UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Praktikum

Wireshark Display Filter Cheat Sheet

Cybersecurity im Sommersemester 2023

Jan Wichelmann, Anja Köhl

In Wireshark kann man von der Vielzahl an Paketen und Informationen regelrecht überflutet werden. Deshalb bietet Wireshark zahlreiche Filtermöglichkeiten an, um aus der überwältigenden Welle einen überschaubaren Fluss zu machen, indem die angezeigten Pakete einer Aufnahme gefiltert werden. Ganz wichtig ist, dass ein Anzeigefilter **kein** Mitschnittfilter ist! Diese beiden haben eine unterschiedliche Syntax.

Das Eingabefenster für Anzeigefilter befindet sich zwischen der Toolbar und der Packet List, hervorgehoben in Abb. 1. Um einen Anzeigefilter anzuwenden kann man entweder direkt in dieses Fenster einen Ausdruck eintippen und bestätigen oder über das Menü `Analyse > Anzeigefilterausdruck` einen Anzeigefilterausdruck zusammenbauen. Anzeigefilterausdrücke bestehen aus einem Feldnamen (`protokollname.feld.subfeld`), Relationen (`==, >, contains, in`), Werten (`0, "uni-luebeck"`) und logischen Operatoren (`&&, ||`). Jeder Filterausdruck filtert implizit nach der Existenz des Ausdrucks. Beispielsweise zeigt `tcp` alle Pakete an, die TCP beinhalten.

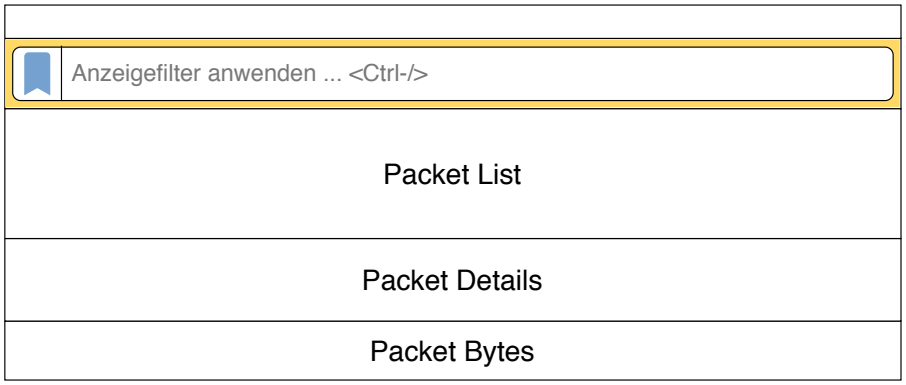

Abbildung 1: Einteilung des Wiresharkfensters

Vergleichsoperatoren

eq	==	Gleich
ne	!=	Ungleich
gt	>	Größer als
lt	<	Kleiner als
ge	>=	Größergleich
le	<=	Kleinergleich

Funktionen

`upper`(String-Feld)	Konvertiert ein String-Feld zu Großbuchstaben
`lower`(String-Feld)	Konvertiert ein String-Feld zu Kleinbuchstaben
`len`(Feld)	Gibt die Bytelänge eines String- oder Byte-Feldes wieder
`count`(Feld)	Gibt die Anzahl von Vorkommen dieses Feldes in einem Frame zurück
`string`(Feld)	Konvertiert ein nicht-String-Feld zu einem String

Aussagenlogische Operatoren

and	`&&`	Aussagenlogisches und
or	`||`	Aussagenlogisches oder
not	`!`	Aussagenlogisches nicht
xor	`^^`	Aussagenlogisches xor

Such- und Übereinstimmungsoperatoren

contains		Prüft ob das spezifizierte Feld, Protokoll oder Slice einen Wert beinhaltet.
matches	`~`	Prüft ob das spezifizierte Feld, Protokoll oder Slice mit einem Perl-kompatiblem RegEx-Ausdruck (PCRE)[1] übereinstimmt.

Slice-Operatoren

Der Slice-Operator funktioniert *ähnlich* wie bei Python, wobei er in Wireshark auf Strings oder Byte-Arrays angewandt wird anstelle von Sequenzen.

`[i:j]`	i = Anfang, j = Länge
`[i-j]`	i = Anfang, j = Ende (inklusiv)
`[i]`	Das i-te Byte oder Zeichen
`[:j]`	Vom Anfang j Zeichen oder Bytes
`[i:]`	Von i bis zum Ende

Mitgliedschafts-Operator

Der Mitgliedschafts-Operator `in {}` überprüft, ob ein Feld einen Wert aus einer Gruppe beinhaltet. Werte werden dabei durch Leerzeichen getrennt oder können durch `..` getrennt einen Wertebereich darstellen.

.. in {0 8}	Feld hat den Wert 0 oder 8
.. in {0 .. 8}	Feld het einen Wert zwischen 0 und 8

Bitwise-Operatoren

Momentan unterstützt Wireshark lediglich einen Bitwise-Operator für das bitwise and. Dieser Operator kann nur auf Felder oder Slices des Typs int angewandt werden. Mit diesem kann man einzelne Bits testen. Beispielsweise filtert man mit `tcp.flags & 0x04` auf TCP Pakete mit einer gesetzten RST Flag (das 3. Bit ist gesetzt).

`bitwise_and`	`&`	bitwise and

Integer-Darstellungen

Integer können in dezimal, oktal, hexadezimal oder wie in C als character constant geschrieben werden. Im folgenden ein Beispiel an der Dezimalzahl 10.

`10`	Dezimalsystem
`012`	Oktalsystem
`0xa`	Hexadezimalsystem
`'\n'`	C ASCII
`'\xa'`	C hexadezimal
`'\012'`	C oktal

Boolean-Darstellungen

Booleans werden mit 0 für false und 1, bzw. eine beliebige von 0 verschiedene Zahl, für true kodiert.

0	boolean false
1	boolean true

Byte-Array-Darstellungen

Byte-Arrays werden hexadezimal dargestellt. Zulässige Trennsymbole sind {`:` `.` `-`}.

ff:ff:ff:ff	mit Doppelpunkten
a.b.c.d.e	mit Punkten
0-1-0-1-0	mit Bindestrichen

IPv4-Darstellungen

IPv4-Adressen können entweder mit Punkten getrennten Dezimalzahlen oder als Hostname geschrieben werden.

141.83.104.159	dotted decimal notation
univis.uni-luebeck.de	aufgelöster Hostname

Beispiele

`ip.addr == 141.83.104.159`	Pakete von oder zu der IP-Adresse
`tcp.flags & 0x04`	Pakete mit der TCP RST Flag
`tcp.flags.reset == 1`	Pakete mit der TCP RST Flag
`udp.port == 53`	UDP-Pakete mit Port 53 (DNS)
`!(eth.addr == ff:ff:ff:ff:ff:ff)`	Alle nicht Broadcast-Pakete
`icmp.type in {9 10}`	ICMP-Pakete des Typs 9 oder 10
`frame contains 666c-3467-2d78-466e-5133-6e`	Pakete, die dieses Bytearray beinhalten

Weitere Informationen und Beispiele finden sich in der Wireshark-Dokumentation:
`https://www.wireshark.org/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html`.

[1] PCRE = Perl Compatible Regular Expressions, hilfreich hierbei sind `https://regex101.com/` und `https://regexr.com/`