Basit CrackMe Uygulamalarının Tersine Mühendislik Analizine Karşı Web Uygulamalarında Güvenlik Teknikleri

Yönetici Özeti

Web uygulamalarının giderek artan karmaşıklığı, WebAssembly (Wasm) ve yapay zeka/makine öğrenimi (AI/ML) gibi yeni teknolojilerin benimsenmesiyle birlikte, tersine mühendislik ortamı önemli ölçüde dönüşmektedir.[1] Geleneksel çevre savunmaları artık yeterli olmamakta, çünkü saldırganlar giderek artan bir şekilde istemci tarafını ve API mantığını hedef almaktadır.[4] Bu durum, güvenlik stratejilerinde çok katmanlı, uyarlanabilir ve proaktif bir yaklaşıma olan ihtiyacı ortaya koymaktadır.[5] Bu rapor, 2025 yılı için web uygulamalarında tersine mühendislik girişimlerine karşı en etkili on güvenlik tekniğini derinlemesine inceleyerek, bu gelişen tehdit ortamında sağlam savunma mekanizmaları oluşturmanın önemini vurgulamaktadır.

Giriş: Web Uygulamalarında Tersine Mühendisliğin Gelişen Ortamı

Tersine mühendislik, tamamlanmış bir üründen bilgi çıkarmayı amaçlayan bir süreçtir; bu bilgi, ürünü geliştirmek, kopyalamak veya bileşenlerini analiz etmek için kullanılabilir.[6] Bu süreç, fikri mülkiyet hırsızlığı, rekabet analizi, güvenlik açıklarının tespiti, lisanslama mekanizmalarının aşılması (CrackMe uygulamalarında olduğu gibi) ve kötü amaçlı yazılım davranışlarının anlaşılması gibi çeşitli motivasyonlarla yönlendirilir.[6] Web uygulamaları bağlamında, bu hedefler gizli API'leri ortaya çıkarmak, istemci tarafı mantığını aşmak için kullanmak ve hassas verileri çıkarmak gibi ek boyutlar kazanır.[9]

Tersine mühendislik, hedeflenen programın türüne ve çalıştığı platforma bağlı olarak

farklı yaklaşımlar ve araçlar gerektirir.[11] Yaygın tersine mühendislik hedefleri ve kullanılan araçlar şunlardır:

- **İkili Çalıştırılabilir Dosyalar:** IDA Pro, Ghidra ve OllyDbg gibi araçlar, derlenmiş kodu analiz etmek için temel teşkil eder.[6] Bu disassembler'lar, makine kodunu insan tarafından okunabilir assembly diline veya sözde koda dönüştürerek statik analize (kodu çalıştırmadan inceleme) ve dinamik analize (çalışma zamanındaki davranışları gözlemleme) olanak tanır.[6] Özellikle x64dbg, Windows ortamlarında dinamik kötü amaçlı yazılım analizi için çok yönlü bir açık kaynak hata ayıklayıcıdır ve sistem çağrılarını ve kötü amaçlı yazılım davranışlarını gerçek zamanlı olarak incelemeye olanak tanır.[8]
- **WebAssembly (Wasm):** C, C++ ve Rust gibi üst düzey dillerden derlenen düşük seviyeli bir bayt kodu formatı olan Wasm, web'de performans açısından kritik ve güvenliğe duyarlı görevler için giderek daha fazla benimsenmektedir.[1] İkili formatı, JavaScript'e kıyasla okunmasını ve tersine mühendisliğini doğal olarak daha zor hale getirir.[14] Manuel yorumlama ve anlama, veri türü bilgilerini gizlemesi ve karmaşıklığı artırması nedeniyle zorlu ve hataya açık kalmaktadır.[1]
- **İstemci Tarafı JavaScript:** Kaynağı kolayca erişilebilir olduğu için tersine mühendisliği genellikle daha basit olsa da, JavaScript kodu genellikle amacını gizlemek için çeşitli gizleme işlemlerinden geçer.[15] Minifikasyon (genel bayt sayısını azaltma) yaygın olsa da, bir güvenlik önlemi olarak tasarlanmamıştır.[15] Tarayıcı geliştirici araçları (örneğin, Chrome DevTools), istemci tarafı JavaScript'i analiz etmek ve manipüle etmek için birincil araçlardır.[16]
- **API'ler:** Mitmproxy, Fiddler, Burp ve Postman gibi web proxy araçları, API işlevlerini, uç noktalarını, veri formatlarını ve kimlik doğrulama yöntemlerini tersine mühendislik yoluyla anlamak için HTTP trafiğini yakalamak ve analiz etmek için vazgeçilmezdir.[9]

"CrackMe" uygulamaları, tersine mühendisleri güvenlik mekanizmalarını (genellikle parola doğrulama veya lisans kontrolleri) aşmaya zorlamak için tasarlanmıştır.[11] Web bağlamında, bu, uygulamanın temel mantığının, Wasm'e derlenmiş veya yoğun şekilde gizlenmiş JavaScript olsa bile, gizli anahtarları veya kontrolleri ortaya çıkarabilecek analize karşı korunması gerektiği anlamına gelir. Kullanıcının C dilindeki parola doğrulama uygulamasını Ghidra ve x64dbg kullanarak analiz etme projesi, derlenmiş kod için sağlam savunmalara olan ihtiyacı doğrudan yansıtmaktadır; bu tür kodlar giderek artan bir şekilde WebAssembly aracılığıyla web ortamlarında dağıtılmaktadır.

2025'te Web Uygulaması Güvenliğini Şekillendiren Temel Eğilimler

Web uygulaması güvenliği ortamı, sürekli gelişen tehditler ve teknolojik ilerlemelerle dinamik bir değişim içindedir. 2025 yılına doğru, çeşitli temel eğilimler, tersine mühendislik saldırılarına karşı savunma yaklaşımlarını yeniden şekillendirmektedir.

Yapay Zeka ve Makine Öğreniminin Web Geliştirme ve Güvenliğine Entegrasyonu

Yapay zeka (AI) ve makine öğrenimi (ML) artık niş teknolojiler olmaktan çıkmış, web geliştirmede kişiselleştirme, otomasyon ve gelişmiş karar verme süreçleri sunarak ayrılmaz birer parça haline gelmiştir.[2] Güvenlik alanında ise, AI gerçek zamanlı tehdit tespiti, otomatik anomali tespiti, davranışsal biyometri ve olay yanıtı için kritik bir rol oynamaktadır.[5]

Bu teknolojinin güvenlik üzerindeki etkisi iki yönlüdür. Bir yandan, AI savunmaları önemli ölçüde güçlendirirken, diğer yandan "AI destekli siber saldırılar" ve düşmanca AI tekniklerinin yükselişiyle yeni saldırı vektörleri ortaya çıkmaktadır.[19] Örneğin, AI'nın gizlenmiş kod üretme veya güvenlik mekanizmalarını atlatma yeteneği, savunucuların AI'ya karşı AI kullanmasını gerektiren sürekli bir silahlanma yarışını tetiklemektedir.[20] Bu durum, güvenlik ekiplerinin yalnızca savunma için AI'ya yatırım yapmakla kalmayıp, aynı zamanda düşmanca AI tekniklerini anlamaları ve AI kırmızı takım testlerini güvenlik uygulamalarına entegre etmeleri gerektiğini göstermektedir.[19]

WebAssembly (Wasm) Benimsenmesinin Artması ve Yeni Bir Hedef Olarak Ortaya Çıkışı

Wasm, web'de performans açısından kritik ve güvenliğe duyarlı görevler için giderek daha fazla benimsenmektedir ve C, C++ ve Rust gibi üst düzey dillerden derlenmektedir.[1] Düşük seviyeli, ikili formatı, manuel yorumlamayı ve anlamayı zorlaştırır, veri türü bilgilerini gizler ve tersine mühendisler için karmaşıklığı artırır.[1]

Wasm'ın ikili doğası, JavaScript'e kıyasla okunmasını ve tersine mühendisliğini doğal olarak daha zor hale getiren bir gizleme katmanı sağlar.[14] Bu doğal gizleme, ölü kod ekleme, yazmaç değiştirme ve kontrol akışı değişiklikleri gibi açık gizleme teknikleriyle

birleştiğinde, Wasm'ın web uygulamalarında hassas mantığı gizlemek için tercih edilen bir ortam haline geldiğini göstermektedir.[21] Bu durum, savunucuların makine öğrenimi araçları [1] ve dinamik ikili enstrümantasyon [13] kullanımı da dahil olmak üzere özel Wasm tersine mühendislik ve gizleme kaldırma yetenekleri geliştirmesini gerektirmektedir.

İstemci Tarafı Güvenliğinin Birincil Savaş Alanı Olarak Odak Noktası Olması

Modern web uygulamaları, JavaScript yürütme, DOM manipülasyonu ve doğrudan kullanıcının tarayıcısından başlatılan API etkileşimleri gibi istemci tarafı işlemlere giderek daha fazla güvenmekte, kritik iş mantığını kullanıcının tarayıcısına aktarmaktadır.[4] Bu değişim, Tarayıcıda Ortadaki Adam (MitB) saldırıları, oturum ele geçirme, kimlik bilgisi toplama ve API kötüye kullanımı gibi tehditlere karşı savunmasız "gizli bir saldırı yüzeyi" oluşturmaktadır.[4]

Geleneksel güvenlik modeli büyük ölçüde sunucu tarafı doğrulamaya dayanırken, istemci tarafı kontroller ikincil olarak kabul edilirdi.[23] Ancak, daha kritik mantık istemciye taşındıkça, saldırganlar bu yeni alanı istismar etmektedir. "İstemciyi kontrol edemezsiniz, bu nedenle ona güvenemezsiniz" [23] ifadesi temel olarak doğru kalsa da, mimari daha fazla istemci tarafı mantığı gerektirmektedir. Bu durum, istemciye

güvenmek için değil, istemci tarafı ortamını kötü amaçlı manipülasyondan *korumak* için istemci tarafı güvenlik önlemlerinin kritik hale geldiği bir paradoks yaratmaktadır. Kuruluşlar, geleneksel güvenlik duvarlarının tespit edemediği istemci tarafı saldırıları tespit etmek ve önlemek için gizleme, kurcalama önleme ve gerçek zamanlı çalışma zamanı izleme gibi dinamik istemci koruma önlemlerini uygulamalıdır.[4]

OWASP Top 10 Güvenlik Açıklarının Süregelen Önemi

OWASP Top 10, web uygulamaları için en kritik güvenlik risklerini belirlemek ve azaltmak için kritik bir standart olmaya devam etmektedir.[24] 2025 güncellemesi, Kırık Erişim Kontrolü, Kriptografik Başarısızlıklar, Enjeksiyon ve Güvenli Olmayan Tasarım gibi kategorilerin önemini koruyarak "kök nedenlere" odaklanmayı sürdürmektedir.[24]

Tersine mühendislik, Güvenli Olmayan Tasarım, Kriptografik Başarısızlıklar ve Güvenlik Yanlış Yapılandırması gibi birçok OWASP Top 10 güvenlik açığını daha kolay *keşfetmek*

veya *istismar etmek* için kullanılabilir.[24] Örneğin, sabit kodlanmış kriptografik anahtarlar veya güvenli olmayan API uç noktaları, tersine mühendisliğin hızla ortaya çıkarabileceği yaygın yanlış yapılandırmalardır.[4] Bu durum, tersine mühendisliğe karşı güçlü bir savunmanın, temel OWASP Top 10 güvenlik açıklarını ele almayı da içermesi gerektiğini göstermektedir, çünkü bunlar genellikle ilk dayanak noktasını sağlar veya tersine mühendislik önleme tekniklerinin korumaya çalıştığı "sırları" ortaya çıkarır. Bu nedenle, proaktif güvenlik açığı yönetimi ve tehdit modellemesi temel öneme sahiptir.[24]

Web Uygulaması CrackMe Tersine Mühendisliğine Karşı En İyi 10 Güvenlik Tekniği (2025)

2025 yılına girerken, web uygulamalarını CrackMe tersine mühendislik girişimlerine karşı korumak, çok yönlü ve sürekli gelişen bir strateji gerektirmektedir. Aşağıdaki teknikler, bu tehditlere karşı en etkili savunma hatlarını temsil etmektedir.

Tablo 1: Web Uygulaması CrackMe Tersine Mühendisliğine Karşı En İyi 10 Güvenlik Tekniğine Genel Bakış (2025)

Teknik	Tersine Mühendisliğe Karşı Temel Fayda	2025 İçin Önemi	Anahtar Referanslar
1. Gelişmiş Kod Gizleme ve Dinamik Mutasyon	Kodu okunması ve anlaşılması zor hale getirir.	AI odaklı gizleme kaldırmayı engeller, Wasm koruması.	[4]
2. Sağlam Kurcalama Önleyici Mekanizmalar	İstemci tarafı kodunun bütünlüğünü sağlar ve manipülasyonu engeller.	İstemci tarafı saldırı yüzeyini korur, hibrit bütünlük modeli.	[4]
3. Sofistike Hata Ayıklama Önleyici Teknikler	Hata ayıklayıcıların tespitini engeller ve analizi zorlaştırır.	Gelişmiş kötü amaçlı yazılım analizi kaçınmasını engeller.	[17]

4. Yapay Zeka Destekli Tehdit Tespiti ve Yanıtı	Anormallikleri, kötü amaçlı davranışları gerçek zamanlı olarak tespit eder.	AI destekli saldırılara karşı koyar, otomatik savunma sağlar.	[5]
5. Gelişmiş API Güvenliği ve Mantık Koruması	API uç noktalarını ve temel iş mantığını tersine mühendislikten korur.	API'lerin yeni saldırı yüzeyi olmasına karşı koyar.	[4]
6. Kuantum Sonrası Kriptografi (PQC) ve Gelişmiş Kriptografik Yaklaşımlar	Verilerin ve mantığın şifreli kalmasını sağlayarak bilgi çıkarımını engeller.	Kuantum tehditlerine karşı koyar, gizliliği artırır.	[5]
7. Donanım Destekli Güvenlik ve Cihaz Doğrulaması	Yazılım manipülasyonunu zorlaştırarak bütünlük kontrollerini donanıma bağlar.	Köklenmiş cihazlara ve kurcalamaya karşı direnci artırır.	[44]
8. Çalışma Zamanı Uygulama Kendi Kendini Koruma (RASP)	Uygulama içinde saldırıları gerçek zamanlı olarak tespit eder ve engeller.	Dış savunmaları aşan saldırılara karşı koruma sağlar.	[4]
9. Güvenli DevOps (DevSecOps) Entegrasyonu	Güvenlik açıklarının geliştirme yaşam döngüsünün erken aşamalarında ele alınmasını sağlar.	Saldırı yüzeyini azaltır, proaktif güvenlik sağlar.	[5]
10. Proaktif Güvenlik Açığı Yönetimi ve Tehdit Modellemesi	Potansiyel saldırı vektörlerini belirler ve önleyici kontrolleri yönlendirir.	Stratejik risk yönetimini sağlar, AI odaklı saldırıları öngörür.	[7]

1. Gelişmiş Kod Gizleme ve Dinamik Mutasyon

Kod gizleme, bir yazılımın işlevselliğini ve davranışını korurken, kodunu, verilerini ve kaynaklarını anlaşılması, analiz edilmesi veya değiştirilmesi zor bir forma dönüştürme sürecidir.[28] Bu, tersine mühendisliği daha zaman alıcı, maliyetli ve hataya açık hale

getirmeyi amaçlar.[28]

- **Teknikler:**
  - **İsim Gizleme:** Değişken, fonksiyon ve sınıf adlarının anlamsız veya yanıltıcı tanımlayıcılara (örneğin, password yerine a9f3b2c1) dönüştürülmesi, kodun anlaşılmasını zorlaştırır.[15]
  - **Kontrol Akışı Gizleme:** Kod yapısını ve sırasını değiştirerek (örneğin, basit if-else ifadelerini karmaşık switch-case yapılarıyla değiştirme, goto ifadeleri ekleme) yürütme akışını gizler.[21]
  - **Veri Gizleme:** Verilerin (örneğin, dize değişmezleri) kodlanması, şifrelenmesi veya parçalanması, doğrudan erişimi veya değiştirmeyi engeller.[15] Bu, tersine mühendisi uyarabilecek değişmez dize değerlerini gizlemeyi içerir.[15]
  - **Kod Gizleme:** Obscure veya alışılmadık özellikler, operatörler veya ifadeler kullanarak (örneğin, bit düzeyinde işlemler, üçlü operatörler) kodun sözdizimini ve semantiğini değiştirme, kod tanımayı zorlaştırır.[28]
  - **Dinamik Kod Mutasyonu:** Kodun çalışma zamanında yapısını veya davranışını değiştiren teknikler, statik analizi daha az etkili hale getirir ve anlamak için dinamik analiz gerektirir.[4]
- **WebAssembly ve JavaScript Gizleme Gelişmeleri:**
  - **WebAssembly (Wasm):** Wasm'ın ikili formatı, JavaScript'e kıyasla okunması, tehditlerin tespiti veya tersine mühendisliği açısından doğal olarak gizleme sağlar.[14] Ölü kod ekleme, yazmaç değiştirme ve kontrol akışı değişiklikleri gibi gizleme teknikleri Wasm'a uygulanmaktadır.[21]
  - **JavaScript:** Minifikasyon (bayt sayısını azaltma, değişkenleri kısaltma, ifadeleri dönüştürme) yaygın olsa da, öncelikli olarak bir güvenlik önlemi değildir.[15] Amacı gizlemek için özel gizleme araçları ve manuel teknikler kullanılır.[15]
  - **Gizleme için Yapay Zeka:** Büyük Dil Modelleri (LLM'ler), ölü kod, yazmaç değiştirme ve kontrol akışı değişiklikleri dahil olmak üzere gizlenmiş assembly kodu üretme yetenekleri açısından değerlendirilmekte olup, bu alandaki potansiyellerini göstermektedir.[21]
- Statik Analiz Araçlarına (Ghidra) ve Tarayıcı Geliştirici Araçlarına Karşı Etkililik: Gizleme, tersine mühendisliği "daha zaman alıcı, maliyetli ve hataya açık" hale getirir.28 Ghidra için, hata ayıklama bilgisi olmayan (strip edilmiş) ikililer kontrol akışını gösterecek ancak değişken/fonksiyon adlarını tanımakta zorlanacak, analizi zorlaştıracaktır.46 Tarayıcı geliştirici araçları için, gizleme, yürütmeyi takip etmeyi ve mantığı anlamayı zorlaştırır, ancak mantık nihayetinde ortaya çıkarsa istemci tarafı kontrolleri atlamak için araçlar hala kullanılabilir.17

Kod gizleme ve gizleme kaldırma arasındaki sürekli bir silahlanma yarışı mevcuttur.

Gizleme kodu anlaşılması zor hale getirmeyi amaçlarken [28], araştırmalar araçların kodu gizleme kaldırma yeteneğine sahip olduğunu [28] ve AI'nın gizlenmiş JavaScript'teki kötü amaçlı niyeti

tespit etmek için kullanıldığını [29] ve tarayıcılar arası polimorfik gizlemeyi tespit etmek için kullanıldığını [30] göstermektedir. Bu, gizleme tekniklerinin gelişmiş gizleme kaldırma ve tespit yöntemleriyle karşılandığı sürekli bir evrimi işaret etmektedir. Bu durum, 2025'te etkili gizlemenin, otomatik analiz ve gizleme kaldırma araçlarının önünde kalmak için sürekli güncellenen çok katmanlı, dinamik ve potansiyel olarak AI destekli teknikler gerektireceği anlamına gelir. Basit, statik gizlemeye güvenmek yetersiz kalacaktır.

Tablo 2: Gizleme Teknikleri ve Tersine Mühendislik Araçları Üzerindeki Etkileri

Gizleme Türü	Açıklama	Ghidra/IDA Pro Üzerindeki Etki (Statik Analiz)	x64dbg Üzerindeki Etki (Dinamik Analiz)	Tarayıcı Geliştirici Araçları Üzerindeki Etki (İstemci Tarafı JS)	2025'teki Etkililik
İsim Gizleme	Değişken, fonksiyon ve sınıf adlarını anlamsız/yanıltıcı hale getirir.	Sembolleri gizler, kodun amacını anlamayı zorlaştırır.	Değişken/fonksiyon adlarını gizler, bellek takibini zorlaştırır.	Değişken/fonksiyon adlarını okunaksız hale getirir, kod takibini zorlaştırır.	AI destekli sembolik yürütme ile kısmen aşılabilir; sürekli değişim gerekir.
Kontrol Akışı Gizleme	Kodun yürütme sırasını değiştirir (örneğin, atlamalar, karmaşık koşullar).	Kontrol akışı grafiğini karmaşıklaştırır, fonksiyon sınırlarını belirsizleştirir.	Adım adım yürütmeyi ve mantık takibini zorlaştırır, atlama noktalarını gizler.	Kodun doğrusal akışını bozar, hata ayıklayıcıda takibi zorlaştırır.	AI tabanlı kontrol akışı analizi gelişiyor; dinamik mutasyon gereklidir.
Veri	Dizeleri,	Önemli	Bellek	Hassas	Otomatik

Gizleme	sabitleri veya diğer verileri kodlar, şifreler veya böler.	dizeleri/değerleri gizler, veri referanslarını zorlaştırır.	dökümlerinde ve yazmaçlarda veri tanımayı engeller.	verileri okunaksız hale getirir, çalışma zamanı manipülasyonunu zorlaştırır.	şifre çözme araçları ve AI ile kısmen aşılabilir; karmaşık algoritmalar önemlidir.
Kod Gizleme	Obscure sözdizimi, operatörler veya ifadeler kullanır.	Kod kalıplarını tanımayı zorlaştırır, otomatik imza tabanlı analizi engeller.	Kodun davranışını anlamayı zorlaştırır, manuel analizi artırır.	Okunabilirliği azaltır, yerleşik hata ayıklayıcıların otomatik analizini engeller.	AI tabanlı kod anlama gelişiyor; sürekli yeni teknikler keşfedilmelidir.
Dinamik Mutasyon	Kodun çalışma zamanında değişmesini sağlar.	Statik analizi büyük ölçüde etkisiz hale getirir, kodun sürekli değişen yapısını yakalayamaz.	Çalışma zamanı davranışını yakalamayı zorlaştırır, dinamik enstrümantasyon gerektirir.	Çalışma zamanı manipülasyonunu ve hata ayıklamayı son derece karmaşık hale getirir.	En etkili gizleme türlerinden biridir, AI tabanlı dinamik analiz gerektirir.

2. Sağlam Kurcalama Önleyici Mekanizmalar

Kurcalama önleme, bir uygulamanın yetkisiz değişikliklere veya manipülasyonlara karşı direncini ifade eder. Web uygulamalarında, bu özellikle istemci tarafı kodunun ve API etkileşimlerinin bütünlüğünü korumak için kritik öneme sahiptir.

- **İstemci Tarafı Bütünlük Kontrolleri:**
  - Bu kontroller, istemci tarafı kodunun (JavaScript, Wasm) değiştirilmediğini doğrulamayı içerir.[23]
  - **Alt Kaynak Bütünlüğü (SRI):** Üçüncü taraf sunuculardan (CDN'ler gibi) alınan kaynakların kurcalanmadığını sağlamak için bir mekanizmadır.[32]
  - **Dinamik Çalışma Zamanı İzleme:** İstemci ortamına hafif güvenlik ajanları enjekte ederek komut dosyası enjeksiyonunu, DOM kurcalamasını ve kötü

amaçlı ağ çağrılarını tespit eder.[4]

- **Sunucu Tarafı Doğrulama ve Sürekli Oturum Doğrulaması:**
  - "İstemciye asla güvenmemelisiniz ve sunucuda kontrolleriniz OLMALIDIR" temel bir prensiptir.[23] Tüm girdiler ve işlemler için sunucu tarafı doğrulama kritik öneme sahiptir.[31]
  - Sürekli oturum doğrulama mekanizmaları, oturum ele geçirme ve jeton hırsızlığını tespit etmek ve önlemek için kullanıcı eylemlerini, cihaz duruşunu ve çevresel bütünlüğü izler.[4]
- **DOM Manipülasyonu ve Komut Dosyası Enjeksiyonuna Karşı Koruma:**
  - Tarayıcıda Ortadaki Adam (MitB) saldırılarını ve DOM enjeksiyonu yoluyla kimlik bilgisi toplamasını azaltma.[4]
  - İçerik Güvenlik Politikaları (CSP'ler), istenmeyen komut dosyalarını ve davranışları engellemek için temel otomatik teknik kontrollerdir.[32]

İstemci tarafı mantığına artan bağımlılık [4] ve istemcinin doğasında var olan güvenilmezliği [23] göz önüne alındığında, hibrit bir bütünlük modelinin zorunluluğu ortaya çıkmaktadır. İstemci tarafı bütünlük kontrolleri [32] yüzeysel kurcalamayı caydırabilse de, nihayetinde atlatılabilirler.[17] Bu durum, kritik işlemler için sunucu tarafı doğrulamanın [23] vazgeçilmezliğini koruduğu anlamına gelir. Gerçek ilerleme, dinamik istemci tarafı izlemenin [4] sağlam sunucu tarafı doğrulama ve sürekli oturum doğrulaması [4] ile birleşerek daha dirençli, hibrit bir bütünlük modeli oluşturmasında yatmaktadır. Kuruluşlar, istemci tarafı kurcalama önlemenin ilk savunma hattı ve erken uyarı sistemi olarak işlev gördüğü, ancak kritik güvenlik kararlarının ve nihai bütünlük kontrollerinin her zaman sunucuda onaylandığı çok katmanlı bir yaklaşım uygulamalıdır. Bu, istemci tarafı güvenlik ajanları ile arka uç izleme sistemleri arasında sorunsuz entegrasyonu gerektirir.

3. Sofistike Hata Ayıklama Önleyici Teknikler

Hata ayıklama önleyici teknikler, tersine mühendislik süreçlerini engellemek için bir uygulamanın hata ayıklayıcıların varlığını tespit etmesini ve buna göre davranmasını içerir.

- **Hata Ayıklayıcıların Tespiti ve Kaçınması (x64dbg, Tarayıcı Geliştirici Araçları):**
  - Hata ayıklama önleyici teknikler, bir hata ayıklayıcının varlığını tespit etmeyi ve analizi engellemek için program davranışını değiştirmeyi veya yürütmeyi

sonlandırmayı amaçlar.[33]
    - Yaygın teknikler arasında hata ayıklayıcı varlığını kontrol etme, atlama hedeflerini değiştirme veya kontrolleri diğer iş parçacıklarına/süreçlerine aktarma yer alır.[35]
    - Tarayıcı geliştirici araçları için, JavaScript hata ayıklama önleme, denetimi tespit etme ve kaçınma tekniklerini içerir, ancak bunlar genellikle çalışma zamanında yamalanarak veya koşullar değiştirilerek atlatılabilir.[17]
- **Sanallaştırılmış Ortamlarda Gelişmiş Şeffaflık için HyperDbg gibi Teknikler:**
    - Modern kötü amaçlı yazılımlar, analizi atlatmak için giderek daha sofistike hata ayıklama önleyici ve sanallaştırma önleyici teknikler kullanmaktadır.[33]
    - Açık kaynaklı, hipervizör düzeyinde bir hata ayıklayıcı olan HyperDbg, kendi hipervizör ayak izlerini ve temel iç içe sanallaştırma yığınlarının (VMware, Hyper-V, KVM) ayak izlerini azaltmak için mekanizmalar sunar.[33] Bu, araştırmacıların kaçamak kötü amaçlı yazılımları gizlice incelemesine yardımcı olur.[33]
- **Hata Ayıklama Önleyici Kontrolleri Gizleme:**
    - Hata ayıklama önlemeyi atlatmayı daha zor hale getirmek için, geliştiriciler hata ayıklama önleyici kontrollerin kendilerini gizlemelidir (örneğin, bir hata ayıklayıcı tespit edildiğinde atlama hedeflerini farklı şekilde hesaplama).[35] Bu, aksi takdirde kontrol fonksiyonunu basitçe yamalayacak olan tersine mühendisler için çabayı artırır.[35]

Temel sorun, hata ayıklama önleyici tekniklerin genellikle "hata ayıklayıcı kontrol fonksiyonunu yamalayarak kolayca atlatılabilir" olmasıdır.[35] Bu durum, daha sofistike, gizlenmiş hata ayıklama önleyici kontrollere [35] ve buna karşılık, varlıklarını gizleyebilen HyperDbg gibi daha gelişmiş hata ayıklayıcılara olan ihtiyacı doğurmaktadır.[33] Bu, basit kontrollerden karmaşık, çok katmanlı kaçınma ve tespit yöntemlerine doğru ilerleyen açık bir "kedi-fare" oyunudur. Web uygulaması geliştiricileri, özellikle hassas istemci tarafı mantığına sahip olanlar (örneğin, Wasm tabanlı CrackMe'ler), dinamik ve gizlenmiş hata ayıklama önleyici önlemleri entegre etmelidir. Aynı zamanda, güvenlik araştırmacıları ve kırmızı ekipler, bu savunmaları aşmak için gelişmiş hata ayıklama araçları ve teknikleri kullanmalıdır.

4. Yapay Zeka Destekli Tehdit Tespiti ve Yanıtı

Yapay zeka (AI), web uygulaması güvenliğinde tehditleri tespit etme ve yanıtlama

biçimini dönüştürmektedir.

- **Anomali Tespiti, Davranışsal Biyometri ve Otomatik Olay Yanıtı için AI/ML Kullanımı:**
  - AI destekli güvenlik araçları, gerçek zamanlı tehdit tespiti ve azaltma için vazgeçilmezdir.[5]
  - Bu, otomatik anomali tespiti (şüpheli etkinlikler için kullanıcı davranışını izleme) ve davranışsal biyometriyi (dolandırıcılığı tespit etmek için yazma düzenlerini, fare hareketlerini analiz etme) içerir.[5]
  - AI, saldırılara verilen yanıtları otomatikleştirerek azaltma süresini kısaltabilir.[5]
- **Kötü Amaçlı JavaScript Niyetini ve Polimorfik Gizlemeyi Tespit Etmek için AI Modelleri:**
  - Cloudflare'ın Page Shield'ı, gizlenmiş olsa bile JavaScript'in arkasındaki kötü amaçlı niyeti tespit etmek için yeni bir AI modeli kullanır.[29] Bu model, tehditleri (Magecart, kripto madenciliği, kötü amaçlı yazılım) sınıflandırmak için sözdizimi ağaçları üzerinde Grafik Sinir Ağları (GNN'ler) kullanır.[29]
  - Polimorfik JavaScript gizlemesini farklı tarayıcı ortamlarında tespit etmek için transfer öğrenme yaklaşımları kullanılmaktadır, çünkü gizlenmiş kod her teslimatta değişir ve tarayıcı motorları farklılık gösterir.[30]
- **Düşmanca Yapay Zeka ve Yapay Zeka Destekli Siber Saldırılara Karşı Koyma:**
  - "Ajan AI"nın yükselişi, düşmanca AI ile geleneksel siber saldırılar arasındaki çizgileri bulanıklaştırarak, kimlik avı ve veri sızıntısı gibi yeni hedeflenmiş tehditlere yol açmaktadır.[19]
  - Saldırganlar, saldırıların hızını ve ölçeğini artırmak için AI'yı kullanmakta ve yarı otonom saldırı modelleri ortaya çıkabilmektedir.[19]
  - Kuruluşların, AI güvenlik açıklarını proaktif olarak test etmek için düşmanca ML'yi kırmızı takım çalışmalarına entegre etmeleri gerekmektedir.[19] Bu, istem enjeksiyonu ve diğer AI'ya özgü güvenlik açıklarını test etmeyi içerir.[20]

AI, sadece tehditleri *tespit etmek* için bir araç değil; aynı zamanda gizlenmiş kod *üretmek* [21] ve sofistike saldırılar

gerçekleştirmek için de kullanılmaktadır.[19] Bu durum, tersine mühendisliğe karşı gelecekteki savunmanın AI'nın yeteneklerinden büyük ölçüde etkileneceğini göstermektedir. Savunucular, karmaşık gizlenmiş kodu analiz etmek için AI'yı kullanmak zorundadır [1], saldırganlar ise daha dirençli gizleme ve analiz önleyici teknikler oluşturmak için AI'yı kullanacaktır. Bu, bir taraftaki ilerlemelerin diğer taraftaki yeniliği tetiklediği bir geri bildirim döngüsü yaratır. 2025'te etkili tersine mühendislik önleme, kuruluşların yalnızca AI destekli güvenlik çözümleri dağıtmasını değil, aynı zamanda AI güvenliği konusunda dahili uzmanlık geliştirmesini, AI kırmızı takım testlerini ve AI'nın

hem saldırgan hem de savunmacı tersine mühendislik için nasıl kullanılabileceğini anlamasını gerektirecektir.

5. Gelişmiş API Güvenliği ve Mantık Koruması

API'ler, modern web uygulamalarının bel kemiğidir ve tersine mühendislik için önemli bir hedef oluşturur. Bu nedenle, API güvenliğini güçlendirmek, uygulamanın genel güvenliğini artırmak için kritik öneme sahiptir.

- **İstemci Tarafı API Etkileşimlerini Güvenli Hale Getirme:**
  - API tersine mühendisliği, ağ trafiğini izleyerek API çağrılarını ve yanıtlarını yakalamayı, uç noktaları, veri formatlarını ve kimlik doğrulama yöntemlerini ortaya çıkarmayı içerir.[9]
  - Savunmalar, jetonları ve oturum tanımlayıcılarını cihaz parmak izlerine ve bağlamsal faktörlere bağlayarak API etkileşimlerini güçlendirmeyi içerir.[4]
  - OAuth 2.1 ve OpenID Connect gibi güvenli yetkilendirme protokollerinin kullanılması önemlidir.[5]
- **Çalışma Zamanı API İzleme ve Otomatik API Güvenlik Testi:**
  - AI tabanlı araçlar, kötü amaçlı API çağrılarını tespit etmek için çalışma zamanı API izlemesi için kullanılır.[5]
  - Otomatik API güvenlik testi aracılığıyla API güvenlik açıklarının dağıtımdan önce sürekli değerlendirilmesi önemlidir.[5]
  - API hız sınırlama ve kısıtlama, DDoS saldırılarını ve kötüye kullanımı önler.[5]
- **API Mantığını Tersine Mühendislikten Koruma:**
  - API'leri tersine mühendislik yoluyla anlamak, belgelenmemiş API'leri anlamaya, eski sistemlerle entegrasyona ve uygulama performansını iyileştirmeye yardımcı olabilir.[9]
  - Ancak, bu aynı zamanda temel iş mantığını da ortaya çıkarabilir.[10]
  - Odak noktası istemci tarafı tersine mühendislik olsa da, API mantığı, kritik iş mantığının sunucu tarafında kalmasını ve istemciye yönelik API çağrıları aracılığıyla ifşa edilmemesini sağlayarak korunabilir.

İş mantığının istemci tarafına kayması [4], istemci tarafının API'lerle yoğun bir şekilde etkileşim kurduğu anlamına gelir. Bu istemci tarafı etkileşimlerinin tersine mühendisliği [9], temel API yapısını ve dolayısıyla iş mantığını ortaya çıkarabilir. Bu durum, API'leri yalnızca veri çıkarma için değil, aynı zamanda temel uygulama işlevselliğini anlama ve potansiyel olarak çoğaltma veya istismar etme için de birincil tersine mühendislik

hedefi haline getirir. API güvenliği, temel kimlik doğrulama ve hız sınırlamanın ötesine geçmelidir. Sağlam girdi doğrulaması [31], güvenli iletişim (mTLS[5]) ve hassas iş mantığının istemciye yönelik API'ler aracılığıyla ifşa edilmesini en aza indiren dikkatli bir tasarım içermelidir.

6. Kuantum Sonrası Kriptografi (PQC) ve Gelişmiş Kriptografik Yaklaşımlar

Kuantum bilgisayarların ortaya çıkmasıyla birlikte, geleneksel şifreleme yöntemleri risk altındadır, bu da kuantum sonrası kriptografiyi (PQC) web uygulaması güvenliği için kritik hale getirmektedir.[5]

- **Uygulamadaki Veriler için Uçtan Uca Şifreleme (E2EE) ve Homomorfik Şifreleme (FHE):**
  - **Tam Homomorfik Şifreleme (FHE):** Verilerin şifresini çözmeden doğrudan şifreli veriler üzerinde hesaplamalara olanak tanır, hesaplamayı veri erişiminden ayırır.[36] Bu, gizliliği koruyan AI dağıtımı ve güvenilmeyen üçüncü taraf altyapılarla çalışmak için faydalıdır.[39]
  - **Uygulamalar:** Analiz için şifreli hasta geçmişleri, müşteri verileri üzerinde birleşik arama, dolandırıcılık tespiti için güvenli tahmine dayalı modelleme.[39]
- **Gizliliği Koruyan Hesaplamalar için Çok Taraflı Hesaplama (MPC) ve Sıfır Bilgi Kanıtları (ZKP):**
  - **Çok Taraflı Hesaplama (MPC):** Birden fazla tarafın, kendi özel girdilerini birbirlerine ifşa etmeden bir fonksiyonu işbirliği içinde hesaplamasına olanak tanır.[36] SMPC pazarı, finans, sağlık ve AI'daki taleplerle önemli ölçüde büyümektedir.[40]
  - **Sıfır Bilgi Kanıtları (ZKP):** Bir tarafın, bilgiyi kendisi ifşa etmeden diğer tarafa bilgiye sahip olduğunu kanıtlamasına olanak tanır.[37]
  - **Uygulamalar:** Blok zincirlerinde gizliliği koruyan işlemler (örneğin, Zcash), doğum tarihini ifşa etmeden yaş kanıtlama, sağlık/finans sektöründe güvenli veri paylaşımı.[37]
- **2025'teki Standardizasyon Çalışmaları ve Pratik Uygulamalar:**
  - FHE, 2025'te AI ve blok zincirinde benimsenmeye başlayarak "ana akım olma eşiğinde"dir.[36]
  - NIST, 2025'te ZKP'leri standartlaştırmayı hedeflemektedir.[42]
  - Hesaplama açısından yoğun ve dağıtımı karmaşık olsa da, bu teknikler olgunlaşmakta ve erken ticari uygulamalar görmektedir.[36]

Geleneksel tersine mühendislik, kodu *anlaşılması zor* hale getirmeye odaklanırken, gelişmiş kriptografi (FHE, MPC, ZKP) bir paradigma değişimi sunar: bir hesaplama veya doğrulama yapmak için verileri veya mantığı düz metin olarak anlamayı *gereksiz* hale getirir. Bu, tersine mühendisliğin temel amacı olan "bilgi çıkarma"ya doğrudan karşı koyar.[6] Veriler veya mantık şifreli kalırken veya sırları ifşa etmeden işlenebiliyorsa, belirli saldırı vektörleri için tersine mühendisliğin değeri önemli ölçüde azalır. Web uygulamaları, özellikle hassas verileri veya karmaşık algoritmaları işleyenler, temel mantığı ve verileri, temel kod tersine mühendisliği yapılsa bile ifşa olmaktan korumak için bu gelişmiş kriptografik teknikleri giderek daha fazla kullanacaktır. Bu, özel kriptografik uzmanlığa ve altyapıya önemli yatırım gerektirecektir.

Tablo 3: Gelişmiş Kriptografi Teknikleri ve Web Uygulaması Uygulamaları (2025)

Teknik	Temel Özellik	Web Uygulaması Kullanım Durumları (2025)	Olgunluk/Zorluklar (2025)
Tam Homomorfik Şifreleme (FHE)	Şifreli veriler üzerinde şifre çözmeden hesaplama yapma.	Gizliliği koruyan AI dağıtımı, şifreli hasta geçmişleri üzerinde analiz, perakende sektöründe birleşik arama.	Ana akım olma eşiğinde, AI ve blok zincirinde erken benimseme. Hesaplama açısından yoğun, dağıtımı karmaşık.
Çok Taraflı Hesaplama (MPC)	Birden fazla tarafın özel girdilerini ifşa etmeden işbirliği içinde hesaplama yapması.	Finansal hizmetlerde gizliliği koruyan veri analizi, sağlık sektöründe güvenli hasta veri analizi, AI ve makine öğrenimi modellerinde güvenli işbirliği.	Pazarda önemli büyüme, finans, sağlık ve AI'dan gelen talep.
Sıfır Bilgi Kanıtları (ZKP)	Bir tarafın, bilgiyi ifşa etmeden bilgiye sahip olduğunu kanıtlaması.	Blok zincirlerinde gizliliği koruyan işlemler (örneğin, Zcash), yaş kanıtlama (doğum tarihini ifşa	NIST tarafından standartlaştırılması hedefleniyor. Yüksek hesaplama gereksinimleri,

		etmeden), AML/KYC uyumluluğu.	uzmanlık gerektirir.

7. Donanım Destekli Güvenlik ve Cihaz Doğrulaması

Donanım destekli güvenlik, web uygulamalarını tersine mühendislik ve kurcalamaya karşı korumak için temel bir katman sunar.

- **Bütünlük Kararları için Donanım Destekli Sinyallerden Yararlanma:**
  - Google Play'in güncellenmiş Play Integrity API'si, Android 13 ve üzeri cihazlarda bütünlük kararları için daha katı donanım destekli güvenlik sinyalleri uygulamaktadır.[44] Bu, köklenmiş cihazların veya özel ROM'ların güvenlik doğrulamalarını geçmesini önemli ölçüde zorlaştırmaktadır.[44]
  - Bu, "daha agresif cihaz parmak izi"ne doğru bir adımdır.[44]
- **Uygulama Mimarisi için Güvenli Tasarım İlkeleri:**
  - Yazılım mimarisi ve tasarımının erken aşamalarına güvenlik ilkelerinin yerleştirilmesi, güvenlik açıklarını erken aşamada azaltmak ve riskleri en aza indirmek için kritik öneme sahiptir.[45]
  - Bu, güvenliğin sonradan eklenen bir özellik değil, temel bir unsur olmasını sağlamayı içerir.[45]
- **Köklenmiş Cihazlar ve Özel ROM'lar için Çıkarımlar:**
  - Daha katı donanım destekli sinyaller, köklenmiş cihazlara veya özel ROM'lara sahip meşru güç kullanıcılarını olumsuz etkileyerek belirli uygulamalara erişimi potansiyel olarak engelleyebilir.[44]

Geçmişte, yazılım tabanlı kurcalama önleme ve hata ayıklama önleme, yazılım ortamının manipülasyonuyla atlatılabilirdi.[17] Donanım destekli sinyallere geçiş [44], güven sınırında temel bir değişimi temsil etmektedir. Bu, bütünlük kontrolünü daha değişmez ve ele geçirilmesi daha zor bir katmana bağlamaya çalışır, böylece yalnızca yazılım tabanlı tersine mühendisliğin bütünlük kontrollerini atlatmasını zorlaştırır. Mobil platformlarda dağıtılan veya istemci tarafı bütünlükten yararlanan web uygulamaları için donanım destekli güvenlik daha önemli bir savunma haline gelecektir. Tersine mühendisler, donanım düzeyinde atlatmalar veya daha sofistike emülasyon teknikleri gerektiren daha yüksek engellerle karşılaşacaklardır. Bu aynı zamanda kullanıcı kontrolü ve kendi cihazlarındaki gizlilik hakkında soruları da gündeme getirmektedir.

8. Çalışma Zamanı Uygulama Kendi Kendini Koruma (RASP)

Çalışma Zamanı Uygulama Kendi Kendini Koruma (RASP), bir uygulamanın kendi içinde saldırıları gerçek zamanlı olarak algılamasını ve engellemesini sağlayan yenilikçi bir güvenlik tekniğidir.

- **Saldırıları Gerçekleşirken Engellemek için Güvenliği Uygulama Çalışma Zamanına Entegre Etme:**
  - RASP, mobil uygulamalar için bir güvenlik geliştirmesidir.[5]
  - Güvenliği doğrudan uygulama çalışma zamanı ortamına entegre ederek çalışır, böylece saldırıları gerçek zamanlı olarak tespit edebilir ve engelleyebilir.[5]
  - Bu, geleneksel çevre savunmaları olan WAF'lardan farklıdır, çünkü RASP içeriden çalışır.[4]
- **WAF'ları ve Diğer Çevre Savunmalarını Tamamlama:**
  - RASP, uygulamayı içeriden koruyan bir "içten dışa" güvenlik yaklaşımı sağlar ve Web Uygulaması Güvenlik Duvarları (WAF'lar) gibi "dıştan içe" savunmaları tamamlar.[4]
  - WAF'ları atlayan saldırıları, örneğin tehlikeye atılmış istemci tarafı komut dosyalarından veya dahili güvenlik açıklarından kaynaklananları tespit edebilir ve önleyebilir.

Geleneksel güvenlik, ağ ve çevre savunmalarına (güvenlik duvarları, WAF'lar) odaklanırken [4], istemci tarafı mantığının yükselişi ve sofistike saldırılar, tehditlerin uygulamanın yürütme ortamı

içinde ortaya çıkabileceği veya kendini gösterebileceği anlamına gelir ve harici kontrolleri atlar. RASP, uygulamayı saldırılara karşı "kendi kendine farkında" hale getirerek bu durumu doğrudan ele alır.[5] Bu, güvenliğin yalnızca harici koruyucu katmanlara dayanmak yerine, uygulamanın

içine yerleştirilmesine doğru bir geçişi ifade eder. RASP, özellikle karmaşık istemci tarafı mantığına veya hassas arka uçlara sahip web uygulamalarını, tersine mühendislik tarafından kolaylaştırılanlar da dahil olmak üzere çalışma zamanı saldırılarından korumak için giderek daha kritik hale gelecektir. Uygulama katmanında kötü amaçlı davranışları doğrudan izleyerek ve bunlara tepki vererek kritik bir savunma katmanı ekler.

9. Güvenli DevOps (DevSecOps) Entegrasyonu

DevSecOps, güvenlik açıklarının erken aşamada ele alınmasını sağlamak için güvenliği geliştirme hattına entegre etmeyi içerir.

- **CI/CD İşlem Hatlarında Otomatik Güvenlik Testleri (SAST/DAST):**
  - Güvenliğin geliştirme hattına entegre edilmesi, güvenlik açıklarının erken aşamada ele alınmasını sağlar.[5]
  - **Statik Uygulama Güvenlik Testi (SAST):** Güvenlik açıklarını bulmak için kaynak kodunu veya ikilileri çalıştırmadan analiz eder.[5]
  - **Dinamik Uygulama Güvenlik Testi (DAST):** Kötü amaçlı girdileri simüle ederek çalışma zamanı sorunlarını belirlemek için uygulamaları çalışan durumda test eder.[5] Burp Suite ve Fortify WebInspect gibi araçlar önemli DAST araçlarıdır.[18]
- **Kod Olarak Altyapı (IaC) Güvenliği ve Sürekli Uyum:**
  - IaC güvenliği aracılığıyla bulut ortamlarındaki yanlış yapılandırmaları önleme.[5]
  - GDPR, CCPA ve ISO 27001 gibi düzenlemeler için otomatik kontroller.[5]
- **Geliştirme Yaşam Döngüsünün Erken Aşamalarına Güvenlik İlkelerini Yerleştirme:**
  - "Tasarımdan güvenli" olmak, güvenliği yazılım mimarisi ve tasarımının erken aşamalarından itibaren entegre etmek anlamına gelir, sonradan düşünmek yerine.[45] Bu, güvenlik açıklarını proaktif olarak azaltmaya yardımcı olur.[45]

Tersine mühendislik genellikle güvensiz tasarım veya yanlış yapılandırmalar nedeniyle var olan güvenlik açıklarını istismar eder.[24] Kuruluşlar, güvenliği DevSecOps hattının erken aşamalarına yerleştirerek [5], bu güvenlik açıklarının ilk etapta ortaya çıkmasını önleyebilir veya dağıtımdan önce yakalayabilir. Bu, savunma stratejisini reaktif yamalamadan proaktif önlemeye kaydırarak, uygulamanın istismar edilebilir kusurlar için tersine mühendisliğini doğal olarak daha zor hale getirir. DevSecOps, sadece bir eğilim değil, tersine mühendislerin erişebileceği saldırı yüzeyini azaltan temel bir değişimdir. Sağlam DevSecOps uygulamalarıyla oluşturulan uygulamalar, saldırganların tersine mühendislik yoluyla keşfetmesi ve istismar etmesi için daha az kolay hedef içerecektir.

10. Proaktif Güvenlik Açığı Yönetimi ve Tehdit Modellemesi

Proaktif güvenlik açığı yönetimi ve tehdit modellemesi, web uygulaması güvenliğinde stratejik bir katman oluşturur ve tersine mühendislik girişimlerine karşı savunmayı güçlendirir.

- **OWASP Top 10 Güvenlik Açıklarına Karşı Sürekli İzleme:**
  - En kritik web uygulaması güvenlik risklerinin düzenli olarak gözden geçirilmesi ve ele alınması.[24]
  - Bu, girdi doğrulaması, hata işleme ve güvenli API kullanımı gibi güvenli kodlama uygulamalarını içerir.[31]
- **Düzenli Güvenlik Denetimleri ve Sızma Testleri (AI Kırmızı Takım Testleri Dahil):**
  - Güvenlik açıklarını belirlemek için sızma testleri yapılması.[7]
  - AI güvenlik açıklarını proaktif olarak test etmek için düşmanca makine öğreniminin (ML) standart kırmızı takım çalışmalarına entegre edilmesi.[19] Bu, istem enjeksiyonu ve diğer AI'ya özgü güvenlik açıklarını test etmeyi içerir.[20]
- **Üçüncü Taraf Bileşenler için Yazılım Malzeme Listesi (SBOM) Tutma:**
  - Güvenlik açığı bulunan ve güncel olmayan üçüncü taraf SDK'lar ve geçişli bağımlılıkların oluşturduğu riskin ele alınması.[24] Bir SBOM, bu bileşenleri izlemeye ve yönetmeye yardımcı olur.[24]

Belirli tersine mühendislik önleme teknikleri kod ve çalışma zamanına odaklansa da, kapsamlı bir güvenlik açığı yönetimi programı [24] ve proaktif tehdit modellemesi [24] stratejik bir katman görevi görür. Uygulamanın saldırı yüzeyini, potansiyel tehditlerini ve bağımlılıklarını anlayarak [32], kuruluşlar bir tersine mühendisin sisteme nasıl yaklaşabileceğini tahmin edebilir ve hedeflenen savunmaları uygulayabilir. AI kırmızı takım testlerinin dahil edilmesi [19], tersine mühendislerin ortaya çıkarabileceği yeni AI ile ilgili saldırı vektörlerini özel olarak test ederek bu durumu daha da iyileştirir. 2025'te etkili tersine mühendislik önleme, teknik kontrollerin ötesine geçen, stratejik risk yönetimi, sürekli değerlendirme ve AI destekli saldırılar da dahil olmak üzere gelişen tehdit ortamının derinlemesine anlaşılmasını içeren bütünsel bir yaklaşım gerektirmektedir.

Sonuç ve Gelecek Görünümü

Web uygulamalarında CrackMe tersine mühendislik analizine karşı savunma, tek bir tekniğe dayalı olmaktan ziyade, çok katmanlı ve uyarlanabilir bir yaklaşım gerektirmektedir. Bu rapor, gelişmiş kod gizleme, sağlam kurcalama önleyici

mekanizmalar, sofistike hata ayıklama önleyici teknikler, yapay zeka destekli tehdit tespiti ve yanıtı, gelişmiş API güvenliği, kuantum sonrası kriptografi ve ileri kriptografik yaklaşımlar, donanım destekli güvenlik, çalışma zamanı uygulama kendi kendini koruma (RASP), güvenli DevOps (DevSecOps) entegrasyonu ve proaktif güvenlik açığı yönetimi ile tehdit modellemesini içeren on temel tekniği vurgulamıştır. Bu tekniklerin her biri, web uygulamalarının bütünlüğünü, gizliliğini ve kullanılabilirliğini korumak için kritik bir rol oynamaktadır. Özellikle istemci tarafı ve uygulama içi güvenliğe doğru bir kayma gözlemlenmektedir, çünkü geleneksel çevre savunmaları artık modern saldırıların karmaşıklığına karşı yeterli değildir.

2025 ve sonrası için, saldırganlar ve savunucular arasındaki sürekli silahlanma yarışı, özellikle AI'nın hem saldırı hem de savunma yeteneklerinde hızlanmasıyla devam edecektir. WebAssembly ve sunucusuz mimariler gibi yeni teknolojilerin artan benimsenmesi, yeni saldırı yüzeyleri ve savunma zorlukları ortaya çıkaracaktır. Bu durum, daha sağlam, standartlaştırılmış ve performanslı gizliliği artıran teknolojilere olan ihtiyacı daha da artıracaktır.

Bu dinamik ortamda, sürekli adaptasyon ve güvenlik topluluğunda işbirliği hayati önem taşımaktadır. Ortaya çıkan tehditler ve en iyi uygulamalar hakkında bilgi sahibi olmak [19], açık geliştirme ve güvenlik çözümleri ile standartların geliştirilmesinde işbirliği çabalarının değeri [38], web uygulaması güvenliğinin geleceğini şekillendirecektir. Kuruluşlar, bu teknikleri entegre ederek ve sürekli gelişen tehdit ortamına karşı çevik kalarak, CrackMe tarzı tersine mühendislik girişimlerine karşı dayanıklılıklarını önemli ölçüde artırabilirler.

Alıntılanan çalışmalar

1. Multi-modal Learning for WebAssembly Reverse Engineering - arXiv, erişim tarihi Haziran 1, 2025, https://arxiv.org/pdf/2404.03171?
2. 2025 Web Application Development Trends You Need to Know - Direct Impact Solutions, erişim tarihi Haziran 1, 2025, https://www.directimpactsolutions.com/en/2025-web-application-development-trends-you-need-to-know/
3. 16 Game-Changing Web Development Trends for 2025 - Atlas Softweb, erişim tarihi Haziran 1, 2025, https://www.atlassoftweb.com/blog/16-game-changing-web-development-trends-for-2025
4. The Hidden Attack Surface: Why Protecting the Client ... - Codesealer, erişim tarihi Haziran 1, 2025, https://codesealer.com/blog/the-hidden-attack-surface-why-protecting-the-client-side-is-critical-in-2025
5. Security Measures For Web And Mobile Apps In 2025 | Digital One ..., erişim tarihi

Haziran 1, 2025, https://digitaloneagency.com.au/security-measures-for-web-and-mobile-apps-in-2025/
6. The Reverse Engineering: Applications, Techniques, and Industry Impact - E-SPIN Group, erişim tarihi Haziran 1, 2025, https://www.e-spincorp.com/the-reverse-engineering-techniques-impact/
7. Reverse Engineering – Software Engineering | GeeksforGeeks, erişim tarihi Haziran 1, 2025, https://www.geeksforgeeks.org/software-engineering-reverse-engineering/
8. What Is Reverse Engineering in Cyber Security? [2025 Guide], erişim tarihi Haziran 1, 2025, https://www.stationx.net/what-is-reverse-engineering-in-cyber-security/
9. Best Practices of Reverse Engineering an API - Apriorit, erişim tarihi Haziran 1, 2025, https://www.apriorit.com/dev-blog/reverse-engineering-an-api-guide
10. How to Reverse Engineer APIs: The Benefits and Tools - DreamFactory Blog, erişim tarihi Haziran 1, 2025, https://blog.dreamfactory.com/reverse-engineering-apis-the-benefits-and-tools
11. Applied Reverse Engineering with IDA Pro - Infosec, erişim tarihi Haziran 1, 2025, https://www.infosecinstitute.com/resources/reverse-engineering/applied-reverse-engineering-ida-pro/
12. Reversing for mortals: Solving Yoire crackme average challenge - Fluid Attacks, erişim tarihi Haziran 1, 2025, https://fluidattacks.com/blog/reversing-mortals
13. WebAssembly Security Analysis & Reversing | FuzzingLabs 2022, erişim tarihi Haziran 1, 2025, https://fuzzinglabs.com/webassembly-security-training/
14. Next Level Smuggling with WebAssembly - LRQA, erişim tarihi Haziran 1, 2025, https://www.lrqa.com/en/cyber-labs/next-level-smuggling-with-webassembly/
15. Reverse Engineering JS by Example - F5, erişim tarihi Haziran 1, 2025, https://www.f5.com/company/blog/reverse-engineering-by-example-flatmap-stream-payload-a
16. Is the key to scraping reverse-engineering the JavaScript call stack? : r/webscraping - Reddit, erişim tarihi Haziran 1, 2025, https://www.reddit.com/r/webscraping/comments/1kfb3t9/is_the_key_to_scraping_reverseengineering_the/
17. Client-Side Encryption Bypass using DevTools Part-1 - Secjuice, erişim tarihi Haziran 1, 2025, https://www.secjuice.com/client-side-encryption-bypass/
18. Top Dynamic Application Security Testing (DAST) Tools in 2025 - Aikido, erişim tarihi Haziran 1, 2025, https://www.aikido.dev/blog/top-dynamic-application-security-testing-dast-tools
19. AI Security: 2025 Predictions & Recommendations - HiddenLayer, erişim tarihi Haziran 1, 2025, https://hiddenlayer.com/innovation-hub/ai-security-2025-predictions-recommendations/
20. OWASP Gen AI Incident & Exploit Round-up, Jan-Feb 2025, erişim tarihi Haziran 1, 2025, https://genai.owasp.org/2025/03/06/owasp-gen-ai-incident-exploit-round-up-jan-feb-2025/

21. arxiv.org, erişim tarihi Haziran 1, 2025, https://arxiv.org/pdf/2412.16135?
22. Black Hat Spring Trainings 2025 | Trainings Schedule, erişim tarihi Haziran 1, 2025, https://www.blackhat.com/tr-25/training/schedule/index.html
23. Is there a way to check client JavaScript integrity and / or prevent user modification as the server? - Stack Overflow, erişim tarihi Haziran 1, 2025, https://stackoverflow.com/questions/66716161/is-there-a-way-to-check-client-javascript-integrity-and-or-prevent-user-modifi
24. Understanding OWASP Top 10 Vulnerabilities in 2025 with Real ..., erişim tarihi Haziran 1, 2025, https://www.webasha.com/blog/understanding-owasp-top-10-vulnerabilities-with-real-world-examples-and-prevention-tips
25. The OWASP Top Ten 2025, erişim tarihi Haziran 1, 2025, https://www.owasptopten.org/
26. Top Mobile App Security Flaws from RSAC 2025 - Verimatrix, erişim tarihi Haziran 1, 2025, https://www.verimatrix.com/cybersecurity/cybersecurity-insights/rsac-2025-top-mobile-app-security-flaws-discussed/
27. Black Hat USA 2025 | Trainings Schedule, erişim tarihi Haziran 1, 2025, https://www.blackhat.com/us-25/training/schedule/
28. Software: How to Protect Your Software from Reverse Engineering and Hacking, erişim tarihi Haziran 1, 2025, https://fastercapital.com/content/Software--How-to-Protect-Your-Software-from-Reverse-Engineering-and-Hacking.html
29. How we train AI to uncover malicious JavaScript intent and make ..., erişim tarihi Haziran 1, 2025, https://blog.cloudflare.com/how-we-train-ai-to-uncover-malicious-javascript-intent-and-make-web-surfing-safer/
30. (PDF) Transfer-Learning Approaches to Detect Polymorphic ..., erişim tarihi Haziran 1, 2025, https://www.researchgate.net/publication/391827659_Transfer-Learning_Approaches_to_Detect_Polymorphic_JavaScript_Obfuscation_Across_Heterogeneous_Browsers
31. Top techniques for Web app security to use in 2025 - Cpluz, erişim tarihi Haziran 1, 2025, https://cpluz.com/blog/top-techniques-for-web-app-security-to-use-in-2025/
32. Industry News 2025 Guidelines for Operationalization of Web Client ..., erişim tarihi Haziran 1, 2025, https://www.isaca.org/resources/news-and-trends/industry-news/2025/guidelines-for-operationalization-of-web-client-runtime-security
33. Countering Anti-Debugging Techniques: Enhancing Transparency ..., erişim tarihi Haziran 1, 2025, https://2025.ecoop.org/details/debt-2025-papers/2/Countering-Anti-Debugging-Techniques-Enhancing-Transparency-in-Nested-Virtualization
34. Bypassing BlackMatter Anti-Debug With x64dbg [Patreon Unlocked] - YouTube, erişim tarihi Haziran 1, 2025, https://www.youtube.com/watch?v=HlEk7P_VZfg

35. Anti-debug technique discussion : r/AskReverseEngineering - Reddit, erişim tarihi Haziran 1, 2025, https://www.reddit.com/r/AskReverseEngineering/comments/1coic7g/antidebug_technique_discussion/
36. Advanced Cryptography: new approaches to data privacy, erişim tarihi Haziran 1, 2025, https://www.ncsc.gov.uk/blog-post/advanced-cryptography-new-approaches-to-data-privacy
37. Advanced Cryptography - NCSC.GOV.UK, erişim tarihi Haziran 1, 2025, https://www.ncsc.gov.uk/whitepaper/advanced-cryptography
38. Three Homomorphic Encryption Trends for 2025 - The Daily Hodl, erişim tarihi Haziran 1, 2025, https://dailyhodl.com/2025/01/14/three-homomorphic-encryption-trends-for-2025/
39. What is Homomorphic Encryption? | AI21 - AI21 Labs, erişim tarihi Haziran 1, 2025, https://www.ai21.com/glossary/homomorphic-encryption/
40. Growth Opportunities and Trends in the Secure Multiparty ..., erişim tarihi Haziran 1, 2025, https://blog.tbrc.info/2025/03/secure-multiparty-computation-market-demand-2/
41. Secure Multiparty Computation Market Report 2025 - Share, erişim tarihi Haziran 1, 2025, https://www.thebusinessresearchcompany.com/market-insights/secure-multiparty-computation-market-overview-2025
42. Why Zero-Knowledge Proofs Are the Future of Blockchain Security ..., erişim tarihi Haziran 1, 2025, https://builtin.com/articles/zero-knowledge-proof-blockchain-security
43. Rumble Fish | Blog | Top ZK Proof Development Companies to ..., erişim tarihi Haziran 1, 2025, https://www.rumblefish.dev/blog/post/top-zk-proof-dev-companies-2025/
44. Google Play's latest security change may break many Android apps ..., erişim tarihi Haziran 1, 2025, https://www.reddit.com/r/technology/comments/1kwxih8/google_plays_latest_security_change_may_break/
45. Application security at re:Inforce 2025 - AWS - Amazon.com, erişim tarihi Haziran 1, 2025, https://aws.amazon.com/blogs/security/application-security-at-reinforce-2025/
46. Is this obfuscation or just normal pseudocode? - ghidra - Reddit, erişim tarihi Haziran 1, 2025, https://www.reddit.com/r/ghidra/comments/1776j1j/is_this_obfuscation_or_just_normal_pseudocode/