EKİM-ARALIK 2024

SİBER TEHDİT DURUM RAPORU

STM MÜHENDİSLİK TEKNOLOJİ DANIŞMANLIK

thinktech
STM Teknolojik Düşünce Merkezi

İÇİNDEKİLER

ŞEKİLLER

TABLO LİSTESİ

GİRİŞ

2024 yılının dördüncü çeyreğinde Siber Güvenlik Müdürlüğü tarafından hazırlanan raporumuzda yeni konularla karşınızdayız. İlk olarak, "Zararlı Yazılım Analizi" başlığı altında zararlı yazılımların ne tür yöntemler kullanılarak analiz edilebileceği ve analizi engelleme yöntemleri üzerinde duracağız. Bunun yanında yakın zamanda sık sık gündeme gelen mobil uygulamaların daha güvenli hâle getirilmesi konusundaki bilgilerimizi sizinle paylaşıyoruz. Sonrasında NFT satış platformu kullanıcılarını hedef alan oltalama saldırılarından ve bu tür saldırılarından etkilenmemek için ne tür tedbirler alınabileceğinden bahsedeceğiz.

Ardından günümüzün en gözde uygulama mimarisi olan microservis mimarisinin tasarım ve kullanımında dikkat edilmesi gereken önemli noktalara odaklanıyoruz. Bunu kurumların en hassas bilgilerine erişim sağlayan yetkili hesapların yönetilmesini sağlayan PAM güvenlik ürünün önemi hakkında bilgilendirme izliyor. Gün geçtikçe önemi daha fazla anlaşılmaya başlanan, siber olay müdahale sürecini sistematik bir yaklaşımla ele alarak kurumların siber güvenlik olaylarını hızlı ve etkili bir şekilde yönetmelerine rehberlik edecek bilgilere yer verdik. Son olarak kuantum bilgisayarların mevcut gelişim durumunu, çalışma mantığını, getirdikleri avantajlar ve dezavantajları ele alıyoruz.

Bu son çeyreğin dönem konusu olarak "Hareket Eden, Değişen Hedefler (AMTD)" konusunu sizlerle paylaşıyoruz. Siber saldırıların tespit edilmesi ve harekete geçilmesinden çok uygulamaların ve sistemlerin çalıştığı ortamların, konfigürasyonlarının değiştirilerek saldırganlar için karmaşıklığı artırmak, daha fazla efor sarf etmelerini, zaman harcamalarını ve kaydettikleri ilerlemenin boşa çıkarılmasını sağlamayı amaçlayan uygulamalar ve yöntemler olarak açıklayabileceğimiz bu konuyu inceleyeceğiz.

Son olarak raporumuzda güncel honeypot verilerimize yer ayırdık. Bu rapor, bilişim dünyasındaki güvenlik tehditlerini ve korunma stratejilerini anlamak isteyen herkes için önemli bir kaynaktır. Güvenlik bilincinizi artırmak ve siber tehditlere karşı daha hazırlıklı olmanız için bu raporu incelemenizi tavsiye ederiz. Güvende kalın!

TEKNOLOJİK GELİŞMELER VE SİBER GÜVENLİK

1. Zararlı Yazılım Analizi

Zararlı yazılım (malware) analizi, kötü amaçlı yazılımların sistem üzerindeki işleyiş mekanizmalarını, etkilerini ve muhtemel saldırı vektörlerini derinlemesine anlamak için yürütülen teknik bir süreçtir. Bu analiz, siber güvenlik stratejilerinde hem proaktif savunma mekanizmaları oluşturmak hem de tehdit istihbaratı sağlamak için temel bir gerekliliktir.

Zararlı Yazılım Analizi Teknikleri

Zararlı yazılım analizi, genellikle iki ana kategoriye ayrılır. Statik Analiz, yürütülebilir dosyaların veya kod parçalarının çalıştırılmadan incelenmesiyle gerçekleştirilir. Dinamik Analiz ise zararlı yazılımın izole edilmiş bir test ortamında (sandbox gibi) çalıştırılarak, gerçek zamanlı davranışlarının gözlemlenmesiyle yapılır. Bu iki analiz türü birlikte kullanılarak, zararlı yazılımın sistem üzerindeki etkileri ve potansiyel güvenlik açıkları detaylı şekilde ortaya çıkarılabilir.

Statik Analiz

Statik analiz, yürütülebilir dosyanın içeriğinin derinlemesine incelenmesiyle başlar ve dosyanın formatını belirlemek, meta veri analizi gibi adımları kapsar. İlk aşamada, dosyanın header bilgileri incelenerek format, mimari ve hedef işletim sistemi belirlenir. Bu aşamada zararlı yazılımın içerdiği metin verileri çıkarılır ve şifreleme anahtarları, Command-and-Control (C2) sunucu adresleri gibi anlamlı içerikler araştırılır.

Disassembly (makine kodunun assembly diline dönüştürülmesi) ve decompilation (assembly kodunun daha yüksek seviyeli bir programlama diline, örneğin C'ye dönüştürülmesi) teknikleri, makine kodunun insan tarafından okunabilir forma dönüştürülmesi için kullanılan kritik yöntemlerdir. Disassembly, işlemciye doğrudan verilen talimatların daha okunabilir bir şekilde incelenmesini sağlar ve bu işlem, yazılımın düşük seviyeli çalışmasını anlamak için temel bir adımdır. Örneğin, bir PE (Portable Executable) dosyasının Import Table'ındaki API çağrılarının nasıl işlediğini görmek için disassembly kullanılabilir.

Decompilation, assembly kodunun mantık ve algoritmalarını anlamak için daha yüksek seviyeli bir dile dönüştürülmesini ifade eder. Bu işlem, kodun işlevselliğini detaylı bir şekilde inceleyerek, karmaşık yapıları ve algoritmaları daha anlaşılır hâle getirir. Örneğin, bir şifreleme algoritmasının ayrıntılı yapısını analiz etmek veya kod içinde sıkıştırılmış veri yapılarını çözmek için decompilation sıklıkla kullanılır.

IDA Pro (1996 yılında Hex-Rays tarafından piyasaya sürüldü), gelişmiş bir disassembler ve debugger aracıdır. Disassembly işlemini otomatikleştirir ve kodun görsel bir akış diyagramına dönüştürülmesini sağlar. Genellikle

Windows, Linux, macOS gibi platformlarda çalışır ve zararlı yazılımların yapısını detaylı şekilde analiz etmek için kullanılır. Bu araç, analistin karmaşık fonksiyonları daha kolay anlamasına yardımcı olur. Örneğin, zararlı yazılımın hangi modülleri yüklediğini veya hangi sistem çağrılarını gerçekleştirdiğini görselleştirmek mümkündür.

Radare2 (2006 yılında açık kaynaklı bir proje olarak başlatıldı), güçlü betikleme yetenekleriyle disassembly işlemlerini otomatikleştirmek ve geniş ölçekli analizleri gerçekleştirmek için tercih edilir. Çapraz platform desteği sunar ve gömülü sistemlerden masaüstü işletim sistemlerine kadar çeşitli ortamlarla uyumlu çalışabilir.

Hopper Disassembler (2012 yılında piyasaya sürüldü), özellikle macOS ve iOS hedeflerine odaklanan bir araç olarak bilinir. Hem disassembly hem de decompilation yetenekleri sunar ve zararlı yazılım analizinde, ARM veya x64 gibi mimarilerdeki kodu çözümlemek için sıklıkla kullanılır.

Decompilation ise assembly kodunun daha yüksek seviyeli bir programlama diline (örneğin, C) dönüştürülmesi sürecidir. Bu, özellikle zararlı yazılımın iç mantığını ve algoritmalarını anlamak için kullanışlıdır. Ghidra (2019 yılında NSA tarafından açık kaynak olarak yayınlandı), gelişmiş interaktif düzenleme ve analiz özellikleri sunan bir disassembler ve decompiler aracıdır. Platform bağımsız çalışabilir ve kullanıcı dostu arayüzü sayesinde karmaşık kod yapılarının çözümlenmesinde oldukça etkilidir. Örneğin, bir şifreleme algoritmasının kod detaylarını incelemek veya sıkıştırılmış veri yapılarının nasıl çalıştığını görmek için decompilation kritik öneme sahiptir. Hex-Rays Decompiler, IDA Pro'ya entegre olarak çalışır ve assembly kodunu C benzeri bir formata dönüştürerek analizi daha anlaşılır hale getirir.

Statik analizin seviyeleri aşağıdaki şekilde detaylandırılabilir.

Temel Statik Analiz

Zararlı yazılım örneğinin temel yapısal özelliklerinin analiz edilmesiyle başlar. Bu seviyede hedef, dosyanın genel özelliklerini ve temel bilgilerini hızlıca ortaya çıkarmaktır. Dosya başlığı (header) ve ilgili metadata bilgileri incelenir. Portable Executable (PE) formatındaki bir dosyanın Import Table, Export Table veya Resource Section gibi bölümleri detaylı olarak analiz edilir.

Temel statik analizde aşağıdaki araçlar kullanılabilir.

- PEview: PE dosyalarının header ve section bilgilerini görselleştirir.
- Strings: Dosyada gömülü metinleri (ör. C2 sunucuları veya şifreleme anahtarları) çıkarır.
- Detect It Easy (DIE): Dosyadaki packing veya obfuscation tekniklerini tespit eder.
- Exeinfo PE: Zararlı yazılımda kullanılan packing algoritmalarını hızlıca tanımlar.
- Binwalk: Gömülü dosyaları, sıkıştırılmış içerikleri ve firmware yapısını çözmek için kullanılır.

Orta Seviye Statik Analiz

Assembly seviyesinde kod analizine geçilir. Disassembly işlemiyle kodun daha okunabilir bir hâle getirilmesi sağlanır. Burada amaç, temel işlevleri anlamak ve kodun içindeki kritik noktaları belirlemektir. Packing veya obfuscation kullanılan durumlarda kodun çözülmesi gerekebilir.

Orta seviye statik analizde aşağıdaki araçlar kullanılabilir.

- IDA Pro: Assembly kodunu görsel akış diyagramlarına dönüştürerek analizi kolaylaştırır.
- Ghidra: Disassembly ve decompilation işlemlerini interaktif düzenleme özellikleriyle birleştirir.
- Radare2: Geniş betikleme desteği ile büyük ölçekli analizlere olanak sağlar.
- OllyDbg: Kod akışını incelemek ve basit disassembly işlemleri için kullanılan bir debugger.

İleri Düzey Statik Analiz

Zararlı yazılımın tüm algoritmalarını ve mantığını detaylı bir şekilde anlamayı hedefler. Decompilation işlemiyle kodun yüksek seviyeli programlama dili formatına dönüştürülmesi sağlanır. Ayrıca, şifreleme algoritmalarının çözülmesi ve API çağrılarının detaylı analiz edilmesi gibi işlemler gerçekleştirilir.

İleri statik analizde aşağıdaki araçlar kullanılabilir.

- Ghidra: Decompilation yetenekleri sayesinde kodun işleyişini detaylı olarak anlamayı sağlar.
- Hex-Rays Decompiler: IDA Pro ile entegre çalışan bu araç, assembly kodunu yüksek seviyeli dile dönüştürerek analizi kolaylaştırır.
- x64dbg: 64-bit yazılımlar için ileri seviye analiz ve hata ayıklama yetenekleri sunar.
- YARA: Dosyada belirli tehdit imzalarını tespit etmek için özel kurallar yazılarak uygulanır.
- FlareTools: Özellikle şifreleme algoritmalarını çözmek için araçlar içeren bir koleksiyon.

Statik analiz, bu seviyeler boyunca sistematik bir şekilde ilerler ve kullanılan araçlar zararlı yazılımın türüne, karmaşıklığına ve kullanılan anti-analiz tekniklerine göre seçilir. Bununla birlikte, kullanılan araçların internetten indirilen farklı sürümleri zararlı yazılım içerebilir ve bu durum analisti tehdit altına sokabilir. Bu nedenle, araçlar yalnızca resmi kaynaklardan indirilmeli ve hash değerleri doğrulanmalıdır. Aksi halde, analiz sırasında "ava giderken avlanma" riski ortaya çıkabilir.

Dinamik Analiz

Dinamik analiz, zararlı yazılımın bir sistem üzerinde çalıştırılarak gerçek zamanlı davranışlarının, sistem üzerindeki etkilerinin ve dışa dönük bağlantılarının detaylı bir şekilde gözlemlenmesine dayanan bir analiz yöntemidir. Bu analiz, özellikle zararlı yazılımın şüpheli veya zararlı aktivitelerini anlamak ve etkisiz hâle getirmek için kritik öneme sahiptir. Genellikle iki seviyede uygulanır.

Temel Dinamik Analiz

Sandbox (kum havuzu) ortamında gerçekleştirilir. Sandbox ortamları, zararlı yazılımın izole bir ortamda otomatik olarak çalıştırılmasını ve davranışlarının gözlemlenmesini sağlar. Sandbox'lar genellikle zararlı yazılım analizinde ilk seviye değerlendirme aracı olarak kullanılır ve bu seviyede zararlı yazılımın genel özellikleri hızlıca ortaya çıkarılır. Kum havuzları, şüpheli dosyaların güvenli bir ortamda çalıştırılması için emülatör tabanlı veya sanallaştırma tabanlı mimariler kullanır. Örneğin, Hybrid Analysis veya Any.Run gibi çevrimiçi sandbox araçları, zararlı yazılımın dosya etkilerini, ağ trafiğini ve API çağrılarını analiz eder. Ancak, zararlı yazılımın anti-sandbox teknikleri kullanarak tespit mekanizmalarını atlatma riski bulunduğundan, daha detaylı analizler için ileri seviyedeki yöntemlere başvurulması gerekebilir.

İleri Düzey Dinamik Analiz

Sanal makinelerde manuel analiz araçlarıyla gerçekleştirilir. Bu seviyede, zararlı yazılımın etkilerini daha detaylı incelemek için özel araçlar kullanılır.

Process analizi, Windows ortamında dosya, kayıt defteri ve işlem etkinliklerini izlemek için kullanılan bir yöntemdir. Process Monitor, bu analiz için en yaygın araçlardan biridir. Process Monitor, dosya sistemi etkinlikleri, kayıt defteri değişiklikleri ve süreç oluşturma gibi sistem düzeyinde gerçekleşen olayları gerçek zamanlı olarak izler. Zararlı yazılımın hangi dosyaları değiştirdiği, hangi kayıt defteri anahtarlarını kullandığı ve hangi işlemleri başlattığı bu araçla kolayca tespit edilebilir. Örneğin, zararlı yazılımın kendisini sistem başlatma dizinine ekleme girişimleri veya kritik sistem dosyalarını maniple etme davranışları analiz edilebilir. Sysinternals Suite içindeki Process Explorer, çalışan işlemlerin daha detaylı incelenmesini sağlar. Özellikle, bir işlemin yüklü modüllerini, ağ bağlantılarını ve bağlı DLL'lerini görselleştirmek için oldukça etkilidir. Autoruns, zararlı yazılımın kalıcılık sağlama mekanizmalarını analiz etmek için kullanılır. Sistem başlatma sırasında çalışan işlemler, kayıt defteri anahtarları ve hizmetleri detaylı şekilde listeler. Daha ileri düzeyde, CAPE (Config And Payload Extraction) gibi araçlar, zararlı yazılımın süreç içine yerleştirdiği kod enjeksiyonlarını ve davranışlarını analiz eder. Örneğin, zararlı yazılımın bir süreçte gizlenmiş yüklerini tespit etmek ve çıkarma işlemi yapmak için kullanılır. Procdump (Windows), bellek dökümü almak için kullanılır ve zararlı yazılımın bellek üzerindeki yapısını incelemede yardımcıdır. Örneğin, şifreleme anahtarlarını veya gizlenmiş zararlı kodları çıkarmak için kullanılır. Strace ise Linux ortamında çalışan işlemleri sistem çağrıları seviyesinde izler. Zararlı yazılımın hangi dosyaları açtığını, ağ çağrılarını ve kullanılan sistem kaynaklarını analiz etmek için kullanılır. HTop (Linux) sistem kaynaklarını görselleştiren bir araçtır. CPU, bellek ve işlem aktivitelerini izlerken, zararlı yazılımın sistem üzerindeki yükünü belirlemeye yardımcı olur. Plist Explorer, macOS üzerinde çalışan işlemler ve plist dosyalarını analiz eder. Zararlı yazılımın sistem başlangıç öğelerini manipüle edip etmediğini kontrol eder. Bu araçlar, işlem analizi sırasında zararlı yazılımın hangi yöntemlerle sistem üzerinde çalıştığını ve hangi kaynakları manipüle ettiğini detaylı bir şekilde ortaya koyar. İşletim sistemine özel araçların kombinasyonu, analistlerin zararlı yazılımın davranışlarını anlamasını ve etkili karşı önlemler geliştirmesini sağlar.

Ağ trafiği analizi, zararlı yazılımın dış iletişimlerini ve veri transferlerini anlamada kritik bir aşamadır. Wireshark, ağ üzerindeki paketleri gerçek zamanlı olarak yakalayıp analiz etmek için yaygın olarak kullanılan güçlü bir araçtır. Zararlı yazılımın Command-and-Control (C2) sunucularıyla olan bağlantılarını, hangi IP adreslerine veya domain'lere eriştiğini ve transfer edilen verilerin boyutlarını belirlemek için kullanılır. Bunun yanı sıra, Tshark, Wireshark'ın komut satırı sürümü olarak daha otomatikleştirilmiş analizlerde tercih edilir. Özellikle büyük hacimli ağ trafiğini hızlıca işlemek ve script'lerle entegre analizler yapmak için etkilidir. Zeek (eski adıyla Bro), ağ trafiğini protokol seviyesinde analiz eden bir güvenlik aracı olarak öne çıkar. Zeek, ağdaki anormallikleri belirlemek ve zararlı yazılımın kullandığı özel protokolleri anlamak için geniş veri sağlar. Örneğin, zararlı yazılımın TLS şifreli trafik içinde hangi anahtarları kullandığını çözümlemek mümkündür. Suricata, imza tabanlı bir ağ izleme ve saldırı tespit sistemi (IDS) olarak, zararlı yazılımın ağ üzerindeki davranışlarını gerçek zamanlı olarak analiz eder. Suricata, aynı zamanda ağdaki veri akışını log'layarak anormallikleri uzun vadede incelemek için detaylı bilgiler sağlar. Arkime (eski adıyla Moloch), büyük ölçekli ağ yakalama ve depolama için kullanılan bir platformdur. Zararlı yazılım analizinde ağ trafiğinin geçmişine dair geniş bir perspektif sunar, böylece analiz sırasında eksik kalan veriler tamamlanabilir. Bu araçların kombinasyonu, ağ tabanlı zararlı yazılımların iletişim protokollerini, veri transfer yöntemlerini ve kullandıkları altyapıyı daha detaylı bir şekilde anlamaya olanak tanır.

Bellek analizi, zararlı yazılımın çalıştığı ortamın dinamik verilerini anlamak için kritik bir aşamadır. Bu süreçte kullanılan araçlardan Volatility, açık kaynaklı bir araç olarak öne çıkar. Zararlı yazılımın bellekte çalışan modüllerini, kod enjeksiyonlarını ve şüpheli işlemleri analiz etmede oldukça güçlüdür. Bellek dökümünden şifreleme anahtarları, sistem oturumları ve zararlı yazılımın gerçekleştirdiği komutlar gibi önemli bilgiler elde edilebilir. Bunun

yanı sıra, Rekall, Volatility'nin bir forku olarak geliştirilmiştir ve platformlar arası uyumluluk sunar. Analistlerin bellek dökümündeki anormallikleri belirlemesini ve süreçleri detaylandırmasını sağlar. Örneğin, bir zararlı yazılımın bellek üzerindeki şifrelenmiş veri depolarını çözmek için sıklıkla tercih edilir. MemProcFS, bellek dökümünü bir dosya sistemi olarak göstermesiyle dikkat çeker ve özellikle hızlı inceleme gerektiren durumlarda analistler için oldukça kullanışlıdır. Zararlı yazılımın bellek üzerindeki değişikliklerini gerçek zamanlı incelemek için güçlü bir araçtır. Ek olarak, LiME (Linux Memory Extractor), canlı Linux sistemlerinden bellek dökümü almak için kullanılırken, FTK Imager, Windows tabanlı sistemlerde hem bellek hem de disk dökümü oluşturmak için etkili bir alternatif sunar. Bu araçlar, bellek tabanlı analizlerde zararlı yazılımın çalışma ortamını ayrıntılı olarak anlamak, şüpheli işlemleri izlemek ve kod manipülasyonlarını analiz etmek için bir arada kullanılabilir. Bu araçlar, bellek tabanlı analizlerde zararlı yazılımın çalıştığı ortamın detaylarını ortaya çıkarmak, şüpheli işlemleri izlemek ve kod manipülasyonlarını incelemek için kullanılır.

Disk üzerinde yazılan veriyi takip etmek ve raporlamak, zararlı yazılımın etkilerini anlamak için önemli bir analiz adımıdır. Bu süreçte kullanılan araçlar, zararlı yazılımın sistem üzerindeki izlerini detaylı bir şekilde tespit etmeye olanak tanır. Sysinternals Disk Usage, diskte hangi dosyaların yazıldığını ve bu dosyaların boyutlarını analiz eder. Zararlı yazılımın hangi dosyaları oluşturduğunu ve bunların ne kadar disk alanı tükettiğini hızlıca raporlar. Auditpol, Windows güvenlik denetim politikalarını yapılandırmak ve disk erişimlerini denetlemek için kullanılır. Zararlı yazılımın belirli dosyalara erişim girişimlerini ve bu dosyalardaki değişiklikleri detaylı bir şekilde kaydeder. inotify-tools, Linux ortamında, belirli dizinlerdeki dosya değişikliklerini gerçek zamanlı olarak izlemek için etkili bir araçtır. Özellikle zararlı yazılımın kritik sistem dosyalarını maniple edip etmediğini tespit etmek için kullanılır. Fslogger macOS için bir dosya sistemi monitörüdür. Filebeat log dosyalarını toplar ve analiz için merkezi bir sisteme yönlendirir. DTrace macOS ve Solaris için disk I/O işlemlerini ve dosya sistemi etkinliklerini izler. Tripwire bir dosya bütünlüğü izleme (File Integrity Monitoring) aracıdır. Disk üzerindeki dosya değişikliklerini kontrol eder ve yetkisiz değişiklikleri raporlar. Bu araçlar, zararlı yazılımın disk üzerinde bıraktığı izlerin, manipülasyonların ve oluşturduğu dosyaların kapsamlı bir şekilde analiz edilmesini sağlar. İşletim sistemine özgü araçlar ve platform bağımsız çözümler, analiz sürecini daha etkili hâle getirir.

Analist, bunlar gibi araçları kullanarak zararlı yazılımın etkilediği dosyaları, bellek manipülasyonlarını ve sistem kaynakları üzerindeki değişiklikleri detaylı bir şekilde takip eder. Snapshot özellikleri, analiz sırasında sistemi geri yüklemek ve zararlı yazılımın farklı davranışlarını test etmek için kritik bir avantaj sağlar. Bu yöntem, zararlı yazılımın karmaşık tekniklerini, örneğin anti-debugging mekanizmalarını veya kod enjeksiyon yöntemlerini derinlemesine anlamak için kullanılır.

Online Analiz Araçları

Online analiz araçları da yaygın olarak tercih edilmektedir. Hybrid Analysis, VirusTotal ve Any.Run gibi platformlar, zararlı yazılım örneklerini analiz etmek ve elde edilen sonuçları paylaşmak için etkili bir yöntem sunar. Bu araçlar, özellikle hızlı geribildirim ve geniş veritabanı sayesinde, tehditlerin özelliklerini ve yayılma yöntemlerini anlamada önemli rol oynar. Ancak bu tür araçları kullanırken dikkat edilmesi gereken bazı teknik ve güvenlik önlemleri vardır.

İlk olarak, analiz edilecek dosyaların hash değerlerinin kontrol edeilrek işlem yapılması önerilir. Hash doğrulaması, dosyanın bütünlüğünü garanti altına alır ve dosyanın beklenmedik bir şekilde değiştirilip değiştirilmediğini anlamayı sağlar. Ayrıca, bir dosyanın hash değeriyle sorgulanması, dosyayı fiziksel olarak yüklemeden mevcut bilgilerin elde edilmesini sağlar. Bu, zararlı yazılım örneğini analiz platformlarına yüklemeden önce potansiyel güvenlik risklerini en aza indirmek açısından kritik öneme sahiptir.

İkinci olarak, analiz edilmesi gereken dosyaların doğrudan bu platformlara yüklenmesi, gizlilik ve güvenlik açısından ciddi riskler oluşturabilir. Örneğin, zararlı yazılımın bir örneği bu platformların veritabanında üçüncü taraflar tarafından erişilebilir hâlde saklanabilir. Bu durum, kurumsal bilgilerin veya hassas verilerin istenmeden ifşa edilmesine yol açabilir. Ayrıca, saldırganlar bu platformları zararlı yazılımlarını test etmek, bunların tespit edilme olasılıklarını azaltmak ve güvenlik mekanizmalarını aşmak için kötüye kullanabilirler.

Bu nedenle, dosyaları önce güvenli bir ortamda analiz etmek ve gerekirse offline yöntemleri tercih etmek daha güvenlidir. Bunun yanı sıra, örneklerin yüklenmeden önce hash değeriyle kontrol edilmesi, hızlı bir ön analiz yapılması ve dosyanın daha önce analiz edilip edilmediğinin doğrulanması hem zamandan tasarruf hem de bilgi güvenliği açısından fayda sağlar.

Bunun yerine, offline analiz yöntemlerini tercih etmek veya dosya üzerinde güvenli bir ortamda ön analiz yapmak daha güvenlidir. Ayrıca, bu araçların, saldırganlar tarafından zararlı yazılımlarını test etmek ve algılama mekanizmalarını aşmak için de kullanılabileceği unutulmamalıdır. Bu nedenle, online analiz araçlarını kullanmadan önce, örneklerin güvenli bir laboratuvar ortamında test edilmesi ve analiz sonuçlarının gizliliğini sağlamak için gerekli önlemlerin alınması kritik önem taşır.

Anti-Analiz Teknikleri

Zararlı yazılımlar, analiz sürecini zorlaştırmak için çeşitli anti-analiz teknikleri kullanabilir. Bu teknikler zararlı yazılımın tespit ve analiz edilmesini önlemek veya zorlaştırmak amacıyla geliştirilir. Öne çıkan anti-analiz teknikleri şunlardır:

Kod Manipülasyonu Teknikleri

Obfuscation (Bulanıklaştırma), kodun karmaşıklaştırılarak okunmasını ve anlaşılmasını zorlaştırır. Örneğin, değişken isimlerinin anlamsız hâle getirilmesi veya kodun akışının karıştırılması. Bu teknik, analiz sırasında kullanılan disassembler ve decompiler araçlarının çıktılarında kafa karışıklığı yaratmayı hedefler. Değişken adlarının anlamsız hâle getirilmesi, döngü yapılarının değiştirilmesi ve gereksiz kod parçalarının eklenmesi gibi yöntemler içerir. Obfuscation tekniklerinin etkisini azaltmak için kodun manuel olarak analiz edilmesi veya özel script'ler yazılarak karmaşıklaştırılmış bölümlerin açığa çıkarılması gerekebilir. Bu süreçte kullanılan araçlar arasında de4dot, UnpackMe, ve Ghidra gibi platformlar yer alır. Bu araçlar, bulanıklaştırılmış kod yapılarının çözülmesini kolaylaştırır. Ayrıca, kontrol akışı düzeltme (Control Flow Recovery) ve kod optimizasyon analizi, obfuscation tekniklerinin etkisini azaltmak için yaygın olarak kullanılan ileri analiz yöntemleridir.

Packing, kodun sıkıştırılarak veya şifrelenerek çalıştırılmadan önce anlaşılmasını engeller. Bu teknik, özellikle runtime unpacking ile zararlı yazılımın yalnızca çalıştırıldığında çözülmesini sağlar. Packing yöntemleri, analiz araçlarının kodun içeriğine doğrudan erişimini engelleyerek, zararlı yazılımın işlevselliğini gizler. Farklı işletim sistemlerinde packing'i aşmak için özel araçlar ve yöntemler kullanılır. PEiD veya Detect It Easy gibi araçlar, sıkıştırma algoritmalarını belirlemek için kullanılır. Bu süreçte, sıkıştırılmış bir dosyanın unpacking işleminden sonra IDA Pro veya x64dbg gibi araçlarla analiz edilmesi gerekebilir. UPX (Ultimate Packer for Executables) gibi yaygın araçlar tarafından sıkıştırılmış dosyalar için upx -d komutu ile çözümleme yapılabilir. Ayrıca Radare2, sıkıştırılmış ELF dosyalarını analiz etmek için kullanılabilir. Mach-O dosyaları için Hopper Disassembler veya Ghidra gibi araçlarla packing tespit edilip, manuel olarak çözümleme yapılabilir. İleri seviyede, packing'i aşmak için runtime analiz tercih edilir. Zararlı yazılım izole bir ortamda çalıştırılarak bellek dökümü alınır ve sıkıştırılmış kodun çözülmüş hâli analiz edilir. Bu yöntem, şifrelenmiş veya polimorfik kodları çözmek için en etkili tekniklerden biridir.

Paketli bir dosyanın çalışabilmesi için runtime sırasında açılması gerekmektedir. Packing, zararlı yazılımın kodunu çalıştırılmadan önce şifreli veya sıkıştırılmış bir biçimde tutarak analiz araçlarının içeriğe erişimini engeller. Bu süreç, çalıştırıldığında dosyanın kendini bellek üzerinde çözmesiyle tamamlanır. Paketleme işlemi genellikle zararlı yazılımın orijinal kodunu bir "stub" (küçük bir yükleyici program) içine sıkıştırır veya şifreler. Stub, çalışma zamanında kodu çözmekten ve çalıştırmaktan sorumludur. Örneğin, bir PE dosyası üzerinde packing, dosyanın bölümlerini gizler ve normal disassembler araçlarının kodu analiz etmesini zorlaştırır. İleri seviyede, bellek dökümü (memory dump) alınarak kodun tam açık hâli elde edilebilir. Runtime sırasında dinamik analiz araçları ile kodun işleyişi izlenir, ardından açılan kod statik analiz araçları ile detaylandırılır. Bu süreç, packing tekniklerinin etkisiz hâle getirilmesi için kritik öneme sahiptir.

Dinamik Analiz Karşıtı Teknikler

Anti-Sandbox tekniğinde zararlı yazılım, kendisinin bir sandbox ortamında çalıştırıldığını algılayabilir ve zararlı aktivitelerini devre dışı bırakabilir. Bu teknikler, zararlı yazılımın sandbox içinde analiz edilmesini önlemek veya yanlış sonuçlar üretmek için tasarlanmıştır. Örneğin, zararlı yazılım CPU özelliklerini kontrol ederek fiziksel bir makinede mi yoksa sanal bir ortamda mı çalıştığını belirleyebilir. CPU hızının düşük olması veya donanım seviyesindeki belirli özelliklerin eksikliği, sandbox varlığına işaret edebilir. Ayrıca, dosya sistemindeki belirli imzalar (örneğin, "vmware.log" veya "VirtualBox Guest Additions" gibi sanal makineye özgü dosyalar) taranarak analiz ortamı tespit edilebilir. Zararlı yazılım, bu tür imzaları bulduğunda zararlı aktivitelerini devre dışı bırakabilir veya tamamen pasif bir hâle geçebilir. Sandbox'ların genellikle sınırlı veya yerel bir ağ yapılandırmasına sahip olmasını kullanarak, sahte ağ etkinliklerini tespit eder. Örneğin, zararlı yazılım, belirli bir DNS sunucusuna bağlanamıyorsa sanal bir ortamda olduğunu anlayabilir. Windows ortamında, sandbox yazılımına özgü kayıt defteri anahtarlarını arar. Örneğin, "HKEY_LOCAL_MACHINE\Software\VMware, Inc." anahtarlarının varlığı sandbox ortamını işaret edebilir. Sandbox'lar genellikle düşük donanım kaynağına sahiptir ve işlemler daha yavaş çalışır. Zararlı yazılım, belirli işlemleri zamanlayarak bu tür gecikmeleri ölçebilir ve analiz ortamında olduğunu anlayabilir. Fare hareketleri, klavye girişleri gibi kullanıcı etkileşimlerini algılayarak, otomatik bir analiz ortamında çalıştığını tespit edebilir. Kullanıcı etkinliği algılanmazsa zararlı aktivitelerini devre dışı bırakabilir.

Anti-debug teknikleri, zararlı yazılımın analiz edilmesini zorlaştırmak için tasarlanmış koruma mekanizmalarıdır. Bu teknikler, yazılımın bir hata ayıklama (debugging) ortamında çalıştığını tespit ederek çalışmasını durdurabilir, yanıltıcı sonuçlar üretebilir veya zararlı aktivitelerini gizleyebilir. Zararlı yazılım, Windows platformunda IsDebuggerPresent veya CheckRemoteDebuggerPresent gibi API çağrılarını kullanarak bir debugger'ın çalışıp çalışmadığını kontrol eder. Benzer şekilde, Linux ortamlarında ptracesistem çağrısı ile hata ayıklama işlemleri algılanabilir. Bazı zararlılar, hata yakalama mekanizmalarını kullanarak debugger'ın tepkisini test eder. Örneğin, yazılım, bir istisna (exception) oluşturarak debugger'ın bu istisnayı nasıl işlediğini gözlemler. Yazılımın belirli kod parçalarını çalıştırmak için geçen süre ölçülür. Debugger ortamlarında bu işlemler genellikle daha yavaş gerçekleştiğinden, zararlı yazılım gecikmeleri algılayarak analiz edildiğini anlayabilir. Yazılım, belirli bir bellek adresindeki kodun değiştirilip değiştirilmediğini kontrol ederek yazılıma eklenmiş bir breakpoint (durma noktası) olup olmadığını tespit edebilir. Zararlı yazılım, kodunu şifreli veya

karıştırılmış bir biçimde tutarak debugger tarafından analiz edilmesini zorlaştırır. Çalışma zamanında yalnızca belirli koşullar altında kodun şifresi çözülür. Bazı durumlarda zararlı yazılım, bilinen hata ayıklama araçlarının (örneğin, OllyDbg, x64dbg) çalıştığını algılar ve bu araçları devre dışı bırakmak için özel kodlar içerir. Anti-debug tekniklerini aşmak için analistler, debugger araçlarının tespit edilmesini önleyecek yöntemler uygular. Örneğin, debugger tespit mekanizmalarını atlatan yamalar geliştirilir veya dinamik analiz sırasında zararlı yazılımın davranışlarını izlemek için sanallaştırma tekniklerinden faydalanılır. Ayrıca, bellek döküm araçları ve simülasyon tabanlı ortamlar kullanılarak, anti-debug tekniklerinin etkileri analiz edilebilir ve bertaraf edilebilir.

Akış Kontrol Manipülasyonu

Control Flow Flattening yani kodun akış yapısını değiştirme, döngüler ve çağrı zincirleri gibi yapıları karmaşıklaştırma tekniğidir. Bu yöntem, programın kontrol akışını düzleştirerek, kodu analiz eden bir kişinin veya otomatik araçların, kodun asıl yürütme sırasını anlamasını zorlaştırır. Örneğin, döngüler yerine bir dizi koşullu ifade eklenebilir veya bir "switch-case" yapısı, kontrol değiştiricilerle karmaşık hâle getirilebilir. Zararlı yazılım, bu teknikle, disassembler ve decompiler araçlarının işini zorlaştırır. Bu tekniklere karşı analiz yaparken, statik analiz araçlarıyla kodun akışını manuel olarak yeniden yapılandırmak gerekebilir. Dinamik analiz sırasında, debugger kullanılarak kodun çalışma sırasını izlemek ve "breakpoint" ile kritik noktaları analiz etmek etkili bir yöntemdir. Ayrıca, Ghidra ve IDA Pro gibi araçların "control flow recovery" hususuna yardımcı olan özellikleri, bu tür manipülasyonları daha kolay çözmeye olanak tanır.

Code Injection

Zararlı kodun başka bir prosese veya bellek alanına enjekte edilmesiyle zararlı aktivitelerin gizlenmesidir. Bu teknik, zararlı yazılımın algılanmasını ve analiz edilmesini önlemek için kullanılır. En yaygın yöntemlerden bazıları şöyledir. DLL Injection, zararlı yazılım, hedef bir süreçte belirli bir DLL'nin yüklenmesini zorlar. Windows sistemlerinde CreateRemoteThread ve LoadLibrary API çağrıları kullanılarak bu işlem gerçekleştirilir. Bu yöntem, zararlı aktivitelerin bir sistem sürecinin içinde saklanmasını sağlar. Process Hollowing ise yasal bir süreç başlatıldıktan sonra, o sürecin yürütülebilir içeriği zararlı kod ile değiştirilir. Örneğin, CreateProcess API'si ile bir süreç başlatılır, ardından ZwUnmapViewOfSection çağrısıyla bellekteki yasal kod boşaltılarak zararlı kod enjekte edilir. APC Injection (Asynchronous Procedure Call) yönteminde zararlı kod, bir sürecin zamanlaması maniple edilerek çalıştırılır. Bu yöntem, QueueUserAPC API'si ile uygulanır ve kodun hedef sürecin yürütme bağlamında çalıştırılmasını sağlar. Code injection tekniklerini tespit etmek için gelişmiş bellek analizi araçları kullanılır. Örneğin Volatility veya Rekall, bellek dökümlerini analiz ederek süreçler arası anormallikleri ortaya çıkarabilir. Process Explorer, bir sürecin yüklü modüllerini kontrol ederek beklenmeyen DLL'leri tespit edebilir. YARA kuralları, zararlı imzaları tespit etmek için özelleştirilmiş kurallar yazarak bellek ve dosya taraması yapılmasını sağlar.

Şifreleme ve Gizleme

Polimorfizm ve metamorfizm, zararlı yazılımın her çalıştırıldığında farklı bir form almasını sağlayarak imza tabanlı tespiti engelleyen ileri düzey anti-analiz tekniklerdir.

Polimorfizm, zararlı yazılım, her çalıştırıldığında kodunun farklı bir şekilde yeniden şifrelenmesiyle kendisini değiştirir. Ancak, bu süreçte zararlı yazılımın ana mantığı ve işlevi değişmez. Polimorfik zararlı yazılımlar genellikle bir şifreleme algoritması ve bir "decryption stub" (şifre çözme mekanizması) içerir. Zararlı yazılım çalıştırıldığında, şifreli bölüm çözülerek asıl zararlı kod yürütülür. Bu teknik, imza tabanlı antivirüs sistemlerini atlatmak için kullanılır.

Metamorfizm, polimorfizme benzer şekilde, zararlı yazılımın her yeniden derlenmesinde veya çalıştırılmasında kendisini yeniden yazması sağlanır. Ancak, metamorfik yazılımlar kod yapısını tamamen değiştirir, bu da yalnızca imza tabanlı değil, davranış tabanlı tespit sistemlerini de zorlaştırır. Metamorfik zararlı yazılımlar genellikle kodun yapısını bozarak, talimat sırasını değiştirerek veya gereksiz talimatlar ekleyerek çalışır. Örneğin, bir döngüyü koşullu ifadelerle değiştirebilir veya aynı işlevi gerçekleştiren farklı talimatlar ekleyebilir.

Bu teknikleri tespit etmek ve analiz etmek için gelişmiş araçlar ve yöntemler gereklidir. Zararlı yazılımın çalışma zamanındaki davranışlarını izlemek için sandbox veya debugger kullanılır. Bu süreçte, zararlı yazılımın bellek üzerindeki çözülmüş hâlini yakalamak önemlidir. Örneğin, x64dbg veya OllyDbg gibi debugger araçlarıyla decryption stub'ların nasıl çalıştığı izlenebilir. Disassembly araçları (ör. IDA Pro, Ghidra) ve imza eşleştirme araçları (ör. YARA) kullanılarak kodun değişen ve sabit kısımları analiz edilir. Polimorfik zararlı yazılımlar için özel olarak tasarlanmış UnpackMe, PEiD gibi araçlar kullanılarak şifrelenmiş bölümler çözülebilir. Metamorfik yazılımlarda ise bellek dökümü alarak kodun orijinal yapısını ortaya çıkarmak etkili bir yöntemdir. Zararlı yazılımın ağ trafiği, dosya manipülasyonları ve sistem çağrıları gibi dışa dönük davranışlarını analiz eden Zeek veya Suricata gibi araçlar kullanılır. Polimorfik ve metamorfik yazılımların değiştiremeyeceği sistem etkileri bu şekilde tespit edilebilir. Bu tür anti-analiz tekniklerine karşı statik ve dinamik analizin güçlü bir kombinasyonu gereklidir. Özellikle, kodun çözülmüş hâlini bellekte yakalamak ve bu yapıları yeniden oluşturmak, zararlı yazılımın etkili bir şekilde incelenmesini sağlar. Bu tür anti-analiz tekniklerine karşı manuel unpacking, sanal makine imzalarının kaldırılması, breakpoint yönetimi, özel analiz araçları ve davranış

tabanlı analiz gibi çözümler uygulanabilir. Analistlerin bu tekniklere hazırlıklı olması, zararlı yazılımın etkili bir şekilde incelenmesi için kritik öneme sahiptir.

Custom encryption, zararlı yazılımın iletişim ve yapılandırma verilerini özel bir algoritma ile şifreleyerek analiz edilmesini zorlaştırmasıdır. Algoritma türünün tespitinin kolay olmamasıyla birlikte özelleştirilmiş şifreleme algoritmaların zafiyetlerinin açık kaynak algoritmalardan daha yüksek olması genel olarak rastlanan bir durumdur.

Dosya Formatları ve İşletim Sistemlerine Göre Analiz Zorlukları

Zararlı yazılım analistleri, farklı işletim sistemleri ve dosya tiplerini analiz ederken çeşitli zorluklarla karşılaşır. Bu dosyaların her biri, analiz sırasında farklı metodolojiler ve araçlar gerektirir.

Yürütülebilir Dosyalar

Windows PE (Portable Executable) dosya tipinde .exe ve .dll formatları sıkça hedef alınır. Analiz sırasında, import/export tabloları, kod bölümleri, gömülü kaynaklar ve özellikle entegre edilen sertifikalar detaylı şekilde incelenir. Debugging sürecinde, PE dosyalarındaki "Section Header" ve "Optional Header" gibi meta bilgiler, dosyanın yapısal bütünlüğünü doğrulamak için kritik öneme sahiptir. Ayrıca, packer kullanımı belirtileri ve anti-debugging teknikleri tespit edilmelidir. Farklı Windows sürümleri (örneğin, Windows 7, 10 ve Server versiyonları) arasında uyumluluk analizleri yapılmalıdır, çünkü API çağrılarındaki farklılıklar zararlı yazılımın davranışını etkileyebilir. 32-bit ve 64-bit yapılar arasında analiz yapılırken, özellikle "WOW64" (Windows-on-Windows 64-bit) uyumluluk katmanı üzerinden çalışan uygulamalar detaylı şekilde incelenmelidir. Bu süreçlerde IDA Pro, PEiD, CFF Explorer ve x64dbg gibi araçlar yaygın olarak kullanılır. 64-bit PE dosyalarında, genişletilmiş adresleme ve farklı kernel yapılarına dikkat edilmelidir, çünkü bunlar analizi zorlaştırabilir.

Linux ELF (Executable and Linkable Format) dosya tipinde dinamik bağlayıcıların analiz edilmesi, ELF dosyasının ".text", ".data," ve ".bss" gibi bölümlerinin yapısının incelenmesi ve "dynamic linker" işlemlerinin detaylı şekilde analiz edilmesi kritik önemdedir. Farklı Linux dağıtımları (örneğin, Ubuntu, CentOS, Arch Linux ve Alpine) arasındaki farklılıklar da analiz süreçlerinde etkili olabilir. Örneğin, CentOS ve Red Hat Enterprise Linux gibi RPM tabanlı sistemlerde kullanılan SELinux politikaları, zararlı yazılımın çalıştırılmasını kısıtlayabilir, ancak bu aynı zamanda analizin karmaşıklığını artırabilir. Ubuntu ve Debian gibi DEB tabanlı sistemlerde, "AppArmor" güvenlik politikaları ve paket yöneticisi yapıları analiz sırasında dikkate alınmalıdır. Hafif sistemler (örneğin, Alpine Linux), farklı C kütüphaneleri (musl libc gibi) kullanarak analizde beklenmedik davranışlara neden olabilir. Bu nedenle, dağıtıma özgü araçlar ve yapılandırmaların detaylı incelenmesi gereklidir. Analiz süreçlerinde GDB, Radare2, Binwalk, ve Strace gibi araçlar önemli roller üstlenir. Özellikle "PLT/GOT" (Procedure Linkage Table/Global Offset Table) gibi yapılar, zararlı kodun yüklenme mekanizmasını anlamak için derinlemesine analiz edilmelidir. Anti-debugging tekniklerinin tespiti ve "ptrace" tabanlı sistem çağrılarının izlenmesi için dinamik hata ayıklama araçları kullanılabilir. Analiz süreçlerinde GDB, Radare2, Binwalk, ve Strace gibi araçlar önemli roller üstlenir.

Mach-O (Mach Object) dosyaları, macOS ve iOS sistemlerinde kullanılan yürütülebilir dosya formatıdır. Dosya yapısı, Header, Load Commands, ve Segments/Sections olmak üzere üç ana bileşenden oluşur. Header, dosyanın temel özelliklerini (örneğin, 32-bit mi yoksa 64-bit mi olduğu) belirler. Load Commands, dosyanın belleğe yüklenmesi sırasında kullanılan talimatları içerir ve segmentler/seksiyonlar ise kod ve veri gibi yürütülebilir içeriği barındırır. Özellikle __TEXT, __DATA, ve __LINKEDIT segmentleri, zararlı kodların analizinde önemli ipuçları sağlar. macOS'un farklı sürümleri (örneğin, macOS Mojave, Catalina, ve Big Sur) arasında API değişiklikleri ve sistem güvenlik politikaları, zararlı yazılımların çalıştırılabilirliğini ve analizin karmaşıklığını etkileyebilir. Örneğin, modern macOS sürümleri "System Integrity Protection" (SIP) gibi koruma mekanizmaları ile analiz süreçlerini sınırlandırabilir. iOS sistemlerinde ise "App Sandbox" ve "Code Signing" politikaları, zararlı uygulamaların yürütülmesini engeller ancak analizde de zorluklar yaratır. Bu platformlar arasında, kullanılan araçlarda bazı farklılıklar görülebilir. Örneğin, otool, lldb, ve MachOView, macOS üzerinde statik analiz için kullanılırken, iOS analizinde Frida, IDA Pro, ve Hopper Disassembler gibi araçlar daha sık tercih edilir. Ayrıca, entegre sertifikaların doğrulanması ve kod imzalarının geçerliliği kontrol edilmelidir. Özellikle entegre sertifikalar, zararlı yazılımların sahte imzalarla meşru gibi görünmesini sağlayabilir.

Mobil Platformlar

Android APK (Android Application Package -Android Uygulama Paketi) ve DEX (Dalvik Executable -Dalvik Çalıştırılabilir Dosyası) şifrelenmiş içeriklerin çözülmesi ve anti-analiz tekniklerinin tespiti, zararlı yazılım analizi sırasında önemli zorluklar yaratır. Örneğin, APK dosyalarında gömülü şifreleme anahtarlarının tespit edilmesi zordur. APK dosyalarının içeriği, genellikle AES veya XOR gibi algoritmalarla şifrelenmiştir. Anahtarların tersine mühendislik yoluyla çıkarılması veya bellek dökümünden alınması gerekebilir.

Manifest dosyası (AndroidManifest.xml) ve kaynak dosyalarının (resources.arsc) doğru şekilde ayrıştırılması gereklidir. AndroidManifest.xml dosyası, uygulamanın izinleri, servisleri ve başlangıç noktalarını tanımladığı

için kritik öneme sahiptir. Zararlı yazılımlar genellikle bu dosyada gizli servisler veya kötü amaçlı ağ bağlantıları tanımlar. resources.arsc dosyası ise uygulama içindeki kaynakların (örneğin, metinler ve görseller) organize edildiği bir veritabanı olarak kullanılır ve gizli komutlar veya veri göstergeleri barındırabilir.

Bazı APK'lar, kaynak dosyalarını şifreler veya özel sıkıştırma algoritmaları kullanır, bu da kaynakların okunmasını zorlaştırır. Örneğin, şifreleme yöntemleri arasında AES (Advanced Encryption Standard -Gelişmiş Şifreleme Standardı) veya RSA (Rivest-Shamir-Adleman) gibi algoritmalar bulunabilir. Bu yöntemler, anahtar yönetimi ile birleşerek kaynak dosyalarına erişimi engeller. Ayrıca, sıkıştırma işlemleri sırasında APK dosyasının bölümleri karmaşık veri yapılarına dönüştürülerek tersine mühendislik zorluğu artırılır. Analiz sırasında, şifrelenmiş içeriklerin çözümü için bellek dökümünden şifreleme anahtarlarının çıkarılması veya runtime analiz yöntemleri uygulanabilir. Binwalk, APKTool, ve özel çözümleme script'leri bu tür zorlukların üstesinden gelmekte sıkça kullanılır.

Kod gizleme teknikleri kapsamında resources.arsc içindeki veriler, genellikle Base64 veya XOR gibi algoritmalarla kodlanabilir, bu da statik analizle çözümlemeyi zorlaştırır. Bu teknikler zararlı yazılımların analiz edilmesini zorlaştırmak için kullanılan yöntemlerdir. Base64, verileri ASCII formatında kodlayarak gizlerken, XOR şifreleme basit de olsa farklı uygulama teknikleri ile daha da karmaşık bir mantıkla her bitin üzerine birden fazla anahtar uygulayabilir. Statik analiz araçları ile kodun içeriği okunabilir hale getirilmeye çalışılır. Özellikle CyberChef veya özel yazılmış Python script'leri ile Base64 kodlarını çözmek mümkündür. XOR kodlamasını çözmek için şifreleme anahtarının tespit edilmesi gerekir. Bu işlem, bellek dökümü çıkarılarak veya kodun yürütülmesi sırasında anahtarın kaynağının izlenmesiyle yapılabilir. Dinamik analiz sırasında verilerin çözülmüş halini gerçek zamanlı olarak incelemek için Frida veya Xposed gibi araçlar kullanılabilir. Bu yöntemlerin doğru bir şekilde uygulanması, gizlenmiş kodların ortaya çıkarılmasını ve analiz edilmesini kolaylaştırır.

Metin gizleme (String Obfuscation) olarak incelediğimizde, DEX formatındaki kodlarda genellikle metin gizleme (String Obfuscation) ve kontrol akışı düzleştirme (Control Flow Flattening) gibi tekniklerle karşılaşılır. Metin gizleme, Base64 kodlaması veya özel şifreleme yöntemleri kullanılarak metinlerin okunabilirliğini engeller ve bu kodlar, dinamik olarak çalıştırılmadan önce statik analizle çözümlenmelidir. Kontrol akışı düzleştirme ise kod akışının mantığını gizleyerek fonksiyonların takibini zorlaştırır ve bu tür kodlar için disassembler veya decompiler araçları ile derinlemesine analiz gereklidir. Statik analiz için APK dosyaları üzerinde JADX (Java Decompiler) ve APKTool (APK Çözücü) kullanılarak kaynak kodu yeniden oluşturulabilir, bu araçlarla kodun mimarisi ve veri akışı analiz edilebilir. Dinamik analiz sırasında, kod çalıştırılarak davranışı izlenebilir ve bu süreçte Frida, Xposed Framework veya Magisk gibi araçlarla zararlı davranışlar tespit edilebilir. Ayrıca, ileri araçlar arasında Smali ve Bytecode Viewer yer alır; bu araçlar ile DEX kodunun düşük seviyede işlenmesi sağlanır ve bellek dökümleriyle şifrelenmiş anahtarların veya string'lerin gerçek zamanlı olarak çözülmesi mümkündür. iOS sistemlerinde IPA dosyaları üzerinde imzalama ve entegre sertifikaların sahte olup olmadığının tespiti önemlidir. Disassembly ve decompilation süreçlerinde, Android için Smali kodları incelenmelidir, iOS için ise Frida veya Cycript gibi araçlar kullanılabilir.

IoT ve Firmware Analizi

SquashFS (Squash File System) veya CramFS (Compressed ROM File System) dosya sistemlerinin şifre çözümü kritik önem taşır, çünkü bu tür dosya sistemleri IoT cihazlarının firmware'lerinde sıkça kullanılır ve sıkıştırılmış yapıları nedeniyle içeriklerin analiz edilmesini zorlaştırır. IoT cihazlarında MQTT (Message Queuing Telemetry Transport) ve CoAP (Constrained Application Protocol) gibi protokolleri analiz etmek için Wireshark veya özel sniffer araçları tercih edilir. MQTT, cihazlar arası haberleşmede düşük bant genişliği gereksinimi sunarken CoAP, RESTful iletişim için optimize edilmiştir; ancak her iki protokol de "man-in-the-middle" saldırılarına açık olabilecek yapılar barındırır. JTAG (Joint Test Action Group) ve SWD (Serial Wire Debug) gibi yöntemler kullanılarak cihazın gerçek zamanlı durumu incelenebilir. Bu yöntemlerle donanım seviyesinde hata ayıklama yapılabilir ve cihazın bellek, CPU ve diğer kaynaklarına erişim sağlanarak zararlı kodların davranışları derinlemesine incelenebilir.

IoT cihazları, akıllı ev uygulamalarında son kullanıcıya daha yakın çalıştıkları için zararlı yazılım analizinde özel zorluklar sunar. Bu cihazlarda kullanılan protokoller ve donanımlar, analiz süreçlerini doğrudan etkileyebilir. Örneğin, Apple HomeKit, Zigbee tabanlı Philips Hue, Matter (Apple, Google, Amazon ve diğerleri tarafından desteklenen yeni bir standart) ve IKEA'nın Thread destekli cihazları gibi popüler akıllı ev ekosistemleri, çeşitli haberleşme protokolleri kullanır. Zigbee ve Thread gibi protokoller düşük güç tüketimi ve kısa mesafeli iletişim için optimize edilmişken, Matter daha geniş kapsamlı bir ekosistem sağlar. Bu protokoller genellikle cihazlar arasında güvenli iletişim sağlamak için AES şifreleme gibi yöntemler kullanır; ancak, şifreleme anahtarlarının ele geçirilmesi durumunda protokoller sahte komutlara açık hale gelir.

Zararlı aktiviteler, genellikle gömülü dosya sistemlerinde saklanan kötü niyetli kodlar, yetkisiz ağ trafiği ve maniple edilmiş firmware güncellemeleri şeklinde ortaya çıkar. Örneğin, saldırganlar, firmware üzerinde değişiklik yaparak cihazın kontrolünü ele geçirebilir veya MQTT gibi protokolleri kullanarak sahte komutlar gönderebilir.

Matter protokolü ile çalışan cihazlarda, kimlik doğrulama sürecindeki zayıflıklar zararlı aktiviteler için bir giriş noktası oluşturabilir. Zigbee tabanlı cihazlarda ise "replay attack" gibi yöntemlerle cihaz kontrolü ele geçirilebilir.

Bu aktiviteleri analiz etmek için, cihazların ağ trafiği Wireshark, Zigbee Sniffer veya OpenThread Sniffer ile izlenir. Firmware dosyaları ise Binwalk, Firmware Mod Kit veya Ghidra gibi araçlarla ayrıştırılarak detaylı şekilde incelenir. Ayrıca, donanım seviyesinde analiz için JTAG (Joint Test Action Group) veya SWD (Serial Wire Debug) bağlantıları kullanılarak cihazın bellek ve işlemci durumu gerçek zamanlı olarak izlenebilir. Güvenlik açıklarını belirlemek ve cihazın tehlikelere karşı korunmasını sağlamak için hem yazılım hem de donanım düzeyinde entegre bir analiz yaklaşımı uygulanmalıdır. Akıllı ev cihazlarında kullanılan protokollerin detaylı incelenmesi, zararlı aktiviteleri tespit etmek ve olası saldırıları önlemek için kritik öneme sahiptir.

Bir IoT zararlı yazılımın laboratuvar ortamındaki analizi kapsamlı ve sistematik bir yaklaşımla yapılmalıdır. Öncelikle, cihazın ağ trafiği Wireshark veya özel bir protokol analiz aracı (örneğin, Zigbee Sniffer veya OpenThread Sniffer) ile izlenir. Cihazın iletişim kurduğu IP adresleri, protokol davranışları ve potansiyel komut kontrol sunucuları tespit edilir. Firmware analizi için, cihazın ROM görüntüsü Binwalk veya Firmware Mod Kit gibi araçlarla ayrıştırılarak, dosya sistemindeki zararlı yazılımlar ve şüpheli dosyalar araştırılır. Zararlı kodun bulunduğu dosyalar Ghidra veya IDA Pro gibi tersine mühendislik araçları ile statik analiz edilir. Dinamik analiz aşamasında, JTAG veya SWD bağlantıları kullanılarak cihazın belleği ve işlemci durumu gerçek zamanlı olarak izlenir. Bu süreçte, şüpheli kod parçalarının yürütülmesi sırasında bellek davranışları ve sistem çağrıları detaylı bir şekilde gözlemlenir. Eğer cihaz bir haberleşme protokolü (örneğin, MQTT, CoAP veya Matter) kullanıyorsa, bu protokoller üzerindeki veri akışı deşifre edilerek potansiyel zararlı komutlar tespit edilir. Analiz sırasında kimlik doğrulama ve şifreleme süreçlerinin zayıflıkları, zararlı aktivitelerin giriş noktaları olarak incelenmelidir. Donanım seviyesindeki detaylarla birlikte entegre bir yazılım ve ağ analizi gerçekleştirildiğinde, IoT zararlı yazılımının davranışı tam olarak anlaşılabilir ve etkili bir çözüm üretilebilir.

Script Dosyaları

Script dosyaları Python (.py), Batch (.bat), JavaScript (.js), PowerShell (.ps1), Bash (.sh), Ruby (.rb), Perl (.pl), PHP (.php), TCL (.tcl), Lua (.lua), VBS (.vbs) ve Shellcode (.bin). Bu script türleri genellikle platforma özgü veya bağımsız şekilde kullanılabilir. Her biri belirli sistemlere veya uygulama senaryolarına özel tehdit vektörleri sunar. Örneğin, genellikle Windows sistemlerinde kullanılan PowerShell otomasyon görevleri, sistem yönetimi veya zararlı yazılım yükleme işlemleri gerçekleştirebilir. Bash ise Linux tabanlı sistemlerde sıkça tercih edilir ve komut dosyası saldırılarında (command injection) önemli rol oynayabilir. JavaScript ise tarayıcı tabanlı tehditlerde veya Node.js ortamında hedef sistemlere erişim sağlamak için yaygın olarak kullanılır. Lua ve TCL gibi diller, genellikle gömülü sistemlerde kullanılır ve zararlı kod yüklemeye yönelik tehditler oluşturabilir. Shellcode ise özellikle bellek bazlı saldırılarda kullanılır ve binary formatında çalışarak sistem zafiyetlerinden faydalanabilir. Her script türü, farklı kodlama ve şifreleme teknikleriyle analizden kaçınmaya çalışabilir.

Platforma özel script dosyaları, ilgili işletim sistemlerine özgü tehdit vektörleri oluşturabilir. Örneğin, PowerShell genellikle Windows sistemlerinde tercih edilir ve .NET Framework ile derin entegrasyonu sayesinde ileri düzey otomasyon ve zararlı yazılım yükleme işlemleri gerçekleştirebilir. PowerShell script'lerinde sıkça rastlanan Base64 kodlaması, payload'ların okunmasını engellemek için kullanılır ve zararlı komutlar çalıştırılmadan önce dinamik olarak çözülür. Bash ise genellikle Linux tabanlı sistemlerde tercih edilir ve sistem komutlarına erişim sağlayarak genişletilmiş saldırı senaryoları oluşturabilir. Bash script'lerinde ise obfuscated (gizlenmiş) komutlar ve çevresel değişken manipülasyonları yaygındır. Kodlama ve Şifreleme: Script dosyalarında yalnızca Base64 değil, XOR, AES gibi daha karmaşık şifreleme yöntemleri de kullanılabilir. Bu şifreleme yöntemleriyle kodların çözülmesi ve statik analizle okunabilir hâle getirilmesi kritik öneme sahiptir. Statik analiz araçları ile (örneğin, CyberChef, Visual Studio Code Extensions) bu tür şifrelemeler çözülüp analiz edilebilirken, dinamik analizde Frida veya Objection gibi araçlar zararlı script'in gerçek zamanlı davranışlarını incelemek için kullanılabilir. Bu süreçlerde, script'in hangi dış kaynaklara erişim sağladığı veya hangi API çağrılarını kullandığı detaylı şekilde analiz edilmelidir.

Kodun çalıştırılmadan önce statik analiz ile çözülmesi gerekir. Bu süreçte Rabin2 (Radare2'nin bir bileşeni), Hopper Disassembler, veya Binary Ninja gibi araçlar kullanılabilir. Rabin2, ELF, PE ve Mach-O gibi farklı dosya formatlarını destekler ve dosyanın meta bilgilerini hızla analiz etmeyi sağlar. Hopper Disassembler, özellikle macOS ve iOS platformları için optimize edilmiştir ve kodun fonksiyonel akışını anlamak için görsel grafikler sunar. Binary Ninja, gelişmiş API entegrasyonu ve scripting desteğiyle statik analiz sırasında özelleştirilmiş analiz yapmanızı sağlar. Statik analiz sırasında karşılaşılan zorluklar genellikle sıkı obfuscation (gizleme) yöntemleri veya çok katmanlı packer kullanımıyla ilişkilidir. Bu durumlarda, dinamik analiz araçlarıyla (Unicorn Engine veya Qiling Framework) birleşik bir yaklaşım benimsenerek zararlı kodun çalışması sırasında bellek izleme ve sistem çağrıları yakalama gibi yöntemlerle analiz tamamlanabilir. Özellikle script tabanlı analizlerde, R2pipe veya Binary Ninja API gibi araçlar kullanılarak kod bölümleri arasında ilişkiler kurulabilir ve şifreleme mantıkları daha etkili bir şekilde çözülür.

Ofis Belgeleri ve PDF Dosyaları

Makro Destekli Ofis Dosyaları: Ofis belgeleri, ilk kez 1980'lerin sonunda yaygın olarak kullanılmaya başlanmış, ancak özellikle 1990'larda makro desteğinin eklenmesiyle zararlı yazılım taşıyıcılarına dönüşmüştür. Her yeni ofis belge türü, zararlı yazılım analizi açısından farklı zorluklar sunar. Örneğin, eski .doc formatında makro kodları düz metin olarak bulunabilirken, modern .docx dosyaları XML tabanlı bir yapıya sahiptir ve şifrelenmiş içerikler barındırabilir. Ayrıca, .xlsm gibi makro destekli dosyalar, VBA (Visual Basic for Applications) makrolarıyla zararlı kodları taşımak için kullanılır. Bu tür belgelerde, makrolar sık sık Base64 ile kodlanmış veya şifrelenmiş olarak saklanır ve çalıştırılmadan önce çözülmeleri gerekir.

Bu belgelerin analizi sırasında kullanılan araçlardan Oletools, OLE (Object Linking and Embedding) formatındaki zararlı içerikleri statik olarak analiz etmek için kullanılırken, OfficeMalScanner makroların ve gömülü objelerin davranışlarını analiz edebilir. Modern .docx veya .pptx dosyalarının analizinde ise bu belgelerin ZIP formatında saklandığı dikkate alınarak, arşiv dosyasının içindeki içeriklerin manuel olarak ayrıştırılması gerekebilir. Ek olarak, gömülü ActiveX bileşenleri veya dış bağlantılar gibi yapıların, sistem üzerinde nasıl bir etki yaratacağı değerlendirilmelidir. Her yeni belge formatı, daha karmaşık şifreleme veya kod gizleme yöntemleri sunarak analiz sürecini zorlaştırabilir, bu nedenle hem statik hem de dinamik analiz araçları birlikte kullanılmalıdır.

Ofis belgelerini zararlı yazılım kapsamında analiz ederken teknik olarak şu hususlara dikkat etmek gerekir: Zararlı yazılım geliştiricileri, belgeleri yalnızca taşıyıcı olarak değil, aynı zamanda saldırının aktif bir parçası olarak kullanır. Örneğin, makro destekli belgelerde (örneğin, .docm, .xlsm), zararlı yazılım yaratıcıları, kodları Base64 ile gizler, dış bağlantılar kullanarak kötü amaçlı dosya indirir veya dinamik olarak oluşturulmuş kodlarla analizden kaçınmaya çalışır. Bu nedenle, VBA kodlarının dinamik davranışlarının incelenmesi ve özellikle "on open" gibi olay tabanlı tetikleyicilerin araştırılması kritik öneme sahiptir. ZIP tabanlı modern ofis formatlarında (örneğin, .docx, .xlsx), içerik genellikle sıkıştırılmış ve şifrelenmiş halde saklanır. Zararlı yazılım geliştiricileri, içeriğin içine gizli dosyalar veya meta veriler aracılığıyla komut kontrol (C2) altyapılarını saklayabilir.

Ek olarak, ActiveX bileşenleri veya gömülü JavaScript gibi içerikler, kullanıcı sisteminde beklenmedik etkiler yaratarak ağ bağlantıları oluşturabilir veya dosya sistemine erişebilir. Örneğin, kötü amaçlı bir ActiveX bileşeni, belgenin açılmasıyla birlikte zararlı bir payload'u çalıştırabilir. Bu tür saldırıları analiz etmek için statik analiz araçları (örneğin, Oletools, OfficeMalScanner) ile belge içeriği detaylı olarak incelenmelidir. Dinamik analiz araçları ise zararlı yazılımın çevresel etkilerini ve davranışlarını gözlemlemek için kullanılabilir. Ayrıca, dosyanın hash değerlerinin bilinen zararlı imzalarla karşılaştırılması, şifrelenmiş veya sıkıştırılmış içeriklerin açılarak incelenmesi ve entegre sertifikaların doğrulanması gibi adımlar da tehditleri daha etkili bir şekilde tespit etmek için önemlidir.

PDF (Portable Document Format): 1993 yılında Adobe tarafından tanıtılmıştır ve zaman içinde birçok sürümü geliştirilmiştir. Başlangıçta yalnızca metin ve temel görseller içeren bir format olarak kullanılsa da, zamanla gömülü JavaScript, form verileri ve multimedya içeriği desteği eklenmiştir. Her sürüm, zararlı yazılım geliştiricilerine yeni fırsatlar sunmuştur.

PDF 1.4 sürümü, zararlı yazılım geliştiricilerinin kullanabileceği birçok yeni özellik getirmiştir. Bu sürümde JavaScript desteğinin eklenmesi, dosyanın açılmasıyla birlikte çalıştırılabilecek otomatik komutların PDF içine gömülmesine olanak sağlamıştır. Özellikle, JavaScript motorları aracılığıyla kötü amaçlı komutlar dosya açılır açılmaz çalıştırılabilir hâle gelmiştir. Bu özellik, kullanıcı etkileşimine gerek kalmadan zararlı yazılımın yüklenmesini sağlamıştır. Ek olarak, saldırganlar bu dönemde JavaScript'i şifreleme, kodlama (örneğin, Base64) ve dinamik yükleme yöntemleriyle gizleyerek analiz süreçlerini zorlaştırmıştır. PDF'nin farklı nesne ve akış yapıları (streams) kötü amaçlı kod saklama amacıyla kullanılmıştır. Statik analiz araçları (örneğin, PDF-Parser) ve dinamik analiz yöntemleri (örneğin, sandbox ortamları) bu zararlı davranışları tespit etmek için gereklidir. Ayrıca, dosya içindeki JavaScript'i manuel olarak incelemek ve potansiyel ağ bağlantılarını analiz etmek için Peepdf ve PDF Stream Dumper gibi araçlar kullanılabilir.

PDF 1.5 sürümü ile birlikte, dosya yapısına gömülü multimedya içerikleri (örneğin, video ve ses dosyaları) ve daha karmaşık form yapılandırmaları dahil edilmiştir. Bu sürümde, içerik akışları (streams) daha karmaşık hâle gelmiş ve sıkıştırılmış veri ile şifrelenmiş nesneler yaygın olarak kullanılmaya başlanmıştır. Zararlı yazılım geliştiricileri, özellikle multimedya içeriklerine gömülü komut dosyaları ve payload'ları saklayarak tespiti zorlaştırmıştır. PDF'nin katalog nesnesi, çapraz referans tabloları (xref tables), ve gömülü dosya özellikleri, kötü amaçlı bağlantılar ve kodların saklanabileceği kritik alanlardır. Bu yapıların analizinde, PDF-Parser ve PDF Stream Dumper gibi araçlar kullanılarak dosyanın içeriği açığa çıkarılabilir. Dinamik analiz süreçlerinde, sandbox ortamında dosyanın çalıştırılması ile sistem üzerindeki etkiler gözlemlenebilir ve kötü niyetli davranışlar detaylı bir şekilde incelenebilir.

Gömülü JavaScript: Zararlı JavaScript, PDF'nin Katalog ve Interactive Elements yapılarında yer alan eklemelerle dosya açıldığında çalışacak şekilde entegre edilir. Bu kod, genellikle /Names veya /OpenAction gibi aksiyon tetikleyicilerinde saklanır ve sistemde kötü amaçlı yazılımların indirilmesine veya kullanıcı verilerinin çalınmasına olanak tanır. PDF'nin Cross-Reference Table (Xref Tablosu), bu tür nesnelerin dosya içinde hangi konumda saklandığını belirler ve analiz sırasında özel dikkat gerektirir. Ayrıca, JavaScript kodları sıklıkla şifrelenir veya

sıkıştırılarak (örneğin FlateDecode filtresi) statik analiz araçlarının tespitini zorlaştırır. Analiz sırasında, PDF-Parser ile JavaScript nesneleri çıkarılabilir ve kodun dinamik etkileri, bir sandbox ortamında çalıştırılarak gözlemlenebilir. Potansiyel ağ bağlantıları veya kötü amaçlı komut dosyaları, dosyanın /URI veya /Action nesneleri üzerinden tespit edilebilir.

PDF form alanları, zararlı kodların veya komutların saklanması için etkili bir yöntem olarak kullanılabilir. Bu form alanları, PDF yapısında /Fields nesnesi içinde saklanır ve genellikle AcroForm nesnesine bağlıdır. Zararlı yazılım geliştiricileri, form alanlarına JavaScript gömerek veya zararlı komutları metin kutularında saklayarak kötü amaçlı aktiviteler gerçekleştirebilir. Örneğin, bir form alanına entegre edilmiş /Action komutu, formun açılmasıyla birlikte otomatik olarak kötü amaçlı bir JavaScript kodunun çalışmasını sağlayabilir. Ayrıca, form alanlarına bağlı /SubmitForm gibi işlemler, saldırganların kullanıcı verilerini toplayarak belirli bir sunucuya göndermesine olanak tanır. Bu tür içeriklerin analizinde PDF-Parser ve Peepdf gibi araçlar kullanılarak form nesnelerinin içeriği detaylı bir şekilde incelenebilir. Dinamik analiz süreçlerinde, sandbox ortamında PDF dosyasının çalıştırılmasıyla form alanlarının etkinliği ve potansiyel zararları tespit edilebilir.

PDF dosyalarının içerikleri genellikle sıkıştırılmış veya şifrelenmiş biçimde saklanır. Bu, zararlı yazılım geliştiricilerinin analiz araçlarından kaçınmasını kolaylaştırır. Sıkıştırma için FlateDecode veya RunLengthDecode gibi filtreler kullanılır. Bu filtreler, dosya boyutunu küçültmek için verileri farklı biçimlerde sıkıştırır, ancak bu aynı zamanda zararlı kodların tespit edilmesini zorlaştırır. Şifreleme ise, genellikle RC4 veya AES gibi algoritmalar kullanılarak uygulanır ve dosyanın belirli bölümlerini (örneğin, metin akışları veya gömülü nesneler) yalnızca yetkili kullanıcıların erişebilmesini sağlar. Zararlı yazılım geliştiricileri, özellikle /Encrypt girişini kullanarak, dosyanın bazı bölümlerini şifreleyebilir ve analiz süreçlerini engelleyebilir.

PDF'nin sıkıştırılmış veya şifrelenmiş bölümlerinin analiz edilmesi için özel araçlar ve yöntemler gereklidir. QPDF (Quality PDF), şifrelenmiş içerikleri çözmek ve sıkıştırılmış akışları açmak için etkili bir araçtır. Ayrıca, PDF Stream Dumper (PDF akış analiz aracı) veya PDF-Parser (PDF ayrıştırıcı) gibi araçlarla sıkıştırılmış nesneler çıkarılabilir ve incelenebilir. Bu süreçte, sıkıştırma filtrelerinin (örneğin, FlateDecode veya RunLengthDecode) çözülmesi ve akış verilerinin düz metin biçiminde analiz edilmesi kritik öneme sahiptir. Şifrelenmiş içeriklerde ise, genellikle RC4 (Rivest Cipher 4) veya AES (Advanced Encryption Standard) gibi algoritmalarla şifrelenmiş bölümlerin anahtarlarının çıkarılması gerekebilir. Bu işlem, bellek dökümü veya dosyanın oluşturulma sürecine ilişkin analiz yoluyla yapılır. Dinamik analiz araçları, şifrelenmiş içeriğin açılmasını ve davranışının gözlemlenmesini sağlamak için kullanılabilir. Özellikle, sandbox (korumalı alan) ortamları, dosyanın açılması sırasında gerçekleştirilen deşifreleme işlemlerini gözlemlemek ve zararlı içeriği ortaya çıkarmak için kullanılabilir.

Zararlı Yazılım Analiz Raporu

Bir zararlı yazılım analiz raporu, zararlı yazılımın genel özelliklerini, etkilerini ve alınabilecek karşı önlemleri detaylı bir şekilde sunmalıdır. Raporun başlangıcında, zararlı yazılımın amacı ve hedef kitlesi açıklanarak analizin neden yapıldığına dair bir özet verilmelidir. Bu bölümde, zararlı yazılımın hangi platformları etkilediği, nasıl yayıldığı ve genel özellikleri hakkında bilgi sunulmalıdır. Örneğin, rapor TrickBot gibi modüler yapıya sahip bir zararlı yazılımı analiz ediyorsa, modüllerin işlevlerinden ve zararlı yazılımın hedeflerine ulaşmak için kullandığı tekniklerden bahsedilmelidir.

Bir sonraki aşamada, zararlı yazılımın incelendiği dosyalar ve kullanılan analiz yöntemleri açıklanmalıdır. Statik ve dinamik analiz yöntemleriyle elde edilen bulgular raporun bu bölümünde yer almalıdır. Statik analiz kısmında zararlı yazılımın yapısı, packer veya şifreleme yöntemleri gibi teknik özellikleri; dinamik analiz kısmında ise sistem üzerindeki davranışları, ağ aktiviteleri ve hangi kaynaklara eriştiği gibi detaylar sunulmalıdır. İnceleme sırasında kullanılan araçlar ve test ortamı da (örneğin, sanal makineler veya sandbox ortamları) belirtilmelidir.

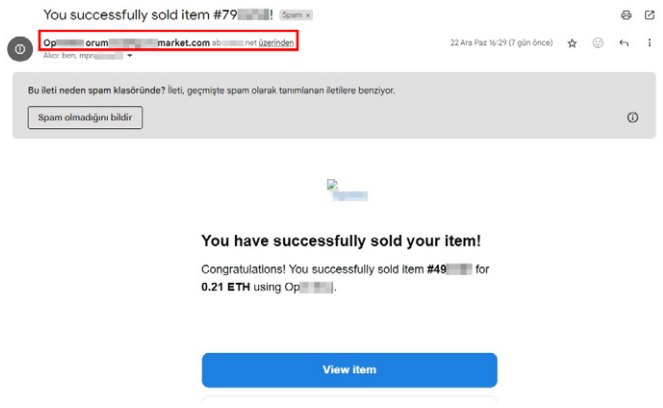
Zararlı yazılımın etkileri, raporun en kritik bölümlerinden biridir. Bu bölümde, zararlı yazılımın sistemde neden olduğu değişiklikler, hedef sistemlerin hangi bileşenlerini etkilediği ve potansiyel zararları detaylı bir şekilde açıklanmalıdır. Ayrıca, zararlı yazılımın neden olduğu kalıcılık mekanizmaları ve saldırı sonrası faaliyetleri (örneğin, veri sızdırma, kontrol sunucularına bağlantı) net bir şekilde belirtilmelidir. IoC'ler (Indicators of Compromise) gibi tespit edici göstergeler liste halinde sunulmalı ve zararlı yazılımın tespit edilmesine yönelik YARA kuralları, IDS/IPS imzaları gibi somut teknik öneriler de eklenmelidir. Raporun sonunda, analizin sonuçları özetlenmeli ve bu tür tehditlere karşı alınabilecek stratejik önlemler sunulmalıdır.

2. Dijital Varlıklarda Büyüyen Tehdit: NFT Pazar Yerlerini Hedef Alan Oltalama (Phishing) Saldırıları

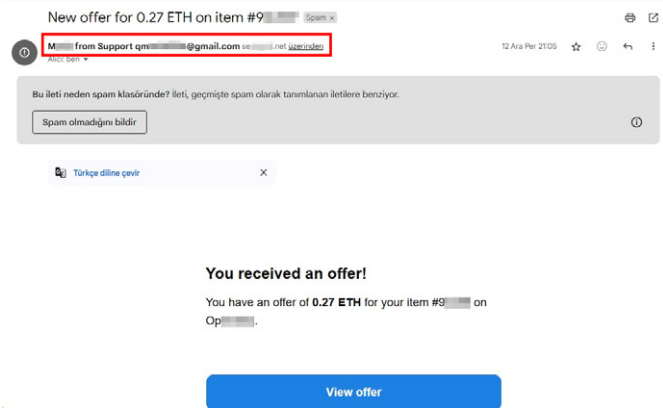
NFT (Non-Fungible Token), dijital dünyada benzersiz varlıkları temsil eden bir token türüdür. Her NFT, kendine özgü bir kimliğe sahiptir ve dijital sanat eseri, müzik, video gibi içeriklerin orijinalliğini ve sahipliğini blok zinciri üzerinde doğrular. Blok zinciri teknolojisiyle kayıt altına alınan NFT'lerin sahipliği ve geçmişi şeffaf bir şekilde izlenebilir. Bu yapı, NFT'lerin güvenli bir şekilde alınıp satılmasını sağlar. Genellikle Ethereum gibi popüler blok zinciri ağlarında işlem gören NFT'ler, günümüzde dijital koleksiyoncular için önemli bir yatırım aracı hâline gelmiştir.

NFT'ler için artan talep, NFT'lerin değerine olumlu yönde yansımaktadır. Ancak siber saldırganlar bu durumu fırsat olarak değerlendirmektedir. Özellikle oltalama (phishing) saldırıları, en büyük tehlikelerden biri olarak dikkat çekmektedir. NFT pazar yerlerinde deneyimli kullanıcılar, bu saldırıları tespit etme konusunda daha dikkatli olabilirken yeni kullanıcılar bu tehditleri fark etmekte zorlanabilmektedir. Saldırganlar genellikle bilinen NFT pazar yerlerinin bir kopyasını oluşturarak oltalama e-postaları aracılığıyla insanları oluşturdukları sahte sitelere çekmeye çalışmaktadır. Hedef aldıkları kişilere bir NFT için teklif aldıkları veya bir NFT'nin yüksek bir değerden satıldığı gibi konularda sahte e-postalar gönderilmektedir. Şekil 1 ve Şekil 2' de iki örnek oltalama e-postası bulunmaktadır. Bu örneklerde e-postalar spam klasörüne düşmüş olsalar da birçok durumda spam filtrelerini başarıyla geçilebildikleri bilinmektedir [1].

Hedef alınan kişi bu e-postalarda yer alan butona/adrese tıkladığında, kontrolü saldırganların elinde olan, orijinalinden neredeyse hiçbir farkı olmayan sahte siteye yönlendirilmektedir. Bu sitede kurbanın e-postalarda bahsi geçen konuyu teyit edebilmesi için kripto cüzdanını bağlaması istenmektedir. Cüzdan bağlama ve imza işleminin ardından kurbanın NFT'leri ve hatta diğer kripto varlıkları çalınmaktadır.



Şekil 1: Bir NFT'nin satıldığına dair gönderilen oltalama e-postası.



Şekil 2: Bir NFT için teklif alındığına dair gönderilen oltalama e-postası.

Nasıl Güvende Kalabiliriz?

Oltalama saldırıları çoğu zaman basit kurgulansa da etkileri büyük olabilmektedir. Bu tür saldırılardan korunmak için alınabilecek bazı önlemler var:

- Gelen e-postalara spam klasöründe olmasa bile her zaman temkinli yaklaşılmalı ve e-postanın kimden geldiği dikkatlice incelenmelidir. Resmî platformlardan geliyormuş gibi görünen, ancak tamamen farklı veya küçük değişiklikler içeren gönderici adreslerine dikkat edilmelidir.
- Benzer şekilde, yönlendirilmek istenen web sitesinin adresi de kontrol edilmelidir. Bağlantılara tıklanmadan önce resmî web sitesi güvenilir birkaç farklı kaynaktan doğrulanmalı, eşleşmiyorsa adrese gidilmemelidir.
- Diğer oltalama saldırılarında da çok sık karşılaşılan, kurbanı acele etmeye yönelten, panik veya heyecan yaratan durumlarda dikkatli olunmalıdır.
- Yazım hataları ve görselliği bozan hatalar içeren e-postalara şüpheyle yaklaşılmalıdır. Resmî kaynaklardan gelen e-postalarda genellikle yazım hataları veya görsel hatalar yer almaz. Ancak bir e-postada hata bulunmamasının tek başına e-postanın güvenli olduğu anlamına gelmeyeceği unutulmamalıdır.
- E-postalarda gönderilen ekler, kaynağı ve güvenilirliği doğrulanmadan asla açılmamalıdır.
- Saldırganlar, insanları oluşturdukları sahte sitelere çekebilmek için sosyal platformları da sıklıkla kullanmaktadır. Bu sebeple sosyal platformlarda gezinirken de dikkatli olunmalı ve paylaşılan bağlantılara tıklamadan önce güvenilirlikleri sorgulanmalıdır.
- Sosyal platformlardaki bir diğer tehdit ise resmî ve güvenilir hesapların saldırganların eline geçmesidir. Böyle bir durumda saldırgan, oltalama saldırısını resmi hesap üzerinden yapacağı için bu hesaptan paylaşılacak gönderilere sorgusuz güvenen kişiler ciddi kayıplarla karşı karşıya kalacaktır.

2020 yılında siber saldırganlar tarafından Twitter'a yönelik benzer bir saldırı gerçekleştirilmişti. Barack Obama, Bill Gates ve Elon Musk gibi çok sayıda tanınmış kişi ve şirketin hesaplarından saldırganlar tarafından paylaşımlar yapılmıştı. Şekil 3'te bunun bir örneği bulunmaktadır.

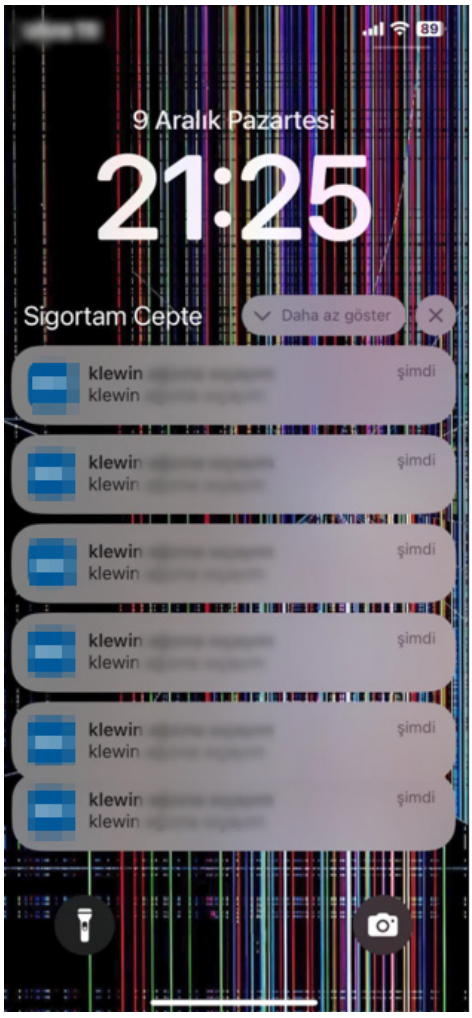
Şekil 3: Saldırganlar tarafından Bill Gates'in hesabı üzerinden gönderilen bir tweet.

Bu paylaşımlar Twitter yetkilileri tarafından kaldırılmadan önce saldırganın cüzdan adresine 110.000 dolar değerinde BTC gönderilmişti [2]. Bu sebeple resmî kaynakların paylaşımlarına da sorgusuz sualsiz güvenmemek ve temkinli yaklaşmak gerekmektedir.

Oltalama saldırıları, tek bir dikkatsizlikle geri dönüşü olmayan varlık kayıplarına yol açabilir. Dijital varlıklarınızı korumanın en etkili yollarından biri, her adımda dikkatli olmak ve her şeye şüpheyle yaklaşmaktır.

3. Mobil Uygulama Güvenliğinde Dikkat Edilmesi Gerekenler

Geçtiğimiz günlerde bir sigorta şirketinin mobil uygulamasından ve hızlı geçiş sisteminin mobil uygulamasından kullanıcılara peş peşe kötü içerikli bildirimler gönderildi. Bunun bir örneği Şekil 4'te gösterilmiştir. Kötü niyetli bildirimlerin yanı sıra belirli bir cüzdana binlerce dolarlık Bitcoin gönderilmesi istendi. İki kurum da yaptıkları açıklamalarda bildirimlerin mesaj servisine yetkisiz erişim yapılarak gerçekleştirildiğini bildirdiler.



Şekil 4: Uygulamalara gönderilen kötü niyetli bildirimler[3].

Uygulamaların mesaj bildirimlerinin OneSignal kütüphanesiyle yapıldığı tespit edildi. OneSignal'ın yöneticisi ve kurucu ortağı George Deglin, sosyal medyada Türk kullanıcıların konuyla ilgili tweet'lerinden dolayı sorunun OneSignal genelinde görülmediğini, belirtilen iki uygulamada API Keylerin yetkisiz erişim sağlanmasından kaynaklandığı belirtti. Uygulamalara ait hem apk hem de ipa dosyaları incelendiğinde sadece api key değil, app id ve proje id gibi bilgilerin env dosyasında veya kaynak kod içinde yer alan sınıfta açık metin şekilde yer aldığı tespit edilmiştir. Bu tür saldırılara karşı daha dikkatli olabilmek için, mobil uygulama güvenliğiyle ilgili dikkat edilmesi gereken bazı maddeler aşağıda sıralanmıştır[4], [5].

1. **Hassas Verilerin Güvenli Depolanması**

   Hassas bilgiler açık metin olarak saklanmamalıdır. Güçlü şifreleme algoritmaları kullanılmalıdır. Veriler mümkünse cihazda saklanmamalı, harici depolamadan kaçınılmalıdır.

2. **Uygulama Girdi Kontrollerinin Yapılması**

   SQL Injection, XSS gibi saldırı türlerine karşı önlem alınmalı, kullanıcı girdileri mutlaka doğrulanmalıdır. Sunucu tarafında giriş doğrulama mekanizmaları uygulanmalıdır.

3. **Veri İletimi Güvenliği**

   Tüm veri iletimleri HTTPS protokolü ile yapılmalıdır. SSL/TLS sertifikalarının geçerli ve güncel olduğundan emin olunmalıdır.

4. **Kod Karıştırma Kullanılması**

   Kaynak kod içinde kritik bilgilere yer verilmemelidir. Aynı zamanda kod karıştırma (obfuscation) teknikleri ile korunmalıdır.

5. **Kullanıcı Yetkilendirmesi ve Yetkilendirme Kontrolleri**

   Kullanıcılar yalnızca kendi yetki alanlarındaki verilere erişebilmelidir. API'lerde kapsamlı yetkilendirme kontrolleri uygulanmalıdır.

6. **Güçlü Kimlik Doğrulama Mekanizmaları**

   Uygulama, kullanıcılar için güçlü parolaların ve çok faktörlü kimlik doğrulamanın (MFA) zorunlu olması şekilde yapılandırılmalıdır. Kayıt işlemleri sırasında kullanılan mail adresi veya telefon numarası gibi bilgilerin doğruluğu teyit edildikten sonra hesap açılmalıdır. Parola değiştirme veya unuttum yöntemleri bu adres veya telefona gelecek zaman limitli linkler ile yapılmalıdır.

7. **Oturum Yönetimi**

   Oturum süre aşımı uygulanmalı ve kullanıcının oturumu, belirli bir süre sonra otomatik olarak sonlandırılmalıdır. Oturum belirteçleri (tokens) güvende tutulmalı ve kötüye kullanılmasının önlenmesi için şifrelenmelidir.

8. **Güvenli Üçüncü Taraf Kütüphaneler Kullanımı**

   Uygulamada kullanılan üçüncü taraf kütüphanelerin güvenli ve güncel olduğundan emin olunmalıdır. Kütüphanelerin zafiyetleri kontrol edilmelidir. Var olan bir zafiyetten uygulamanın etkilenip etkilenmediği kontrol edilmelidir.

9. **Jailbreak ve Root Kontrolleri**

   Jailbreak veya root yapılmış cihazlarda uygulamanın çalışması sınırlandırılmalıdır. Bu tür cihazlarda hassas verilere erişim kontrolü artırılmalıdır.

10. **Uygulama Güncellemeleri**

    Güvenlik açıklarını gidermek için düzenli olarak güncelleme yapılmalıdır. Kullanıcıların eski sürümlerde kalmaması için güncelleme politikaları uygulanmalıdır.

11. **İzin Yönetimi**

    Kullanıcı cihazındaki izinler minimum düzeyde tutulmalı ve yalnızca uygulama çalışması için gerekli izinler istenmelidir.

4. Mikroservis Mimarilerde Güvenlik

Mikroservis mimarisi, yazılım sistemlerinin küçük, bağımsız ve birbirinden bağımsız olarak geliştirilip dağıtılabilen servislere bölünmesi prensibine dayanır. Daha esnek, ölçeklenebilir ve bakımı kolay uygulamalar geliştirilmesine olanak tanır. Her mikroservis, belirli bir işlevi yerine getiren, kendi içinde stabil ve bağımsız dağıtım mekanizmasına sahip bir bileşendir. Mikroservis mimarisi ile kullanılan çeşitli sistemler ve uygulamalara dair bazı örnekler aşağıdaki gibidir:

- E-ticaret Platformları
- Sosyal Medya Uygulamaları
- Finansal Yazılımlar
- Medikal ve Sağlık Uygulamaları
- Haber ve İçerik Yönetimi Platformları
- Oyun Platformları
- IoT Uygulamalar

Büyük ölçekli uygulamalar için iş süreçlerini kolaylaştıran mikroservisler, bazı ek güvenlik tedbirlerinin alınmasını zorunlu kılar. Güvenlik açığı oluşturan hatalar, sistemin yapılandırılması, hizmetlerin birbirleriyle etkileşimi, veri yönetimi ve kullanıcı erişimi gibi bir dizi farklı alanda ortaya çıkabilir. Monolitik mimari için gerekli güvenlik tedbiri hem ayrı ayrı mikroservis bileşenleri için hem de mimarinin bütünsel altyapısı ve birbirleri arasındaki iletişim için alınmalıdır. Mikroservis mimarisinde sıkça karşılaşılan güvenlik açıklarına yol açabilecek bazı yazılım veya konfigürasyon hataları aşağıdaki gibidir:

Kimlik Doğrulama ve Yetkilendirme Eksiklikleri

Servisler arasındaki iletişimde kimlik doğrulama (authentication) ve yetkilendirme (authorization) eksiklikleri varsa, bir mikroservis, diğer mikroservislere erişim sağlayabilir. Kullanıcı ve servis rollerine dayalı olarak doğru erişim izinlerinin verilmemesi, yetkisiz erişimlere yol açabilir.

Zayıf Şifreleme ve Veri Gizliliği

Mikroservisler arası veri iletiminde şifreleme eksikliği, ağ üzerinden geçen hassas bilgilerin ele geçirilmesine neden olabilir. TLS/SSL gibi güvenli protokoller kullanılmalıdır. Veritabanlarında veya dosya sistemlerinde yeterince güvenli şifreleme kullanılmaması, hassas verilerin sızmasına yol açabilir.

Servisler Arası Güvenlik Açıkları

Mikroservisler birbirleriyle genelde API'ler aracılığıyla iletişim kurar ve dışa açık uçlar güvenlik açığı yaratabilir. API'lerin yeterli güvenlik önlemleriyle korunmaması (ör: rate limiting, IP filtreleme vb.) DDoS saldırılarına veya kötü niyetli kullanıma neden olabilir. API anahtarları, şifreler ve diğer gizli bilgiler yanlışlıkla kaynak kodunda veya loglarda saklanabilir. Bu durum, servisler arası güvenliği tehlikeye atabilir.

Mikroservislerin İzlenebilirlik Eksiklikleri

Mikroservislerin aktiviteleri hakkında yeterince loglama yapılmaması, saldırganların kötü niyetli faaliyetlerini tespit etmeyi zorlaştırır. Ayrıca, merkezi bir loglama ve izleme sistemi kurulmazsa, güvenlik ihlallerini izlemek ve müdahale etmek zor olabilir. Aynı şekilde kaydedilen logların şifrelenmeden depolanması, saldırganların hassas bilgilere erişmesini sağlayabilir.

Yanıt Verme ve Hata Yönetimi

Mikroservislerdeki hatalar, sistemin iç yapısı hakkında bilgi verebilir. Detaylı hata mesajları ve istisnalar, saldırganlara sistemin zafiyetlerini gösterebilir. Saldırı durumlarında mikroservisler, güvenlik ihlallerini hızlı bir şekilde algılayamıyorsa, saldırganların sistemde uzun süre kalması mümkün olabilir.

Bileşenlerin ve Bağımlılıkların Güvenlik Açıkları

Mikroservisler, eski veya güncellenmemiş dış kütüphaneleri ve bileşenleri kullanıyorsa, bilinen güvenlik açıklarına sahip olabilir. Bağımlılık yönetimi yetersiz olduğunda bu tür açıklar ortaya çıkabilir. Mikroservislerin çalıştığı altyapılarda, düzgün yapılandırılmamış güvenlik duvarları, servislerin gereksiz yere dışa açık hâle gelmesine neden olabilir.

DevOps ve CI/CD Süreçlerindeki Hatalar

DevOps süreçlerinde güvenlik zafiyetlerini tespit etmek için yeterli güvenlik testlerinin yapılmaması, prodüksiyon ortamında keşfedilmemiş açıkların kalmasına yol açabilir. Yazılım güncellemeleri, güvenlik yamaları ve yamanın dağıtımı düzgün şekilde yapılmazsa, bilinen güvenlik açıkları aktif kalabilir.

Veri Sızıntıları ve Yanlış Yapılandırmalar

Mikroservisler arasındaki veri paylaşımında yanlış yapılandırılmış API izinleri, kullanıcıların veya hizmetlerin daha fazla veriye erişmesini sağlar. Bu durum, hassas bilgilerin sızmasına yol açabilir. Bir mikroservisin başka bir servisi gereksiz yere tam erişimle çağırması, veri sızıntısına yol açabilir.

Zayıf Hedef Tabanlı Kimlik Doğrulama ve Yetkilendirme

Kullanıcılar veya servisler için kullanılan kimlik doğrulama token'larının kötü yönetimi, bu token'ların çalınmasıyla ciddi güvenlik tehditlerine yol açabilir.

Mikroservis mimarilerinde, servisler arasındaki yazılım ve konfigürasyon güvenliği kadar, servislerin üzerinde konuşlandıkları altyapı güvenliği de önemlidir. Özellikle konteyner tabanlı uygulamalarda, imaj, konfigürasyon, konteyner ve network güvenliği, hem servislerin birbirine açık oluşturmaması hem de sistemin güvenliğinin bir bütün hâlinde sağlanması açısından oldukça önemlidir.

Image Registry Güvenliği

Image registry, konteyner imajlarının depolandığı ve hızlı bir şekilde oluşturularak dağıtıldığı bir yerdir (örneğin, Docker Hub, Google Container Registry, Amazon ECR gibi). Bu registry'ler, güvenli ve doğru bir şekilde yapılandırıldığında, mikroservisleri sağlıklı ve güvenli bir şekilde çalıştırmak için kritik öneme sahiptir. Image registry güvenliği ile ilgili bazı önemli konular alttaki gibidir:

Erişim Kontrolü ve Kimlik Doğrulama

Image Registry'lere erişimi sınırlandırmak için kimlik doğrulama mekanizmaları kullanmak çok önemlidir. Kimlik doğrulama için OAuth, LDAP, SAML gibi yöntemler kullanabilir. Yalnızca belirli kullanıcılara veya sistemlere registry üzerindeki imajları yükleme, silme veya çekme yetkisi verilmelidir.

İmajların Güvenliği (Tarama ve İnceleme)

Kötü amaçlı yazılımlar, zafiyetler veya güvenlik açıkları içerebilecek imajları tespit etmek için otomatik güvenlik taramaları yapılmalıdır. Bazı araçlar, Docker imajlarını veya diğer konteyner imajlarını tarayarak bilinen güvenlik açıklarını raporlar. Örneğin:

- Clair: Konteyner imajlarında güvenlik açıklarını tarar.
- Trivy: Güvenlik açıkları, zafiyetler ve yanlış yapılandırmaları tespit eder.
- Anchore: İmajlarda güvenlik denetimi yapar.

İmajlarınızı oluştururken kullanılan baz imajların da güncel zafiyet içermemesi önemlidir. Resmi ve güncel imajları kullanarak inşa edilen imajlar güvenlik açıklarını azaltabilir. İmajların içerdiği yazılım, yazılım bağımlılıkları ve işletim sistemleri açısından ilgili CVE kayıtlarının kontrol edilmesi ile güvenlik kontrolü yapılmalıdır.

Kapsayıcı İmzalama ve Geçerlilik Kontrolü

Docker imajlarını imzalamak, yalnızca belirli geliştiriciler tarafından sağlanan imajların kullanıldığından emin olmaya yardımcı olur. Bu, imajın içeriğinin değiştirilip değiştirilmediğini doğrulamak için kullanılır. Docker Content Trust (DCT) gibi araçlar, imzalı imajların kullanılmasını zorunlu kılabilir. İmajların yalnızca geçerli ve onaylanmış kaynaklardan alınmasını sağlamak, dışarıdan gelen kötü niyetli yazılımların engellenmesine yardımcı olur.

Şifreleme

Image registry'ye bağlantılar HTTPS aracılığı ile şifreli olarak yapılmalıdır. Bu, imajların dinlenmesini engeller ve güvenli bir şekilde iletilmesini sağlar. Registry'ye yüklenen imajlar, kötü niyetli kişilerin erişimini engellemek amacıyla şifrelenmiş olarak saklanabilir.

Güncelleme ve Yamama Yönetimi

Image registry ve ilgili araçlar düzenli olarak güncellenmeli ve yeni güvenlik yamaları uygulanmalıdır. Yeni güvenlik tehditlerine karşı registry'ler sık sık izlenmeli ve gerektiğinde hızlıca yama uygulanmalıdır. Artık kullanılmayan, eski veya hatalı imajlar güvenlik risklerini artırabileceği için sistemde tutulmamalıdır.

İmajların Güncellenebilirliği ve Dağıtımı

Mikroservis mimarisinde, imajlar sürekli olarak güncellenir ve dağıtılır. Bu süreçlerin güvenli ve izlenebilir olmasını sağlamak önemlidir. CI/CD araçları, yalnızca güvenli ve onaylı imajların dağıtılmasını sağlayacak şekilde yapılandırılmalıdır. Özellikle, ürün/üretim ortamına gönderilecek imajlar net ve izlenebilir bir şekilde etiketlenmelidir. Versiyon kontrolü yapılmalı ve stabil olmayan imajlar üretim ortamına alınmamalıdır. İmajlar sadece güvenilir kaynaklardan alınmalı ve yalnızca resmi olarak onaylanmış imajlar kullanılmalıdır. Güvenlik amacıyla proxy çözümler kullanılabilir. Bu, dışarıdan gelen potansiyel tehditleri engelleyebilir.

Audit ve İzleme

Image registry'ye erişimle ilgili tüm işlemler yerine getirilmeli ve kayıt altına alınmalıdır (loglanmalıdır). Kim, ne zaman hangi imajı kullanmış veya değiştirmiş, her işlem kaydedilmeli ve analiz edilmelidir. Image registry üzerinde yapılan şüpheli işlemleri tespit etmek için anomali tespiti çözümleri kullanılabilir.

Private Registry Kullanımı

Güvenlik açısından, private registry kullanmak genellikle daha güvenlidir. Bu şekilde, imajlara yalnızca güvenilir kişiler erişebilir ve dış tehditler sınırlandırılmış olur. Private registry kullanıldığında, sadece belirli ağlardan erişimi sağlamak için IP bazlı kısıtlamalar veya Virtual Private Cloud (VPC) ile entegrasyon yapılabilir.

Konfigürasyon Yönetimi

İmajlar, konfigürasyon dosyaları ve hassas bilgiler içerebilir. Bu dosyaların doğru bir şekilde güvenliğini sağlamak ve kimlik bilgilerini veya şifreleri imajlar içinde tutmamak önemlidir.

Konteyner ve Ağ Güvenliği

Konteynerler, mikroservislerin izolasyonunu ve taşınabilirliğini sağlar, ancak güvenlik açısından bazı önlemler alınmasını da gerektirir. Konteyner güvenliği, kritik bir rol oynar, çünkü her bir konteynerin kendi başına çalıştığı ve ağ üzerinden diğer mikroservislerle iletişim kurduğu bir ortamda saldırı yüzeyleri çoğalır. Dikkat edilmesi gereken bazı güvenlik konuları aşağıdaki gibidir:

Konteynerin Temiz ve Güvenli Olması

Daha önce anlatılan Image Security konusu aynı zamanda konteyner güvenliği için en temel konudur. İmajlar, bağımlılıklar, işletim sistemleri ve uygulama bileşenlerini içerir ve bunlara ait güvenlik açıkları veya kötü amaçlı yazılım içerebilir. İmajların güvenli kaynaklardan alınması, düzenli olarak güncellenmesi ve taranması önemlidir.

Konteyner İzolasyonu

Konteynerler, birbirinden izole edilmiş bir ortamda çalışmalıdır. Konteynerlerin sistem kaynakları, dosya sistemleri ve ağ bağlantıları, yalnızca kendi servislerinden erişebilecek şekilde izole edilmelidir. Bu, bir konteynerin başka bir konteynere veya ana sisteme zarar vermesini engellemeye yardımcı olur. Konteynerlerin konfigürasyonları, özellikle ağ ve dosya sistemi izinleri, dikkatlice yönetilmelidir. Konteynerlere sadece ihtiyaç duydukları izinler verilmeli, aşırı yetkilendirmeden kaçınılmalıdır.

Konfigürasyon Yönetimi ve Sırlama

Konteynerlerde kullanılan hassas veriler (API anahtarları, şifreler, gizli bilgiler) güvenli bir şekilde yönetilmelidir. Çevre değişkenleri veya konfigürasyon dosyaları aracılığıyla saklanmamalıdır. Bunun yerine bir yönetim aracı (HashiCorp Vault, Kubernetes Secrets vb.) kullanılmalıdır.

Konteynerlerin güvenlik yamaları için düzenli olarak güncellenmesi önemlidir. Ayrıca, mikroservislerin bileşenlerini takip etmek ve güvenlik güncellemelerini anında uygulamak gerekir.

Gözlemleme ve Denetleme

Konteynerler ve mikroservislerin tüm aktiviteleri loglanmalıdır. Bu, güvenlik olaylarını tespit etmek ve geçmişteki olayları analiz etmek için önemlidir. Güvenlik ihlalleri için anlık bildirim sistemleri oluşturulmalıdır. Mikroservisler ve konteynerlerin davranışları izlenmeli ve denetlenmelidir. Anormal aktiviteler tespit edildikçe, otomatik güvenlik politikaları devreye girmelidir.

Kubernetes Güvenliği

Kubernetes ortamlarında, servisler ve kullanıcılar için Role-Based Access Control (RBAC) politikaları uygulanmalıdır. Böylece sadece gerekli yetkileri olan kullanıcılar ve servisler ilgili yerlere erişim sağlayabilir. Kubernetes'te pod güvenliği, her pod için uygun güvenlik politikalarının oluşturulması anlamına gelir. Bu, her pod'un çalışacağı ortamın güvenli olmasını sağlar.

5. Privileged Access Management (PAM): Ayrıcalıklı Erişim Yönetiminin Önemi

Ayrıcalıklı hesaplar, organizasyonların en hassas sistemlerine erişim sağlar. Bu hesapların ele geçirilmesi durumunda, saldırganlar kritik sistemler üzerinde tam kontrol elde ederek mali kayıplara, veri ihlallerine ve itibar zedelenmesine yol açabilir. Kapsamlı bir PAM çözümünün uygulanması, şu faydaları sağlar:

- **Hesapları tanımlama ve yönetme:** Tüm ayrıcalıklı hesapların keşfedilmesi ve düzenlenmesi.
- **Şifrelerin merkezi olarak saklanması:** Hassas bilgilerin yetkisiz erişimden korunması ve kazara ifşa edilmesinin önlenmesi.
- **Erişimin izlenmesi ve denetlenmesi:** Ayrıcalıklı erişim faaliyetlerinin izlenmesi ve kaydedilmesi.
- **Güvenlik süreçlerinin otomasyonu:** Şifre değiştirme ve erişim sağlama gibi görevlerin insan hatasını azaltacak şekilde otomatikleştirilmesi.

Etkili bir PAM çözümü, ayrıcalıklı erişim yönetimi ile ilgili birçok zorluğu ele alabilecek özellikler sunar:

Keşfetme ve Envanter Yönetimi

Tüm ayrıcalıklı hesapları belirlemek ilk adımdır. Otomatik keşfetme aracı, hizmet hesapları ya da script'lere gömülü hesaplar gibi genellikle fark edilmeyen hesapları ortaya çıkarabilir. Bu sayede organizasyonlar, tutarlı politikalar uygulayabilir ve yönetilmeyen kimlik bilgisi riskini azaltabilir.

Merkezi Şifre Kasası

Merkezi bir kasa, şifreler, anahtarlar ve sertifikalar gibi hassas bilgileri saklamak için güvenli bir depolama alanı sunar. Şifreleme, bu bilgilerin korunmasını sağlarken, gerçek zamanlı izleme ve otomatik şifre değiştirme gibi özellikler bu bilgilerin güvenliğini artırır.

Rol Tabanlı Erişim Kontrolü (RBAC)

Rol tabanlı erişim kontrolü, belirli kimlik bilgilerine yalnızca yetkilendirilmiş kişilerin erişim sağlamasını garantiler. Politikalar, roller ve sorumluluklara göre oluşturularak çok daha güvenli bir erişim yönetimi sağlanabilir.

Çok Faktörlü Kimlik Doğrulama (MFA)

MFA, kullanıcılardan birden fazla kimlik doğrulama adımı talep ederek ek bir güvenlik katmanı sağlar. PAM çözümüne erişim veya ayrıcalıklı hesapları kullanma aşamalarında MFA uygulamak, çalıntı bilgilerle yetkisiz erişim sağlanmasını önler.

Oturum İzleme ve Yönetim

Ayrıcalıklı oturumların gerçek zamanlı izlenmesi ve yönetilmesi, kötüye kullanım risklerini azaltır. Oturum proxy'leri, kullanıcıları kimlik bilgilerini görmeden sistemlere bağlarken oturum kaydı ve çevrimiçi denetim gibi özellikler şeffaflık ve hesap verebilirlik sağlar.

Otomatik Süreçler ve Entegrasyonlar

Otomasyon, BT ekiplerinin üzerindeki yükü azaltır ve hata riskini düşürür. Şifre dönüşümü, uyumluluk raporlaması ve anomali uyarıları gibi süreçler otomatikleştirilebilir. Üçüncü taraf araçlarla entegrasyonlar da PAM çözümünün etkinliğini artırır.

Denetim ve Uyumluluk

Değiştirilemez denetim kayıtları, ayrıcalıklı hesaplarla ilgili tüm faaliyetleri belgelendirerek yasal düzenlemelere ve iç politikalarına uygunluğu kanıtlar. Özelleştirilebilir raporlar, kimlik bilgisi kullanımı, politika uyumu ve iyileştirme alanlarına dair öngörüler sunar.

Sonuç

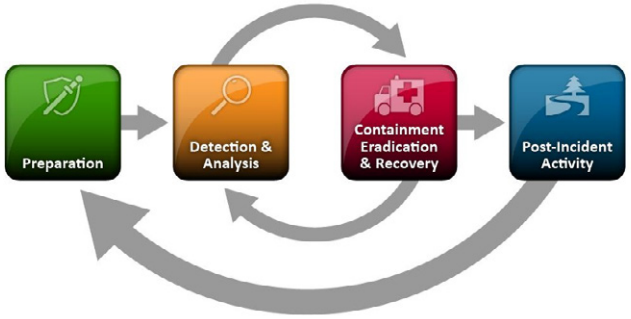
Günümüz tehdit ortamında, ayrıcalıklı erişimlerin korunması zorunluluktur. Şekil 5'te görüldüğü gibi PAM çözümü yetkisiz erişimlerin önünde, girişleri denetleyen ve yetkisiz olanları engelleyen bir kontrol noktası gibidir.

Şekil 5: Ayrıcalıklı erişim faaliyetleri izlenmelidir.

Kapsamlı bir PAM çözümü uygulayarak organizasyonlar, kritik varlıklarını güvence altına alabilir, operasyonel sürekliliği sağlayabilir ve paydaşların güvenini koruyabilir. Ayrıcalıklı kimlik bilgilerinin önemini fark etmek ve bunları etkin şekilde yönetmek, daha güvenli bir geleceğe doğru atılan ilk adımdır.

6. NIST SP 800-61 Standardına Göre Siber Olay Müdahale Yönetimi

NIST SP 800-61 Bilgisayar Güvenliği Olayı İşleme Kılavuzu[6], siber olay müdahale sürecini sistematik bir yaklaşımla ele alarak kurumların siber güvenlik olaylarını hızlı ve etkili bir şekilde yönetmelerine rehberlik eder. Bu çerçevede, müdahale süreci dört temel aşamada incelenir: **Hazırlık, Tespit ve Analiz, Müdahale ve Kurtarma, Raporlama ve Öğrenme**. Aynı zamanda bu süreci güçlendiren uygulamalar da sürecin etkinliğini artırır. Sürecin adımları ve aralarındaki ilişki Şekil 6'da gösterilmiştir.



Şekil 6: Siber Olay Müdahale Yaşam Döngüsü[7].

1. Hazırlık (Preparation)

Siber olaylara karşı güçlü bir savunma için hazırlık aşaması kritik öneme sahiptir. Bu aşamada başlıca adımlar aşağıda yer almaktadır.

- **Olay Müdahale Planları:** Organizasyonlar, farklı olay türlerine göre özelleştirilmiş planlar geliştirmelidir. Örneğin, bir fidye yazılımı saldırısı ve DDoS saldırısı için ayrı süreçler hazırlanmalıdır.
- **Ekip Yapılandırması:** Teknik uzmanlar, iletişim sorumluları ve karar vericilerden oluşan bir ekip oluşturulmalıdır. Bu ekibin görev dağılımı ve sorumlulukları uygun şekilde tanımlanmalıdır.
- **Araç ve Teknoloji Kullanımı:** SIEM, EDR, DLP gibi araçlar, olayların tespiti ve analizi için kullanılabilir, SOAR çözümleri ise otomasyonu artırarak süreci hızlandırır.
- **Tatbikatlar ve Eğitimler:** Düzenli tatbikatlar ekiplerin siber olaylara karşı daha hazır hâle getirir. Güncel tehditler ve araçlarla ilgili eğitimler ise sürekli iyileşmeyi destekler.

2. Tespit ve Analiz (Detection and Analysis)

Bu aşama, olayı belirlemek ve etkisini anlamak için kritik süreçleri içerir.

- **Proaktif İzleme ve Log Yönetimi:** Sistem ve ağ loglarının düzenli olarak analiz edilmesi şüpheli etkinliklerin tespit edilmesini sağlar. Örneğin, SIEM kullanarak şüpheli oturum açma saldırılarının tespit edilmesi.
- **Tehdit İstihbaratı Kullanımı:** MISP gibi tehdit paylaşım platformları, zararlı yazılım ve saldırgan IP bilgileri ile erken tespit sağlar.
- **Analiz Süreci:** Olayın kaynağı, saldırının kapsamı ve etkilenen sistemlerin belirlenmesi gereklidir. Örneğin, bir sistemde tespit edilen zararlı yazılımın hangi verilere eriştiği ve ne tür bir tehdit oluşturduğu analiz edilmelidir.

3. Müdahale ve Kurtarma (Containment, Eradication and Recovery)

Olayın etkisinin kontrol altına alınması ve sistemlerin güvenli bir şekilde eski hâline döndürülmesi bu aşamanın odak noktasıdır.

- **Sınırlama (Containment):** Olayın yayılmasını önlemek için hızlı önlemler alınır. Örneğin, saldırıya uğramış cihazların ağdan izole edilmesi.
- **Ortadan Kaldırma (Eradication):** Zararlı yazılımların sistemden temizlenmesi ve güvenlik açıklarının kapatılması gerekir.
- **Kurtarma (Recovery):** Sistemlerin ve operasyonların eski işlevselliğine güvenli bir şekilde döndürülmesi sağlanır. Örneğin, yedekten veri geri yükleme veya yeniden yapılandırma.

4. Raporlama ve Öğrenme (Post-Incident Activity)

Olay sonrası süreçte müdahale etkinliği değerlendirilerek süreçler iyileştirilir:

- **Raporlama:** Olay sırasında alınan aksiyonlar, etkiler ve öneriler detaylı bir şekilde kayıt altına alınır.
- **Ders Çıkarma:** Süreçteki eksiklikler tespit edilerek müdahale planları revize edilir. Örneğin, DDoS saldırısı sırasında iletişim eksiklikleri fark edilirse, iletişim prosedürleri yeniden düzenlenir.

NIST SP 800-61 Standartlarına Göre En İyi Uygulamalar

NIST SP 800-61, olay müdahale süreçlerinin etkinliğini artırmak için aşağıdaki uygulamaları önermektedir[6].

Olay Kategorileri ve Önceliklendirme

Olayların önceliklendirilmesi, müdahale süreçlerinin hızını ve etkinliğini artırır:

- **Kritik Olaylar:** Örneğin, kişisel verilere yönelik bir veri ihlali.
- **Orta Seviyeli Olaylar:** Örneğin, bir çalışanın cihazında zararlı yazılım tespiti.
- **Düşük Öncelikli Olaylar:** SPAM mailler.

Otomasyon ve Playbook Kullanımı

Önceden tanımlanmış playbook'lar, tekrarlayan siber olaylar için mevcut süreçleri hızlandırır ve otomatize hâle getirir.

- **SOAR Araçları:** Tespit, analiz ve müdahale işlemlerini otomatikleştirerek insan hatalarını azaltır.
- **Olay Müdahale Şablonları:** Örneğin, fidye yazılımı saldırıları veya DDOS için özel bir playbook oluşturulması.

Tehdit İstihbaratı Entegrasyonu

Tehdit istihbaratı, saldırganların tekniklerini anlamada kritik bir araçtır.

- **MISP ve OSINT:** Tehdit bilgilerini paylaşma ve açık kaynak raporları analiz etme araçları olarak kullanılabilir.

Eğitim ve Tatbikatlar

Düzenli eğitim programları ve tatbikatlar, ekiplerin olaylara hızlı ve etkili yanıt verebilmesini sağlar:

- **Tatbikat Senaryoları:** Örneğin, bir DDoS saldırısı veya insider tehdidi gibi durumların simülasyonu.
- **Eğitim Programları:** Güncel saldırı yöntemleri ve teknolojilere yönelik bilgi paylaşımı.

İletişim Yönetimi

Doğru ve etkili iletişim, olay müdahale sürecinin kritik bir parçasıdır:

- **İç İletişim:** Teknik ekipler ve üst yönetim arasında koordinasyonu sağlar.
- **Dış İletişim:** Müşteriler, iş ortakları ve ilgili regülasyon kurumları ile şeffaf bilgi paylaşımı yapılır.

Sonuç

NIST SP 800-61, siber olay müdahale sürecinin her aşamasında kapsamlı bir rehber sunar. Bu standart, organizasyonların tehditlere karşı hazırlıklı olmasını sağlarken, iyi uygulamalarla sürecin etkinliğini artırır. Olay müdahale ekibinin rol dağılımı, proaktif izleme, otomasyon ve tehdit istihbaratı gibi unsurlar, yalnızca mevcut tehditlere karşı değil, gelecekteki tehditlere karşı da güçlü bir savunma oluşturur.

Bu bütünsel yaklaşım, organizasyonların siber güvenlik olgunluğunu artırarak hem operasyonel sürekliliği hem de veri güvenliğini sağlamada önemli bir rol oynar.

7. Quantum Bilgisayarlar ve Geldikleri Son Nokta

Kuantum Bilgisayarların Kısa Tarihçesi

Kuantum bilgisayar fikri 1980'li yıllardan beri gündemde olmakla birlikte doğası gereği içerdiği zorluklardan dolayı görece yavaş ilerlemektedir. Günümüzde bile hâlen kabul edilebilir hata oranları ve çalışma süreleri sunan bir kuantum bilgisayar mevcut değildir[8]. Başta Çin olmak üzere birçok ülke bu teknolojiye sahip olmak için büyük yatırımlar yapmaktadır. 2022 yılında yapılan bir çalışmaya göre Çin tarafından kuantum bilgisayarlara yapılan yatırım 15 milyar dolar seviyelerindedir. Çin'i, 8,4 milyar dolarla Avrupa Birliği ve 3,7 milyar dolarlık bir yatırımla ABD izlemektedir [9]. Bu yatırımların amacı Q-day olarak ifade edilen çok az hatayla uzun süre kararlı çalışabilen bir kuantum bilgisayara sahip olmaktır. Düşük hata oranında ve uzun süre kararlı çalışabilecek seviyede bir kuantum bilgisayarın kullanılması şifreleme altyapılarından yapay zekâ uygulamalarına kadar birçok alanda çok ciddi ilerlemeler ve değişimler getirecektir. Çin, 1.024 kübitlik bir kuantum bilgisayara 2025 yılı sonunda sahip olacağını yol haritasına eklemiş olsa da[10] bazı çevreler bunun çok daha uzun süreceğini ama bu yüzyıl içinde herhangi bir anda olabileceğini düşünmektedir.

Bu yoğun çalışmaların bir sonucu olarak 2024'ün Aralık ayında Google tarafından 105 kübit kapasiteye sahip Willow kuantum bilgisayarı tanıtıldı. Daha önce de hem Google hem de büyük kuantum bilgisayarlar geliştirme çalışması yapan diğer firmalardan buna benzer açıklamalar gelmişti. Willow, kuantum bilgisayardaki en büyük sorunlardan biri olan yüksek hata oranlarına bir çözüm bulduğu iddiasıyla karşımıza çıktı [11].

Ülkemizde de 2024'ün Kasım ayında TOBB Ekonomi ve Teknoloji Üniversitesinin (TOBB ETÜ) çalışmalarıyla Quantum Computer of TOBB ETÜ (QuanT) devreye alındı. QuanT henüz çok yüksek kübit kapasitesine sahip olmasa da geliştirme çalışmaları devam etmektedir[12]. Ayrıca Türkiye Kuantum Teknolojileri Geliştirme Merkezi kurulacağı da açıklanmış bulunuyor.

Kuantum Fiziği Prensipleri

Klasik fizikten farklı olarak atom ve atom altı parçacıkların etkileşimlerini inceleyen kuantum fiziğini anlaması gerçekten de çok zordur. Kısaca fikir vermesi için kuantum bilgisayarlarda kullanılan temel prensipler şu şekildedir:

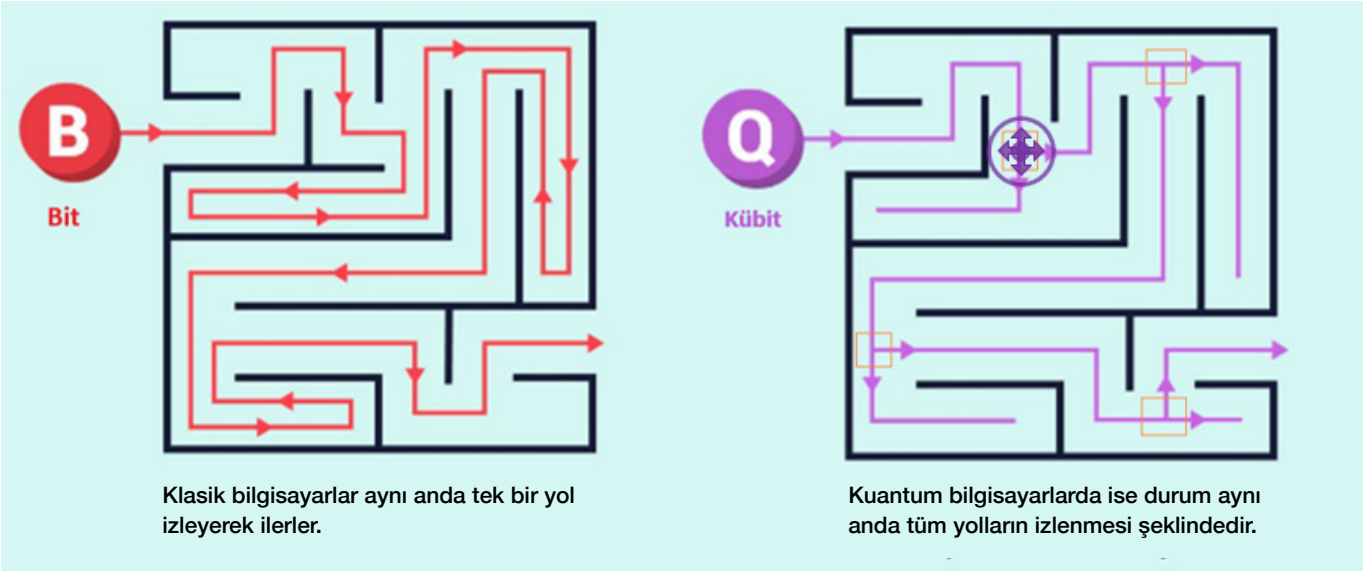
- **Süperpozisyon (Superposition)**: Klasik bitler sadece 0 veya 1 durumunda bulunabilirken, bir kübit aynı

anda hem 0 hem de 1 durumunda olabilir. Bu, kuantum bilgisayarın birden fazla hesaplamayı paralel olarak yapabilmesine olanak tanır. Bu durumu bir benzetme ile anlamak adına yazı tura için atılan madeni bir para hayal edin. Para kendi etrafından dönerken yazı ve turalar arasında geçişler yapar. Bu dönüş esnasında anlık olarak yüzde 30 yazı yüzde 70 tura şeklinde de olabilir. İşte süperpoziyon kuantum fiziğinde buna benzer bir durumu ifade etmektedir. Bu durum bir bitin 0 ile 1 arasında olması durumuna benzerdir. Kuantum biti olarak ifade ettiğimiz bir kübit 0 ve 1 arasında birçok ara değerde de bulunabilir. Kuantum bilgisayarlarda hızın asıl kaynağı bu özelliktir.

- **Kuantum Dolanıklığı (Entanglement):** Birden fazla kübit birbiriyle dolanık hâle geldiğinde, biri hakkında da bilgi sahibi olmak diğerini kontrol etmeye gerek kalmadan o kübit hakkında da bilgi verir. Bunu, yazı tura için atılan iki para için düşünecek olursak, birbiri ile dolanık olan bu iki paradan biri tura geldiğinde diğerini kontrol etmeye gerek kalmadan tura gelmiş demektir.
- **Gözlemlenebilirlik**: Kuantum durumlar ölçüldüğü anda 0 veya 1 gibi klasik bir değere dönüşür. Bu nedenle kuantum algoritmaları, hesaplama yapıldıktan sonra ölçüm işlemine dikkatle yaklaşır. Örneğimizde olduğu gibi havada dönen madeni para sonuçta yere düştüğünde sonuç yazı veya turadan biri olacaktır.

Kuantum Bilgisayarın Çalışma Mantığı:

Kuantum bilgisayarlar, klasik bilgisayarlardan farklı olarak bitler yerine kübitler kullanır. Kübit, bir kuantum sisteminin temel enformasyon birimidir ve kuantum mekaniğinin prensiplerinden faydalanır. Şekil 7, bit kullanan klasik bilgisayarlar ve kübit kullanan kuantum bilgisayarların çalışma mantıklarını örneklemektedir. Klasik bilgisayarlarda anlık olarak bir işlem biriminde sadece bir işlem yapılabilir. Klasik bir bilgisayarın bir labirentte doğru yolu bulmaya çalıştığını düşünecek olursak, Şekil 7'de de görüldüğü gibi, çıkışa gitmek için tek tek tüm yolları dener. En iyi senaryoda doğrudan yolu bulurken, en kötü senaryoda dört farklı yol ayrımında, her seferinde yanlış yola girerek 16 (2^4) farklı yolun hepsini dener. Kuantum bilgisayarlarda ise tüm yollar aynı anda denendiğinden en kötü senaryoda bile çıkış yolunu en kısa şekilde bulur. Kuantum bilgisayarda her ayrım noktasından sadece bir yol seçilerek değil her iki yol da aynı anda denenerek ilerlenir. Asıl hızın kaynağı da budur. 4 farklı yolun olduğu şekilde klasik bilgisayarlarda 2^4 tane yol varken, kuantum bilgisayarda sonuca götürecek tek yol üzerinden ilerlenmektedir. Geriye kalan 15 yol da aynı anda denendiğinden, çıkışa giden doğru yol tek adımda bulunabilmektedir. Örneğimizde bir değişiklik yapıp bir yolun sadece 2 farklı yöne değil 100 farklı yöne ayrıldığını kabul edelim. Bu durumda klasik bir bilgisayarda denenebilecek yol sayısı her seferinde 100 farklı yol ayırımı ile 100 milyon (10^8) olur. Klasik bir bilgisayar en kötü senaryoda 100 milyon denemenin hepsini; ortalama olarak 50 milyon **art arda** deneme yaparak çıkışa ulaşacaktır. Kuantum bilgisayar ise sonuca ulaşmak dört ayrım noktasını da doğru şekilde bulmasıyla tek bir yoldur. Kuantum bilgisayar 100 milyon farklı yolun hepsini **aynı anda** denediği için tek deneme ile sonuca ulaşır. Kuantum bilgisayarların sağladığı üstel fayda muazzam boyutlardadır. Verdiğimiz küçük örnekte bile durum böyleyken birçok değişkenin olduğu gerçek hayat problemlerinde sonuçların hayret verici boyutlara ulaşması beklenmektedir.



Şekil 7: Klasik bilgisayarlar ve kuantum bilgisayarlar[13].

Kuantum Bilgisayarları Geliştirmekteki Zorluklar:

Kuantum bilgisayarlar gerçek hayatta henüz anlamlı bir çözüm olamamışlardır. Kuantum bilgisayarların geliştirilmesinde karşılaşılan zorluklar şunlardır:

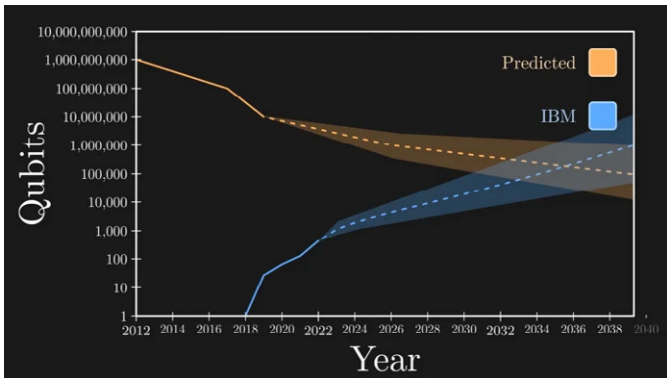
- **Hata Düzeltme ve Koherens Süresi**: Kübitlerin kararlı bir durumda kalabilme süresi, yani "koherens süresi" çok kısadır. Kuantum hesaplamaları esnasında çevreden gelen gürültüler (noise) kübitlerin hassas dengesini bozabilir. Bu nedenle, kuantum hata düzeltme algoritmaları ve kararlı çalışmayı artıran çözümler geliştirilmektedir. Yukarda bahsettiğimiz Google tarafından geliştirilen Willow kuantum bilgisayarların çok ses getirmesinin nedeni, bu hata düzeltme yöntemlerinde kayda değer bir ilerleme kaydettiklerini iddia etmeleridir.
- **Düşük Sıcaklık Gereksinimi**: Kübitler, genellikle süperiletken malzemelerle çalışır ve bu malzemeler mutlak sıfıra yakın sıcaklıklarda kararlı hâle gelir. Bu kadar düşük sıcaklıkları sağlamak ve korumak, son derece maliyetlidir ve teknik olarak karmaşıktır.
- **Kübit Sayısı ve Bağlantıları**: Günümüzde geliştirilen kuantum bilgisayarlarda kübit sayısı artırma çabaları kübitler arasındaki bağlantıların kontrolünü çok zorlaştırmaktadır.
- **Kuantum Algoritmalar**: Kuantum bilgisayarların potansiyelini tam anlamıyla kullanabilecek algoritmalar hâlen gelişme aşamasındadır. Shor ve Grover algoritmaları gibi belli başlı örneklerin yanında özel problemler için daha fazla algoritmaya ihtiyaç duyulmaktadır.

Kuantum Bilgisayarların Getireceği Avantajlar ve Dezavantajlar:

Bahsettiğimiz bu üstün hesaplama gücüyle kuantum bilgisayarların birçok alanda kayda değer ilerlemelere imkân vereceği düşünülmektedir. Bunların bazıları şu şekildedir:

- **İlaç Endüstrisinde Devrim:** Kuantum bilgisayarlar, moleküler yapıları ve kimyasal reaksiyonları eşsiz bir hassasiyetle simüle edebilir. Bu yetenek ile yeni ilaçların keşfinde büyük bir hızlanma sağlayarak kişiselleştirilmiş tedaviler ve karmaşık hastalıklar için daha etkili tedaviler geliştirilmesine olanak tanır.
- **Karmaşık Hesaplama Problemlerini Çözme**: Klasik bilgisayarlarla neredeyse imkânsız olan hesaplamaları yapmada kuantum bilgisayarlar çok iyi sonuçlar vermektedir. Örneğin, küresel lojistik optimizasyonu, karmaşık diferansiyel denklemlerin çözülmesi ve büyük veri kümesi analizleri, klasik bilgisayarlara göre çok daha hızlı bir şekilde yapılabilir.
- **Hassas Simülasyonlar:** Kuantum bilgisayarlar kimyasal reaksiyonları, malzeme bilimi ile ilgili süreçleri ve hatta evrenin fiziksel modellerini simüle etmede çok başarılıdır. Bu yetenek yeni enerji kaynaklarının keşfi ve daha dayanıklı malzemelerin geliştirilmesi gibi yeniliklere olanak tanır.
- **Bilimsel Araştırmalara Katkı:** Kuantum bilgisayarların sağladığı hesaplama gücü, astrofizik, genetik ve yapay zekâ gibi bilimsel alanlarda önemli ilerlemeler sağlayabilir. Özellikle yapay zekânın durdurulamaz yükselişini çok daha hızlandıracak katkılar sunabilirler. Bu durum, yapay zekâyı gerçekten hayal gücümüzün çok ötesine taşıyacak bir potansiyele sahiptir.

Sağladığı avantajların yanında kuantum bilgisayarlar mevcut şifreleme altyapısı için de büyük bir tehdit oluşturmaktadır. Belli bir işlem gücüne erişmiş olan kuantum bilgisayarlar ile belli şifrelerin çözülmesi çok daha kolaylaşacaktır. Şekil 8'de bu çerçevede yapılan bir tahmin paylaşılmaktadır. Şekilde gösterilen RSA algoritması asal çarpanlara ayırmanın zorluğunu temel alan bir şifreleme algoritmasıdır. Kuantum bilgisayarlar asal çarpanlara ayırmayı çok kolaylaştıracağından, bunu kullanan şifreleme algoritmaları ile şifreleniş verileri kırmak çok daha kolay olacaktır. 2036 yılında öngörüler ile yapılan çalışmaların sonuçlarının çakıştığı ve şifrelerin artık kırılabileceği düşünülmektedir



Şekil 8: RSA şifreleme algoritmasının tahmini kırılma zamanı[14].

Kuantum bilgisayarların mevcut şifreleme altyapısını tehdit etmesine karşı kuantum sonrası (post-Quantum) olarak isimlendirilen algoritmalar da geliştirilmekte ve kullanılmaktadır. Bu çerçevede kullanılabilecek şifreleme ve özet (hash) alma algoritmalarından bazıları şu şekildedir.

- **Lattice Tabanlı Kriptografi:** Çok boyutlu kafes problemlerine dayanır. Kırılması kuantum bilgisayarlar için bile zordur.
- **Kod Tabanlı Kriptografi:** Hata düzeltme kodlarına dayanır, dayanıklılığı yıllardır bilinmektedir.
- **Hash Tabanlı Kriptografi:** Dijital imzalar için hash fonksiyonlarının zincirleme kullanımına dayanır.
- **Çok Değişkenli Polinomlar:** Çözülmesi zor polinomsal denklemlere dayanır. Özellikle dijital imzada kullanılır.

- **Isogeny Tabanlı Kriptografi:** Eliptik eğri isojenileri üzerinde çalışır. Anahtar değişimi için uygundur.

Sonuç olarak, henüz günümüz sorunlarına çözümde kullanılabilecek bilinen bir kuantum bilgisayar bulunmamaktadır. Çok uzun süredir üzerinde çalışılmaktadır ve devletler bu gücü elde etmek için çok büyük yatırımlar yapmaktadır. Geleceğin en büyük dijital silahlarından biri olacağına kesin gözüyle bakılmaktadır.

DÖNEM KONUSU

Hareket Eden, Değişen Hedefler -Automated Moving Target Defence (AMTD)

Kuruluşlarda günümüzde siber tehditlere karşı önlem almak, tespit etmek ve tespit edilen siber tehditlere müdahale etmek için Güvenlik Duvarı Sistemleri (Firewall), Güvenli E-Posta Ağ Geçidi (Mail GW), Zararlı Yazılım Tespit ve Engelleme Yazılımları (EDR,XDR), Yeni Nesil Antivirüs (NGAV), Ağ ve Sunucu Tabanlı Saldırı Tespit ve Engelleme Sistemleri (IDS, IPS), Veritabanı Güvenlik Duvarı Sistemleri (Database Firewall), SIEM uygulamaları, Zafiyet Tarama Uygulamaları, PAM Çözümleri, Kullanıcı ve Varlık Davranış Analiz Teknolojileri (UEBA), Kimlik ve Erişim Yönetim Sistemleri (IAM), Bal Küpü Sistemleri (Honey Pot), Kum Havuzu Sistemleri (SandBox), Güvenlik Orkestrasyonu Otomasyonu ve Yanıt Sistemleri (SOAR) gibi birçoğu yapay zekâ ile de desteklenmeye başlanan yazılımlar ve sistemler kullanılmaktadır.

Bunlara ek olarak siber güvenlik organizasyonlarının etkinliğinin güvence altına alınması ve sistemlerin yönetişimi ve bilgi güvenliği, için Güvenlik Operasyonları Merkezi/Takımları (SOC), Siber Olaylara Müdahale Ekipleri (SOME), ISO 27001, Siber Güvenlik Olgunluk Modelleri, Bilgi Güvenliği Denetimleri de kullanılmaktadır.

Şekil 9: Yeteri kadar güvende miyiz?*

Peki, daha neler yapılabilir? Bu kadar teknoloji, insan gücü yeterli olmayacak mı? Automated Moving Target Defence bir sistem mi, uygulama mı? Ne işe yarıyor? Siber güvenlik için kullandığımız uygulama ve sistem envanterine bunu da ekleyecek miyiz? AMTD Şekil 9'da görüldüğü gibi, bu sistemlere kıyasla farklı bir bakış açısına sahiptir.

İşletmeler ve siber güvenlik için kullanılan sistemler ve uygulamalar gibi, yapay zekânın daha ulaşılabilir olmasıyla, bu teknoloji de siber saldırganlar tarafından kullanılmaktadır. Saldırganlar sistemlere daha kolay ve hızlıca sızmak için kullanacakları yöntemleri yapay zekâ ile belirleyebilmekte, uygulayacakları yöntemleri geliştirmek için zaman kazanmakta, bu yöntemler için küçük yazılımları, kod parçaları ve komutları daha hızlı üretebilmekteler.

Konuya detaylı girmeden önce, Otomatik Hareketli Hedef Savunması olarak çevirebileceğimiz Automated Moving Target Defence için AMTD kısaltmasını kullanacağımızı belirtelim.

AMTD uygulamalarının ve metotlarının çıkış noktası olarak dilimize Önleyici Siber Savunma olarak çevirebileceğimiz Preemptive Cyber Defense [15], kavramını gösterebiliriz. Preemptive Cyber Defense kavramını bir siber saldırının olmasını beklemeksizin, sistemlerde gerekli değişikliklerin ve konfigürasyonların yapılması olarak tanımlayabiliriz.

MTD temel düşüncesi, mükemmel güvenliğe ulaşılamayacağının varsayılmasıdır [16].

Otomatik ifadesinden de bu sürecin kendiliğinden tetiklenebilir olmasını, sistemin, uygulamanın bunun için yapılandırılmış olduğunu anlıyoruz.

Moving Target Defence (MTD) kavramını, siber saldırıların tespit edilmesi ve karşı harekete geçilmesinden çok uygulamaların ve sistemlerin çalıştığı ortamların, konfigürasyonlarının değiştirilerek saldırganlar için karmaşıklığı artırarak daha fazla çaba sarf etmelerini, zaman harcamalarını ve sistem içinde o ana kadar ilerledikleri ortamda, konfigürasyonun değişmesi ile kaydettikleri ilerlemenin boşa çıkarılmasını sağlamayı amaçlayan uygulamalar ve yöntemler olarak açıklayabiliriz.

AMTD, saldırganların uyguladıkları atakları etkisiz hâle getirmeyi, bozmayı hedefler. Bunun için uygulamalardaki, sistemlerdeki, ağdaki atak yüzeyinin saldırganlar tarafından erişilmesini engellemek için sistemlerin ve konfigürasyonların değiştirilmesi, taşınması, gizlenmesi, başka bir şekle getirilmesi metotlarını kullanır.

Uygulamalarda, işlemlerde, servislerde, ağda ve çalışan uygulamaların yaşam döngüsünde (runtime) ve altyapıda periyodik, bir siber tehdit durumunda ise belirli sürelerde ya da rasgele zamanlarda değişiklikler yapılır [17].

Değişiklikler sunucuların, bilgisayarların hafıza alanlarında, ağda, ağ iletişim protokollerinde, uygulamalarda, işletim sistemlerinde ve donanım konfigürasyonlarında gerçekleştirilir.

Uygulamalar İçin Kullanımı

Uygulamalar hafızaya yüklenirken, kullanılan AMTD ürünü dinamik olarak uygulamanın hafızadaki yapısını dönüştürerek (morp) gizlenmesini (conseal), anlaşılabilir olmamasını sağlar. Bu uygulamanın kullandığı diğer sistem kaynaklarını hafızada adreslerken de aynı yöntemi kullanır. Buna ek olarak sanki uygulama normal olarak hafızada yer alıyormuş gibi hafızadaki yapısını (memory structure) az bir kapasiteyle hafızada konumlandırır. Güvenilmeyen alan (untrusted), tuzak (trap) hâline gelen bu hafıza alanı AMTD uygulaması ile izlenir. Aldatma (deception), AMTD uygulamalarının kullandığı temel unsurlardan biridir. Bu alana olabilecek bir erişim, kötü niyetli kod olarak kabul edilerek asıl kaynaklara erişemez. Bu erişim AMTD tarafından yapılmadığından bir siber atak olarak değerlendirilir ve bu atak engellenir. Erişime ait tüm kayıtlar kapsamlı bir şekilde (evidence) oluşturulur ve araştırma, soruşturma için (investigation) diğer sistemlere aktarılabilir. Bir yanda da kullanılan AMTD üreticisinin yetenekleri dahilinde sistemde değişiklikler otomatik ya da elle tetiklenebilir. Bu değişiklikler ilgili uygulamanın, servisin başka bir ağa kaydırılması, başka sunucularda, konteynerlerde çalıştırılması, erişim parolalarının, SSL/TLS sertifikalarının, erişim doğrulama anahtarlarının değiştirilmesi olabilir.

EDR, XDR, NGAV uygulamaları imza tabanlı, davranış analizi, makine öğrenmesi, kum havuzu metot ve teknolojileriyle atakları tespit eder. AMTD yukarıda bahsedilen metodu kullanır ve aktiviteleri izlemek yerine eninde sonunda saldırganların erişeceği hafıza alanını gizler ve erişimleri izlemek için tuzak olarak yapısal bir alan oluşturur. Ayrıca EDR, XDR, NGAV uygulamalarını atlayarak hafızaya yapılan erişimler içinde koruma sağlanır.

Sistemler Üzerinde Kullanımı

Diğer bir örnekte TLS sertifikalarının sistemlerde 3-6 ayda bir, belki de ayda bir değiştirilmesi önümüzdeki zamanlarda gündeme gelecek bir başka konu. AMTD ile ne ilgisi var denecek olursa, kuantum bilgi işlemin ilerlemesiyle şifreleme anahtarların çok daha kısa sürelerde çözülme olasılığı ortaya çıktı. Bulut ve kuruluş BT sistemlerinde, API bağlantı noktalarında periyodik olarak sistem yöneticilerinin daha az etkileşimiyle sertifikaların değiştirilmesi de AMTD başlığı altında bir süreç olarak yer almaya başlamış durumda.

AMTD ile ilgili olarak belirtmekte fayda var; bir siber saldırının olmasına gerek olmadan da sistemde değişiklikler yapılabilir. Örnek olarak bir uygulamanın API ile eriştiği kaynaklarda her bir bağlantı için ya da belirli sürelerde veya rasgele sürelerde erişim için kullanılan kimlik doğrulama anahtarlarının değiştirilmesi verilebilir.

Gartner tarafından 25 Eylül 2024 tarihinde yayınlanan "Emerging Tech Impact Radar: Preemptive Cybersecurity" [18] raporunda AMTD için teknoloji, ürün geliştiren üreticilerin bazıları Agita Labs, Arms Cyber Defense, Canauri, Dispel, Dispersive Networks, HOPR, Morphisec olarak, 7 Ekim 2024 tarihinde yayınlanan "Emerging Tech: Build Preemptive Security Solutions to Improve Threat Detection (Part 1)" [19] raporunda da Arms Cyber, Cloudbrink, Hopr.co, HyperSphere Technologies, Jscrambler, Morphisec, Quarkslab, R6 Security olarak verilmektedir. Bu yazının çıkış noktasının da aslında bu "Gartner, Emerging Tech Impact Radar: Preemptive Cybersecurity 25 November 2024 - IDG00811452" raporunun olduğunu söylemeliyiz. Gartner raporlarında belirtilen üreticilerin web sitelerinde bu yazıda anlatılan AMTD metotları ve daha fazlası için bilgi bulabilirsiniz. Bu üreticilere ek olarak Sophos [20] da ürünlerinde AMTD yöntemlerini, teknolojilerini kullandığını web sitesinde belirtmektedir.

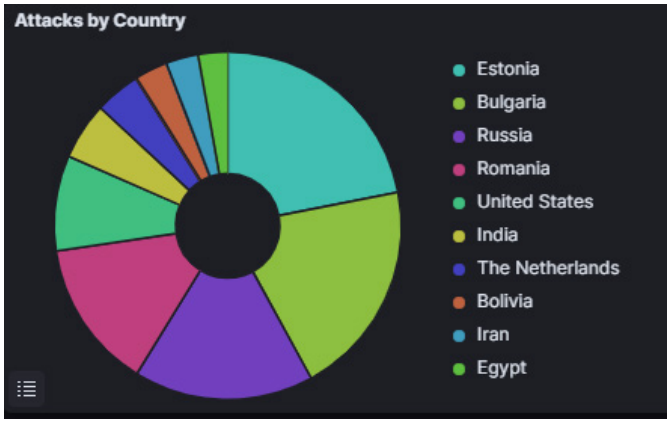
Sonuç olarak, önümüzdeki dönemde siber güvenlik çözümleri arasında Preemptive Cyber Defense -Önleyici Siber Savunma teknoloji ve yöntemleri de kullanılmaya başlayacak. Tehdit aktörleri tarafından yapay zekâ ve kuantum bilişim teknolojilerinin kullanılmasıyla bu teknolojilerin ve yöntemlerin giderek daha yaygınlaşacağını ve gelişeceğini söylemek yanlış olmayacaktır.

Honeypot Verileri

Bu rapor üç ay içinde Honeypot sensörlerimizden topladığımız verilerle oluşturulmuştur. Saldırıların en çok toplandığı ülkeler, portlar, en çok denenen kullanıcı adları ve parolalar, veriler azalan sırada listelenerek inceleme için sunulmuştur. Ekim, Kasım ve Aralık 2024 ayları boyunca honeypot sensörlerimize toplam 1.551.849 saldırı gelmiştir.

Şekil 10 ve Tablo 1'de gösterilen veriler incelendiğinde, en çok saldırı gelen 10 ülkenin ilk sırasında Estonya'nın (yüzde 21,91), sonrasında sırasıyla Bulgaristan (yüzde 20,10), Rusya (yüzde 16,77), Romanya (yüzde 13,91) ve ABD'nin (yüzde 8,87) yer aldığı görülmektedir.

Tablo 2'de de görüldüğü üzere en çok saldırı 5900 portuna gelmiştir. 5900 numaralı port VNC (Virtual Network



Şekil 10: Gelen saldırıların ülkelere göre dağılımı.

Saldırıların Geldiği Ülke	Saldırı Sayısı
Estonya	270.629
Bulgaristan	248.320
Rusya	207.119
Romanya	171.794
ABD	109.553
Hindistan	65.790
Hollanda	51.991
Bolivya	37.811
İran	37.150
Mısır	35.053

Tablo 1: En çok saldırı gelen 10 ülke ve saldırı sayıları.

Saldırılan Port	Saldırı Sayısı
5900	790.314
445	491.232
25	258.420
1080	2.797
21	1.830
5432	615
46869	7
45699	6
34535	5
35743	5

Tablo 2: En çok saldırı gelen portlar, bu portları kullanan servisler ve saldırı sayıları.

Denenen Parola	Deneme Sayısı
123	1.449
password	427
princess	390
canadian	336
12345678	189
Password	185
123456	183
business	147
postgres	131
sunshine	108

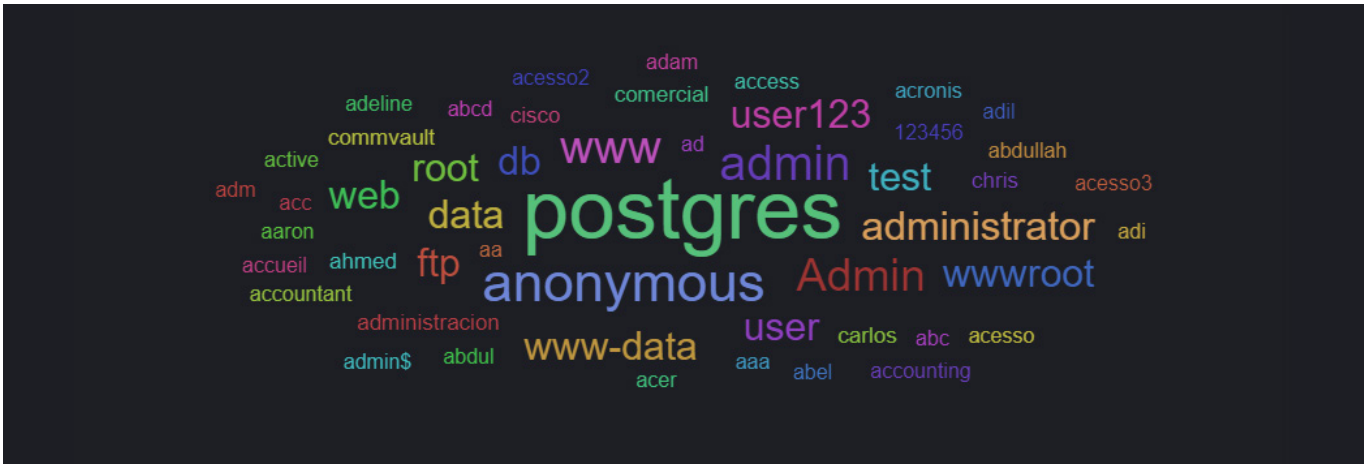
Tablo 3: SSH ve RDP honeypot'larımız üzerinde en çok denenen parolalar ve deneme sayıları.

Computing) yer almaktadır. VNC uzaktan masaüstü erişimi sağlayan bir protokol olduğundan, bu porta yapılan saldırılar sıklıkla yetkisiz erişim ve kötü amaçlı yazılımların bulaşması amacıyla gerçekleştirilmiş olabilir.

İkinci sırada 445 numaralı port yer almaktadır. 445 portunda sunucuların yazıcı ve paylaşılan dosyalar için kullandığı SMB servisi çalışmaktadır. Bu yüzden SMB servisinin diğer servislerden daha çok saldırı alması beklenen bir durum olarak kabul edilebilir.

Üçüncü sırada 25 numaralı port yer almaktadır. 25 portunda, e-posta iletimi için kullanılan Simple Mail Transfer Protocol (SMTP) hizmeti çalışmaktadır. Bu nedenle SMTP servisinin, spam gönderimi, adres sahteciliği ve hizmet reddi saldırıları gibi risklerden dolayı diğer servislerden daha çok saldırı alması beklenen bir durum olarak kabul edilebilir.

Şekil 11: Parola etiket bulutu.



Şekil 12: Kullanıcı adı etiket bulutu.

Denenen Kullanıcı Adı	Deneme Sayısı
postgres	617
anonymous	191
admin	124
www	116
Admin	103
administrator	64
data	62
db	62
ftp	62
test	62

Tablo 4: SSH ve RDP honeypot'larımız üzerinde en çok denenen kullanıcı adları ve deneme sayıları.

Denenen parolalar incelendiğinde, birçok yönetim arayüzünün standart olarak kullandığı parolalar olan 123, password, princess gibi terimler gözlemlenmektedir. Tablo 3'te bu en çok kullanılan şifreler gösterilmiştir. Bu parolaların, test veya deneme süreçleri tamamlanır tamamlanmaz değiştirilmesi ve karmaşık, 12-16 karakterli, özel karakter içeren parolalarla değiştirilmesi analistlerimiz tarafından tavsiye edilmektedir. Ayrıca kolay hatırlanması ve girilmesi için herhangi bir harf, özel karakter içermeden sadece sıralı sayılar ile oluşturulmuş parolalar kullanmaktan kaçınılmalıdır.

Denenen kullanıcı adları incelendiğinde, yeni kurulan sistemlerin sıklıkla kullandığı postgres, anonymous, admin gibi kullanıcı adlarının saldırganlar tarafından tercih edildiği görülmektedir. En çok kullanılan kullanıcı isimleri Tablo 4 ve Şekil 12'de gösterilmektedir. Kurulumu tamamlanan servislerin ve yönetim panellerinin kullanıcı adlarının en kısa zamanda değiştirilmesi ve kurulan sistemlerin kendi isimlerinin (örn. ubuntu, postgres, oracle, testuser) kullanılmaması tavsiye edilmektedir.

KAYNAKÇA:

[1] "Cofense," 25 Kasım 2024. [Çevrimiçi]. Available: https://cofense.com/blog/phish-swimming-in-the-opensea-the-opensea-phishing-threat.

[2] R. Iyengar, "CNN Business," 16 Temmuz 2020. [Çevrimiçi]. Available: https://edition.cnn.com/2020/07/15/tech/twitter-hack-elon-musk-bill-gates/index.html.

[3] "Mobil Uygulamadan Gelen Bildirim Ekran Görüntüsü (2024,12 9).," [Çevrimiçi].

[4] "Mobile App Security Top 12 Strategies to Secure Your Apps," 28 10 2024. [Çevrimiçi]. Available: https://www.uptech.team/blog/mobile-app-security.

[5] "Mobile App Security: Top 11 Threats and Ways to Mitigate Them," 29 07 2024. [Çevrimiçi]. Available: https://learn.g2.com/mobile-app-security.

[6] Nist. [Çevrimiçi]. Available: https://csrc.nist.gov/publications/detail/sp/800-145/final.

[7] "Computer Security Incident Handling Guide," NIST , 2024 12 25. [Çevrimiçi]. Available: https://csrc.nist.gov/pubs/sp/800/61/r2/final.

[8] "The quantum computer: it doesn't exist yet, but still we understand increasingly better what problems it can solve," 25 12 2024. [Çevrimiçi]. Available: https://www.universiteitleiden.nl/en/news/2024/04/the-quantum-computer-it-doesnt-exist-yet-but-still-we-underst.

[9] "Quantum technology historic public funding as of 2022, by country," 25 12 2024. [Çevrimiçi]. Available: https://www.statista.com/statistics/1319273/planned-public-funding-quantum-computing-country/.

[10] ""China's Origin Quantum Sets Roadmap, Sees 1,024-Qubit Device by 2025," 25 12 2024. [Çevrimiçi]. Available: https://thequantuminsider.com/2021/09/14/chinas-origin-quantum-sets-roadmap-sees-1024-qubit-device-by-2025/.

[11] "Introducing Willow, the next generation of quantum chips," 25 12 2024. [Çevrimiçi]. Available: https://quantumai.google/.

[12] "Türkiye'nin İlk Kuantum Bilgisayarı QuanT Tanıtım Töreni Gerçekleştirildi!," 25 12 2024. [Çevrimiçi]. Available: https://www.etu.edu.tr/tr/haber/turkiye-nin-ilk-kuantum-bilgisayari-quant-tanitim-toreni-gerceklestirildi.

[13] "The Qubit," 25 12 2024. [Çevrimiçi]. Available: https://qns.science/thequbit/.

[14] "Quantum-Proofing Our Secrets: Safeguarding Communication in the Age of Quantum Computers," [Çevrimiçi]. Available: https://www.swidch.com/resources/blogs/quantum-proofing-our-secrets-safeguarding-communication-in-the-age-of-quantum-computers .

[15] "Proactive Cyber Defence," 23 12 2024. [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/Proactive_cyber_defence.

[16] "csd-mtd Erişim Tarihi: 22.12.2024," 22 12 2024. [Çevrimiçi]. Available: https://www.dhs.gov/archive/science-and-technology/csd-mtd .

[17] "Gartner, Emerging Tech Impact Radar: Preemptive Cybersecurity," Gartner, 2024.

[18] Gartner, "Emerging Tech Impact Radar: Preemptive Cybersecurity IDG00811452," 2024.

[19] Gartner, "Emerging Tech: Build Preemptive Security Solutions to Improve Threat Detection (Part 1) ID G00810682," Gartner, 2024.

[20] "Pioneering Automated Moving Target Defense," Sophos, 25 12 2024. [Çevrimiçi]. Available: https://news.sophos.com/en-us/2023/10/19/pioneering-automated-moving-target-defense-amtd/.

* https://ussphoenix.substack.com/are-we-secure-yetthe-mirage-of-security

STM
www.stm.com.tr
/STMDefence

thinktech
STM Teknolojik Düşünce Merkezi
thinktech.stm.com.tr
/STMThinkTech