Bilgisayar Ağlarında Yazılımcı Tespiti için Wireshark Kullanımına Yönelik 'Developer Hunter' Projesi: 2025 ve Sonrası için En Etkili 10 Teknik ve Trend

Yönetici Özeti

Bu rapor, modern siber güvenlik ortamında yazılımcıların ağ davranışlarını tespit etmeye odaklanan 'Developer Hunter' projesinin kritik önemini vurgulamaktadır. Proje, Wireshark'ın gelişmiş yeteneklerini kullanarak yazılımcılara özgü ağ etkinliklerini (Git/SSH, IDE trafiği, geliştirme/test sunucusu erişimi, API test araçları, sanal makine/konteyner etkileşimleri, özel port/protokol kullanımları) analiz ederek ayırt edici dijital izlerini belirlemeyi amaçlamaktadır. Rapor, 2025 ve sonrasına yönelik en etkili 10 tekniği ve trendi derinlemesine incelemekte, şifrelenmiş trafiğin analizi, yapay zeka ve makine öğrenimi destekli davranışsal analizler, bulut tabanlı ve uzaktan çalışma ortamlarındaki zorluklar gibi konulara odaklanmaktadır. Ayrıca, etik ve yasal uyumluluğun bu tür projelerin başarısı için vazgeçilmez bir unsur olduğunun altını çizmektedir.

1. Giriş: 'Developer Hunter' Projesinin Stratejik Gerekliliği

Geliştirici Dijital Ayak İzlerini Tanımlama Zorluğu

'Developer Hunter' projesi, modern siber güvenlik ortamında kritik bir ihtiyacı karşılamaktadır: bir bilgisayar ağındaki yazılımcıların hassas bir şekilde tanımlanması ve izlenmesi. Yazılımcılar, rolleri gereği genellikle kritik fikri mülkiyete (kaynak kodu, tescilli algoritmalar), hassas verilere ve üretim altyapısına yüksek düzeyde erişime sahiptir. Bu ayrıcalıklı erişim, onları harici saldırganlar için potansiyel hedefler haline getirirken, daha da önemlisi, kötü niyetli veya kasıtsız iç tehditler için bir vektör oluşturmaktadır.[1] Proje, yazılımcıların Git/SSH işlemleri, Entegre Geliştirme Ortamı (IDE) trafiği, geliştirme ve test sunucularına erişim, API test araçlarının kullanımı, sanal makineler ve konteynerlerle etkileşimler ve özel port veya protokol kullanımları gibi karakteristik ağ etkinliklerini derinlemesine analiz ederek benzersiz dijital ayak izlerini belirlemeyi hedeflemektedir. Bu girişim, sadece teknik bir egzersiz olmanın ötesinde, bir kuruluşun ağ görünürlüğünü artırmak ve güvenlik duruşunu güçlendirmek için stratejik bir zorunluluktur.

Ağ Trafiği Analizinde Wireshark'ın Rolü

Wireshark, ağ profesyonellerinin ve siber güvenlik uzmanlarının araç setinde temel ve vazgeçilmez bir araç olarak öne çıkmaktadır. Güçlü, açık kaynaklı bir ağ protokolü analizörü olarak, ağ trafiğini gerçek zamanlı olarak yakalama, inceleme ve etkileşimli

olarak tarama konusunda eşsiz yetenekler sunar.[2] Wireshark'ın temel gücü, her yakalanan paketi ayrıştırarak protokoller, başlıklar ve yükler hakkında mikroskobik düzeyde derinlemesine bilgi sunmasıdır.[3] Bu ayrıntılı görünüm, ağ sorunlarını gidermek, iletişim kalıplarını anlamak ve güvenlik açıklarını veya tehditleri tespit etmek için paha biçilmezdir.[2] Wireshark'ın açık kaynak yapısı ve binlerce ağ protokolüne sağladığı geniş destek, onu ağ yöneticileri, güvenlik uzmanları ve hatta yazılımcılar için vazgeçilmez bir varlık haline getirmektedir.[2]

Geliştirici faaliyetleri için "ağ çevresi" tanımının değişimi, bu projenin temelini oluşturmaktadır. Wireshark'ın kablolu veya kablosuz arayüzlerden trafik yakalama yeteneği, yazılımcıların faaliyetlerinin artık geleneksel ofis ağlarıyla sınırlı olmadığını göstermektedir.[2] Bulut tabanlı geliştirme, mikro hizmetler, konteynerleşme ve uzaktan çalışma modellerinin artan yaygınlığı, yazılımcıların "ağının" fiziksel ofis sınırlarının çok ötesine geçtiği anlamına gelmektedir.[9] Bu durum, basit bir çevre tabanlı Wireshark yakalamasının yetersiz kalacağı anlamına gelmektedir. Bunun yerine, 'Developer Hunter' projesi için veri toplama stratejisi, genellikle geçici ve coğrafi olarak dağınık olan çeşitli, dağıtılmış uç nokta ortamlarına (dizüstü bilgisayarlar, uzaktan sunucular, bulut altyapıları) odaklanmalıdır. Bu, veri toplamanın karmaşıklığını önemli ölçüde artırsa da, daha zengin ve ilgili bilgiler sağlayacaktır.

'Developer Hunter' projesi, yalnızca reaktif sorun gidermenin ötesine geçerek proaktif bir güvenlik önlemi olarak konumlanmaktadır. Wireshark, ağ sorunlarını gidermek ve mevcut güvenlik açıklarını tespit etmek için yaygın olarak kullanılsa da [2], bu projenin amacı "yazılımcıları tespit etmek" ve "karakteristik ağ etkinliklerini" belirlemektir. Bu, genel anomali tespitinden belirli kullanıcı profillemesine doğru bir geçişi ifade etmektedir. Yazılımcıya özgü davranış kalıplarını proaktif olarak tanımlayarak, kuruluşlar "normal" geliştirici davranışları için temel referans çizgileri oluşturabilirler. Bu referans çizgileri, iç tehditleri, tehlikeye atılmış hesapları veya yetkisiz faaliyetleri gösterebilecek sapmaları tespit etmek için hayati öneme sahiptir. Bu yaklaşım, güvenliği ihlallere reaktif olarak yanıt vermekten, kritik kullanıcı gruplarını (yazılımcılar) proaktif olarak anlamaya ve güvence altına almaya kaydırmaktadır. Bu, sürekli izleme ve davranışsal analiz ihtiyacını vurgulamaktadır.

2. Ağ Keşfi için Temel Wireshark Yetenekleri

Paket Yakalama ve Arayüz Seçimi

Herhangi bir Wireshark analizinin ilk adımı, uygulamayı başlatmak ve yakalama için uygun ağ arayüzünü seçmektir.[2] Bu, fiziksel bir Ethernet adaptörü, bir Wi-Fi arayüzü veya bir sanal makine (VM) veya konteyner ile ilişkili bir sanal arayüz olabilir. Wireshark, Ağ Arayüz Kartını (NIC) karışık moda (promiscuous mode) alarak çalışır ve bu, yalnızca

yerel makineye yönelik trafik değil, o arayüzden geçen tüm trafiği yakalamasına olanak tanır.[4] Sanal makineler gibi sanallaştırılmış ortamlar için, Wireshark'ın ilgili trafiği yakalaması, VM'nin IP adresine giden veya gelen paketleri izlemek için özel filtreleme gerektirir.[16]

Ekran ve Yakalama Filtrelerinde Uzmanlaşma: Sözdizimi ve En İyi Uygulamalar

Etkili analiz, alakasız verileri filtrelemeyi gerektirir. Wireshark, iki temel filtreleme mekanizması sunar:

- **Yakalama Filtreleri:** Bu filtreler, paketler yakalama dosyasına (PCAP) yazılmadan *önce* uygulanır ve yüksek trafikli yakalamalar sırasında dosya boyutunu ve sistem kaynakları üzerindeki yükü önemli ölçüde azaltır.[17] Berkeley Paket Filtresi (BPF) sözdizimini kullanırlar. Örnekler arasında belirli bir IP adresinden gelen veya giden trafiği yakalamak için host <IP_adresi>, belirli bir TCP portunu yakalamak için tcp port <port_numarası> veya belirli bir MAC adresini yakalamak için ether host <MAC_adresi> yer alır.[17]
- **Ekran Filtreleri:** Yakalamadan *sonra* uygulanır ve temel yakalama dosyasını değiştirmeden görüntülenen paketlerin etkileşimli olarak filtrelenmesine olanak tanır.[2] Mantıksal operatörler (and, or, not) ve karşılaştırma operatörleri (==, !=, <, >) gibi daha ifade edici bir sözdizimi kullanırlar.[6] 'Developer Hunter' projesi için temel ekran filtreleri, belirli protokolleri (ssh, http, mysql), portları (tcp.port == 22, tcp.port == 3306), IP adreslerini (ip.addr == <geliştirici_IP>) ve hatta yükler veya başlıklardaki dize eşleştirmelerini (http.user_agent contains "Mozilla", dns.qry.name contains "example.com", ssh.username == "admin") hedefleyecektir.[2]

Wireshark'ın büyük hacimli trafik için "kaynak yoğun" olabileceği göz önüne alındığında [3], "gürültü" sorunu ve ön-filtreleme zorunluluğu ortaya çıkmaktadır. Karışık modda "tüm trafiği" yakalamak [4] ve ardından filtrelemek, özellikle yoğun bir ağda, verimsizdir ve paket kaybına veya sistem zorlanmasına yol açabilir. Ekran filtreleri güçlü olsa da, yakalama dosyasının boyutunu azaltmazlar.[20] Bu nedenle, yazılımcı trafiğini etkili bir şekilde analiz etmek için, ilk veri hacmini yalnızca ilgili olanla sınırlamak amacıyla *yakalama filtrelerinin* uygulanmasına öncelik verilmelidir. Bu, yakalama öncesinde beklenen yazılımcı trafik kalıplarının güçlü bir şekilde anlaşılmasını gerektirir ve 'Developer Hunter' projesinin başarısı için bir ön koşuldur. Yazılımcı makinelerinin (MAC/IP/ana bilgisayar adı aracılığıyla) erken tespiti, geniş ağ çapında yakalamalar yerine hedeflenmiş yakalama filtrelerinin uygulanması için kritik öneme sahiptir.

İlk Ana Bilgisayar ve Cihaz Tanımlaması

Ağ keşfinde temel adım, trafiği oluşturan cihazları ve kullanıcıları tanımlamaktır. Bu, Dinamik Ana Bilgisayar Yapılandırma Protokolü (DHCP) trafiğini analiz ederek MAC adreslerini, IP adreslerini ve ana bilgisayar adlarını ilişkilendirmekle başlayabilir.[25] Bir MAC adresinin ilk üç baytı genellikle bir satıcı kimliğine (OUI) çözümlenir ve bu, cihaz üreticisi hakkında ipuçları sağlayabilir (örneğin, Apple MacBook Pro).[25] Microsoft Windows veya macOS ana bilgisayarlarının bulunduğu ortamlarda, NetBIOS Ad Hizmeti (NBNS) trafiği de ana bilgisayar adlarını tanımlamak için kullanılabilir.[25] Şifrelenmemiş web trafiği için, HTTP istek başlıklarındaki User-Agent satırı işletim sistemini (OS) ve bazen cihaz modelini ortaya çıkarabilir.[25] Ancak, bu yöntem User-Agent dizelerinin manipüle edilebileceği için dikkatli kullanılmalıdır.[25] Active Directory (AD) ortamlarında, Kerberos trafiğindeki CNameString alanı incelenerek kullanıcı hesap adları belirlenebilir.[25]

Ana bilgisayar tanımlamasının hedeflenmiş izleme ile etkileşimi, projenin verimliliğini artırmaktadır. [25]'te belirtilen ana bilgisayar tanımlama yöntemleri (MAC, IP, ana bilgisayar adı, işletim sistemi, AD'deki kullanıcı) sadece envanter için değil, aynı zamanda son derece spesifik ve etkili Wireshark filtreleri oluşturmak için de temeldir. Bir yazılımcının makinesi (IP/MAC/ana bilgisayar adı aracılığıyla) tanımlandıktan sonra, sonraki filtreler o belirli cihaza daraltılabilir.[6] Bu, analiz edilecek trafik hacmini önemli ölçüde azaltır ve sinyal-gürültü oranını iyileştirir. Bu durum, "ana bilgisayarı tanımla -> hedeflenmiş filtreleri uygula -> yazılımcı profilini iyileştir" şeklinde kritik bir geri bildirim döngüsü oluşturmaktadır.

Tablo: İlk Keşif için Temel Wireshark Filtreleri

Etkinlik Türü	Filtre Türü	Filtre İfadesi	Açıklama
Genel Ana Bilgisayar Trafiği	Yakalama	host <IP_Adresi>	Belirli bir IP adresine/adresinden gelen tüm trafiği yakalar.
Genel Ana Bilgisayar Trafiği	Ekran	ip.addr == <IP_Adresi>	Belirli bir IP adresine/adresinden gelen tüm trafiği görüntüler.
DHCP	Yakalama	port 67 or port 68	DHCP isteklerini ve yanıtlarını yakalar.

DHCP	Ekran	dhcp	Yakalanan DHCP paketlerini görüntüler.
NetBIOS	Ekran	nbns	Windows/macOS ana bilgisayar adlarını tanımlamak için NBNS trafiğini görüntüler.
HTTP Kullanıcı Ajanı	Ekran	http.user_agent contains "Windows"	HTTP trafiğinden işletim sistemi/cihaz bilgilerini ortaya çıkarır.
Kerberos Kullanıcı Tanımlaması	Ekran	kerberos ve ardından CNameString sütunu ekle	Active Directory ortamlarında kullanıcı hesap adlarını belirler.
MAC Adresi Tanımlaması	Yakalama	ether host <MAC_Adresi>	Belirli bir MAC adresine/adresinden gelen trafiği yakalar.
MAC Adresi Tanımlaması	Ekran	eth.addr == <MAC_Adresi>	Belirli bir MAC adresine/adresinden gelen trafiği görüntüler.

3. Geliştirici Faaliyet Tespiti için En İyi 10 Gelişmiş Teknik ve Trend (2025 ve Sonrası)

3.1. Gelişmiş Git/SSH Trafik Analizi ve Şifre Çözme

Git işlemleri genellikle SSH (TCP port 22) veya HTTP/HTTPS (TCP port 80/443) üzerinden gerçekleşir. Wireshark, ssh ekran filtresi [28] ve http veya tls filtreleri [23] kullanarak SSH ve HTTP/HTTPS trafiğini tanımlayabilirken, temel zorluk bu protokollerin şifrelenmesinde yatmaktadır.

SSH Şifre Çözme ve TLS/SSL Stratejileri: Wireshark'ın SSH ayrıştırıcısı bağlantı kurulum paketlerini analiz edebilir, ancak şifrelenmiş SSH oturumlarının gerçek veri yükü, paylaşılan gizli anahtar olmadan okunamaz durumdadır.[28] Mevcut Wireshark sürümlerinde, TLS'ye benzer bir anahtar günlüğü dosyası aracılığıyla SSH yüklerinin doğrudan şifre çözümü yerel olarak desteklenmemektedir, ancak bu izlenen bir

özelliktir.[28] Bu durum, SSH içeriğinin derinlemesine incelenmesi için meta verilere güvenmeyi veya kontrollü, şifrelenmemiş ortamlarda (mümkünse) trafik yakalamayı gerektirmektedir.

TLS/HTTPS trafiğinin şifresini çözmek için en etkili ve evrensel yöntem, SSLKEYLOGFILE ortam değişkenini kullanmaktır.[30] Git'in HTTPS kullandığı durumlarda ve diğer HTTPS iletişimlerinde bu yöntem kritik öneme sahiptir. Git istemcisi veya web tarayıcısı gibi uygulamayı başlatmadan önce bu ortam değişkeni ayarlanarak, oturum anahtarları belirtilen bir dosyaya kaydedilir. Wireshark daha sonra bu anahtar günlüğü dosyasını okuyacak şekilde yapılandırılabilir ve HTTPS yükünün tam olarak şifresinin çözülmesini ve incelenmesini sağlar.[30] Bu yöntem, Chrome, Firefox, Node.js ve Java (belirli yardımcılarla) gibi yaygın olarak kullanılan birçok araç için çalışmaktadır.[30]

Şifrelemenin yaygınlaşmasıyla oluşan "görünürlük boşluğu", siber güvenlik analizi için önemli bir zorluktur. TLS 1.3 ve SSH gibi güçlü şifrelemenin tüm ağ katmanlarında varsayılan olarak artan kullanımı, içerik düzeyinde analiz için geleneksel paket incelemesini büyük ölçüde etkisiz hale getirmektedir. 'Developer Hunter' projesi için, yazılımcı iş istasyonlarında ve kritik geliştirme sunucularında SSLKEYLOGFILE gibi anahtar kaydetme mekanizmalarının uygulanması, hassas Git işlemleri, API çağrıları ve diğer veri aktarımları hakkında anlamlı bilgi edinmek için temel bir gerekliliktir. Bu olmadan, analiz meta verilerle sınırlı kalır ve bu, ayrıntılı davranışsal profil oluşturma ve fikri mülkiyet koruması için yetersizdir. Bu durum, dikkatli bir şekilde yönetilmesi gereken önemli bir operasyonel ve gizlilik sorununu ortaya koymaktadır.

Şifrelenmiş trafik analizinde "ne"den "nasıl"a geçiş, analitik yaklaşımda bir evrimi temsil etmektedir. İçerik şifrelendiğinde ve şifre çözme her zaman mümkün olmadığında (örneğin, anahtar günlüğü dosyası olmayan SSH yükleri), analizin odak noktası şifrelenmiş trafiğin kendi özelliklerine kaydırılmalıdır. Bu, bağlantı sıklığı, süresi, paket boyutları ve hedef IP'ler gibi trafik kalıplarını analiz etmeyi içerir.[1] Git/SSH için bu, bilinen Git sunucusu IP'lerine sık bağlantıları (tcp.port == 22 and ip.addr == <Git_Sunucusu_IP>), yaygın Git komutlarıyla ilişkili karakteristik paket boyutlarını (örneğin, git clone veya git push için büyük paketler) veya SSH oturumlarının tipik süresini tanımlamak anlamına gelebilir. Bu "nasıl" analizi, "ne" için bir vekil haline gelir ve şifre çözmenin mümkün olmadığı ortamlarda davranışsal profil oluşturma için kritik öneme sahiptir.

3.2. Entegre Geliştirme Ortamı (IDE) Ağ İmzalarının Profillenmesi

VS Code, IntelliJ IDEA ve Eclipse gibi modern IDE'ler, sadece kaynak kodu yönetiminin ötesinde farklı ağ trafik kalıpları üretir. Bu kalıplar şunları içerir:

- **Yazılım Güncellemeleri ve Eklenti İndirmeleri:** IDE'ler, genellikle HTTPS üzerinden güncellemeleri veya uzantıları sık sık kontrol eder ve indirir. Bunlar, belirli satıcı alan adlarına (örneğin, update.code.visualstudio.com, plugins.jetbrains.com) giden trafik ve karakteristik HTTP User-Agent dizeleri aracılığıyla tanımlanabilir.[2]
- **Telemetri ve Analiz:** Birçok IDE, anonim kullanım verilerini veya çökme raporlarını satıcılara gönderir. Genellikle düşük hacimli olsalar da, bunlar belirli uç noktalar ve yük kalıpları aracılığıyla tanımlanabilir.
- **Uzaktan Geliştirme Trafiği:** VS Code Remote-SSH veya JetBrains Gateway gibi özellikler, uzaktan sunucularda geliştirmeyi kolaylaştırır. Bu trafik genellikle SSH (TCP port 22) veya TCP/IP üzerinden tünellenmiş özel protokolleri kullanır.[28] Bu trafiğin analizi, SSH şifre çözmeyi (özel protokoller kullanılıyorsa ve anahtarlar mevcutsa) veya uzaktan geliştirme sunucularına giden ve bu sunuculardan gelen bağlantı kalıplarına ve veri hacimlerine odaklanmayı gerektirir.[35]
- **Dil Sunucusu Protokolü (LSP) / Hata Ayıklayıcı Protokolü Trafiği:** Bu protokoller, genellikle WebSocket veya TCP üzerinden çalışır ve IDE ile bir dil sunucusu (yerel veya uzaktan) arasında gerçek zamanlı kod analizi, otomatik tamamlama ve hata ayıklamayı kolaylaştırır. Bunları tanımlamak, standart olmayan portlar kullanılıyorsa özel ayrıştırıcılar veya port tabanlı filtreleme gerektirebilir.[36]

IDE'ye Özgü Trafik için Filtreleme Stratejileri:

- tcp.port == 22 (SSH tabanlı uzaktan geliştirme için) [28]
- http | | tls.handshake.type eq 1 (HTTPS üzerinden güncellemeler, telemetri için, şifre çözme mümkünse) [22]
- http.user_agent contains "VSCode" or http.user_agent contains "IntelliJ" (IDE'ler belirli kullanıcı ajanları kullanıyorsa) [6]
- dns.qry.name contains "visualstudio.com" or dns.qry.name contains "jetbrains.com" (güncellemeler/telemetri için alan adı tabanlı filtreleme) [6]
- IDE'ler tarafından kullanılan özel protokoller veya standart olmayan portlar için, tcp.port == <özel_port> veya udp.port == <özel_port> yük analizi ile birleştirilir.[17]

Uzaktan geliştirmenin "gizliliği" ve uç nokta düzeyinde yakalama ihtiyacı, projenin kapsamını genişletmektedir. Uzaktan geliştirme araçları, yazılımcı faaliyetinin birincil konumunu yerel iş istasyonundan uzaktan bir sunucuya kaydırmaktadır. Wireshark yalnızca seçilen arayüzün gördüğü trafiği yakalayabilir.[35] Bu durum, yazılımcılar IDE'leri aracılığıyla uzaktan sunucularda çalışıyorsa, karakteristik ağ faaliyetlerinin (örneğin, derleme, test, dosya aktarımları) yerel makinelerinden değil, uzaktan sunucudan kaynaklanabileceği anlamına gelmektedir. Bu durum, 'Developer Hunter' projesinin yakalama yeteneklerini bu uzaktan geliştirme ortamlarına genişletmesini zorunlu kılmaktadır. Bu, uzaktan sunucuda tshark çalıştırmayı ve yakalamaları merkezi bir analiz

noktasına yönlendirmeyi veya bu sunuculara bağlanan ağ cihazlarında ayna/span portlarını kullanmayı içerebilir.[35] Bu, yerel uç nokta izlemesinden dağıtılmış uç nokta izlemesine doğru kritik bir geçişi temsil etmekte, veri toplamanın karmaşıklığını artırmakta ancak yazılımcı davranışının daha eksiksiz bir resmini sunmaktadır.

IDE telemetrisinin profil oluşturma için potansiyel bir "altın madeni" olması, aynı zamanda gizlilik endişelerini de beraberinde getirmektedir. IDE'ler, kullanım kalıpları hakkında değerli bilgiler içerebilecek telemetri verileri gönderirler. Şifresi çözülüp analiz edildiğinde, bu veriler bir yazılımcının tercih ettiği diller, çerçeveler, eklentiler veya hatta projeye özgü etkileşimler hakkında ayrıntılı bilgiler sağlayabilir. Bu ayrıntılı veriler, bireysel yazılımcılar için daha hassas bir "normal" davranış profili oluşturmaya yardımcı olabilir. Ancak, bu tür kişisel kullanım verilerinin toplanması ve analizi, GDPR gibi düzenlemeleri ihlal etme potansiyeli taşıyan önemli etik ve yasal gizlilik endişelerini hemen tetiklemektedir.[37] Bu nedenle, 'Developer Hunter' projesi, derinlemesine bilgi edinme arzusunu, gizlilik yasalarına sıkı sıkıya bağlı kalma ve yazılımcılarla şeffaf iletişim kurma arasında dikkatli bir denge kurmalıdır. Bu, toplanan telemetri verilerinin anonimleştirilmesi veya yalnızca toplu, tanımlanamayan verilere odaklanılması gibi uygulamaları içerebilir.[37]

3.3. Geliştirme/Test Sunucusu ve API Aracı Etkileşimlerinin Derinlemesine Analizi

Yazılımcılar, API'leri test etmek için Postman ve SoapUI gibi araçları yoğun bir şekilde kullanır. Bu araçlar, Wireshark tarafından yakalanabilen HTTP/HTTPS trafiği üretir.

- **API Çağrılarının Yakalanması ve Ayrıştırılması:** API sunucusunun hedef IP'sine ve ilgili portlara (HTTP 80, HTTPS 443 veya özel API portları) göre filtreleme yapmak anahtardır.[15] Postman bir proxy olarak yapılandırılmışsa (varsayılan port 5555), Wireshark bu proxy portundaki trafiği izleyebilir.[39] http.request.method == "POST" veya http.request.uri contains "api/v1" gibi belirli filtreler API çağrılarını izole etmek için kullanılabilir.[23] Genellikle HTTP üzerinden SOAP/XML gönderen SoapUI için, Wireshark'ı ilgili portta HTTP olarak "Çözümle" (Decode As) olarak yapılandırdıktan sonra xml veya belirli URI kalıpları (http.request.uri contains "MyService.asmx") etkili olabilir.[40]
- **Şifre Çözme Kritik Öneme Sahiptir:** Çoğu modern API HTTPS kullandığından, API istek/yanıt yüklerini incelemek için TLS trafiğinin SSLKEYLOGFILE yöntemiyle şifresini çözmek esastır.[30] Bu yükler genellikle hassas veriler veya fikri mülkiyet içerir.
- **Veritabanı Bağlantılarının (örneğin, MySQL) İzlenmesi ve Sorgu Çıkarımı:** Yazılımcılar, geliştirme ve test sırasında veritabanlarıyla sık sık etkileşim kurar. Tipik olarak TCP port 3306'yı kullanan MySQL trafiği derinlemesine analiz edilebilir. Wireshark'ın port 3306'daki trafiği MySQL protokolü olarak "Çözümle" olarak

yapılandırılması gerekir (Analyze > Decode As > tcp_port 3306 > MySQL).[42] mysql veya mysql.query!= "" gibi ekran filtreleri, SQL sorgularının doğrudan görüntülenmesini sağlar.[42] Kolay okunabilirlik için mysql.query için özel bir sütun eklenebilir.[42]

- **Şifreleme Zorluğu:** MySQL bağlantısı SSL/TLS kullanıyorsa, sorgular şifrelenecektir. Test/analiz için, MySQL istemcisinde SSL'yi geçici olarak devre dışı bırakmak (örneğin, mysql --ssl-mode=DISABLED) veya ideal olarak, TLS oturum anahtarlarını yakalamak (istemci uygulaması SSLKEYLOGFILE destekliyorsa) şifre çözme için gereklidir.[42]

Şifrelenmiş dahili iletişimlerin "kör noktası", derinlemesine ağ analizi için önemli bir zorluktur. Harici trafik genellikle şifrelenirken, yazılımcıdan sunucuya ve uygulamadan veritabanına olan dahili trafik de varsayılan olarak artan bir şekilde şifrelenmektedir (örneğin, hizmet ağlarında mTLS).[43] Bu durum, şifre çözme anahtarları sistematik olarak toplanmazsa, geleneksel Wireshark analizi için önemli bir "kör nokta" oluşturmaktadır. 'Developer Hunter' projesi, kritik dahili iletişim yolları için bu anahtarları elde etmeye yönelik politikalar ve teknik mekanizmalar oluşturmalı veya bir yedek olarak meta veri analizine ve davranışsal kalıplara güvenmelidir.[1] Bu, ağ forensiği için kapsamlı bir anahtar yönetim stratejisi ihtiyacını vurgulamaktadır.

Fikri mülkiyet koruması ve iç tehditler için sorgu düzeyinde görünürlüğün değeri çok büyüktür. mysql.query [42] veya API yüklerini (şifre çözüldükten sonra) görebilmek, manipüle edilen *gerçek veriyi* görmek anlamına gelmektedir. Fikri mülkiyetin korunması [44] ve iç tehdit tespiti [1] için, veritabanı sorgularının ve API çağrılarının içeriğini inceleme yeteneği paha biçilmezdir. Bu, yetkisiz veri erişimi, hassas bilgilerin sızdırılması veya şüpheli kod enjeksiyonu girişimlerinin tespit edilmesini sağlar. Bu ayrıntı düzeyi, temel ağ izlemesinin çok ötesine geçmekte ve 'Developer Hunter' projesinin temel hedeflerine doğrudan katkıda bulunmaktadır. Bu, şifre çözme mekanizmalarını uygulama çabasının güvenlik ekipleri için yüksek getirili bir faaliyet olduğunu göstermektedir.

3.4. Sanal Makine ve Konteyner Ağ Etkinliklerinin İzlenmesi

Geleneksel Wireshark, VM'ler, Docker konteynerleri ve Kubernetes podları gibi modern, dinamik ortamlarda önemli zorluklarla karşılaşmaktadır.

- **Sanallaştırılmış Ortamlardaki Zorluklar:**
  - **Geçici Yapı:** Konteynerler ve podlar kısa ömürlüdür, sık sık başlatılır ve kapatılır, bu da sürekli değişen IP adreslerine ve ağ bağlamlarına yol açar.[47] Bu, belirli örnekleri izlemeyi zorlaştırır.
  - **Ağ Soyutlaması:** Kubernetes, Wireshark'ın doğal olarak farkında olmadığı

soyutlama katmanları (podlar, hizmetler, dağıtımlar) sunar, bu da ham paket verilerini belirli uygulama bileşenleriyle ilişkilendirmeyi zorlaştırır.[47]
    - **Trafik Hacmi:** Mikro hizmet mimarileri içindeki doğu-batı trafiğinin (hizmetler arası iletişim) hacmi, geleneksel yakalama yöntemlerini bunaltabilir.[1]
- **Çözümler ve Özel Araçlar:**
    - **VM Trafiği:** Wireshark, uygun sanal arayüzü seçerek veya VM'nin IP adresine göre filtreleme yaparak VM'lerden trafik yakalayabilir.[16]
    - **Docker: Edgeshark** gibi araçlar, Docker konteynerlerinin içindeki ve dışındaki iletişimi tek tıklamayla Wireshark'a bağlayan bir eklenti sağlar ve manuel yapılandırma karmaşıklıklarını ortadan kaldırır.[48]
    - **Kubernetes:**
        - **Kubeshark:** "Kubernetes için Wireshark" olarak konumlandırılmış olup, Kubernetes kümeleri içindeki API çağrılarını izlemeye odaklanır. kubectl komutları ve diğer dahili API trafiğini gözlemlemek için görsel bir arayüz sağlar.[49] Bu, yazılımcıların orkestrasyon katmanıyla nasıl etkileşim kurduğunu anlamak için kritik öneme sahiptir.
        - **Falco + tshark Entegrasyonu:** Kubernetes'te hedeflenmiş ağ forensiği için, Falco (bulut tabanlı bir tespit motoru) ile tshark entegrasyonu, olay odaklı paket yakalamalarına olanak tanır. Falco, Kubernetes bağlamına dayalı şüpheli etkinlikleri tespit eder ve ilgili podlar/hizmetlerden yalnızca ilgili trafiği yakalamak için tshark'ı tetikler, bu da gürültüyü önemli ölçüde azaltır ve olay müdahalesini iyileştirir.[47]
    - **Hizmet Ağları (Istio, Linkerd):** Hizmet ağları [11] güvenliği (mTLS, sıfır güven) ve gözlemlenebilirliği artırırken, varsayılan olarak hizmetler arası iletişimi de şifreler.[43] Bu, küme içinde bile trafiğin genellikle şifrelendiği anlamına gelir ve şifre çözme stratejileri veya meta veri/davranışsal analize güvenmeyi gerektirir.

Bulut tabanlı ortamlarda "bağlam farkındalıklı" ağ analizinin gerekliliği, geleneksel araçların yetersiz kaldığı bir alandır. Geleneksel Wireshark, geçici IP'ler ve Kubernetes soyutlamalarıyla zorluklar yaşar.[47] Kubeshark ve Falco gibi araçlar "Kubernetes bağlamı" sağlar.[47] 2025 ve sonrasında, yazılımcı faaliyetleri büyük ölçüde bulut tabanlı olacaktır. Yalnızca ham paketleri yakalamak yetersizdir; verilerin podların, hizmetlerin ve dağıtımların dinamik bağlamıyla ilişkilendirilmesi gerekir. Bu, yalnızca ham paket analizinden, bu bağlamı sağlayan entegre çözümlere doğru bir geçişi ifade eder. 'Developer Hunter' projesi, bulut tabanlı soyutlamaları anlayan ve daha üst düzey olaylara dayalı hedeflenmiş yakalamaları tetikleyebilen araçlara yatırım yapmalıdır. Bu, manuel Wireshark işlemlerinden otomatik, bağlam açısından zengin izlemeye geçişi temsil eder.

Hizmet ağı şifrelemesinin "çift taraflı kılıç" etkisi, güvenlik ve görünürlük arasında bir denge kurma ihtiyacını ortaya koymaktadır. Hizmet ağları, dahili trafiği şifreleyerek mikro hizmetlerin güvenlik duruşunu önemli ölçüde iyileştirirken, aynı zamanda Wireshark gibi geleneksel paket analizörlerinin görünürlüğünü azaltmaktadır.[43] Bu durum, 'Developer Hunter' projesinin, hizmet ağı gözlemlenebilirlik özellikleriyle entegre olmanın yollarını bulması (bu, trafik meta verilerini veya şifresi çözülmüş yükleri ortaya çıkarabilir) veya ağ içinde şifre çözme stratejileri uygulaması (örneğin, yan araba proxy'lerinden anahtarları kaydetmek, destekleniyorsa) gerektiği anlamına gelmektedir. Zorluk, ağın güvenlik faydalarından ödün vermeden görünürlüğü sürdürmektir.

3.5. Özel Protokol ve Olağandışı Port Kullanımının Tespiti

Yazılımcılar, çeşitli nedenlerle (örneğin, gizleme, belirli geliştirme kurulumları, port çakışmalarını önleme) genellikle özel uygulama protokolleri kullanır veya standart hizmetleri standart olmayan portlarda çalıştırır. Bu "olağandışı" kalıpları tespit etmek, yazılımcı profil oluşturmanın önemli bir yönüdür.

- **Olağandışı Port Kullanımının Tanımlanması:** Wireshark, port numarasına göre trafiği filtreleyebilir (tcp.port == <port>, udp.port == <port>).[17] "Olağandışı" portları bulmak için, yaygın portlar filtrelenebilir (örneğin, !(tcp.port == 80 or tcp.port == 443 or tcp.port == 22 or tcp.port == 3306)) ve ardından kalan trafik incelenebilir.[27] İletişim kalıplarındaki anomaliler, örneğin standart olmayan portlardaki trafik, yapay zeka/makine öğrenimi tabanlı NDR çözümleri için göstergelerdir.[6]
- **Özel Protokol Ayrıştırması:** Özel bir protokol tanımlandığında (örneğin, port analizi veya gözlemlenen kalıplar aracılığıyla), Wireshark'ın **"Çözümle" (Decode As)** özelliği paha biçilmezdir.[51] Bu, bir analistin Wireshark'a belirli bir porttaki trafiği, varsayılan olmasa bile bilinen bir protokol olarak yorumlamasını talimat vermesine olanak tanır. Gerçekten tescilli veya yeni geliştirilen protokoller için, Wireshark'ın Lua API'si kullanılarak **özel ayrıştırıcılar** yazılabilir.[36] Bu, özel protokolün alanlarının ve yüklerinin derinlemesine incelenmesini sağlayarak yazılımcıya özgü iletişimlere ilişkin ayrıntılı bilgiler sunar.

Özel protokollerin potansiyel bir "iç tehdit" göstergesi olarak kullanılması, dikkatli bir izleme gerektirir. Bir yazılımcı tarafından özel veya olağandışı protokollerin/portların kullanılması, iş akışının meşru bir parçası olabilir (örneğin, dahili RPC, hata ayıklama). Ancak, aynı zamanda bir iç tehdidin verileri sızdırmaya veya gizlenmiş kanallar kullanarak komuta ve kontrol (C2) sunucularıyla iletişim kurmaya çalıştığının bir göstergesi de olabilir.[1] 'Developer Hunter' projesi, geliştirme ekipleri içinde beklenen özel protokol kullanımına ilişkin temel referans çizgileri oluşturmalıdır. Bu temel çizgiden sapmalar veya tamamen yeni, belgelenmemiş özel protokollerin keşfi, yüksek

öncelikli uyarıları ve daha fazla araştırmayı tetiklemelidir. Bu, meşru özel trafiği anlamak için güvenlik ve geliştirme ekipleri arasında güçlü bir işbirliği gerektirir.

Gizleme ve ayrıştırma arasındaki "silahlanma yarışı", sürekli adaptasyon ihtiyacını vurgulamaktadır. Güvenlik izleme daha sofistike hale geldikçe, yazılımcılar (kötü niyetli veya başka türlü) ağ faaliyetleri için daha gelişmiş gizleme tekniklerine başvurabilirler. Bu durum, 'Developer Hunter' projesinin ayrıştırma yeteneklerini sürekli olarak uyarlaması gereken sürekli bir "silahlanma yarışı" yaratmaktadır. Özel Lua ayrıştırıcıları yazma becerilerine yatırım yapmak ve gizli protokolleri (örneğin, yaygın protokoller üzerinden tünelleme, steganografi) nasıl tanımlayacağını anlamak, potansiyel kaçınma taktiklerinin önünde kalmak için kritik öneme sahip olacaktır. Bu, sürekli bir öğrenme ve adaptasyon zorluğudur.

3.6. Modern Protokoller için Gelişen TLS/SSL Şifre Çözme (TLS 1.3, DoH/DoT)

Ağ trafiğinin büyük çoğunluğu, yazılımcıyla ilgili faaliyetler (Git, API çağrıları, paket yöneticisi indirmeleri) dahil olmak üzere artık şifrelenmiştir ve TLS 1.3 modern standarttır.[30] Bu şifreleme, güvenlik için faydalı olsa da, derin paket incelemesi için önemli bir zorluk oluşturmaktadır.

- **Gelişmiş Şifre Çözme Yöntemleri:**
  - **SSLKEYLOGFILE:** Bu, TLS trafiğinin şifresini çözmek için en sağlam ve evrensel yöntem olmaya devam etmektedir.[30] Yazılımcı makinelerinde (tarayıcılar, Node.js, Java uygulamaları vb. için) bu ortam değişkeni ayarlanarak, oturum anahtarları kaydedilir ve Wireshark'ın yakalanan verileri şifresini çözmesine olanak tanır. Bu, yazılımcıya özgü HTTPS trafiğinin içeriğini incelemek için kritik öneme sahiptir.
  - **RSA Özel Anahtarları:** Modern TLS için daha az etkilidir (TLS 1.3 veya Diffie-Hellman anahtar değişimi ile çalışmaz), ancak belirli eski senaryolarda hala uygulanabilir.[31]
  - **Ön Paylaşımlı Anahtarlar (PSK):** Öncelikle IoT cihazları ve belirli gömülü sistemler için kullanılır.[31]
- **Şifrelenmiş DNS (DoH/DoT):** DNS over HTTPS (DoH) ve DNS over TLS (DoT), DNS sorgularını şifrelemek ve gizliliği artırmak için tarayıcılar ve işletim sistemleri tarafından giderek daha fazla benimsenmektedir.[52]
  - **DoT:** Özel bir TCP portu olan 853'ü kullanır, bu da tanımlanmasını kolaylaştırır (tcp.port == 853).[52]
  - **DoH:** Düzenli HTTPS ile aynı olan TCP port 443'ü kullanır, bu da diğer web trafiğinden ayırt edilmesini zorlaştırır (tcp.port == 443).[52] DoH'yi tanımlamak, açık metin DNS'nin (port 53) *yokluğuna* ve potansiyel olarak şifreli akış içindeki paket boyutlarının veya HTTP istek kalıplarının analizine dayanır.[52] DoH'nin

şifresini çözmek, diğer TLS trafiğiyle aynı anahtar günlüğü dosyası yöntemini gerektirir.[54]

Güçlü şifrelemenin yaygınlaşması, güvenlik ekipleri için "gizlilik ve görünürlük" ikilemini yaratmaktadır. Şifreleme (TLS 1.3, DoH/DoT) gizlilik için tasarlanmıştır [30], ancak güvenlik ekiplerinin ağ görünürlüğüne ihtiyacı vardır. SSLKEYLOGFILE teknik bir çözüm sunsa da, uygulanması etik ve yasal sonuçların dikkatli bir şekilde değerlendirilmesini gerektirir, özellikle çalışan izlemesi konusunda.[37] 'Developer Hunter' projesi, bu hassas dengeyi yönetmeli, izleme uygulamaları hakkında yazılımcılarla şeffaflığı sağlamalı ve veri gizliliği düzenlemelerine sıkı sıkıya uymalıdır. Bu sadece teknik bir zorluk değil, aynı zamanda bir politika ve güven oluşturma sorunudur.

Şifrelenmiş ortamlarda meta verilerin gelişen rolü, analitik yaklaşımların adaptasyonunu zorunlu kılmaktadır. Tam şifre çözme mümkün olmadığında bile, meta veri analizi giderek daha önemli hale gelmektedir. Örneğin, bilinen kötü niyetli DoH çözümleyicilerine bağlantıların, olağandışı DoH sorgu hacimlerinin veya belirli paket boyutu kalıplarının tanımlanması [53], DNS sorgusunun içeriğini açığa çıkarmadan şüpheli etkinliği hala gösterebilir. Bu durum, 'Developer Hunter' projesinin, mümkün olduğunca tam paket incelemesini tamamlayarak, şifrelenmiş trafik meta verilerine dayanarak niyeti çıkarabilen ve anomalileri tanımlayabilen sofistike davranışsal analizler geliştirmesi gerektiği anlamına gelmektedir.

Tablo: Wireshark TLS/SSL Şifre Çözme Yöntemleri

Yöntem	Açıklama	Uygulanabilirlik (TLS Sürümleri, Anahtar Değişimi)	Gereksinimler	Sınırlamalar
Anahtar Günlüğü Dosyası (SSLKEYLOGFILE)	Uygulama (tarayıcı, Node.js, Java) tarafından ortam değişkeni aracılığıyla oturum anahtarlarının bir dosyaya kaydedilmesi. Wireshark bu dosyayı	Evrensel (TLS 1.2, TLS 1.3, Diffie-Hellman (DH) dahil) [30]	SSLKEYLOGFILE ortam değişkeninin ayarlanması. Uygulamanın anahtar günlüğü özelliğini desteklemesi.	Safari, eski Edge sürümleri için yerel destek yok. Güvenlik riski (anahtarların kaydedilmesi). [30]

	okuyarak şifreyi çözer.			
RSA Özel Anahtarı	Sunucu özel anahtarının Wireshark'a sağlanmasıyla şifre çözme.	Sınırlı durumlar: DH kullanılmayan şifreleme paketleri, SSLv3, (D)TLS 1.0-1.2 (TLS 1.3 için çalışmaz). Sunucu sertifikasıyla eşleşmeli. [31]	RSA özel anahtar dosyası (PEM veya PKCS#12 formatında).	TLS 1.3 ile uyumlu değil. DH anahtar değişimiyle çalışmaz. Oturum yeniden başlatıldığında çalışmayabilir. [31]
Ön Paylaşımlı Anahtar (PSK)	Önceden paylaşılan bir anahtarın Wireshark'a sağlanmasıyla şifre çözme.	Genellikle IoT cihazları ve belirli gömülü sistemler tarafından kullanılır. [31]	PSK'nin hex formatında elde edilmesi ve Wireshark'ta yapılandırılması.	Daha az yaygın kullanım.

3.7. Davranışsal Anomali Tespiti için Yapay Zeka/Makine Öğrenimi Kullanımı

2025 ve sonrasında, yazılımcı faaliyetleri için yalnızca imza tabanlı tespit yöntemlerine güvenmek yetersiz kalacaktır. Geliştirmenin dinamik doğası ve tehditlerin (iç tehditler dahil) karmaşıklığı, **yapay zeka (YZ) ve makine öğrenimi (ML) destekli davranışsal analize** geçişi zorunlu kılmaktadır.[33]

- **Normal Temel Çizgilerin Oluşturulması:** YZ/ML çözümleri, bireysel yazılımcılar, geliştirme ekipleri ve ilgili cihazları ile uygulamaları için "normal davranışsal bir profil" oluşturacaktır.[33] Bu temel çizgi, tipik trafik hacimlerini, bağlantı kalıplarını, sıkça erişilen kaynakları, yaygın protokolleri ve hatta günün belirli saatlerindeki faaliyetleri içerecektir.
- **Anomali Tespiti:** ML algoritmaları (denetimli ve denetimsiz öğrenme), bu belirlenmiş normal profilden sapmaları sürekli olarak gerçek zamanlı ağ trafiğinde izleyecektir.[33] Yazılımcı faaliyetleriyle ilgili anomalilere örnekler şunlardır:
  - **Olağandışı Veri Aktarımları:** Ağdan çıkan veya olağandışı harici hedeflere (örneğin, kişisel bulut depolama, bilinmeyen IP'ler) aktarılan büyük miktarda veri.[1]
  - **Hassas Bilgilere Erişim:** Bir çalışanın hassas kod depolarına, üretim veritabanlarına veya fikri mülkiyete ani ve olağandışı erişim artışları.[1]
  - **Standart Olmayan Protokol/Port Kullanımı:** Geliştirme işiyle tipik olarak

ilişkilendirilmeyen portlarda veya protokollerle iletişim.[6]
    - **Olağandışı Bağlantı Kalıpları:** Komuta ve kontrol (C2) sunucularına bağlantılar, botnet etkinliği veya ağ içinde yanal hareket.[1]
    - **Mesai Dışı Faaliyet:** Belirli bir yazılımcı veya ekip için olağandışı saatlerde geliştirme faaliyeti.
- **YZ/ML Yetenekleri:** Bu sistemler, büyük veri kümelerini (meta veriler ve tam paket yakalamaları) işleyebilir, ince kalıpları tanımlayabilir ve hatta şifreli trafikteki tehditleri tespit edebilir.[33] Tehdit azaltmayı otomatikleştirebilir ve tahmine dayalı bakım bilgileri sağlayabilirler.[56]

Yazılımcı profillemesinde "insan unsuru", YZ/ML'nin temel bir katkısıdır. YZ/ML, *kullanıcılar* için "normal davranışsal profiller" oluşturur.[33] İç tehditler karmaşıktır ve genellikle davranışsaldır.[1] Yazılımcıları tanımlamak sadece teknik imzalardan ibaret değildir; teknik bir bağlamda insan davranış kalıplarını anlamakla ilgilidir. YZ/ML'nin davranışsal analizdeki gücü, 'Developer Hunter' projesinin statik kurallardan dinamik profillemeye geçmesine olanak tanır. Bu, ağda *kimin ne yaptığını* tanımak ve kötü niyetli veya kasıtsız bir iç tehdidi, hesap ele geçirmeyi veya hatta yetkisiz araçlarla deney yapan bir yazılımcıyı gösterebilecek sapmaları işaretlemek anlamına gelir. Bu, "normal"in dikkatli bir şekilde tanımlanmasını ve yanlış pozitifleri azaltmak için ML modellerini iyileştirmek için bir geri bildirim döngüsünü gerektirir.

Ağ forensiği için ölçeklenebilirlik zorunluluğu, YZ/ML'nin kritik bir rol oynadığı diğer bir alandır. YZ/ML, "büyük hacimli verileri hızlı ve verimli bir şekilde" işler [38], oysa geleneksel araçlar kaynak yoğunudur.[3] Ağlar karmaşıklık ve veri hacmi (özellikle bulut tabanlı mimarilerle) arttıkça, manuel paket analizi ölçeklenemez hale gelir. YZ/ML destekli NTA/NDR çözümleri, 'Developer Hunter' projesinin 2025 ve sonrasında yazılımcı trafiğinin muazzam hacmini yönetmesi için esastır. Wireshark, derinlemesine incelemeler için güçlü olsa da, giderek YZ/ML sistemleri tarafından işaretlenen belirli anomalileri *araştırmak* için bir adli araç olarak hizmet edecektir. Bu entegrasyon (bir sonraki noktada tartışılacaktır) operasyonel verimlilik için kritik öneme sahiptir.

3.8. Wireshark Analizini Betikleme ve CLI Araçlarıyla Otomatikleştirmek

Büyük hacimli ağ verilerini Wireshark GUI ile manuel olarak incelemek, sürekli izleme veya büyük ölçekli araştırmalar için verimsiz ve sürdürülemezdir.[59] 'Developer Hunter' projesinin çabalarını ölçeklendirmek için otomasyon kritik öneme sahiptir.

- **Komut Satırı İşlemleri için tshark:** Wireshark'ın komut satırı arayüzü (CLI) olan tshark, otomatik yakalamalar, toplu işleme ve betikleme için idealdir.[59] Şunlara olanak tanır:
    - **Hedeflenmiş Yakalamalar:** Arayüzlere, protokollere veya süreye dayalı belirli

trafiği yakalama (örneğin, tshark -i eth0 -Y "ssh" -a duration:3600 -w dev_ssh.pcap).[60]
    - **Çevrimdışı Analiz:** Önceden yakalanan PCAP dosyalarına ekran filtreleri uygulama ve belirli alanları veya istatistikleri çıkarma (örneğin, tshark -r dev_traffic.pcap -Y "http.request.method == POST" -T fields -e http.host -e http.request.uri).[60]
    - **Temel Anomali Tespiti:** En çok konuşanları veya potansiyel port tarama faaliyetini tanımlamak için tshark'ın istatistiksel yeteneklerini kullanma.[60]
- **Gelişmiş Betikleme için PyShark: PyShark**, tshark'ı saran güçlü bir Python kütüphanesidir ve Wireshark'ın ayrıştırma yeteneklerine programatik bir arayüz sağlar.[59] Bu şunları sağlar:
    - **Karmaşık Davranışsal Analiz:** Eylem dizilerini, olağandışı protokol kullanımını veya belirli veri kalıplarını tespit etmek için özel mantık uygulayan Python betikleri yazma.[59]
    - **Otomatik Şifre Çözme ve Kod Çözme:** Şifre çözme mantığını (anahtarlar mevcutsa) ve belirli yükler için özel kod çözmeyi entegre etme.[59]
    - **Diğer Araçlarla Entegrasyon:** Ayrıştırılmış ağ verilerini veritabanlarına, SIEM'lere veya özel raporlama araçlarına kolayca besleme.[59]
    - **Canlı ve Çevrimdışı Analiz:** Canlı arayüzlerde gerçek zamanlı analiz yapma veya büyük PCAP dosyası arşivlerini işleme.[59]

"Manuel incelemeden" "otomatik ağ zekasına" doğru evrim, projenin ölçeklenebilirliği için kritik öneme sahiptir. Wireshark GUI manuel inceleme içindir [2], ancak tshark ve PyShark otomasyonu mümkün kılar.[59] 'Developer Hunter' projesinin 2025'te etkili olabilmesi için, yalnızca insan analistlerinin manuel olarak Wireshark yakalamalarını incelemesine güvenemez. Modern ağların ölçeği ve karmaşıklığı otomasyonu gerektirmektedir. tshark ve PyShark, güvenlik ekiplerinin yazılımcı faaliyetlerini sürekli olarak izleyen, analiz eden ve uyarı veren özel betikler oluşturmasına olanak tanıyarak bu geçişi kolaylaştırır ve ham paket verilerini eyleme dönüştürülebilir ağ zekasına dönüştürür. Bu, insan analistlerini sıkıcı veri eleme yerine daha üst düzey tehdit avcılığı ve olay müdahalesi için serbest bırakır.

Özel betikleme, tehdit tespitinde rekabet avantajı sağlamaktadır. PyShark, "belirli tehditlere göre uyarlanmış özel, yeniden kullanılabilir avcılık betikleri" oluşturmaya olanak tanır.[59] Ticari NDR/SIEM çözümleri geniş yetenekler sunarken, PyShark ile özel betikleme, 'Developer Hunter' projesinin kendi geliştirme ortamının benzersiz özelliklerine ve potansiyel iç tehdit senaryolarına göre son derece uzmanlaşmış tespit mantığı geliştirmesine olanak tanır. Bu, hazır çözümlerin eksik kalabileceği bir ayrıntı ve uyarlanabilirlik düzeyi sağlar ve yazılımcıyla ilgili bir güvenlik olayını gösterebilecek

ince, kuruluşa özgü anomalilerin tespitini mümkün kılar. Bu, dahili güvenlik mühendisliği yeteneklerine stratejik bir yatırımdır.

3.9. Daha Geniş Ağ Güvenliği Araçlarıyla Entegrasyon (SIEM, NDR, EDR)

Wireshark, ağ paketlerine ilişkin derin, ayrıntılı bilgiler sağlasa da, nokta-zamanlı veya yerelleştirilmiş bir yakalama aracıdır. Kapsamlı 'Developer Hunter' yetenekleri için, elde ettiği bilgilerin daha geniş bir güvenlik ekosistemine, özellikle **Güvenlik Bilgileri ve Olay Yönetimi (SIEM)**, **Ağ Tespiti ve Yanıtı (NDR)** ve **Uç Nokta Tespiti ve Yanıtı (EDR)** sistemlerine entegre edilmesi gerekmektedir.[62]

- **SIEM/NDR/EDR'nin Rolü:**
  - **Merkezi Veri Alımı ve Korelasyonu:** SIEM'ler, çeşitli kaynaklardan (günlükler, ağ akış verileri, uç nokta telemetrisi) büyük miktarda güvenlik verisi toplar ve normalleştirir.[64] Bu, ağ olaylarının (Wireshark/tshark yakalamalarından) diğer güvenlik olaylarıyla (örneğin, oturum açma girişimleri, dosya erişimi, süreç yürütme) ilişkilendirilmesini sağlayarak bir olayın bütünsel bir görünümünü oluşturur.
  - **YZ/ML Destekli Anomali Tespiti:** YZ/ML tarafından giderek daha fazla desteklenen NDR çözümleri, anomaliler ve temel çizgilerden sapmalar için ağ trafiğini sürekli olarak analiz ederek gerçek zamanlı tehdit tespiti sağlar.[33] Yazılımcı faaliyetlerinin otomatik izlenmesinin büyük bir kısmı burada gerçekleşecektir.
  - **Gelişmiş Tehdit Avcılığı:** SIEM/NDR platformları, güvenlik analistlerinin büyük veri kümelerinde ince uzlaşma göstergelerini aramasına olanak tanıyarak proaktif tehdit avcılığını kolaylaştırır.[64] Wireshark'tan türetilen PCAP'lar, bir anomali SIEM/NDR tarafından işaretlendikten sonra derinlemesine adli analiz için kritik hale gelir.[63]
  - **Olay Müdahalesi:** Ağ verilerini uç nokta ve günlük verileriyle ilişkilendirmek, saldırının kapsamı, etkisi ve temel nedeni hakkında daha net bir resim sağlayarak olay müdahalesini hızlandırır.[63]
- **Entegrasyon Mekanizmaları:**
  - **PCAP Alımı:** SIEM/NDR platformları, Wireshark veya tshark tarafından oluşturulan PCAP dosyalarını geriye dönük analiz için alabilir.[63]
  - **Uyarı Yönlendirme:** Otomatik betikler (örneğin, PyShark), Wireshark çıktısını ayrıştırabilir ve belirli uyarıları veya özetlenmiş verileri SIEM'lere iletebilir.[59]
  - **API Entegrasyonu:** Bazı gelişmiş araçlar, SIEM uyarılarına dayalı olarak yakalamaların veya veri dışa aktarımının otomatik olarak tetiklenmesine olanak tanıyan programatik etkileşim için API'ler sunar.

Wireshark'ın birincil izleme aracından adli derinlemesine inceleme aracına dönüşümü,

bu entegrasyonun doğal bir sonucudur. YZ/ML destekli NDR/SIEM'ler gerçek zamanlı, büyük ölçekli izleme içindir.[33] Wireshark ise "derin inceleme" ve "sorun giderme" için kullanılır.[2] PCAP analizi "adli soruşturmalar" içindir.[63] 2025'te Wireshark, büyük kurumsal ağlar için, özellikle yazılımcı faaliyetleri için birincil, sürekli gerçek zamanlı izleme aracı olmayacaktır. Gücü, mikroskobik ayrıntıda yatmaktadır. Bu nedenle, 'Developer Hunter' projesindeki rolü evrilecektir: daha üst düzey YZ/ML destekli SIEM/NDR sistemleri tarafından işaretlenen belirli şüpheli olayların *uyarı sonrası soruşturması* ve *derinlemesine adli analizi* için başvurulan araç olacaktır. Bu, 'Developer Hunter' projesinin verimli PCAP toplama ve yönetimine ve Wireshark ile bu daha geniş güvenlik platformları arasında sorunsuz entegrasyona odaklanması gerektiği anlamına gelir.

Otomatik tespit ve insan uzmanlığı arasındaki sinerjik ilişki, güvenlik operasyonlarının geleceğini şekillendirmektedir. YZ/ML anomalileri tespit eder [33], insan analistleri bunları araştırır [64] ve Wireshark ayrıntı sağlar.[2] Wireshark'ın SIEM/NDR/EDR ile entegrasyonu güçlü bir sinerji yaratır. YZ/ML, potansiyel tehditleri ölçekli olarak tanımlayarak "samanlıkta iğne" sorununu çözerken, Wireshark ile donatılmış insan analistleri, tehdidin doğasını doğrulamak ve anlamak için gereken hassas, ayrıntılı araştırmayı gerçekleştirebilir. Bu durum, 'Developer Hunter' projesinin sadece teknik araçlara değil, aynı zamanda hem otomatik sistemleri hem de derin adli yetenekleri etkili bir şekilde kullanabilen yetenekli bir ekibe ihtiyaç duyduğunu göstermektedir.

3.10. Bulut Tabanlı ve Uzaktan Çalışma Ortamları için Ağ Forensiği Gelecek Trendleri

Bulut tabanlı geliştirmeye geçiş, 2025 için belirleyici bir trenddir.[9]

- **Bulut Tabanlı Mimariler (Mikro Hizmetler, Konteynerler, Sunucusuz):** Bağımsız, gevşek bağlı hizmetlerin Docker ve Kubernetes'te yaygınlaşması [11], ağ trafik kalıplarının son derece dinamik ve dahili (doğu-batı trafiği) olduğu anlamına gelir. Geleneksel çevre tabanlı Wireshark yakalamaları yetersizdir. Kubeshark ve Falco/tshark entegrasyonu gibi araçlar, bu geçici ortamlarda bağlam farkındalıklı izleme için vazgeçilmez hale gelmektedir.[47] Sunucusuz işlevler (FaaS), geçici doğaları ve geleneksel sunucu tarafı görünürlüğünün olmaması nedeniyle benzersiz izleme zorlukları sunar.[65] Bulut sağlayıcıları bazı günlük kaydı/metrikler sunsa da [65], Wireshark ile ayrıntılı paket düzeyinde analiz, işlevler son derece soyutlanmış ortamlarda yürütüldüğü için zordur. Odak noktası API ağ geçidi trafiğine, harici hizmet çağrılarına ve bulut sağlayıcı günlükleriyle korelasyona kaymaktadır.
- **Uzaktan ve Hibrit Çalışma Modelleri:** Uzaktan ve hibrit çalışmanın yaygın olarak benimsenmesi [13], yazılımcı ağ faaliyetlerinin artık kurumsal LAN ile sınırlı olmadığı

anlamına gelir. Bu durum şunları gerektirir:
    - **Uç Nokta Merkezli İzleme:** Yazılımcı dizüstü bilgisayarlarında uç nokta düzeyinde paket yakalamaya (örneğin, tshark) veya uzaktan yakalama tekniklerine daha fazla güvenme.[35]
    - **VPN/Güvenli Tünel Analizi:** Trafik genellikle şifreli olan VPN'ler üzerinden geçer. VPN tünellerinin şifresini çözme (anahtarlar mevcutsa) veya VPN meta verilerinin analizi önemli hale gelir.
- **Kalıcı Şifreleme Zorlukları:** Her yerde bulunan şifrelemeye (TLS 1.3, DoH/DoT) yönelik eğilim, derin içerik incelemesini zorlaştırmaya devam edecektir.[30] SSLKEYLOGFILE yöntemi kritik olmaya devam edecek, ancak güvenlik ekipleri şifre çözmenin imkansız olduğu senaryolar için gelişmiş meta veri analiz teknikleri de geliştirmelidir.
- **Siber Güvenlik ve Dijital Adli Tıpın Yakınsaması:** Proaktif siber güvenlik ile reaktif dijital adli tıp arasındaki çizgiler bulanıklaşmaktadır.[38] Adli tıp ekiplerinin olay müdahalesine dahil olduğu gerçek zamanlı adli tıp daha belirgin hale gelecektir. Bu, Wireshark analizinin hızlı olay soruşturması için SIEM/NDR/EDR platformlarıyla sıkı bir şekilde entegre olacağı anlamına gelir.[38]
- **Adli Tıpta YZ/ML:** YZ/ML, dijital adli tıpta veri analizini, anomali tespitini ve tahmine dayalı değerlendirmeleri otomatikleştiren bir oyun değiştirici olacaktır.[38] Bu, yazılımcı profillemesi için büyük hacimli ağ verilerinin daha verimli işlenmesini sağlayacaktır.

"Dağıtılmış dijital ayak izi" ve çok yönlü bir yaklaşıma duyulan ihtiyaç, projenin kapsamını genişletmektedir. Yazılımcı faaliyetleri artık merkezi değildir (uzaktan, bulut tabanlı). Geleneksel Wireshark yereldir.[2] Konteynerler/Kubernetes için özel araçlar mevcuttur.[48] Bulut forensiği karmaşıktır.[38] 'Developer Hunter' projesi, tek, monolitik bir Wireshark dağıtımına güvenemez. Aşağıdakileri birleştiren çok yönlü bir yaklaşım gerektirir:

1. Yazılımcı iş istasyonlarında anında, derinlemesine incelemeler için **yerel Wireshark/tshark**.
2. Konteynerli ortamlar için **özel araçlar** (Kubeshark, Edgeshark).
3. Sunucusuz ve bulut altyapısı için **bulut sağlayıcı günlük kaydı/izleme ile entegrasyon**.
4. Tüm bu dağıtılmış kaynaklardan veri toplamak ve analiz etmek için **merkezi SIEM/NDR çözümleri**. Bu, çeşitli bir araç setine ve bu farklı ortamlarda çalışabilen yetenekli personele önemli bir yatırım yapılmasını gerektirir.

Teknolojinin gelişmesiyle birlikte etik ve yasal çerçevelerin de gelişmesi gerektiği, projenin sürdürülebilirliği için kritik öneme sahiptir. Dijital adli tıp, veri gizliliği ile ilgili

etik zorluklarla karşı karşıyadır.[38] Çalışan izlemesinin yasal sonuçları vardır.[37] Ağ izleme yetenekleri daha yaygın ve ayrıntılı hale geldikçe (özellikle YZ/ML profillemesiyle), yazılımcı faaliyetlerini izlemenin etik ve yasal sonuçları hayati hale gelmektedir. 'Developer Hunter' projesi, hangi verilerin toplandığı, nasıl kullanıldığı, kimlerin erişebildiği ve hangi amaçla kullanıldığına ilişkin açık, şeffaf politikalar proaktif olarak geliştirmelidir. Gelişen veri koruma yasalarına (GDPR vb.) uyum ve güven kültürü oluşturmak, başarılı uygulama ve yasal sonuçlardan veya olumsuz çalışan moralinden kaçınmak için kritik öneme sahiptir. Bu, 2025'te herhangi bir etkili izleme programı için sadece "olması gereken" değil, temel bir gerekliliktir.

Tablo: Gelişen Ağ Trendleri ve Yazılımcı Faaliyet İzlemesi Üzerindeki Etkisi (2025+)

Trend	Yazılımcı Ağ Trafiği Üzerindeki Etki	Gerekli Adaptasyon/Tamamlayıcı Araçlar
Bulut Tabanlı Mimariler (Mikro Hizmetler, Konteynerler, Sunucusuz)	Geçici IP'ler, dinamik doğu-batı trafiği, geleneksel görünürlük eksikliği.	Kubeshark, Edgeshark, Bulut Sağlayıcı Günlük Kaydı/İzleme, Hizmet Ağı Entegrasyonu.
Uzaktan/Hibrit Çalışma	Ağ faaliyetinin kurumsal LAN dışına kayması, VPN trafiği, uç nokta merkezli izleme ihtiyacı.	Uç nokta düzeyinde tshark yakalamaları, Uzaktan yakalama teknikleri, VPN şifre çözme yetenekleri.
Her Yerde Bulunan Şifreleme (TLS 1.3, DoH/DoT)	İçerik incelemesinde zorluklar, meta veri analizine bağımlılık, DNS çözümlemesinin gizlenmesi.	SSLKEYLOGFILE uygulaması, Gelişmiş meta veri analizi, Şifreli DNS tespiti.
YZ/ML Destekli Ağ Forensiği	Büyük veri hacimlerinin işlenmesi, davranışsal anomali tespiti, tahmine dayalı analiz.	YZ/ML destekli NDR/SIEM çözümleri, PyShark otomasyonu, Gelişmiş veri bilimi yetenekleri.
Siber Güvenlik ve Adli Tıpın Yakınsaması	Gerçek zamanlı olay müdahalesi, proaktif tehdit avcılığı, bütünsel güvenlik görünümü ihtiyacı.	SIEM/NDR/EDR entegrasyonu, Olay müdahale ekiplerinde adli tıp uzmanlığı.

4. Geliştirici Faaliyet İzlemesinin Etik ve Yasal Hususları

Ağ faaliyetinin derinlemesine izlenmesini içeren bir 'Developer Hunter' projesinin uygulanması, bir kuruluşun güvenlik gereksinimleri (fikri mülkiyet koruması, iç tehdit tespiti) ile çalışan gizlilik hakları arasında doğal olarak bir gerilim yaratır.[37] Şirkete ait cihazlarda meşru iş amaçları için izleme, ABD yasalarına göre genellikle yasal olsa da [55], etik hususlar ve gelişen küresel veri koruma düzenlemeleri (GDPR gibi) dikkatli ve şeffaf bir yaklaşım gerektirmektedir.[37]

- **Temel Etik Endişeler:**
  - **Gizlilik İhlali:** Yazılımcılar sürekli izlendiklerini hissedebilir, bu da toksik bir çalışma ortamına ve sadakat kaybına yol açabilir.[37]
  - **Toplanan Verilerin Kötüye Kullanımı/Maruz Kalması:** Güvenli bir şekilde işlenmezse hassas çalışan verilerinin yetkisiz erişim veya maruz kalma riski vardır.[37]
  - **Ayrımcılık/Yanlılık:** İzleme eşit olmayan bir şekilde uygulanırsa veya YZ/ML modelleri yanlıysa kasıtsız ayrımcılık potansiyeli vardır.[37]
- **Yasal Uyumluluk:** Kuruluşlar, izleme uygulamalarını geçerli yasa ve düzenlemelerle uyumlu hale getirmelidir. Bu şunları içerir:
  - **Veri Koruma Yasaları:** Veri toplama, depolama ve işleme konusunda katı gereksinimler getiren GDPR (Avrupa) veya HIPAA (hassas sağlık verileri için) gibi çerçevelere uyum.[37]
  - **Yerel Mevzuat:** Çalışan gizliliğinin makul bir düzeyini gerektirebilecek veya şirkete ait olmayan cihazlarda izlemeyi yasaklayabilecek eyalete özgü yasaların farkında olmak.[37]
- **Etik İzleme için En İyi Uygulamalar:**
  - **Ayrıntılı İzleme Politikası:** Veri toplamanın kapsamını, amacını ve kapsamını ana hatlarıyla belirten açık, yazılı bir politika oluşturulmalıdır.[34]
  - **Şeffaflık:** İzleme niyetleri ve gerekçeleri, uygulamadan *önce* çalışanlara açıkça iletilmelidir. Hangi verilerin toplandığı, nasıl kullanılacağı ve kimlerin erişebileceği konusunda onları bilgilendirilmelidir.[37] Kullanıcılara izlendiklerine dair bildirimler gösterilmelidir.[37]
  - **Kapsamı Sınırlandırma:** Yalnızca şirkete ait cihazlardaki faaliyetler izlenmeli ve veri toplama, meşru iş amaçları için kesinlikle gerekli olanla sınırlandırılmalıdır.[37] Kişisel iletişimler veya mesai dışı faaliyetler izlenmemelidir.
  - **Veri Güvenliği ve Anonimleştirme:** Kötüye kullanımı veya yetkisiz erişimi önlemek için toplanan veriler için sağlam güvenlik önlemleri uygulanmalıdır. Mümkün olduğunda veri anonimleştirmesi düşünülmelidir.[37]
  - **Adalet ve Tutarlılık:** Yanlılık algılarını önlemek için izleme, ilgili tüm çalışanlara

veya ekiplere tutarlı bir şekilde uygulanmalıdır.[37]
    - **Hukuk Danışmanlığı:** Gelişen mevzuata uyumu sağlamak için hukuk profesyonelleriyle işbirliği yapılmalıdır.[34]

Güven, 'Developer Hunter' projesinin kritik bir başarı faktörüdür. Etik olmayan izleme, "toksik çalışma ortamına, sadakat kaybına, düşük çalışan tutulmasına" yol açar.[37] Şeffaflık anahtardır.[37] 'Developer Hunter' projesinin başarısı, yalnızca teknik yeteneklere değil, aynı zamanda yazılımcı iş gücüyle güveni sürdürmeye de bağlıdır. Yazılımcılar, işlerinin doğası gereği genellikle esneklik ve erişim gerektirir. İzleme müdahaleci veya güvensiz olarak algılanırsa, morali, üretkenliği ciddi şekilde etkileyebilir ve yazılımcıların izlemeyi atlatma yollarını bulmasına yol açarak yeni kör noktalar yaratabilir. Bu nedenle, etik ve yasal çerçeve, etkili ve sürdürülebilir yazılımcı faaliyet izlemesi için bir ön koşuldur.

Gelişen düzenleyici ortam, sürekli adaptasyonu zorunlu kılmaktadır. Veri koruma yasaları "daha katı" hale gelmektedir [38] ve hukuk danışmanlığına ihtiyaç duyulmaktadır.[34] İzleme ve veri gizliliği etrafındaki yasal ve etik ortam dinamiktir. 'Developer Hunter' projesi, izleme politikaları ve teknolojileri için sürekli bir inceleme süreci oluşturmalı ve yeni düzenlemelere ve en iyi uygulamalara sürekli uyum sağlamalıdır. Bu, projenin hedeflerinden veya kuruluşun değerlerinden ödün vermeden bu değişikliklere uyum sağlamak için güvenlik, hukuk, İK ve geliştirme ekipleri arasında güçlü, sürekli bir işbirliği gerektirir.

5. Sonuç ve Eyleme Geçirilebilir Öneriler

'Developer Hunter' projesi, yazılımcıların benzersiz ağ faaliyetlerini tespit etmek ve izlemek için Wireshark'ın gücünden yararlanarak kuruluşlar için kritik bir siber güvenlik zorunluluğunu temsil etmektedir. Bu raporun analizi, 2025 ve sonrasında bu alandaki çabaları şekillendirecek temel teknikleri ve gelişen trendleri ortaya koymaktadır.

Temel Çıkarımlar:

- **Şifrelemenin Zorluğu:** TLS 1.3 ve DoH/DoT gibi modern şifreleme protokollerinin yaygınlaşması, derin paket incelemesi için önemli engeller oluşturmaktadır. SSLKEYLOGFILE gibi yöntemler, içerik görünürlüğü için vazgeçilmezdir, ancak etik ve yasal hususlar dikkatli bir şekilde yönetilmelidir.
- **Bağlam Farkındalığının Önemi:** Bulut tabanlı mimariler (mikro hizmetler, konteynerler, sunucusuz) ve uzaktan çalışma modelleri, ağ trafiğini dağıtılmış ve geçici hale getirmektedir. Wireshark'ın tek başına yetersiz kalması, Kubeshark, Edgeshark ve Falco/tshark entegrasyonu gibi bağlam farkındalığına sahip araçların benimsenmesini gerektirmektedir.

- **Otomasyon ve YZ/ML'nin Rolü:** Ağların ölçeği ve karmaşıklığı, manuel analizi sürdürülemez kılmaktadır. tshark ve PyShark gibi araçlarla otomasyon, YZ/ML destekli davranışsal analizlerle birleştiğinde, büyük veri hacimlerini işlemek ve ince anomalileri tespit etmek için kritik öneme sahiptir.
- **Entegrasyonun Gerekliliği:** Wireshark, derinlemesine adli analiz için güçlü bir araç olmaya devam ederken, birincil gerçek zamanlı izleme için SIEM, NDR ve EDR gibi daha geniş güvenlik platformlarıyla entegre edilmelidir. Bu, otomatik tespit ve insan uzmanlığı arasında sinerjik bir ilişki yaratır.
- **Etik ve Yasal Uyumluluk:** Yazılımcı faaliyetlerinin izlenmesi, gizlilik endişelerini artırmaktadır. Şeffaf politikalar, kapsamın sınırlandırılması ve veri koruma düzenlemelerine uyum, projenin başarısı ve çalışan güveninin sürdürülmesi için temeldir.

Uygulama ve Sürekli İzleme için Öncelikli Öneriler:

1. **Kapsamlı Şifre Çözme Yetenekleri Geliştirin:**
    - **Eylem:** Yazılımcı iş istasyonlarında ve kritik geliştirme sunucularında SSLKEYLOGFILE mekanizmalarının sistematik olarak uygulanmasını zorunlu kılın.
    - **Gerekçe:** Bu, Git/HTTPS, IDE telemetrisi ve API çağrıları gibi şifreli geliştirici trafiğinin içeriğini incelemek için hayati öneme sahiptir, aksi takdirde bunlar kör noktalar olarak kalacaktır.
2. **Bulut Tabanlı ve Uç Nokta Merkezli Yakalama Stratejilerini Benimseyin:**
    - **Eylem:** Yalnızca çevre tabanlı yakalamalardan uzaklaşarak, yazılımcıların çalıştığı uzaktan sunuculara, VM'lere ve konteynerlere tshark veya özel araçlar (Kubeshark, Edgeshark) dağıtın.
    - **Gerekçe:** Geliştirici faaliyetleri giderek dağıtılmış ve geçici hale gelmektedir; kapsamlı görünürlük, ilgili tüm uç noktalardan veri toplamasını gerektirir.
3. **YZ/ML Destekli Davranışsal Analize Yatırım Yapın:**
    - **Eylem:** Yazılımcıların "normal" ağ davranışları için temel referans çizgileri oluşturmak ve olağandışı veri aktarımları, erişim kalıpları veya mesai dışı faaliyetler gibi sapmaları tespit etmek için YZ/ML destekli NDR çözümlerini uygulayın.
    - **Gerekçe:** Bu, iç tehditleri, tehlikeye atılmış hesapları ve sıfır gün istismarlarını proaktif olarak tanımlamak için ölçeklenebilir ve dinamik bir yaklaşım sağlar.
4. **Wireshark Analizini Otomatikleştirin:**
    - **Eylem:** Büyük hacimli PCAP dosyalarını verimli bir şekilde işlemek, özel tespit mantığı oluşturmak ve diğer güvenlik araçlarıyla entegrasyon için tshark ve PyShark gibi araçlarla betikleme yeteneklerini geliştirin.
    - **Gerekçe:** Otomasyon, manuel incelemenin sınırlamalarını aşar ve güvenlik

analistlerinin daha yüksek değerli tehdit avcılığı ve olay müdahalesi görevlerine odaklanmasını sağlar.

5. **Kapsamlı Bir Güvenlik Ekosistemiyle Entegre Olun:**
    - **Eylem:** Wireshark'tan elde edilen PCAP'ların ve uyarıların SIEM, NDR ve EDR platformlarına sorunsuz bir şekilde alınmasını ve korelasyonunu sağlayın.
    - **Gerekçe:** Bu entegrasyon, olay müdahalesini hızlandırır, tehdit avcılığını geliştirir ve ağ, uç nokta ve günlük verilerini birleştirerek bütünsel bir güvenlik görünümü sağlar.
6. **Özel Protokol ve Olağandışı Port Kullanımını İzleyin:**
    - **Eylem:** Yazılımcıların kullandığı standart olmayan portları ve özel uygulama protokollerini aktif olarak arayın. Bilinmeyen protokolleri ayrıştırmak için Wireshark'ın "Çözümle" özelliğini ve Lua tabanlı özel ayrıştırıcıları kullanın.
    - **Gerekçe:** Bu kalıplar, meşru geliştirme iş akışlarının bir parçası olabileceği gibi, aynı zamanda veri sızdırma veya gizlenmiş C2 iletişimleri için de kullanılabilir.
7. **Gelişen DNS Protokollerini (DoH/DoT) Göz Önünde Bulundurun:**
    - **Eylem:** Şifreli DNS trafiğini tespit etmek ve analiz etmek için özel filtreler geliştirin. DoH'yi diğer HTTPS trafiğinden ayırt etmek için paket boyutu ve HTTP kalıpları gibi meta veri analizine odaklanın.
    - **Gerekçe:** Şifreli DNS, geleneksel DNS tabanlı tehdit tespitini atlayabilir ve ağ görünürlüğünde bir kör nokta oluşturabilir.
8. **Sürekli Yasal ve Etik Uyum Sağlayın:**
    - **Eylem:** İzleme uygulamaları hakkında yazılımcılarla şeffaf bir iletişim politikası oluşturun ve sürdürün. Veri toplamanın kapsamını meşru iş amaçlarıyla sınırlayın ve tüm ilgili veri koruma yasalarına (örneğin, GDPR) uyun.
    - **Gerekçe:** Güven oluşturmak, olumsuz çalışan moralini önlemek ve yasal sonuçlardan kaçınmak, projenin uzun vadeli başarısı için kritik öneme sahiptir.
9. **Sürekli Eğitim ve Yetenek Gelişimi:**
    - **Eylem:** Güvenlik ve geliştirme ekipleri için gelişmiş Wireshark kullanımı, YZ/ML tabanlı analiz ve bulut tabanlı ağ forensiği konularında düzenli eğitimler sağlayın.
    - **Gerekçe:** Ağ ortamının karmaşıklığı ve tehditlerin gelişimi, sürekli öğrenmeyi ve uyarlamayı gerektirmektedir.
10. **Proaktif Tehdit Avcılığı Yaklaşımını Benimseyin:**
    - **Eylem:** 'Developer Hunter' projesini, yazılımcı faaliyetleri için bir "normal" temel çizgi oluşturmaya odaklanan, sürekli ve proaktif bir tehdit avcılığı programının temel bir bileşeni olarak konumlandırın.
    - **Gerekçe:** Reaktif olay müdahalesinden proaktif tehdit tespitine geçiş, kuruluşun genel güvenlik duruşunu önemli ölçüde güçlendirecektir.

Bu önerilerin uygulanması, kuruluşların yazılımcıların dijital ayak izlerini etkili bir şekilde tespit etmelerini, fikri mülkiyeti korumalarını ve 2025 ve sonrasında gelişen siber tehdit ortamında iç tehditleri azaltmalarını sağlayacaktır.

Alıntılanan çalışmalar

1. Digital Forensics for Insider Threats Detection and Response ..., erişim tarihi Haziran 5, 2025, https://fidelissecurity.com/threatgeek/threat-detection-response/digital-forensics-for-insider-threats-in-it-environments/
2. Wireshark Essentials: Mastering Network Traffic Analysis | Lenovo US, erişim tarihi Haziran 5, 2025, https://www.lenovo.com/us/en/glossary/wireshark/
3. Mastering Wireshark: The Ultimate Guide to Network Traffic Analysis -SecureMyOrg, erişim tarihi Haziran 5, 2025, https://securemyorg.com/2024/12/31/mastering-wireshark-securemyorg/
4. Beginner's Guide to Wireshark - zenarmor.com, erişim tarihi Haziran 5, 2025, https://www.zenarmor.com/docs/network-basics/what-is-wireshark
5. Network Traffic Analysis with Wireshark - Blue Goat Cyber, erişim tarihi Haziran 5, 2025, https://bluegoatcyber.com/blog/what-is-wireshark/
6. How to use advanced display filters in Wireshark for complex ..., erişim tarihi Haziran 5, 2025, https://labex.io/tutorials/wireshark-how-to-use-advanced-display-filters-in-wireshark-for-complex-network-traffic-analysis-in-cybersecurity-415201
7. How Is Wireshark Used in Cybersecurity?, erişim tarihi Haziran 5, 2025, https://kings-guard.com/how-is-wireshark-used-in-cybersecurity/
8. Wireshark • Go Deep, erişim tarihi Haziran 5, 2025, https://www.wireshark.org/
9. 9 Cloud Native Security Tools For 2025 - SentinelOne, erişim tarihi Haziran 5, 2025, https://www.sentinelone.com/cybersecurity-101/cloud-security/cloud-native-security-tools/
10. F5 Extends Ability to Scale and Secure Network Traffic Across Kubernetes Clusters, erişim tarihi Haziran 5, 2025, https://cloudnativenow.com/features/f5-extends-ability-to-scale-and-secure-network-traffic-across-kubernetes-clusters/
11. Beyond Microservices: The Emerging Post-Monolith Architecture for 2025 - DZone, erişim tarihi Haziran 5, 2025, https://dzone.com/articles/post-monolith-architecture-2025
12. Microservices Architecture in 2025: Designing Scalable and Maintainable Applications, erişim tarihi Haziran 5, 2025, https://www.nucamp.co/blog/coding-bootcamp-full-stack-web-and-mobile-development-2025-microservices-architecture-in-2025-designing-scalable-and-maintainable-applications
13. 16 Best Network Monitoring Tools in 2025 - 10XSheets, erişim tarihi Haziran 5, 2025, https://www.10xsheets.com/blog/best-network-monitoring-tools/

14. Top 10 Network Monitoring Tools for 2025: Enhance Your Network Performance - Uptrace, erişim tarihi Haziran 5, 2025, https://uptrace.dev/tools/network-monitoring-tools
15. Analyze Network Traffic with Wireshark Display Filters - LabEx, erişim tarihi Haziran 5, 2025, https://labex.io/tutorials/wireshark-analyze-network-traffic-with-wireshark-display-filters-415944
16. wireshark filters - GitHub Gist, erişim tarihi Haziran 5, 2025, https://gist.github.com/githubfoam/08efac0343f98bd727caa32e6c81f655
17. Steps of Filtering While Capturing in Wireshark - GeeksforGeeks, erişim tarihi Haziran 5, 2025, https://www.geeksforgeeks.org/steps-of-filtering-while-capturing-in-wireshark/
18. How to Filter and Search for Specific Network Traffic in Wireshark - LabEx, erişim tarihi Haziran 5, 2025, https://labex.io/questions/how-to-filter-and-search-for-specific-network-traffic-in-wireshark-288913
19. Wireshark - Filter for Inbound HTTP Requests on Port 80 Only - Server Fault, erişim tarihi Haziran 5, 2025, https://serverfault.com/questions/353952/wireshark-filter-for-inbound-http-requests-on-port-80-only
20. 6.3. Filtering Packets While Viewing - Wireshark, erişim tarihi Haziran 5, 2025, https://www.wireshark.org/docs/wsug_html_chunked/ChWorkDisplayFilterSection.html
21. SF16 - 16: Advanced Wireshark Display Filters (Betty DuBois) - YouTube, erişim tarihi Haziran 5, 2025, https://www.youtube.com/watch?v=EKPef0BFTQY
22. Wireshark Tutorial: Display Filter Expressions - Palo Alto Networks Unit 42, erişim tarihi Haziran 5, 2025, https://unit42.paloaltonetworks.com/using-wireshark-display-filter-expressions/
23. How to filter network traffic based on protocol, port, and HTTP method in Wireshark for Cybersecurity | LabEx, erişim tarihi Haziran 5, 2025, https://labex.io/tutorials/wireshark-how-to-filter-network-traffic-based-on-protocol-port-and-http-method-in-wireshark-for-cybersecurity-415199
24. Wireshark Filters Guide Protocol & IP Analysis - Learn with Yasir, erişim tarihi Haziran 5, 2025, https://yasirbhutta.github.io/wireshark/
25. Wireshark Tutorial: Identifying Hosts and Users - Palo Alto Networks Unit 42, erişim tarihi Haziran 5, 2025, https://unit42.paloaltonetworks.com/using-wireshark-identifying-hosts-and-users/
26. How to find type of a device and its behavior within the network? : r/wireshark - Reddit, erişim tarihi Haziran 5, 2025, https://www.reddit.com/r/wireshark/comments/1fh827g/how_to_find_type_of_a_device_and_its_behavior/
27. Filter all packages that not use a specific port - wireshark - Super User, erişim tarihi Haziran 5, 2025, https://superuser.com/questions/867560/filter-all-packages-that-not-use-a-speci

fic-port
28. SSH - Wireshark Wiki, erişim tarihi Haziran 5, 2025, https://wiki.wireshark.org/SSH
29. Compare SSH and Telnet with Wireshark - YouTube, erişim tarihi Haziran 5, 2025, https://www.youtube.com/watch?v=t0iHo90iqKs
30. Using Wireshark to analyze TLS encrypted traffic | Open200, erişim tarihi Haziran 5, 2025, https://www.open200.com/post/using-wireshark-to-analyze-tls-encrypted-traffic
31. TLS - Wireshark Wiki, erişim tarihi Haziran 5, 2025, https://wiki.wireshark.org/TLS
32. Wireshark Tutorial: Decrypting HTTPS Traffic - Palo Alto Networks Unit 42, erişim tarihi Haziran 5, 2025, https://unit42.paloaltonetworks.com/wireshark-tutorial-decrypting-https-traffic/
33. The power of AI and ML in network traffic analysis: next generation ..., erişim tarihi Haziran 5, 2025, https://socwise.eu/the-power-of-ai-and-ml-in-network-traffic-analysis-next-generation-ndr-solutions/
34. Implementing Network Traffic Analysis for Effective Threat Detection - MoldStud, erişim tarihi Haziran 5, 2025, https://moldstud.com/articles/p-implementing-network-traffic-analysis-for-threat-detection
35. How can I sniff the traffic of remote machine with wireshark? - Server Fault, erişim tarihi Haziran 5, 2025, https://serverfault.com/questions/362529/how-can-i-sniff-the-traffic-of-remote-machine-with-wireshark
36. How can I add a custom protocol analyzer to wireshark? - Stack Overflow, erişim tarihi Haziran 5, 2025, https://stackoverflow.com/questions/4904991/how-can-i-add-a-custom-protocol-analyzer-to-wireshark
37. Employee Monitoring Ethics: 8 Best Practices for Organizations - Syteca, erişim tarihi Haziran 5, 2025, https://www.syteca.com/en/blog/employee-monitoring-ethics-best-practices
38. Key Trends Shaping the Future of Digital Forensics in 2025, erişim tarihi Haziran 5, 2025, https://www.oxygenforensics.com/en/resources/digital-forensics-trends-2025/
39. Capture HTTP requests in Postman | Postman Docs, erişim tarihi Haziran 5, 2025, https://learning.postman.com/docs/sending-requests/capturing-request-data/capturing-http-requests/
40. Capturing SOAP messages using wireshark - Muazzam Ali's Blog - WordPress.com, erişim tarihi Haziran 5, 2025, https://muazzamali.wordpress.com/2014/10/22/capturing-soap-messages-using-wireshark/
41. Capturing and playing back web service calls with Wireshark and SoapUI - Code rant, erişim tarihi Haziran 5, 2025, http://mikehadlow.blogspot.com/2009/01/capturing-and-playing-back-web-service.html

42. How to use wireshark to capture mysql query sql clearly - Stack ..., erişim tarihi Haziran 5, 2025, https://stackoverflow.com/questions/38167587/how-to-use-wireshark-to-capture-mysql-query-sql-clearly
43. Enhancing Security of Cloud-Native Microservices with Service Mesh Technologies - PhilArchive, erişim tarihi Haziran 5, 2025, https://philarchive.org/archive/SIDESO
44. Monitoring And Updating Your Intellectual Property Strategy - FasterCapital, erişim tarihi Haziran 5, 2025, https://fastercapital.com/topics/monitoring-and-updating-your-intellectual-property-strategy.html/1
45. ISO 27001:2022 Annex A 8.16 – Monitoring Activities - ISMS.online, erişim tarihi Haziran 5, 2025, https://www.isms.online/iso-27001/annex-a/8-16-monitoring-activities-2022/
46. What is Network Forensics? - Proven Data, erişim tarihi Haziran 5, 2025, https://www.provendata.com/blog/what-is-network-forensics/
47. Optimizing Wireshark in Kubernetes - Sysdig, erişim tarihi Haziran 5, 2025, https://sysdig.com/blog/optimizing-wireshark-in-kubernetes/
48. [HOWTO] Capture the communication of and inside a Docker ..., erişim tarihi Haziran 5, 2025, https://forums.docker.com/t/howto-capture-the-communication-of-and-inside-a-docker-container-using-wireshark-with-edgeshark-plugin/139440
49. Kubeshark: The Wireshark For Kubernetes - DEV Community, erişim tarihi Haziran 5, 2025, https://dev.to/thenjdevopsguy/kubeshark-the-wireshark-for-kubernetes-3a72
50. Linkerd vs Istio, a service mesh comparison - Buoyant.io, erişim tarihi Haziran 5, 2025, https://www.buoyant.io/linkerd-vs-istio
51. What is Control Protocol Dissection in Wireshark? - GeeksforGeeks, erişim tarihi Haziran 5, 2025, https://www.geeksforgeeks.org/what-is-control-protocol-dissection-in-wireshark/
52. How to verify that you're using DNS over HTTPS (DoH) with Quad9 / other non Cloudflare DNS providers - Super User, erişim tarihi Haziran 5, 2025, https://superuser.com/questions/1708212/how-to-verify-that-youre-using-dns-over-https-doh-with-quad9-other-non-clou
53. Detection of HTTPS Encrypted DNS Traffic - University of Twente Student Theses, erişim tarihi Haziran 5, 2025, https://essay.utwente.nl/82085/1/Nijeboer_BA_EEMCS.pdf
54. Sample captures for QUIC, DoH, CommunityID, WPA3 and other protocols in CloudShark 3.10 - QA Cafe, erişim tarihi Haziran 5, 2025, https://www.qacafe.com/resources/sample-captures-for-quic-doh-communityid-wpa3-cloudshark-3-10
55. Employee Monitoring: Ethical Guidelines for Employers - Teramind, erişim tarihi Haziran 5, 2025, https://www.teramind.co/blog/employee-monitoring-ethics/
56. Network Trends & Predictions with AI and GenAI in 2025 - Comparitech, erişim

tarihi Haziran 5, 2025, https://www.comparitech.com/net-admin/ai-genai-network-trends/
57. What is Network Traffic Analysis? - Traceable AI, erişim tarihi Haziran 5, 2025, https://www.traceable.ai/blog-post/network-traffic-analysis
58. What is Network Traffic Analysis? - Traceable - Blog, erişim tarihi Haziran 5, 2025, https://traceable.ai/blog-post/network-traffic-analysis
59. Threat Hunting with Pyshark: Using Open Source Python Libraries to ..., erişim tarihi Haziran 5, 2025, https://insanecyber.com/threat-hunting-with-pyshark/
60. How to analyze Cybersecurity network traffic with Wireshark CLI - LabEx, erişim tarihi Haziran 5, 2025, https://labex.io/tutorials/wireshark-how-to-analyze-cybersecurity-network-traffic-with-wireshark-cli-415094
61. KimiNewt/pyshark: Python wrapper for tshark, allowing python packet parsing using wireshark dissectors - GitHub, erişim tarihi Haziran 5, 2025, https://github.com/KimiNewt/pyshark
62. How to detect network security threats using Wireshark in ... - LabEx, erişim tarihi Haziran 5, 2025, https://labex.io/tutorials/wireshark-how-to-detect-network-security-threats-using-wireshark-in-cybersecurity-415263
63. How Network Monitoring Detects Insider Threats and Compromised Devices - Exabeam, erişim tarihi Haziran 5, 2025, https://www.exabeam.com/blog/security-operations-center/insider-threats-and-compromised-devices-how-network-monitoring-uncovers-security-blind-spots/
64. SIEM Threat Hunting: Comprehensive Guide - SearchInform, erişim tarihi Haziran 5, 2025, https://searchinform.com/cybersecurity/measures/siem/use-cases/threat-hunting/
65. A comprehensive guide to serverless monitoring and debugging - Bejamas, erişim tarihi Haziran 5, 2025, https://bejamas.com/hub/guides/comprehensive-guide-to-serverless-monitoring-and-debugging
66. Serverless Monitoring Guide - Lumigo, erişim tarihi Haziran 5, 2025, https://lumigo.io/serverless-monitoring-guide/
67. Monitoring network traffic in AWS Lambda functions | AWS Compute Blog, erişim tarihi Haziran 5, 2025, https://aws.amazon.com/blogs/compute/monitoring-network-traffic-in-aws-lambda-functions/
68. Key Trends in Digital Forensics 2025: Chanllenges ... - SalvationDATA, erişim tarihi Haziran 5, 2025, https://www.salvationdata.com/knowledge/key-trends-in-digital-forensics-for-2025/