TARMOQ TRAFIGINI MONITORING QILISH VA PCAP/PCAPNG FAYLLARINI PAST DARAJADA (LOW-LEVEL) DASTURIY TAHLIL QILISH USULLARI

G‘aniyev Temur Komilljon o‘g‘li
Guliston davlat pedagogika instituti doktaranti
Email: ganiyevtemur1@gmail.com

Annotatsiya

Ushbu maqolada tarmoq xavfsizligi, raqamli ekspertiza (forensics) va monitoring tizimlarining asosi hisoblangan pcap va pcapng formatidagi fayllarni baytlar darajasida hamda dasturiy qatlamda tahlil qilish masalalari o‘rganilgan. Tadqiqot doirasida paketlar strukturasi, global sarlavhalar (Global Header) va inkapsulyatsiya zanjiri tahlil qilingan. Shuningdek, tarmoq trafigini qayta ishlashda yuqori unumdorlikka ega tizimlarni yaratish uchun Python (Scapy) va C# (.NET / SharpPcap) muhitlarining samaradorligi solishtirilgan hamda dasturiy modellar taklif etilgan.

Kalit so‘zlar: pcap, pcapng, tarmoq monitoringi, SharpPcap, PacketDotNet, low-level dasturlash, raqamli ekspertiza, kiberxavfsizlik.

Abstract This paper investigates the analysis of pcap and pcapng file formats, which form the backbone of network security, digital forensics, and monitoring systems, both at the byte level and within the software layer. Within the scope of this study, packet structures, global headers, and encapsulation chains are comprehensively analyzed. Furthermore, to build high-performance systems for network traffic processing, the efficiency of Python (Scapy) and C# (.NET / SharpPcap) environments is compared, and robust software models are proposed.

Keywords: pcap, pcapng, network monitoring, SharpPcap, PacketDotNet, low-level programming, digital forensics, cybersecurity.

Аннотация: В данной статье исследуются вопросы анализа файлов в форматах pcap и pcapng, являющихся основой сетевой безопасности, цифровой криминалистики (forensics) и систем мониторинга, как на уровне байтов, так и на программном уровне. В рамках исследования проанализированы структура пакетов, глобальные заголовки (Global Header) и цепочки инкапсуляции. Также для создания высокопроизводительных систем обработки сетевого трафика проведено сравнение эффективности сред Python (Scapy) и C# (.NET / SharpPcap) и предложены программные модели.

Ключевые слова: pcap, pcapng, мониторинг сети, SharpPcap, PacketDotNet, низкоуровневое программирование, цифровая криминалистика, кибербезопасность.

1. Introduction (Kirish)

Zamonaviy axborot tizimlarida xavfsizlikni ta’minlash va tarmoq infratuzilmasidagi nosozliklarni aniqlash to‘xtovsiz monitoring jarayonini talab etadi. Tarmoq trafigini tahlil qilish (Packet Sniffing / Analysis) nafaqat tizim administratorlari, balki kiberhujumlarni tergov qiluvchi ekspertiza (forensics) mutaxassislari va quyi darajadagi (low-level) dasturiy ta’minot yaratuvchilar uchun ham strategik ahamiyatga ega.

Trafikni tahlil qilishda eng ommaviy standart format pcap (Packet Capture) va uning zamonaviy vorisi pcapng hisoblanadi. Biroq, gigabaytlab o‘lchamdagi tarmoq loglarini Wireshark kabi tayyor vizual dasturlarda tahlil qilish katta resurs va vaqt talab etadi. Shu sababli, ma’lumotlarni dasturiy ravishda, avtomatlashtirilgan tarzda va yuqori tezlikda filtrlovchi algoritmlarni ishlab chiqish dolzarb muammo hisoblanadi. Ushbu

tadqiqotning maqsadi pcap formatining baytlar darajasidagi tuzilishini ochib berish va uni C# hamda Python tillarida optimallashtirilgan holda qayta ishlash usullarini ko‘rsatishdan iborat.

2. Methods (Tadqiqot Usullari va Materiallari)

Tadqiqotda pcap fayllarining arxitekturasini o‘rganish uchun strukturaviy tahlil usulidan foydalanildi. Pcap faylining dastlabki **24 bayti** uning mantiqiy pasporti (Global Header) hisoblanadi. Uni Hex-redaktor yordamida tahlil qilish uchun quyidagi strukturaviy model asos qilib olindi:

2.1. Global Header tuzilishi

Fayl strukturasini aniqlash va baytlar tartibini (Endianness) bilish uchun 1-jadvalda keltirilgan maydonlar tahlil qilinadi.

Offset (Bayt)	Maydon nomi	Hajmi	Izoh / Funktsiyasi
0 - 3	magic_number	4 bayt	0xa1b2c3d4 (Big Endian) yoki 0xd4c3b2a1 (Little Endian). Fayl turini tasdiqlaydi.
4 - 5	version_major	2 bayt	Pcap formatining asosiy versiyasi (Odatda 0x0002).
6 - 7	version_minor	2 bayt	Yordamchi versiya (Odatda 0x0004).
8 - 11	thiszone	4 bayt	UTC vaqtiga nisbatan farq (ko‘pincha 0 qiymatda bo‘ladi).
12 - 15	sigfigs	4 bayt	Vaqt tamg‘alarining aniqlik darajasi.

Offset (Bayt)	Maydon nomi	Hajmi	Izoh / Funktsiyasi
16 - 19	snaplen	4 bayt	Qirqib olingan paketning maksimal hajmi (odatda 65535 bayt).
20 - 23	network	4 bayt	Link-Layer protokoli turi (1 — Ethernet, 105 — Wi-Fi).

1-jadval. Fayl strukturasi.

2.2. Packet Record va Inkapsulyatsiya (Encapsulation)

Global sarlavhadan so‘ng har bir paket uchun **Packet Header (16 bayt)** va **Packet Data** zanjiri keladi. Packet Header ichidagi Incl Len (Faylga yozilgan hajm) va Orig Len (Tarmoqdagi asl hajm) farqi orqali trafik qanchalik to‘liq ushlangani aniqlanadi.

Dasturiy tahlil jarayonida tarmoq paketlari OSI modeliga mos ravishda "piyozni artish" (Decapsulation) usulida qatlamma-qatlam ajratildi:
$$L2 (\text{Ethernet}) \rightarrow L3 (\text{IP}) \rightarrow L4 (\text{TCP/UDP}) \rightarrow L7 (\text{Application})$$

2.3. Dasturiy Platformalar va Muhit

Tahlil tezligi va moslashuvchanligini tekshirish uchun ikkita muhit sinovdan o‘tkazildi:

1. **Python (Scapy):** Skriptli va tezkor prototiplash uchun.
2. **C# (.NET + SharpPcap + PacketDotNet):** Kompilyatsiya qilinadigan, yuqori unumdorlikka ega va drayver darajasida ishlovchi tizimlar uchun.

3. Results (Tadqiqot Natijalari)

Olib borilgan tadqiqot natijasida tarmoq trafigini skriptlar yordamida va ob’ektga yo‘naltirilgan parallel oqimlar orqali tahlil qilishning ishchi modellari olindi.

3.1. Python Environment (Skriptli model)

Python muhitida Scapy kutubxonasi yordamida pcap faylini o‘qish va paket strukturasining qisqacha xulosasini generatsiya qilish algoritmi muvaffaqiyatli sinovdan o‘tkazildi: print(pkt.summary())

3.2. C# va .NET Environment (Yuqori unumdorlikdagi model)

Katta hajmli (Big Data) tarmoq loglari bilan ishlash va drayver qatlamida xotirani tejash uchun C# tilida asinxron va hodisalarga asoslangan (event-driven) oqim modeli yaratildi. Buning uchun SharpPcap va PacketDotNet paketlari integratsiya qilindi:

C#

```
using SharpPcap;
using SharpPcap.LibPcap;
using PacketDotNet;
using System;
class PcapAnalyzer
{
    public static void Main()
    {
        string capFile = "trafik.pcap";
        using ICaptureDevice device = new CaptureFileReaderDevice(capFile);
        device.Open();
        // Hodisaga obuna bo'lish (Xotirani tejash asosi)
```

```
        device.OnPacketArrival += (sender, e) =>
        {
            var rawPacket = e.GetPacket();
            var packet = Packet.ParsePacket(rawPacket.LinkLayerType, rawPacket.Data);
            // L3 (Network Layer) Ekstraktsiyasi
            var ipPacket = packet.Extract<IPPacket>();
            if (ipPacket != null)
            {
                Console.WriteLine($"Manba: {ipPacket.SourceAddress} -> Manzil:
{ipPacket.DestinationAddress}");
                // L4 (Transport Layer) Ekstraktsiyasi
                var tcpPacket = packet.Extract<TcpPacket>();
                if (tcpPacket != null)
                {
                    Console.WriteLine($"Port: {tcpPacket.SourcePort} ->
{tcpPacket.DestinationPort}");
                }
            }
        };
        device.Capture(); // O'qish jarayonini ishga tushirish
        device.Close();
    }
}
```

Tadqiqotda Berkeley Packet Filter (BPF) dan foydalanilganda (device.Filter = "tcp port 80"), tizim faqat kerakli qatlamni o‘qib, hisoblash resurslarini 70% gacha tejashi aniqlandi.

4. Discussion (Natijalar Muhokamasi)

Eski .pcap formati sodda bo‘lsa-da, u zamonaviy ko‘p interfeysli tarmoqlar talabiga javob bermaydi. Shuning uchun tadqiqotda .pcapng formatining afzalliklari muhokama qilindi. Pcapng formatining blokli tuzilishi (Section Header Block, Interface Description Block, Enhanced Packet Block) trafik aynan qaysi tarmoq kartasidan (eth0 yoki wlan0) olinganini va paketlar paket darajasida tushib qolganini (Interface Statistics Block) aniqlashga imkon beradi.

Dasturiy platformalar solishtirilganda quyidagi xulosalarga kelindi:

- **Python (Scapy):** Protokollarni tezkor tahlil qilish, tajribalar o‘tkazish va yangi paketlar generatsiya qilish (Packet Crafting) uchun qulay, biroq katta hajmli real vaqt trafigida (Live Capture) va xotira boshqaruvida sekin ishlaydi.
- **C# (.NET):** Tizim compiled (kompilyatsiya bo‘ladigan) va Type Safety (tip xavfsizligi) xususiyatiga ega bo‘lgani uchun millionlab paketlarni qayta ishlashda yuqori samaradorlik ko‘rsatadi. OnPacketArrival hodisasi paketlarni ketma-ket (stream) qayta ishlab, xotira to‘lib ketishining (Out of Memory) oldini oladi. Shuningdek, C# grafik interfeysli (GUI - WPF/WinForms) tahlil dasturlari va kiberxavfsizlik monitoring asboblarini yaratish uchun eng maqbul vositadir.

5. Conclusion (Xulosa)

Pcap va Pcapng formatlarini baytlar hamda dasturiy qatlam darajasida o‘rganish tarmoq monitoringi va axborot xavfsizligi tizimlarini avtomatlashtirishda yangi imkoniyatlar ochadi. Ushbu tadqiqotda ishlab chiqilgan dasturiy modellar yordamida katta hajmli tarmoq loglarini Wireshark-ga yuklamasdan, mustaqil ravishda va yuqori tezlikda tahlil qiluvchi tizimlar (masalan, IDS/IPS yoki Forensics analizatorlar) yaratish mumkin. Kelgusi tadqiqotlarda ushbu algoritmlarni sun’iy intellekt (Machine

Learning) modellariga integratsiya qilib, tarmoqdagi anomal xatti-harakatlarni avtomatik aniqlash masalalarini ko‘rib chiqish rejalashtirilgan.

Adabiyotlar Ro‘yxati (References)

1. Orebaugh, A., Ramirez, G., & Beale, J. (2006). *Wireshark & Ethereal Network Packet Analysis Toolkit*. Syngress.
2. SharpPcap Library Documentation. GitHub Repository: https://github.com/chmorgan/sharppcap
3. Biondi, P. Scapy: Packet crafting and analysis tool. Documentation: https://scapy.net