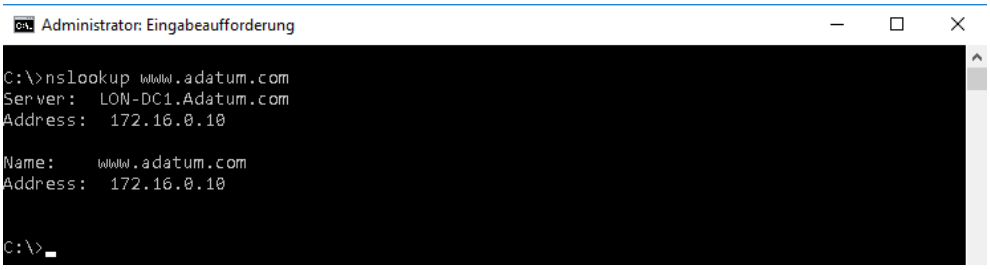
Inhalt

Der Angriff – eine DNS Amplification Attacke

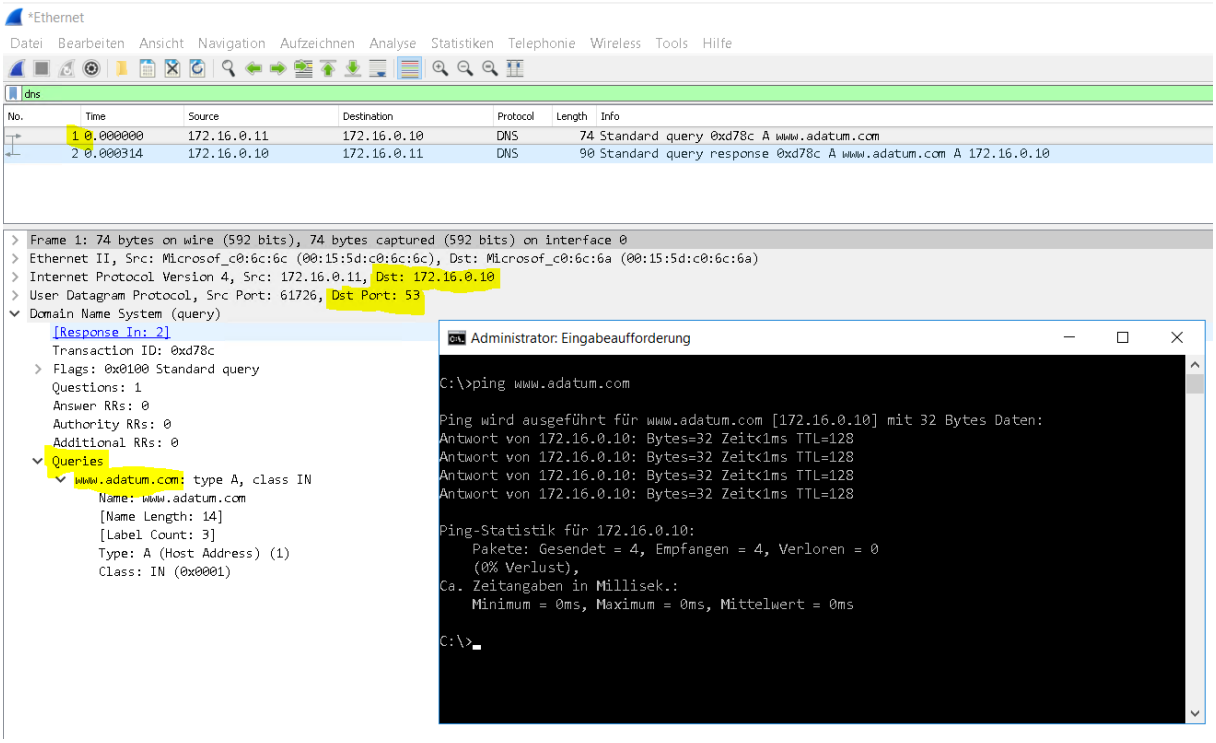
DNS als gutgläubiger Netzwerkdienst

Die Namensauflösung im Netzwerk ist zentraler Bestandteil aller Infrastrukturen (was geht ohne?). Und DNS ist als Service aktuell die Komponente, mit der die Namen in IP-Adressen (und umgekehrt) aufgelöst werden.

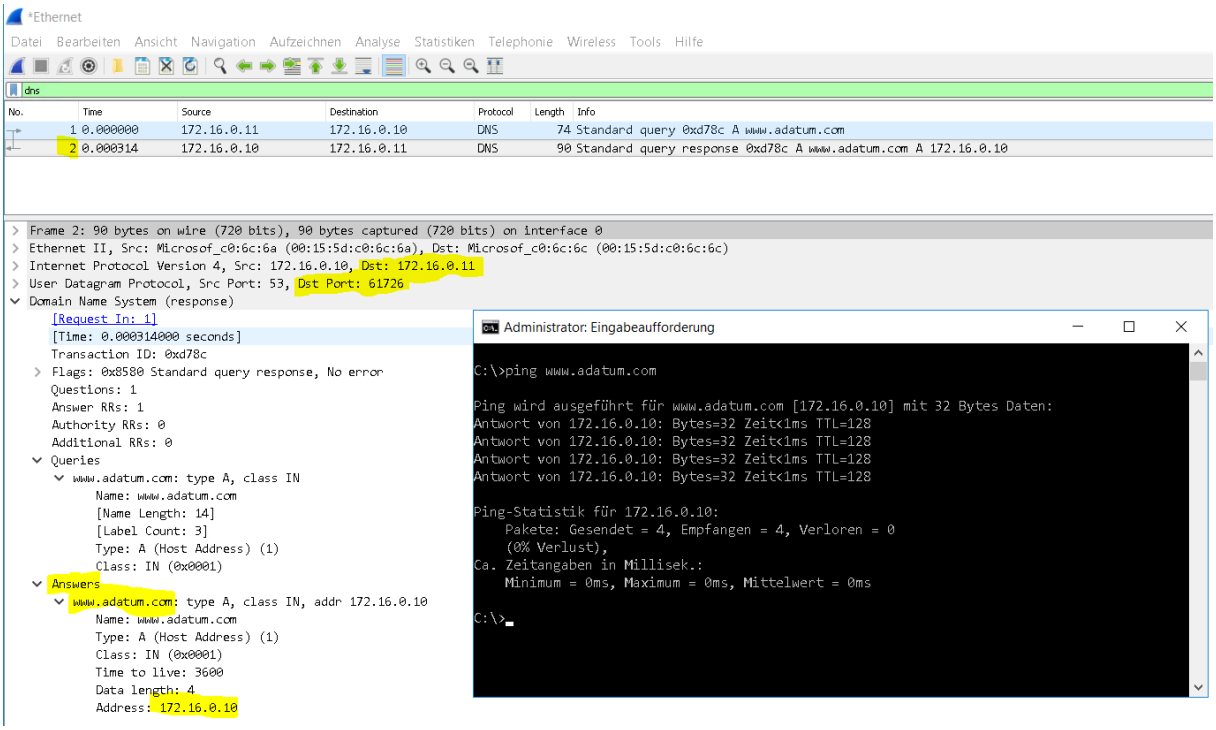
Dennoch ist DNS schon einige Tage alt und verwendet einige Mechanismen, die durch Angreifer ausgenutzt werden können. Ein DNS prüft nicht, welches System ihm eine Anfrage stellt. Er sucht nach einer Antwort und sendet diese an das System zurück:



Über das Netzwerk werden dafür Pakete zwischen dem Client und dem DNS-Server ausgetauscht. Der DNS-Client sendet an seinen DNS-Server via UDP an dessen Port 53 eine Anfrage (hier ausgelöst durch einen Ping auf www.adatum.com):



Der DNS-Server antwortet auf die Anfrage:



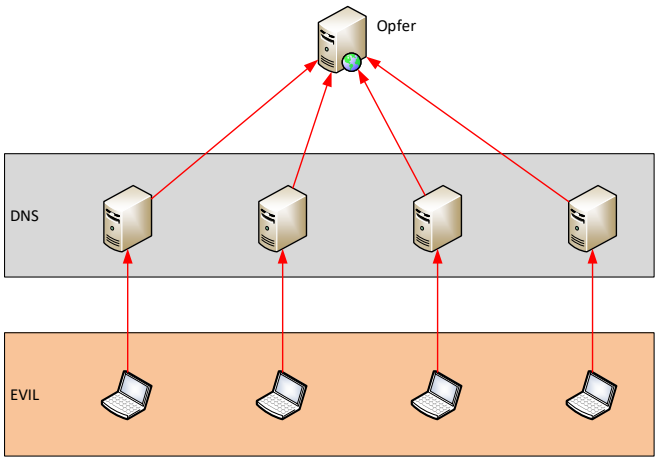
Bei der Anfrage verwendet der Server die IP-Adresse des DNS-Clients und dessen QuellPort. Und natürlich wird im lokalen Netzwerk auch die MAC-Adresse des DNS-Clients verwendet.

Der DNS-Server arbeitet auf diese Weise alle Anfragen brav ab. Er prüft nicht, WER ihn etwas fragt…

Schema einer DNS Amplification Attacke

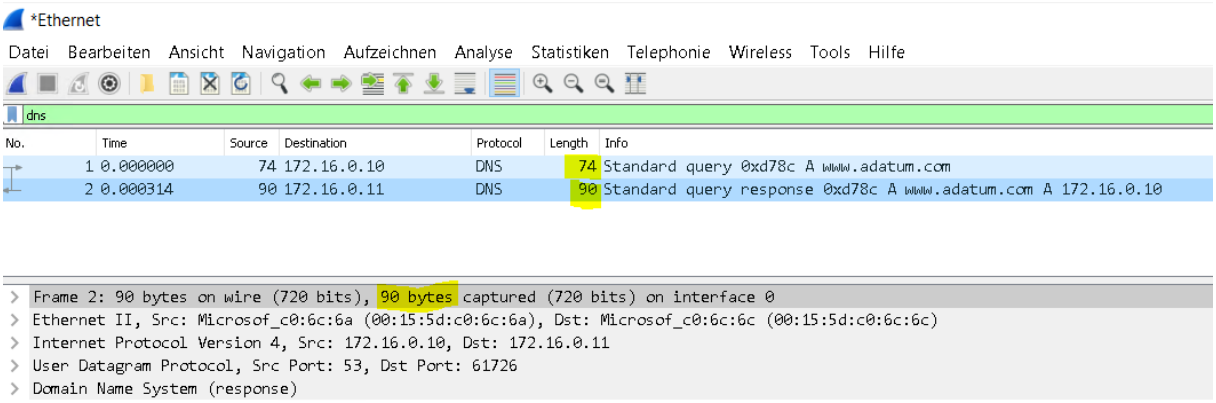
Und das kann ein möglicher Angriffsvektor sein. Angenommen, ein Angreifer sendet ein UDP-Paket mit einer Anfrage an einen DNS und fälscht darin seine IP-Adresse. Wer wird wohl die Antwort auf die Frage erhalten? Richtig: der aktuelle Eigentümer der gefälschten IP!

OK, der Client, der keine Anfrage gestellt hat wird die Antwort des DNS-Servers ignorieren. Was aber, wenn der Angreifer nicht nur eine Anfrage sendet, sondern tausende pro Sekunde. Und was wäre, wenn der Angreifer ein Botnetz verwendet, um JEWEILS tausende Anfragen pro Sekunde an verschiedene DNS-Server zu senden – jedes mal mit der gefälschten IP-Adresse seines Opfers? Richtig: das wird ein Denial of Service…

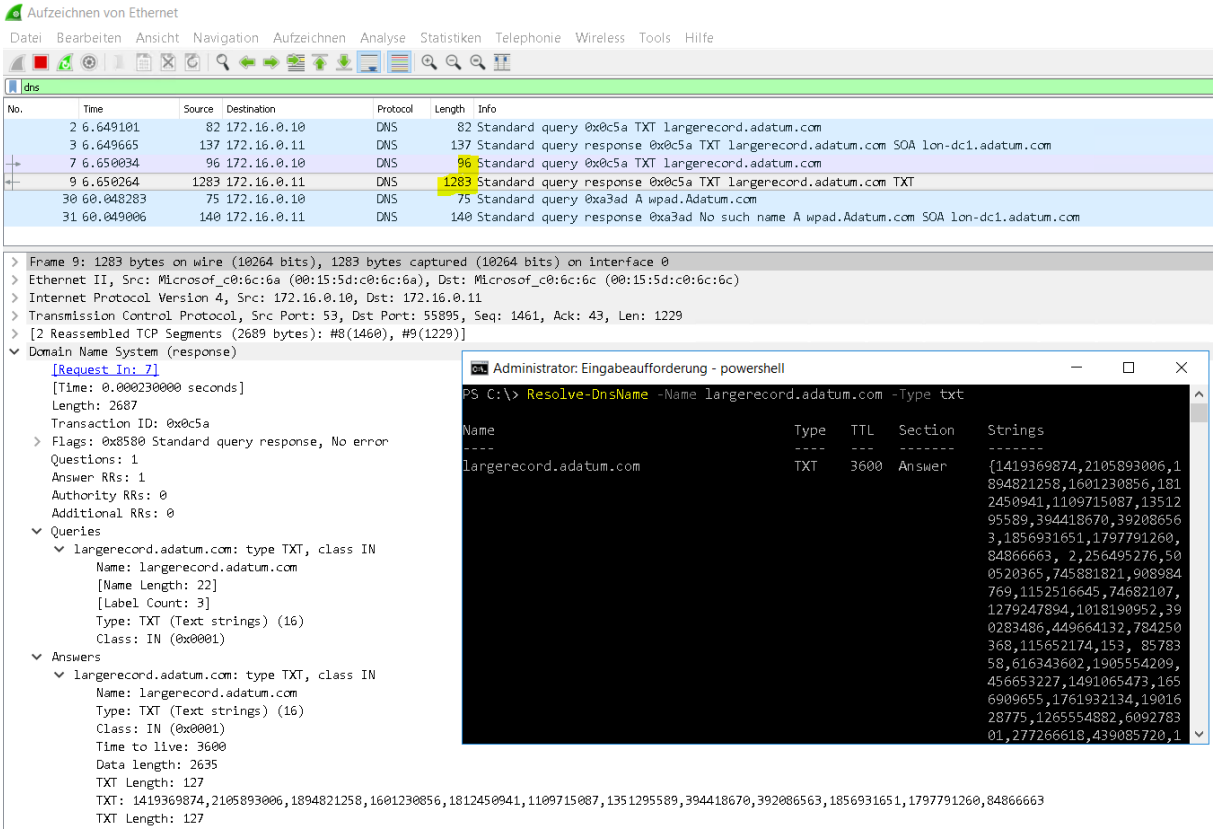


Das ist dann aber nur ein Distributed Denial of Service. Ein Angreifer kann noch weiter gehen und durch die richtigen Abfragen sehr große Antworten von den DNS-Servern provozieren. Er verstärkt den Angriff, was als Amplification bezeichnet wird. Diese Anfrage „kostet“ den Angreifer 74 Bytes.

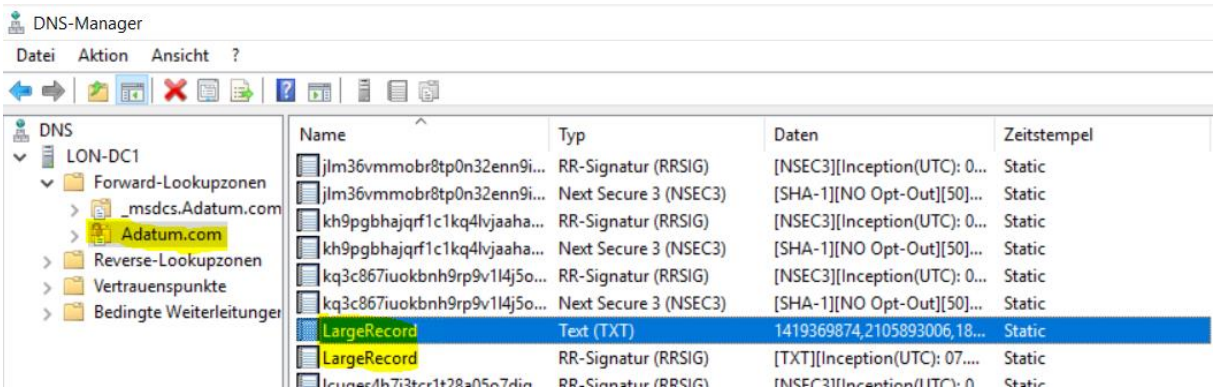
Das Opfer erhält ein 90 Bytes Antwortpaket:



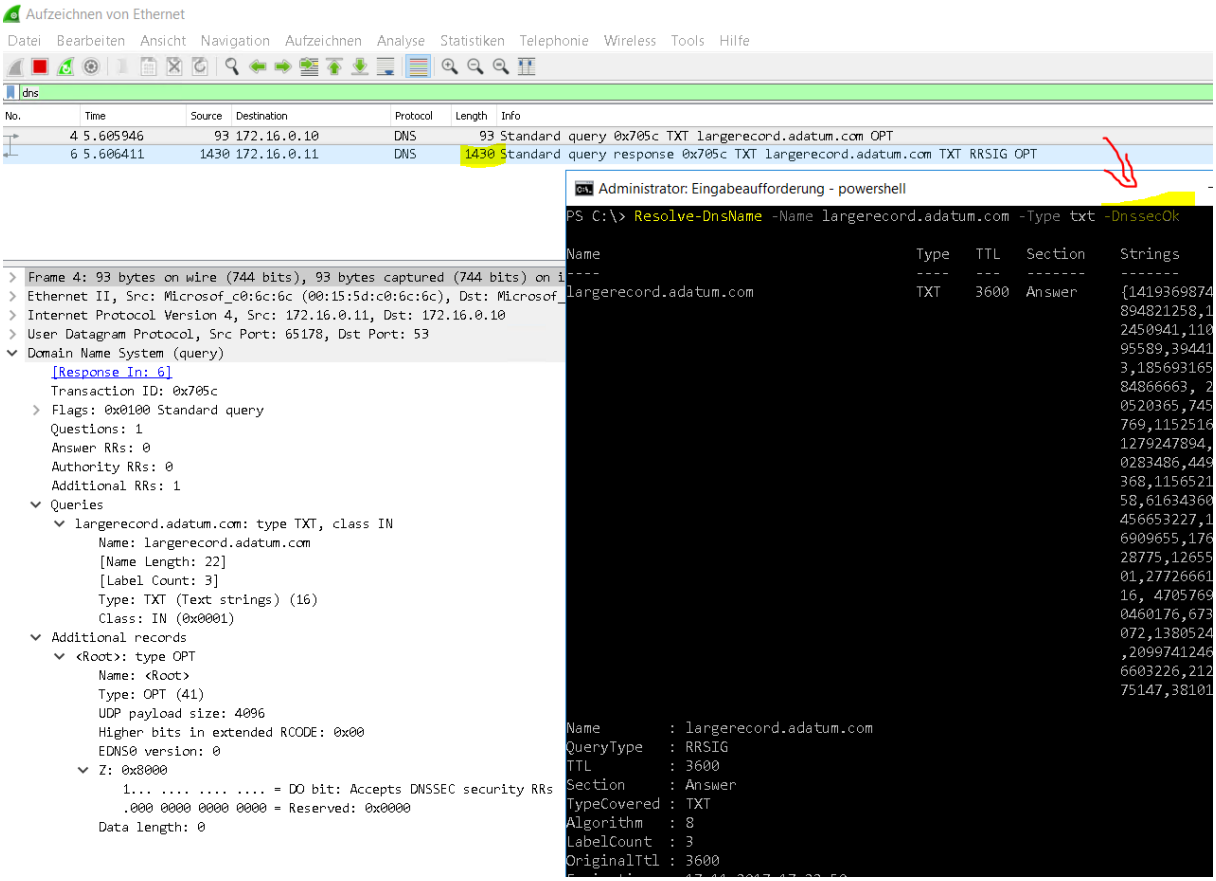
Findet der Angreifer einen großen Record und fragt diesen ab, dann ist seine Anfrage immer noch recht klein. Aber das Opfer wird geflutet. Die Frage kostet den Angreifer 96 Bytes. Die Antwort ist aber schon 1283 Bytes groß…



DNSSEC verschärft das Szenario noch zusätzlich. Die nächste Abfrage ist absolut identisch mit der im vorherigen Bild. Zusätzlich habe ich auf dem DNS-Server DNSSEC konfiguriert:



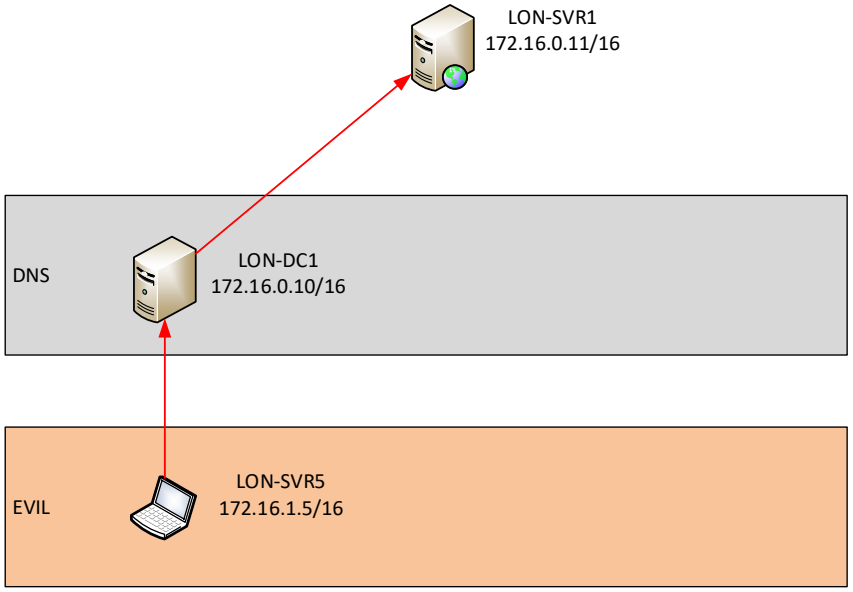
Und hier kommt das Ergebnis beim Client. Nun ist die Antwort schon 1430 Bytes groß:



Beispielszenario eines Angriffes

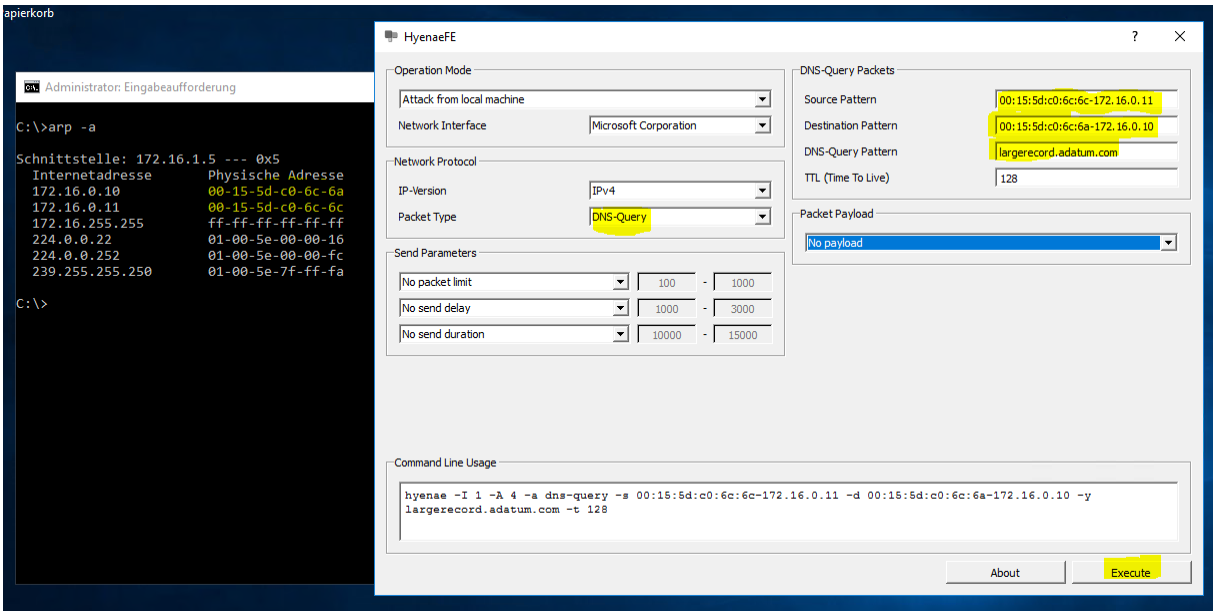
Ein kleines LAB soll das Problem sichtbar machen. Ich verwende 3 Server:

- einen DNS-Server (LON-DC1 mit der IPv4 172.16.0.10),
- ein Opfer (LON-SVR1 mit der IPv4 172.16.0.11)
- und einen Angreifer-Server (LON-SVR5 mit der IP 172.16.1.5)



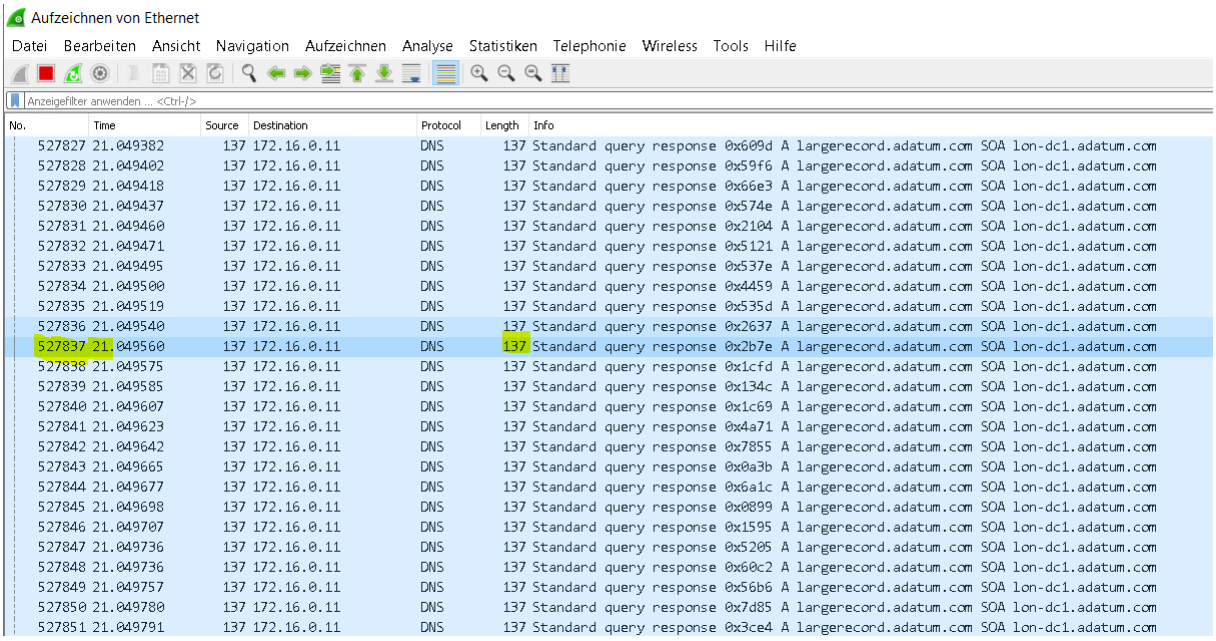
Auf dem Angreifer-Server verwende ich das Tool Hyenae, um die DDOS-Atacke zu starten. Diese sendet extrem viele Anfragen „im Auftrag von Server LON-SVR1“ an den LON-DC1. Und dieser wird LON-SVR1 auch brav antworten.

Und so geht's. Auf LON-SVR5 bringe ich Hyenae in Stellung. Als Source verwende ich die MAC und die IPv4 von LON-SVR1 (mein Opfer). Als Destination verwende ich die MAC und die IPv4 vom DNS-Server:



Der Record ist mein LargeRecord – ein dicker TXT-Eintrag mit DNSSEC bei etwa 1400 Bytes größe – pro Antwort!!!

Auf LON-SVR1 verwende ich WireShark, um die Antworten vom DNS-Server zu sehen. Und los geht's:



Mein Opfer hat Glück, dass die einfache Version nur Host-A-Records abfragen kann. Da sind die über 500.000 Antworten in 21 Sekunden nur etwa 70 MB statt 700 MB groß. Und es war nur 1 DNS!

Ich denke, das Problem ist nun klar…

Ein Schutz - Response Rate Limiting?

Was ist RRL und wie funktioniert es?

Die Antwort hat Microsoft in den DNS-Service ab Windows Server 2016 eingebaut: Response Rate Limiting. Dabei wird der Server prüfen, welcher Client wie oft eine Anfrage stellt. Wenn das Anfrageaufkommen einen Schwellwert überschreitet, dann werden die Antworten für einen Zeitraum gedrosselt. Es wird also nicht mehr jede Anfrage beantwortet.

ACHTUNG: Falsch konfiguriert wird so ein Denial Of Service für alle Clients und Server vorprogrammiert sein!!!

Leider wurde die Option nicht in die grafische Oberfläche eingebaut. Die Steuerung ist nur über die PowerShell möglich:

```
PS C:\> Get-Command -Name *response*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Add-DnsServerResponseRateLimitingExceptionlist     2.0.0.0    DnsServer
Function        Get-DnsServerResponseRateLimiting                  2.0.0.0    DnsServer
Function        Get-DnsServerResponseRateLimitingExceptionlist     2.0.0.0    DnsServer
Function        Remove-DnsServerResponseRateLimitingExceptionlist  2.0.0.0    DnsServer
Function        Set-DnsServerResponseRateLimiting                  2.0.0.0    DnsServer
Function        Set-DnsServerResponseRateLimitingExceptionlist     2.0.0.0    DnsServer


PS C:\>
```

Es sind bereits einige Einstellungen per default gesetzt. Die Aktivierung von RRL fehlt aber noch:

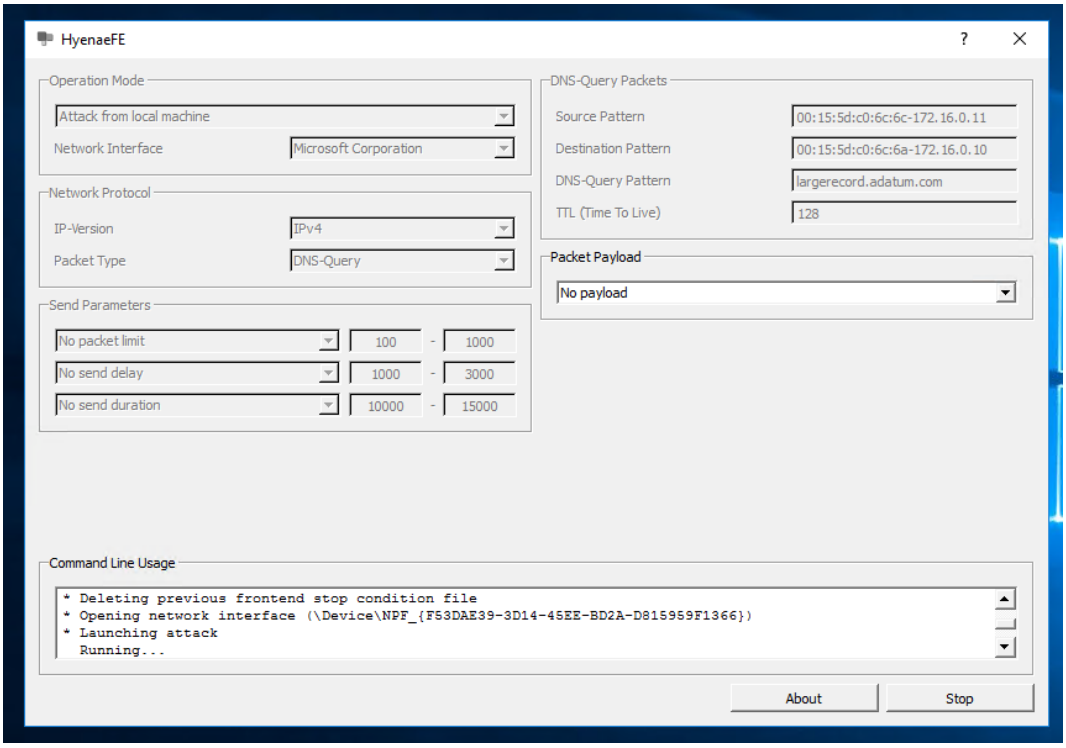
```
PS C:\> Get-DnsServerResponseRateLimiting


ResponsesPerSec           : 5
ErrorsPerSec              : 5
WindowInSec               : 5
IPv4PrefixLength          : 24
IPv6PrefixLength          : 56
LeakRate                  : 3
TruncateRate              : 2
MaximumResponsesPerWindow : 1024
Mode                      : Disable
```

Dann aktivieren wir doch RRL einmal. Nach einer Bestätigung und einer Warnung, das RRL falsch konfiguriert selbst ein Denial of Service für DNS bewirken kann, ist das neue Feature aktiviert:

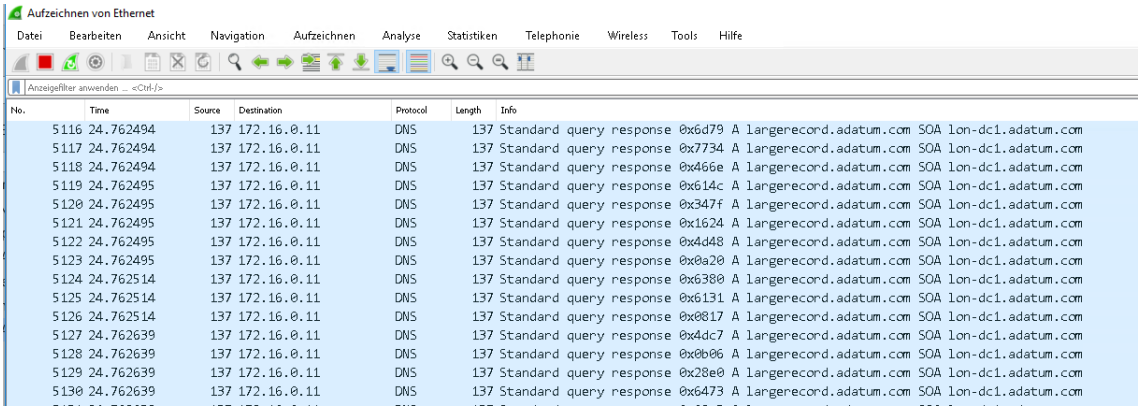
```
PS C:\> Set-DnsServerResponseRateLimiting -Mode Enable
WARNUNG: Response Rate Limiting kann zu Denial-of-Service-Angriffen auf Clients
führen, wenn eine Vielzahl ähnlicher Anforderungen festgestellt wird, die aus ei
ner gemeinsamen Quelle stammen.

PS C:\>
```

Beispielszenario eines Angriffes mit aktiviertem RRL

Dann bleibt nur noch ein ausstehender Test von RRL. Das Szenario ist das gleiche wie im ersten Kapitel: LON-SVR5 startet mit Hyenae eine DOS-Attacke und sendet dabei sehr viele DNS-Anfragen „im Auftrag von LON-SVR1“ an den DNS-Server:



Auf LON-SVR1 läuft derweil wieder WireShark und schneidet für uns die Antworten des DNS-Servers mit. Der Test läuft auch etwas mehr als 20 Sekunden. In dieser Zeit hatte der DNS-Server ohne RRL über 500.000 Antworten gesendet. Und mit RRL?



Jetzt sind es noch etwas über 5000 Antworten von DNS-Server an das Opfersystem! 😊

Und so funktioniert es: der DNS-Server sendet per Default mit aktiviertem RRL nur 1024 identische Antworten in einem Zeitraum von 5 Sekunden an einen Client, der identische Fragen stellt:

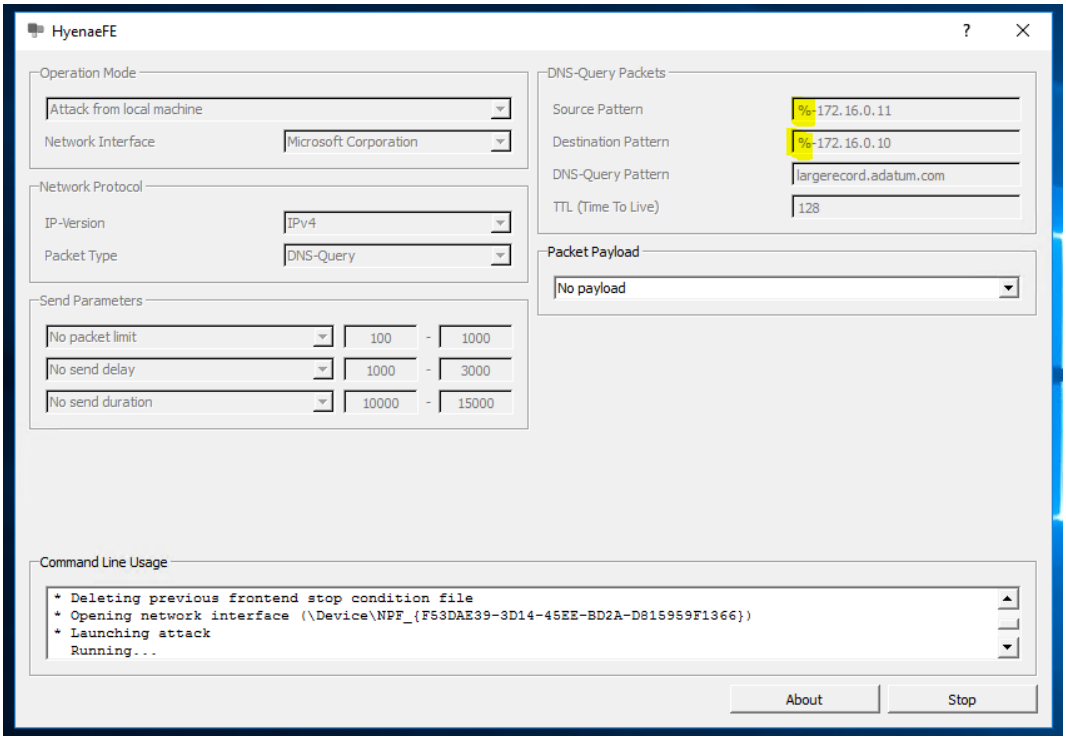
```
PS C:\> Get-DnsServerResponseRateLimiting


ResponsesPerSec             : 5
ErrorsPerSec                : 5
WindowInSec                 : 5
IPv4PrefixLength            : 24
IPv6PrefixLength            : 56
LeakRate                    : 3
TruncateRate                : 2
MaximumResponsesPerWindow   : 1024
Mode                        : Enable
```

Und das sind in 25 Sekunden eben etwas mehr als 5000 Antwortpakete. 😊

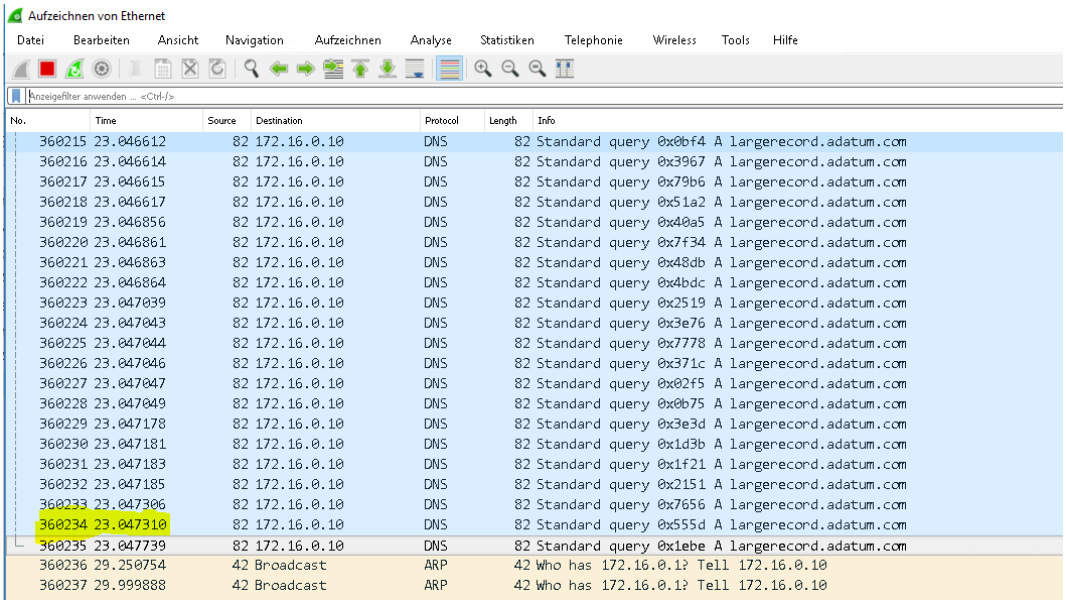
Katz und Maus

Leider genügen die Mechanismen von RRL nur bestimmten Rahmenbedingungen. Und diese setzen neben gleichen Absender-IP-Adressen auch gleiche MAC-Adressen voraus. Nur in der Kombination wird ein Client eindeutig gekennzeichnet.

Wenn ich in Hyenae nun einen Parameter der Attacke ändere (gefälschte MAC-Adressen)…



… dann versagt RRL, da der DNS-Server nicht mehr einen Client sondern ganz viele Clients erkennt. Und jeder bekommt in einem Zeitfenster die 1024 Standardantworten:



Dennoch kann RRL einen Teil des Angriffsvektors vereiteln. Es lohnt sich also, diese Funktion zu evaluieren!