts
```

Und die Antwort von unbound, wo sich das Ziel befindet:

```
-r 0.0.0.0
```

Anstatt mit einem »127.0.0.1« zu antworten, erhält der Client bei einem Treffer innerhalb der Filterliste ein »0.0.0.0« zurückgeliefert. StevenBlack empfiehlt dieses Vorgehen aus folgendem Grund:

Traditionally most host files use 127.0.0.1, the loopback address, to establish an IP connection to the local machine.

We prefer to use 0.0.0.0, which is defined as a non-routable meta-address used to designate an invalid, unknown, or non applicable target.

Using 0.0.0.0 is empirically faster, possibly because there’s no wait for a timeout resolution. It also does not interfere with a web server that may be running on the local PC.

Ein vollständiger Aufruf würde also folgendermaßen aussehen:

```
bash /root/dns_blocklist.sh -s
1,2,3,5,7,8,12,13,14,"https://easylist.to/easylistgermany/easylistgermany.tx
t","https://easylist.to/easylist/easyprivacy.txt","https://easylist.to/easyl
ist/fanboy-
social.txt","https://raw.githubusercontent.com/crazy-max/WindowsSpyBlocker/m
aster/data/hosts/win7/extra.txt","https://raw.githubusercontent.com/crazy-ma
x/WindowsSpyBlocker/master/data/hosts/win7/spy.txt","https://s3.amazonaws.co
m/lists.disconnect.me/simple_tracking.txt","https://s3.amazonaws.com/lists.d
isconnect.me/simple_ad.txt","https://raw.githubusercontent.com/AdguardTeam/A
dguardFilters/master/MobileFilter/sections/adservers.txt","https://raw.githu
busercontent.com/AdguardTeam/AdguardFilters/master/MobileFilter/sections/spy
ware.txt" -b /root/blacklist.hosts -r 0.0.0.0
```

3.3 Automatisierung

Die Filterlisten sollten regelmäßig neu geladen und für unbound aktualisiert werden. Über fcrontab könnt ihr das Updateverhalten auf der IPFire steuern:

```
fcrontab -e
```

Legt dort einen neuen Eintrag an, um bspw. jeden Tag um 23:30 Uhr die aktualisierten Listen zu

laden:

```
# 23 Uhr
# Update DNS blocking lists
30 23 * * *     bash /root/dns_blocklist.sh -s
1,2,3,5,7,8,12,13,14,"https://easylist.to/easylistgermany/easylistgermany.tx
t","https://easylist.to/easylist/easyprivacy.txt","https://easylist.to/easyl
ist/fanboy-
social.txt","https://raw.githubusercontent.com/crazy-max/WindowsSpyBlocker/m
aster/data/hosts/win7/extra.txt","https://raw.githubusercontent.com/crazy-ma
x/WindowsSpyBlocker/master/data/hosts/win7/spy.txt","https://s3.amazonaws.co
m/lists.disconnect.me/simple_tracking.txt","https://s3.amazonaws.com/lists.d
isconnect.me/simple_ad.txt","https://raw.githubusercontent.com/AdguardTeam/A
dguardFilters/master/MobileFilter/sections/adservers.txt","https://raw.githu
busercontent.com/AdguardTeam/AdguardFilters/master/MobileFilter/sections/spy
ware.txt" -b /root/blacklist.hosts -r 0.0.0.0
```

3.4 Modifikation rückgängig machen

Wenn ihr alle blockierten Werbe- und Tracker-Domains wieder entsperren wollt, dann könnt ihr einfach folgende Datei entfernen und unbound neustarten.

```
rm /etc/unbound/local.d/blocklist.conf && /etc/init.d/unbound restart
```

4. Modifikation: Für Fortgeschrittene

Im Normalfall liefert unbound bei einem Treffer eine IP-Adresse an den Client zurück. Das ist entweder die »127.0.0.1« oder die »0.0.0.0«, je nachdem wie ihr den Parameter gewählt habt. Schön wäre allerdings, wenn dem Client der Status Code NXDOMAIN zurückgeliefert wird, wenn eine Domain innerhalb der Liste angefragt wird. Die Bezeichnung NXDOMAIN bedeutet:

Non-Existent Domain

Wenn ihr das Skript nicht modifiziert habt, dann wird euch ein nslookup für die Adresse »facebook.com« folgende Antwort liefern:

```
nslookup facebook.com

Server:   192.168.150.1
Address:    192.168.150.1#53

Name:    facebook.com
Address: 127.0.0.1
```

Das Ziel, also facebook.com, befindet sich demnach auf 127.0.0.1 (localhost) und wird dort angefragt. Das ist ein unnötiger Vorgang, da sich hinter der 127.0.0.1 natürlich nicht das entsprechende Ziel verbirgt, sondern dann einfach dort ins Leere läuft. Die folgende Antwort wäre ideal:

```
nslookup facebook.com

Server:   192.168.150.1
Address:    192.168.150.1#53

** server can't find facebook.com: NXDOMAIN
```

Weitere Versuche das Ziel zu erreichen werden Aufgrund der Antwort NXDOMAIN direkt abgebrochen, da das Ziel nicht aufgelöst werden kann bzw. nicht existiert. Wer das erreichen möchte, der kann an das Skript einfach einen sed-Befehl anhängen, der die unbound Einträge entsprechend modifiziert:

```
sed -i 's/local-data/local-zone/g; s/[[:space:|]]A 127.0.0.1"/" static/g'
/etc/unbound/local.d/blocklist.conf
```

Wenn ihr alles in einem Aufruf kombinieren wollt, dann würde das folgendermaßen aussehen:

```
bash /root/dns_blocklist.sh -s
1,2,3,5,7,8,12,13,14,"https://easylist.to/easylistgermany/easylistgermany.tx
t","https://easylist.to/easylist/easyprivacy.txt","https://easylist.to/easyl
ist/fanboy-
social.txt","https://raw.githubusercontent.com/crazy-max/WindowsSpyBlocker/m
aster/data/hosts/win7/extra.txt","https://raw.githubusercontent.com/crazy-ma
x/WindowsSpyBlocker/master/data/hosts/win7/spy.txt","https://s3.amazonaws.co
m/lists.disconnect.me/simple_tracking.txt","https://s3.amazonaws.com/lists.d
isconnect.me/simple_ad.txt","https://raw.githubusercontent.com/AdguardTeam/A
dguardFilters/master/MobileFilter/sections/adservers.txt","https://raw.githu
busercontent.com/AdguardTeam/AdguardFilters/master/MobileFilter/sections/spy
ware.txt" -b /root/blacklist.hosts && sed -i 's/local-data/local-zone/g;
s/[[:space:|]]A 127.0.0.1"/" static/g' /etc/unbound/local.d/blocklist.conf
&& /etc/init.d/unbound restart
```

Hinweis: Das Skript hat eine ähnliche Funktion bereits integriert. Allerdings wird mit dem Parameter (always_nxdomain) nicht nur die Antwort NXDOMAIN geliefert, sondern auch alle Adressen in Wildcard-Adressen umgewandelt, was bei mir zu Overblocking führte. Siehe Abschnitt »unbound nxdomain EXPERIMENTAL use only«.

5. Fazit

Von Haus aus bietet die IPFire bisher nicht die Möglichkeit Werbung, Tracker und schadhafte Domains mittels DNS zu blockieren. Mit dem Skript von sfeakes lässt sich dies allerdings einfach nachrüsten. Ein zusätzlicher Pi-hole ist dann nicht mehr notwendig, da das Prinzip und auch die Filterlisten nahezu identisch sind. Da die Werbung und Tracker schon auf DNS-Ebene gefiltert werden, sparen die Geräte dahinter einiges an CPU und Rechenpower – was nicht angezeigt wird, muss auch nicht berechnet werden.

Permanent link:
http://wiki.richter-ch.de/doku.php/wiki:computer:ipfire:dnsadblock

Last update: **2020/02/27 11:45**