Presented by Profitap (www.profitap.com) and Chappell University (www.chappellU.com)

WIRESHARK® NETWORK FORENSICS CHEAT SHEET

Created by Laura Chappell (www.chappellU.com) – Wireshark Training for Troubleshooting, Optimization, and Security

BASIC: Filter out 10.1.1.1 traffic from view	`!ip.addr==10.1.1.1`
BASIC: TCP traffic to or from port 443	`tcp.port==443`
BASIC: ICMP or ICMPv6	`icmp || icmpv6`
BASIC: TLS Handshake Packets (including Client Hello, Server Hello)	`tls.record.content_type == 22`
BASIC: TLS traffic - all [Wireshark v3 supports `ssl` and `tls` filters, not just `ssl`]	`tls`
TCP: SYN packets	`tcp.flags.syn==1 && tcp.flags.ack==0`
TCP: SYN/ACK packets (example of a bitwise filter)	`tcp.flags & 0x12`
TCP: SYN packet with non-zero ACK# field	`tcp.flags.syn==1 && tcp.flags.ack==0 && tcp.ack==0`
TCP: To/from port 443 or 4430 through 4434	`tcp.port in {443 4430..4434}`
TCP: Connection refusal or ACK scan	`tcp.flags.reset==1 && tcp.flags.ack==1 && tcp.seq==1 && tcp.ack==1`
TCP: Data in Urgent Pointer field (rarely seen for legitimate purposes)	`tcp.urgent_pointer>0`
TLS: Client Hello [Wireshark v3 supports `ssl` and `tls` filters, not just `ssl`]	`tls.handshake.type == 1`
TLS: Server Hello [Wireshark v3 supports `ssl` and `tls` filters, not just `ssl`]	`tls.handshake.type == 2`
TLS: TLS Encrypted Alert (followed by FIN, it's probably a connection close)	`tls.record.content_type == 21`
TLS: Target server contains "badsite" in server name	`tls.handshake.extensions_server_name contains "badsite"`
DNS: DNS PoinTeR (PTR) query/response	`dns.qry.type == 12`
DNS: Unusually large DNS answer count (check answers - C&C server list?)	`dns.count.answers>10`
DNS: Non port-53 traffic to DNS server (x.x.x.x) – TCP and UDP	`ip.dst==x.x.x.x && !udp.port==53 && !tcp.port==53`
HTTP: HTTP PUT and POST messages	`http.request.method in {PUT POST}`
HTTP: Requested HTTP objects with.exe/.zip/.jar file name extensions (PERL regex)	`http.request.uri matches "\.(exe|zip|jar)$"`
HTTP: Content type "application" from server	`http.content_type contains "application"`
HTTP: Redirections (all response codes starting with 3)	`http.response.code>299 && http.response.code<400`
HTTP: GET command not running over TCP port 80	`frame contains "GET" && !tcp.port==80`
FTP Unusually long FTP USER name (USER command, space and 0d0a take 7 bytes)	`ftp.request.command=="USER" && tcp.len>50`
P2P: Peer-to-peer traffic among hosts on 172.16 subnet, but not subnet broadcasts	`ip.src==172.16.0.0/16 && ip.dst==172.16.0.0/16 && !ip.dst==172.16.255.255`
IRC: Frame contains the JOIN command (upper or lower case) (PERL regex)	`frame matches "join #"`
Nmap: "Nmap" Identified in HTTP User Agent field (case sensitive)	`http.user_agent contains "Nmap"`
Nessus: Frame offset 100-199 contain "nessus" in lower case	`frame[100-199] contains "nessus"`
Nessus: Frame offset 100-199 contains "nessus" in upper or lower case (PERL regex)	`frame[100-199] matches "nessus"`
MISC: Unexpected applications on the network (see reverse)	`tftp || irc || bittorrent`

Profitap and Chappell University are Partners in Technology Education

PROFITAP	CHAPPELL UNIVERSITY
20% Off Products Purchased Directly from Profitap	20% Off Your All Access Pass Subscription
PROFITAP15RSA (exp. 5/1/2019)	**RSA2FYR (exp. 5/1/2019)**
Network Visibility Solutions	Online and Onsite Network Analysis Training

STATISTICS | PROTOCOL HIERARCHY (Tips to Locate Suspicious Traffic)

Look for the following elements under IP, IPv6:

Data: This is an indication that Wireshark could not determine the transport layer protocol in use. Considering the number of protocols for which Wireshark has dissectors, this would be considered unusual. Right click the Data entry and select **Apply as Filter | Selected**. Examine the traffic in the Packet Bytes pane to locate identifiers to help determine the purpose of the traffic.

Look for the following elements under UDP and TCP (try this on *sec-sickclient.pcapng* available at *chappellU.com*):

Undesirable Applications: Look for applications that are not desired or permitted on the network. For example, look for `irc`, `bittorrent`, `tftp`, etc.

Data: This is an indication that Wireshark could not determine the application in use. If an application is using a non-standard port number, it may show up here. If an application is not well-known, it will show up here. Right click on the Data line and select **Apply as Filter | Selected**. After the filter is applied, right click on a packet and select **Follow | UDP Stream/TCP Stream**. Examine the stream to identify the possible purpose of the traffic.

EMBEDDING A PRIVATE KEY IN A TRACE FILE (Wireshark v3 and later)

NEW

Share an encrypted trace file and the session private key only (not the public key).

Step 1: Download and unzip the RSA19 Trace File set from *chappellU.com*. Both *rsasnakeoil2.pcapng* and *rsasnakeoil2.key* are inside this trace file set. Place both files in a directory called */keyembed on your system*.

Step 2: Open *rsasnakeoil2.pcapng* in Wireshark v3 (available at *wireshark.org*). Select **Edit | Preferences | RSA Keys**.

Step 3: Click **Add new keyfile...** and select *rsasnakeoil2.key* from your /keyembed directory. Click **Open**. Click **OK** to close the Preferences window. You may need to click the **Refresh** button on the main toolbar to see the decrypted traffic.

Step 4: Select **File | Export TLS Session Keys...** and save your key (name it *rsasession.key*) into your */keyembed* directory.

Step 5: Return to **Edit | Preferences | RSA Keys**, click on your key listed and select **Remove Key**. Click **OK**.

Step 6: [Ensure you have the Wireshark program directory in your path.] At the command prompt, navigate to your */keyembed* directory. Type the following command:

```
editcap --inject-secrets tls,rsasession.key rsasnakeoil2.pcapng snakeoil-withkey.pcapng
```

Step 7: Open your *snakeoil-withkey.pcapng* file in Wireshark. You will see the trace file unencrypted without use of the public RSA key. Nice, eh?