

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI

**AKILLI ŞEBEKELER İÇİN GÜVENLİ YAZILIM
GELİŞTİRME TEMELLERİNE UYGUN
BİR UTM ARAYÜZ TASARIMI**

**G191210303 – Barış YILMAZ
G191210387 – Cemal ASLAN**

**Bölüm : BİLGİSAYAR MÜHENDİSLİĞİ
Danışman : Dr. Öğr. Üyesi Musa BALTA**

2022-2023 Güz Dönemi

ÖNSÖZ

Günümüzde teknolojinin önemi oldukça artmakta ve bununla birlikte teknolojik, uzaktan takibi yapılmayan ürün ve süreçler için çalışmalar artmaktadır. Bu tarz uzaktan erişim bulunan ve izleme sistemleri gerektiren gelişmeler beraberinde güvenlik endişelerini getirmektedir. Güvenlik endişelerini mümkün olduğunda yok etmek, minimuma indirmek projelerin ana fikri kadar mühimdir.

İÇİNDEKİLER

ÖNSÖZ	ii
İÇİNDEKİLER	iii
SİMGELER VE KISALTMALAR LİSTESİ.....	vi
ŞEKİLLER LİSTESİ	vii
ÖZET.....	viii
BÖLÜM 1. GİRİŞ.....	1
BÖLÜM 2. GÜVENLİ YAZILIM GELİŞTİRME KİLAZUZU	2
2.1. Güvenli Yazılım Geliştirme Yaşam Döngüsü	2
2.1.1. Güvenlik Risk Analizi	3
2.2. Olgunluk Modelleri	5
2.3. Yazılım Güvenliğini Sağlamak İçin Yöntemler.....	5
2.3.1. Static Analiz	6
2.3.2. Sembolik Koşturma.....	6
2.3.3. Dinamik Analiz	6
2.3.4. Negatif Test	6
2.3.5. Rastgele Test	6
2.3.6. Biçimsel Program Analizi	7
2.3.7. Model Doğrulayıcılar	7
2.3.8. Mantık Yöntemleri	7
2.3.9. Model Tabanlı Geliştirme.....	7
2.3.10. Doğrulanmış Araç ve Kod Depoları Kullanma	7
2.3.11. Kod Güçlendirme	8
2.3.12. İspat Taşıyan Kod	8
2.3.13. Sızma Testi	8
2.3.14. Fuzz Testi.....	8
2.4. Yazılım Çalışma Ortamı Güvenliğini Sağlayan Yöntemler	9

2.4.1.	Uygulama Konteyneri (Application Container).....	9
2.4.2.	Mikro Servis Mimarisi İle Yazılımların Geliştirilmesi	9
2.4.3.	Sanallaştırma Ortamlarının Kullanılması.....	9
2.5.	Yazılım Güvenliğini Sağlamada Diğer Yöntemler	10
2.5.1.	Programlama Dilleri ile Güvenliğinin Arttırılması	10
2.5.2.	Değişen Hedef Savunmasının Uygulanması.....	10
2.5.3.	Ağ ve Sistem Seviyesinde Önlem Alınması.....	10
2.6.	Yazılım Güvenliğinin İlkeleri	10
2.7.	Web Uygulama Güvenliği Kavramları	16
2.7.1.	Cross Site Scripting (XSS).....	16
2.7.1.1.	Yansıtılan XSS (Reflected XSS).....	16
2.7.1.2.	Depolanan XSS (Stored XSS)	17
2.7.1.3.	DOM Tabanlı XSS (DOM Based XSS).....	17
2.7.2.	SQL Enjeksiyonu (SQLi).....	18
BÖLÜM 3.	UTM ARAYÜZÜ	19
3.1.	UTM Arayüz Örnekleri	19
BÖLÜM 4.	MODÜLLER	24
4.1.	Log Yönetimi.....	24
4.1.1.	LogSign	24
4.1.2.	ELK	25
4.1.3.	IBM QRadar	26
4.2.	Zararlı Yazılım	26
4.2.1.	Virüs	27
4.2.2.	Solucan (Worm)	27
4.2.3.	Truva Atı (Trojan)	27
4.2.4.	Casus Yazılım (Spyware)	27
4.2.5.	Keylogger	27
4.2.6.	Fidye Yazılımı (Ransomware).....	27
4.2.7.	Reklam Yazılımı (Adware).....	28
4.2.8.	Tarayıcı Korsanları	28
4.2.9.	Rootkit.....	28
4.3.	Anomaly Detection.....	28

4.3.1.	Point Anomalies	29
4.3.2.	Contextual Anomalies	29
4.3.3.	Collective Anomalies	29
BÖLÜM 5.	TASARIM	30
BÖLÜM 6.	SONUÇLAR VE ÖNERİLER.....	33
KAYNAKLAR.....		34
ÖZGEÇMİŞ		35
BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI.....		36

SİMGELER VE KISALTMALAR LİSTESİ

UTM	: Unified Threat Management (Birleşik Tehdit Yönetimi)
OWASP	: Open Web Application Security Project
SAMM	: Software Assurance Maturity Model
BSIMM	: Building Security In Maturity Model
DSL	: Domain Specific Language
ASLR	: Address Space Layout Randomization
SQL	: Structured Query Language
BT	: Bilgi Teknolojileri
ISO	: International Organization for Standardization
DOM	: Document Object Model
API	: Application Programming Interface
URL	: Uniform Resource Locator
P2P	: Peer to Peer
VPN	: Virtual Private Network
SIEM	: Security Information and Event Management
ERD	: Entity-Relationship Diagram

ŞEKİLLER LİSTESİ

Şekil 2.1. Yazılım Güvenlik Gereksinimleri Tanımlama Adımları	3
Şekil 2.2. Yazılım Varlıklarını Değerlendirme Matrisi.....	4
Şekil 2.3. Tehdit Profili Değerlendirme Matrisi	4
Şekil 2.4. OWASP SAMM Olgunluk Modeli Bileşenleri	5
Şekil 2.5. BSIMM Etkinlik Alanları ve Faaliyetleri	5
Şekil 2.6. Reflected XSS	16
Şekil 2.7. Stored XSS.....	17
Şekil 2.8. DOM Based XSS	17
Şekil 3.1. UTM Yapılandırması	19
Şekil 3.2. ManageEngine Log360	20
Şekil 3.3. WatchGuard Firebox.....	20
Şekil 3.4. Exosphere.....	21
Şekil 3.5. AlienVault.....	22
Şekil 3.6. Comodo Dome Firewall.....	22
Şekil 3.7. Security Management Summary Dashboard by Carole Fennelly	23
Şekil 3.8. STMBugShield	23
Şekil 4.1. LogSign.....	25
Şekil 4.2. ELK	25
Şekil 4.3. IBM QRadar.....	26
Şekil 5.1. Login Ekranı	30
Şekil 5.2. Arayüz Görünümü	31
Şekil 5.3. Arayüz Görünümü Devamı.....	31
Şekil 5.4. ERD.....	32

ÖZET

Anahtar kelimeler: Akıllı Sayaç, Web Uygulama, UTM Dashboard, Güvenli Yazılım

Akıllı sayaçlar kullandıkları haberleşme teknolojileriyle beraber uzaktan anlık takip, kontrol ve izleme, esneklik, güvenilirlik, insan etkisi ile oluşabilecek hatanın ortadan kaldırılması, anlık veri transferi şeklinde sıralanabilecek yenilikler sunmaktadır. Bu yeniliklerin beraberinde uzaktan izleme sistemi geliştirilmesi ve bu geliştirilecek sistemin güvenliğinin maksimum seviyede olması elzemdir.

Bu tasarım çalışması ile Güvenli Yazılım Geliştirme Kılavuzlarına uygun olarak akıllı sayacın uzaktan kontrolünün sağlanabildiği, yalnızca yetkili kişiler tarafından sisteme giriş sağlanabilen ve Anomali Tespiti, Zararlı Yazılım Tespiti, Log Sistemi'nin bulunduğu bir UTM Arayüzü web uygulaması geliştirilerek takibe imkan sağlanacaktır.

BÖLÜM 1. GİRİŞ

Günümüzde içerisinde internet erişimi bulunan sistemlerde tasarım, erişilebilirlik, ana fikri kadar sistemin güvenliği, erişimi bulunmayan iyi veya kötü niyetli kişilerin sisteme erişememesi de önemlidir. Bu tasarım ile birlikte uzaktan erişim ile izleme yapılacak olan akıllı sayaç için güvenli yazılım geliştirme prosedürlerine uygun olarak bir ASP.NET Core 7.0 web uygulaması geliştirilmesi sağlanacaktır.

Yazılımın tehdit veya saldırısı altındayken işlevlerini doğru bir şekilde devam ettirmesi yazılımın güvenlidir ve bu bağlamda güvenliği sağlayabilmek için çeşitli organizasyonlar tarafından güvenli yazılım geliştirmenin standartları belirlenmiştir. Geliştirilecek her yazılımın bu standartlara bağlı olarak geliştirilmesi yazılımın güvenliği için önemdir. Bu tasarımda da bu standartlara bağlı olarak bir uygulama geliştirilmesi sağlanacaktır.

Bu tasarım kitapçığında güvenli yazılım geliştirme standartlarıyla ilgili özet bilgiler ve bu bilgilerin akabinde yapılabilecek işlemlerin bir listesi, UTM arayüzünün amacı, arayüzde kullanılacak modüllerin tanımları ve örnekleri, benzer UTM arayüzleri, bir kısmı tamamlanmış arayüzün görselleri ve tamamlanana kadar geçirilecek süreç ile ilgili bilgiler bulunacaktır.

BÖLÜM 2. GÜVENLİ YAZILIM GELİŞTİRME KİLAVUZU

Yazılımın, saldırısı veya tehdit altındayken işlevlerini doğru bir şekilde yerine getirmeye devam edecek şekilde korunmasına yazılım güvenliği denir. Bir yazılımın güvenliği yazılımı ortaya çıkarmak kadar mühimdir. Yazılım güvenliği faaliyetlerinin amacı tüm bilgi güvenliği saldırılara karşı daha dirençli ve hatasız çalışan yazılım üretmektir. Yazılım geliştirirken proje boyunca güvenli yazılım geliştirme kurallarının nasıl ve ne zaman uygulanacağıının belirlenmesi, güvenli yazılım geliştirme yaşam döngüsü faaliyetlerinin izlenmesi ve denetimi ve yazılım güvenliğinin sürekliliğinin sağlanması önem arz etmektedir. Bu sürecin yürütülmemesi durumunda saldırırlara maruz kalabilir ve yazılımın tamamen durdurulması ya da doğru çalışmasının engellenmesi, istenmeyen amaca hizmet etmesi veya bilgi sizıntılarına engel olunamaz [1].

Tasarımı gizli tutarak güvenlik yaklaşımı (Security through obscurity), yeni nesil araç ve yöntemlerle (tersine mühendislik, kod analizi araçları vb.) artık geçerliliğini yitirmiştir. Güvenlik tasarımının karşı tarafça bilindiği varsayılarak güvenlik analizlerinin yapılması gerekmektedir. Bu bağlamda, uygulamanın tehdit modeli tanımlanmalı, güvenlik tasarım doğrulama yöntemleri (tehdit modelleme ve risk analizi, biçimsel yöntemler, tasarım gözden geçirme vb.) projelerde standart olarak uygulanmalıdır.

2.1. Güvenli Yazılım Geliştirme Yaşam Döngüsü

Güvenli bir yazılımın tasarımı, gerçeklenmesi, yapılandırılması, kurulması ve desteklenmesi, yazılımın birçok saldırıyla karşıın doğru bir şekilde çalışmaya devam edecek, karşı koyamadığı saldırıların etkisini sınırlamak için gerekli önlemleri içinde barındıracak ve kısa sürede normal çalışmaya devam edecek şekilde olabilmesini sağlayacak bir süreçte hayata geçirilir. Yazılım ve çevresindeki ortam arasındaki etkileşimlerden dolayı ortaya çıkabilecek zafiyetleri en aza indirmek için, geçmiş zafiyetlerin incelenmesi, dış inceleme, sizma (penetrasyon) testi ve olay müdahalesi

gibi önlemler uygulanır. Ayrıca geliştiriciler güvenli yazılım geliştirme hususunda eğitilmeli ve güvenlik gereksinimlerinin tanımlamaları da yapılmalıdır [2].



Şekil 2.1. Yazılım Güvenlik Gereksinimleri Tanımlama Adımları

2.1.1. Güvenlik Risk Analizi

Yazılım geliştirme sürecinde bulunan varlık türlerine örnek olarak şunlar verilebilir: Bilgi varlıkları – İş Kuralları – Servisler veya metodlar – Yazılım kodu – Firmaya özgü formüller – Şifreleme yöntem ve anahtarları – Veritabanları – Kişiler veya kişilerin sahip olduğu belirli bilgi ve yetenekler – Hesap bilgileri ve hesapla ilişkilendirilmiş fonlar – İş kayıtları.

Varlıkların tespit edilmesinde Veri Akış Şemalarının (Data-Flow Diagram) oluşturulması yönetimi uygulanabilir. Bu yöntemde süreçler, harici öğeler, veri depoları, veri akışları ve güven sınırları belirlenir. Varlıklar tespit edildikten sonra Varlık Değerlendirme (Asset Valuation) adımı gerçekleştirilir. Bu adımda matris, Delphi yöntemi, Lineer değerlendirme (ISO 27005) gibi yöntemler izlenebilir.

Maddi değer?	Düşük	Orta	Yüksek
İş sürecine etkisi?			
Düşük	(Genel) Bilgi varlıkları	İş kayıtları	Hesap bilgileri
Orta	İş Kuralları	Yazılım Kodu	Kişiler Müşteri veritabanları
Yüksek	Kullanıcı bilgileri veritabanı	Yazılım Servisleri Şifreleme yöntem ve anahtarları	Firmaya özgü formüller

Şekil 2.2. Yazılım Varlıkları Değerlendirme Matrisi

Daha sonra tehdit profili değerlendirme analizi yapılır.

Kaynak	Düşük	Orta	Yüksek
Yetenek/Teknik Zorluk			
Düşük	Meraklı çocuk	Siber savaşçı	Siber terörist
Orta	Çevrimiçi sosyal hacker	Siber suçlu	Kötü niyetli geliştirici/operatör
Yüksek	İç tehdit	Hacker organizasyonları	Siber casusluk

Şekil 2.3. Tehdit Profili Değerlendirme Matrisi

Son olarak ise buradan, etki ve olasılık değerleri riski hesaplamak için kullanılabilir.

Bu amaçla aşağıdaki formüller örnek olarak kullanılabilir.

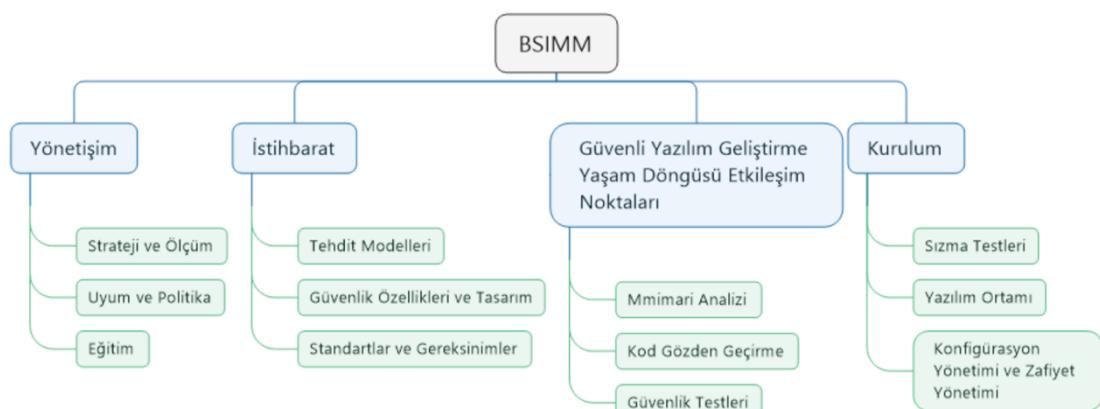
- Olasılık = Zayıflık * Tehdit derecesi
- Etki = MAX (Varlığın kuruma olan değeri, Saldırının dolaylı etkileri)
- Risk = Olasılık * Etki

2.2. Olgunluk Modelleri

Güvenli yazılım geliştirme olgunluk modelleri bir kuruluşun yazılım güvenliği pratiklerini ne derecede uyguladığını ortaya koyan derecelendirmeye dayalı ölçüm modelidir. OWASP tarafından ortaya konulmuş olan SAMM modeli ve BSIMM modeli bulunmaktadır ve bunlar aşağıdaki şekillerde özetlenmiştir.



Şekil 2.4. OWASP SAMM Olgunluk Modeli Bileşenleri



Şekil 2.5. BSIMM Etkinlik Alanları ve Faaliyetleri

2.3. Yazılım Güvenliğini Sağlamak İçin Yöntemler

Statik analiz ve sembolik koşturma yöntemleri için yazılımin çalıştırılan halini yansıtan bir modeli veya benzetimini oluşturarak, bilinen yazılım hata türleri bu model üzerinde aranır.

2.3.1. Statik Analiz

Bir yazılımın belli başlı özelliklerini onu çalıştırmadan analiz etmeyi amaçlayan yöntemlerden biridir. Statik analiz ile yazılımları karşılaştırırken hangi tür yazılım hatalarını çözebildikleri harici kaynaklardan doğrulanmalıdır. Örneğin Carnegie Mellon Üniversitesi'nin <https://www.securecoding.cert.org> sitesinde C/C++, Java gibi dillere ilişkin yazılım kodlama hataları ve bu hataların hangi araçlarla ne oranda bulunabildiğine ilişkin detaylı incelemeler mevcuttur. Statik analizde Boon ve Find Security Bugs gibi araçlar kullanılabilir.

2.3.2. Sembolik Koşturma

Kodun derilerindeki gizli hataların bulunması için kullanılır. Bu yöntemde yazılımın kodu üzerinde bir soyutlama yapılarak tüm olası çalışmalar modellenir. Yazılım testi kullanılarak sadece tanımlanan akış kapsamındaki hatalar bulunabilir.

2.3.3. Dinamik Analiz

Uygulamadaki açıklıkları tespit etmek için çalışan yazılım üzerinde ve çalışma ortamında gerçekleştirilen analiz araç ve yöntemleridir. Dinamik analiz için kullanılan uygulamalara örnek olarak Nmap, Wireshark, sqlmap verilebilir.

2.3.4. Negatif Test

Hatalı veya güvenlik kurallarına uymayan durumların ve istisnaların yazılım tarafından kontrol edildiğinin test edilmesine denir. Örneğin; en az 8 karakterlik bir parola gerektiren bir sayfanın 7 karakterli parola ile test edilmesi ve sonuçta yazılımın bu hususta hata mesajı vermesi.

2.3.5. Rastgele Test

Bir yazılıma geçersiz, beklenmeyen veya rastgele veri girdisinde bulunarak istenmeyen davranışta bulunup bulunmayacağı tespit edilmesine denir.

2.3.6. Biçimsel Program Analizi

Statik analizlerin matematik ve mantık temelli olarak gerçekleştirilmesine denir.

2.3.7. Model Doğrulayıcılar

Yazılımda mevcudiyeti tespit edilmesi istenen gizlilik, kimlik doğrulama, erişim denetimi, bütünlük gibi güvenlik özelliklerinin bir yazılımda mevcut olup olmadığıının doğrulanması için kullanılan yöntemdir.

2.3.8. Mantık Yöntemleri

Kodun içerisinde yapılan varsayımların ve beklenen sonuçların mantıksal ifadelerle tanımlanmasından sonra otomatik doğrulayıcı araçlar yardımıyla beklenen sonuçların doğrulanması yöntemidir.

2.3.9. Model Tabanlı Geliştirme

Yazılımın güvenlik davranışını uygulama alanına özel üst seviye bir dille DSL tanımlanarak kaynak kodun büyük bir kısmının veya tamamının oluşturulan modelden üretilmesi yöntemidir. Daha çok gömülü sistemlerde uygulanan bir yöntemdir. Kullanıcı arayüzü yoğun yazılımlar için bu yöntemin kullanımı uygun değildir.

2.3.10. Doğrulanmış Araç ve Kod Depoları Kullanma

Yazılımcılar genellikle genel amaçlı işlemler için internette paylaşılan kod parçacıklarını kullanmaktadır ve çoğunlukla da bu kod parçacıkları herhangi bir güvenlik testine veya gözden geçirmeye tabi tutulmadan yazılımın kod deposuna eklenmektedir. Dolayısıyla güvenilir olduğundan emin olmadığımız kütüphaneleri ve kod parçacıklarını kullanmamamız gereklidir. Doğrulanmış kütüphanelerin oluşturulma çabalarına bir örnek olarak, OWASP ESAPI projesi verilebilir.

2.3.11. Kod Güçlendirme

Önceden geliştirilmiş bir yazılımın güvenlik bakış açısı ile kaynak kodlarının, yapılandırma dosyalarının, derleyicisinin, dinamik kütüphanelerin ve özel amaçlı diğer kütüphanelerin biçimsel yöntemlerle doğrulanması ve güçlendirilmesidir.

2.3.12. İspat Taşıyan Kod

Uygulamanın çalıştırılabilir kodunu takip ederek biçimsel yöntemlerle analiz ederek kodun güvenli şekilde çalışıp çalışmayacağına karar veren yöntemdir.

2.3.13. Sızma Testi

Yöntem olarak kara kutu, gri kutu ve beyaz kutu yaklaşımları kullanılmaktadır. Beyaz kutu yaklaşımda kaynak kod dahil sistem hakkında tüm bilgilere sahipken, gri kutu yaklaşımda ise kapsam hakkında sınırlı bilgi ile sızma testi gerçekleştirilir. Kara kutu yaklaşımda test edilen sistem hakkında hiçbir ön bilgi bulunmadığı kabul edilmektedir ve ayrıca testi gerçekleştiren ekibin kaynak koda erişimi bulunmamaktadır.

Sızma testinde öncelikle keşif yapılır ve bu aşamada pasif ve aktif bilgi toplama aşamaları bulunur. Keşif aşamasında Cain & Abel, Nmap, Scapy, Wireshark gibi programlar kullanılabilir. Keşif aşamasından sonra ise zafiyet taraması yapılır, burada keşiften toplanan bilgilerden yararlanılarak keşifte bir zafiyet olasılığı bulunmuşsa o zafiyet sömürülmeye çalışılır. Bu aşamada zafiyet testi için Metasploit kullanılabilir.

2.3.14. Fuzz Testi

Fuzz Testi ile uygulamanın girdi noktalarına beklenmedik rasgele veriler sistematik olarak gönderilerek uygulamanın ürettiği hatalar gözlemlenir. Öncelikle hedef yazılım üzerinde testi gerçekleştirecek girdi noktalarının tespiti edilir. Bu tespit yapıldıktan sonra istenen verinin formatına uygun olarak uygun veri kümesi

oluşturulur ve uygulamada kullanılır. Ardından sistemin tepkisi incelenir ve son aşamada hata oluşturan girdiler tespit edilerek bulunan hataların güvenlik açısından sömürülebilir (exploitable) olup olmadığı tespit edilir. Fuzz testi için OWASP WSFuzzer ve Wfuzz kullanılabilir.

2.4. Yazılım Çalışma Ortamı Güvenliğini Sağlayan Yöntemler

Bazı güvenlik işlevlerinin sistem tarafından ortak kullanılabilir mekanizmalarla sağlanması, saldırılara karşı dayanıklı ve güvenlik hizmetlerini varsayılan olarak sağlayan çalışma ortamı oluşturulması, yazılımın güvenliğini artıracak önlemler arasındadır.

2.4.1. Uygulama Konteyneri (Application Container)

Kullanımı üzerinde çalıştığı bilgisayarın bazı kaynaklarını, kendi içerisinde çalışan uygulama veya sistem için yalıtan bir nesnedir. Konteynırlar, yeterince yalıtıldığında yazılım zayıflıklarını ciddi şekilde azaltabilmektedir.

2.4.2. Mikro Servis Mimarisi İle Yazılımların Geliştirilmesi

Yazılımın küçük servisler halinde ve her bir servisin kendi süreci içerisinde birbirleri ile konteynerler ile iletişim kuracak şekilde tasarlanmasına mikro servis mimarisi denir. Bu mimari ile tasarlanan bir yazılımda yer alan her bir servis de mikro servis olarak adlandırılır. Her mikro servis iş veya süreç gerçekleştirir.

2.4.3. Sanallaştırma Ortamlarının Kullanılması

Tek bir donanım ya da fiziksel sunucu üzerinde birden çok program yığınının (computing stack) sanallaştırmaya özgü bazı yazılımlar ve donanımlar sayesinde çalıştırılmasını sağlayan bir çözümdür.

2.5. Yazılım Güvenliğini Sağlamada Diğer Yöntemler

2.5.1. Programlama Dilleri ile Güvenliğinin Arttırılması

Yazılımın kendinden beklenenleri tam anlamıyla yapmasının dışında yapması istenmeyen şeyleri de ihtimalı ne kadar küçük olursa olsun gerçekleşmesini engelleyebilmesi gerekmektedir. Her dil için o dile uygun güvenli kodlama standartı oluşturulmalıdır.

2.5.2. Değişen Hedef Savunmasının Uygulanması

Bir yazılımın normal çalışmasını bozmadan, yapısının ve özelliklerinin sürekli olarak değiştirilerek saldırganın zayıflıkları sömürme olasılığı en aza indirilmeye çalışılmasına değişen hedef savunması denir. Buna örnek olarak bellek taşıımı saldırularını engellemek için kullanılan ve yiğin belleğini rastgele hale getiren ASLR yöntemi gösterilebilir.

2.5.3. Ağ ve Sistem Seviyesinde Önlem Alınması

Her ne kadar yazılım geliştirme sürecinin parçası olmasa da yazılımın koştuğu ortamdaki ağ ve sistem seviyesinde önlemlerin alınması, yazılımdaki açıklıkların sömürülmüşünü zorlaştıran bir unsur olduğundan göz ardı edilmemelidir. Gözden geçirme, sızma testi, sıklaştırma yöntemleri ile testler yapılabilir.

2.6. Yazılım Güvenliğinin İlkeleri

- Her bir prosedür, modül kendi işini bitirmesi için gerekli en az haklarla çalıştırılmalı ve bu süreç esnasında bu haklara gereken en az süre boyunca sahip olmalı
- Her bir nesneye yapılan erişimlerde yetki kontrolü yapılmalı ve bu kontrol normal durumların dışında başlatma, kapatma veya istek, yanıt aşamalarında da yapılmalı

- Yüksek güvenlik gerektiren işlemlerde tek bir koşula bağlı olarak izin verilmemeli
- Yazılımda kullanılan tüm varsayılan değerler güvenliği artıracak şekilde seçilmeli ve her bir nesnenin varsayılan erişimi hiç olmalı
- Sistem ve yazılım içerisindeki modüller arasında yalıtım sağlanmalıdır
- Yazılımın güvenliği o yazılımdaki en zayıf halkanın güvenliği kadardır. Bu halka tespit edilerek yeniden tasarlanıp gerçeklenmeli
- Gereksiz özellikler eklenmemelidir
- Yetkisiz bir kullanıcının yazılım ortamından veri alabileceği, girebileceği veya yetkisiz işlemlerde bulunabileceği noktalar tespit edilmeli ve bu noktalar izlenmeli, sınırlandırılmalıdır
- Savunma mekanizmaları peş peşe uygulanmalıdır
- Güvenlik mekanizmaları mümkün olduğunda basit olmalıdır ki bu şekilde hata oluşturduğunda yetkililer tarafından kolay anlaşılmalı ve düzeltilebilmeli
- Kullanıcılar işini en kolay şekilde ancak en az yetkiyle yapabilmeli
- Uygulamanın güvenlik katmanları birbirinden ayrılmalı ve güvenlik işlevlerinin normal işlemlerin gerçekleştirildiği çalışma ortamından farklı bir izole ortam içerisinde gerçekleştirilmesi sağlanmalıdır.
- Uygulama veritabanında kişisel veri içeren birincil anahtar (kimlik no, e-posta adresi vb.) kullanılmamalıdır.
- Sistem mimarisi zayıf yönleri veya zayıf noktaları bulmak için saldırmanın bakış açısı ile incelenmeli.
- Kaynak kodun zayıflığı ve faaliyet analizleri yapılmalıdır.
- Güvenli diller, güvenli ve onaylanmış kütüphaneler kullanılmalıdır. Eski ve güvensiz kütüphaneler kullanılmamalıdır.
- Kasıtlı veya kasıtsız uygulamada güvenlik açığı yatacak kodu kimin yarattığının kontrolü için sürüm kontrolü yapılmalıdır.
- Kod kalitesi ölçümünde pep8 vb. kodlama stilleri üzerinden kontroller yapılır. Bu kontroller kodlamanın yazım biçimlerini, kod içinde bulunan boşluklar, değişken isimleri, fonksiyonların girdi sayılarını, satır sayılarını ve benzeri özellikleri kontrol eder.

- Otomatik araçlar ürettiği hata çıktılarının bir kısmı yanlış alarm (false pozitif) hatalarıdır. Bundan dolayı otomatik araçların çıktıları manuel olarak da incelenmelidir.
- Koruma gerektiren kritik verinin tam olarak korunabilmesi için, aynı zamanda uygulama kaynak kodu, binary kodu ve çalışma zamanı kodunun da korunması gerekmektedir.
- Mimarideki tüm yazılım bileşenleri tanımlı olmalı ve ihtiyaç duyulmayan bileşenler kaldırılmalıdır.
- Uygulama saldırıyla uğradığında izlenecek saldırı karşılık planı oluşturulmalıdır.
- Periyodik olarak veritabanları, uygulamalar ve uygulama verisi yedeklemesi yapılmalıdır.
- Gereksinimler açık, tutarlı, tam, uygulanabilir, takip edilebilir ve doğrulanabilir olmalıdır.
- Tüm parola alanlarında kullanıcı giriş yaparken kullanıcının parolası maskelenmeli ve açık olarak görünmemelidir.
- Kimlik doğrulama başarısız olduğu takdirde güvenli bir duruma geçilmeli ve saldırganların yetkisiz oturum açmaları engellenmelidir.
- Hesaba yeniden erişebilecek tüm hesap kimlik doğrulama işlevleri (profil güncelleme, parolamı unuttum vb.) en az ana kimlik doğrulama mekanizması kadar saldırılara dayanıklı olmalıdır.
- Hassas işlevler gerçekleştirilmeden önce, yeniden kimlik doğrulama, daha güçlü bir mekanizmayla kimlik doğrulama, ikinci faktör veya işlem imzalama gibi yöntemler uygulanmalıdır.
- Kaynak kodunda veya kaynak kodu depolarında gizli bilgiler, API anahtarları ve parolalar mevcut olmamalıdır.
- Uygulamanın yönetim arayüzlerine güvenilmeyen taraflarca erişilmesi engellenmelidir.
- Uygulama kullanıcının son başarılı oturum açma tarih ve saatini görüntülemelidir.
- Güvenilmeyen kaynaklardan alınan dosyaların türü doğrulanmalı ve zararlı bir içeriğe sahip olup olmadığı kontrol edilmelidir.
- Oturumlar belirli bir süre etkinlik olmadığından kendiliğinden sonlanmalıdır.

- Kimlik doğrulamaya erişilen tüm sayfalardan oturum kapatma işlevine erişilebilmelidir.
- Oturum kimliğinin URL, hata mesajları ve iz kayıtları içerisinde yer almaması sağlanmalıdır. URL içerisinde oturum kimliğinin yeniden yazılması engellenmelidir.
- Tüm kimlik doğrulama ve yeniden kimlik doğrulama işlemleri sonucunda yeni bir oturum ve yeni bir oturum kimliği üretilmelidir.
- Kullanıcı sadece yetkilendirildiği uygulama bileşenlerine ve kaynaklara erişebilmeli ve bunları kullanabilmelidir.
- Bellekte tutulan önemli veriler gereksinimi sona erdiğinde güvenlik ihlali oluşturamayacak şekilde silinmelidir.
- Uygulama, hassas veri ve kişisel verileri içeren hata mesajı veya iz kaydı üretmemelidir.
- Uygulama tarafından, istemci ve sunucu tarafında, kabul edilen her bir veri tipi için girdi doğrulama denetimi yapılmalıdır.
- HTML form alanlarının veri girdileri, REST çağrıları, HTTP üst başlıklar, cerezler, toplu işlem dosyaları, RSS beslemeleri gibi veri girdileri için doğrulama denetimi yapılmalıdır.
- Uygulama, sunucu ve istemci tarafında dil kodlaması (encoding) saldırılara karşı dayanıklı olmalıdır.
- Uygulama, kişisel verileri şifreli olarak saklamalı ve bu verilerin taşınmasında korumalı iletişim kanallarını kullanmalıdır.
- Uygulamanın istemci tarafında çalışan kodları, kişisel verileri başka ortamlara aktarmamalı (konsola yazma, başka dosya olarak kaydetme, yerel veya uzak uygulamalara transfer etme vb.), güvensiz ortamlarda (ortak dizin, USB disk vb.) güvensiz yöntemlerle (açık metin olarak, zayıf şifreleme algoritma kullanarak şifreleme vb.) saklamamalıdır.
- Kullanılan veritabanının dışarıya aktarımı ancak veritabanı yönetim yetkisi olan hesaplarla yapılmalı ve öncesinde veritabanındaki kişisel verilerin silinmesi sağlanmalıdır.
- Değişen parola fonksiyonu eski parolayı, yeni parolayı ve bir parola onayını kapsamalıdır.

- Kaba kuvvet saldırıları ya da servis dışı bırakma saldırıları gibi otomatik yapılan yaygın kimlik doğrulama saldırılarını önlemek için istekler azaltılmalıdır. Aşırı kimlik doğrulama denemelerini engellenmeli.
- Parolalar için bir en uzun geçerlilik süresi tanımlanmış olmalıdır.
- Uygulama, ayar ve denetim dosyaları kullanıcı verisiyle aynı konumda depolamamalıdır.
- Desteklenmeyen, güvensiz veya teknolojisi zaman aşımına uğramış istemci teknolojileri kullanılmamalıdır.
- Uygulama tarafından etkin ve aynı zamanlı oturumların sayısı sınırlanırılmalıdır. Sızma testi ile test edilebilir.
- Her bir kullanıcının uygulamadaki etkin oturumları görüntülenebilmeli, kullanıcı herhangi bir etkin oturumunu sonlandırmalıdır.
- Web uygulamalarında oturum cerezlerinde HTTPOnly bayrağı etkin olmalıdır.
- Uygulama, başarısız sistem başlatma, başarısız sonlandırma veya kapatma gibi işlemlerde güvenli bir duruma geçmelidir. Negatif test ile test edilebilir.
- Tüm kriptografik modüllerin, güvenli bir şekilde hataya düşüğü doğrulanmalıdır. Hata yönetimi “Oracle Padding” atağına imkan tanımayacak şekilde olmalıdır. Negatif test ile test edilebilir.
- Tüm rastgele üretilen sayılar, dosya isimleri, global eşsiz değerler (GUID) ve karakter dizilerinin saldırgan için tahmin edilemez olması sağlanmalıdır. Rastgele sayıların yüksek entropiye sahip olarak üretilmelidir. Negatif test ile test edilebilir.
- Tüm anahtar ve şifreler kullanımları tamamlandığında, tamamen sıfırlanarak yok edilmelidir. Negatif test ile test edilebilir.
- Tüm formlarda istemci tarafında yapılan ön bellekleme işlevselliği önemli veriler için kapatılmalıdır.
- Siteler arası komut dosyası çalışma (XSS) engellemeye uygulama-istemci arasındaki hassas veri iletişim için HTTP başlığı veya gövdesi kullanılmalıdır. Postman, Wireshark vs. ile test edilebilir.
- Uygulama, saklama gereksinimi sona erdikten sonra önemli verileri güvenlik sonunu yaratmayacak şekilde silinmelidir.

- Sunucuya gelen isteklerin öngörülmeyen bir sayıda ya da büyüklikte olup olmadığı kontrol edilebilmelidir. Negatif test ile test edilebilir.
- İz kayıtlarında olayların zaman sıralamasına ilişkin araştırma yapılabilecek şekilde zaman bilgisi yer almalıdır.
- Uygulama tarafından üretilen iz kayıtları hassas bilgi içermemelidir.
- Uygulama hassas bilgileri formlarda bulunan gizli alanlarda saklamamalıdır.
- Uygulama Cross-Site Request Forgery (CSRF)'dan kaynaklanan açıklıklardan korunma mekanizmasına sahip olmalıdır. Fuzz testi, kaynak kod zayıflık analizi ile test edilebilir.
- Uygulamanın çalışma ortamı, bellek taşması saldırılara dayanıklı olmalıdır veya mevcut güvenlik mekanizmaları bellek taşmasını engellemelidir.
- Sunucuda yapılan girdi doğrulama hataları, isteğin reddi ile sonuçlanmalı ve iz kaydı oluşturulmalıdır.
- SQL Injection engellemek için bütün veritabanı sorguları, parametre olarak yapılmalı ve veritabanına erişimde kullanılan dile karşı önleyecek denetimler yapılmalıdır. Veritabanı açıklık tarama aracı, web uygulama zayıflık tarayıcı veya negatif test ile test edilebilir.
- LDAP-Active Directory Injection için uygulama, yetki onaylama hizmetlerinin enjeksiyonu açıklıklarını önleyici güvenlik denetimlerini yapmalıdır. Tasarım gözden geçirme, işlevsel test, negatif test ve kaynak kod gözden geçirme ile test edilebilir.
- İşletim sistemi komut enjeksiyonu için uygulama, işletim sistemi komut enjeksiyonu açıklıklarını önleyici güvenlik denetimlerini yapmalıdır. Kaynak kodu açıklık tarayıcısı, Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod gözden geçirme ile test edilebilir.
- Uygulama, girdi alınan içerik bir dosya yolu içeriyorsa, uzak ya da yakın dosya işleme açıklıklarını önleyici güvenlik denetimlerini yapmalıdır.
- Uygulama, XML açıklıklarını (XPath sorgu saldırıları, XML harici öğe saldırıları, XML enjeksiyonu vb.) önleyici güvenlik denetimlerini yapmalıdır.
- Uygulama, web servislerini iyi yapılandırılmış en az TLS v1.2 veya SSL gibi muadil güvenlik önleme sunan bir protokol ile sunacak şekilde tasarılanmalıdır.

- Uygulama, web servis kimlik doğrulama ve yetkilendirmesi için oturum temelli yapılar kullanacak şekilde tasarlanmalıdır.
- Uygulama, web servisi ile gönderilen veride betik (script) içermeyecek şekilde tasarlanmalıdır. Statik analiz ile test edilebilir.
- Uygulama, web servislerinden şifreli olarak paylaşılan verileri yine şifreli olarak saklayacak şekilde tasarlanmalıdır.
- Uygulama, kişisel veriler üzerinde işlem yapılması ana amaç olmayan durumlarda kişisel verileri maskeleyerek görüntülemeli, aktarmalı veya işlemelidir.

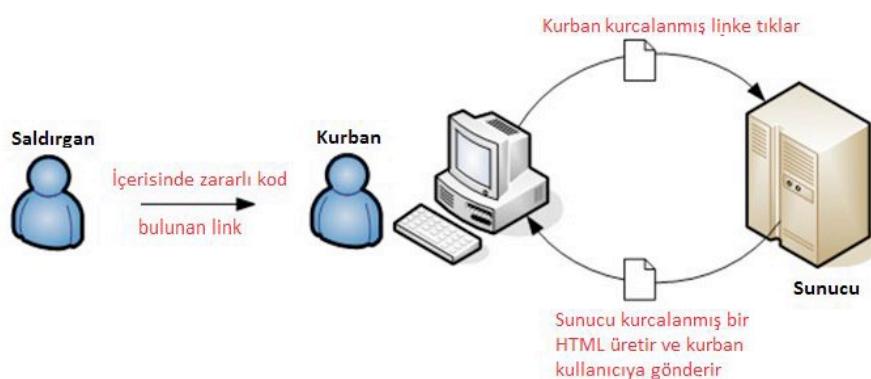
2.7. Web Uygulama Güvenliği Kavramları

2.7.1. Cross Site Scripting (XSS)

Bir tarayıcıda izinsiz olarak kod çalıştırılmaktır. 3 tipi vardır [3].

2.7.1.1. Yansıtılan XSS (Reflected XSS)

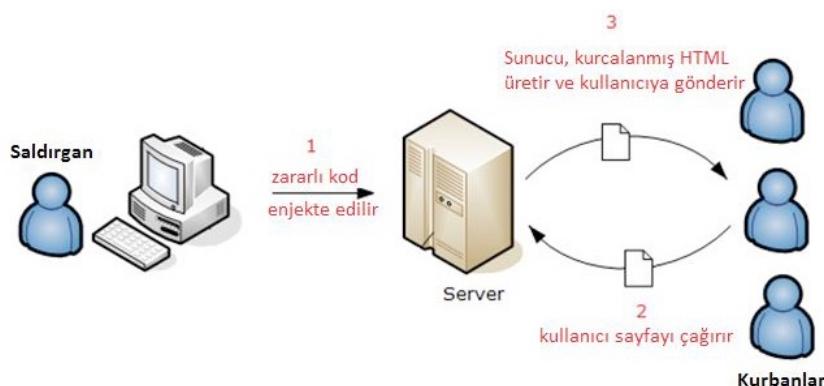
Saldırgan zararlı bir link hazırlar ve kurban kullanıcısını bu linke tıklamaya ikna eder. Kurban kullanıcının tarayıcısından zararlı bir talep gönderilir. Linkteki zararlı HTML, web sayfası içerisine eklenerek kullanıcıya geri gönderilir. Kurban kullanıcının tarayıcısı zararlı HTML’i çalıştırır.



Şekil 2.6. Reflected XSS

2.7.1.2. Depolanan XSS (Stored XSS)

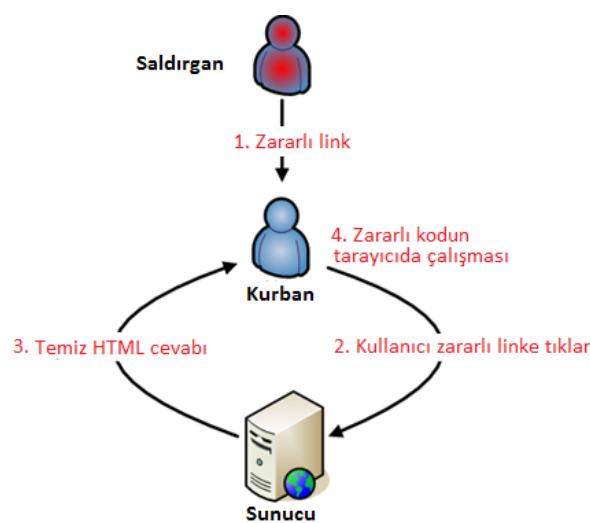
Saldırgan zararlı kodunu veritabanı veya sunucuya enjekte eder. Sayfayı ziyaret eden tüm kullanıcıların tarayıcıları bu zararlı kodu çalıştırır.



Şekil 2.7. Stored XSS

2.7.1.3. DOM Tabanlı XSS (DOM Based XSS)

Kullanıcıya gönderilen HTML zararlı bir kod içermez fakat tarayıcı DOM objesini çağrıır ardından DOM objesi zararlı kodu çalıştırır.



Şekil 2.8. DOM Based XSS

2.7.2. SQL Enjeksiyonu (SQLi)

Veritabanında izinsiz SQL sorgusu çalıştırılmaktır. Kullanıcıdan alınan girdinin SQL sorgusu olarak işlenmesi sonucu ile gerçekleşir.

SELECT name FROM products WHERE id='100'; yerine
SELECT name FROM products WHERE id='100' UNION SELECT password FROM users WHERE userid='1' gibi bir sorgu çalıştırılması ile olur.

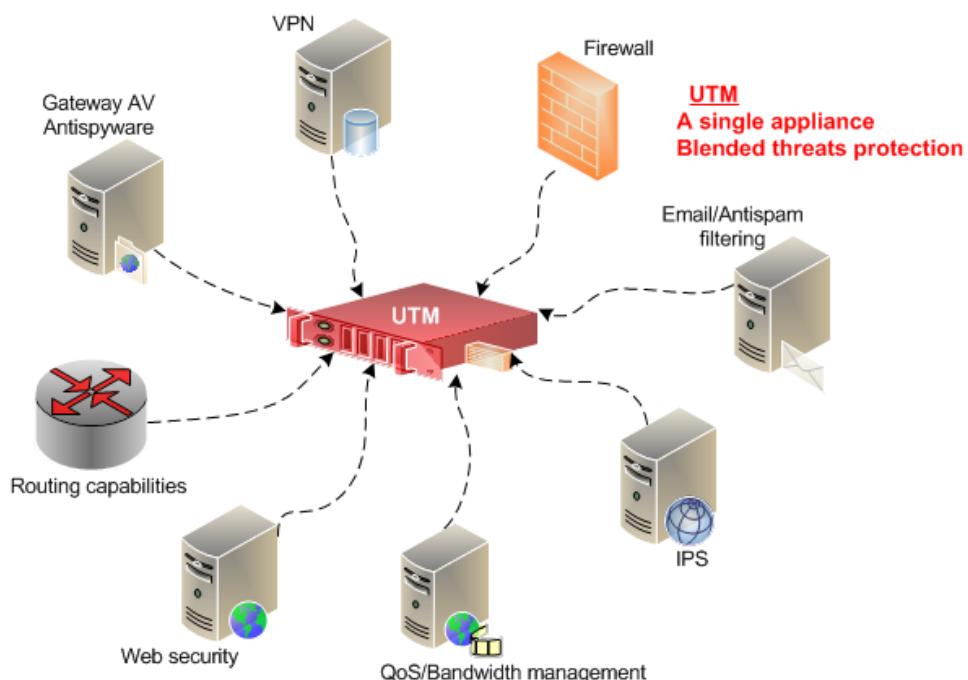
SQL enjeksiyonuna karşı en etkili önlem string eklemedir. Örneğin;

```
String username = Session["username"];
String password = Session["password"];
String selectCommand = "SELECT * FROM users WHERE username=" + username
+ "" AND password=" + password + "";
SqlCommand myCommand = new SqlCommand(selectCommand, DataConnection);
SqlDataReader dr = myCommand.ExecuteReader();
```

Şeklinde bir kullanımla SQLi önüne geçilebilir.

BÖLÜM 3. UTM ARAYÜZÜ

UTM, birden çok güvenlik özelliği ve servislerinin bir cihazda veya hizmette sunularak kullanıcıların basitleştirilmiş bir şekilde korunması ve kontrolünün sağlanmasıdır. Dolayısıyla merkezi bir yönetim sağlanarak kontrol ve organizasyon kolaylaştırılmış olur. İçerisinde Güvenlik Duvarı, VPN, Saldırı Tespit ve Önleme, Web Filtreleme, Antivirüs Gateway, Anti-Spam, İçerik Filtreleme, Trafik Ayarlama gibi özellikler ve daha fazlasını barındırabilir. Bunların yanı sıra yönetim konsolu sayesinde verileri ve olayları raporlayarak bir sorun durumunda bunu giderme aşamasında yöneticilere otomatik olarak rehberlik edebilir ve ayrıca maliyeti de azaltabiliriz [4].

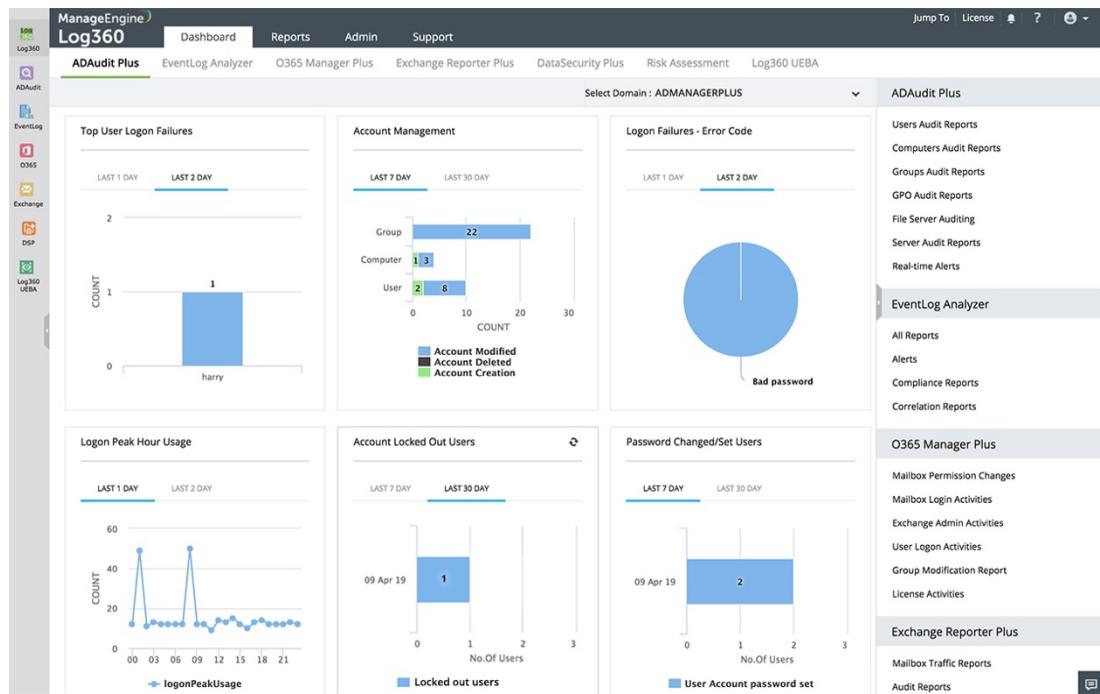


Şekil 3.1. UTM Yapılandırması

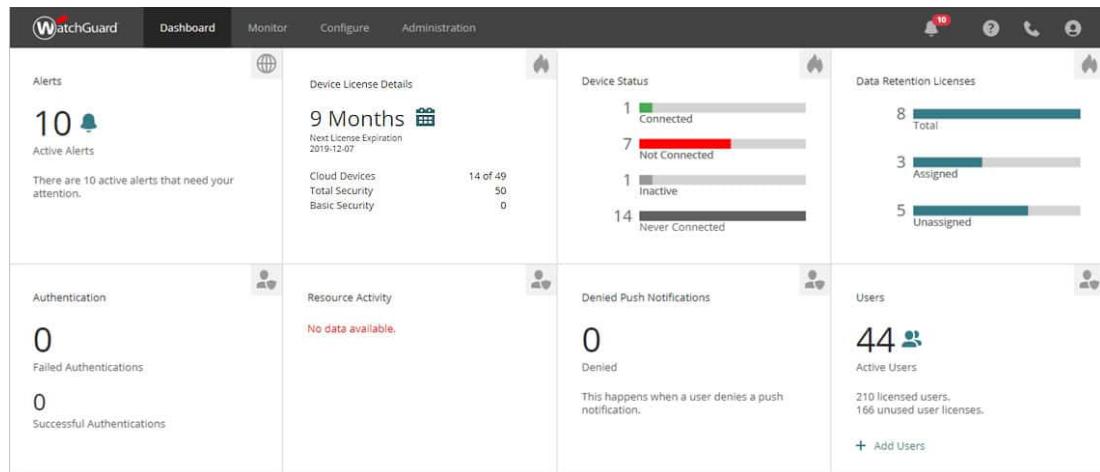
3.1. UTM Arayüz Örnekleri

Sektördeki yapacağımız arayüze benzer grafiksel arayüz ve özellikler barındıran gözlemlenebilirliği kolay olan bazı UTM arayızları örnek olması açısından aşağıda bulunmaktadır. Planladığımız muhtemel tamamlanmış arayüz de örneklerdeki gibi

sade ve tek sayfada tüm özelliklerini barındıran, kontrolün ve gözlemin kolay olacağı, esinlenmiş ve harmanlanmış bir arayüz olacak.



Şekil 3.2. ManageEngine Log360



Şekil 3.3. WatchGuard Firebox

Exosphere

Home > Dashboard

Dashboard

Select Period: 2020-01-23 ~ 2020-01-30

Select Group: Exosphere | Select Sub-Group

Security Scores

Score Range	Persons
0-10	~1
10-20	~1
20-30	~1
30-40	~1
40-50	~1
50-60	~1
60-70	~1
70-80	~1
80-90	~1
90-100	~1

Month	Points
Jan	~95
Feb	~90
Mar	~92
Apr	~94
May	~95
Jun	~93
Jul	~96
Aug	~98
Sep	~95
Oct	~96
Nov	~97
Dec	~94

Outsider Threat Protection

Anti-Malware

189 Disinfected, 2 Quarantined, 0 Skipped

Anti-Ransomware

8 Unknown Process, 11 Blocked Access to Files, 4 Blocked Applications

Web Protection

10 Blocked, 21 Warned

Data Loss Prevention

Device Control

4 Devices Blocked, 2 File Copy Blocked, 11 File Copied

Application Control

2 Applications Blocked, 5 File Transfer Blocked, 2 File Transfer Allowed

Screen Capture Control

11 Blocked

Data Discovery

3 Real Time Detection

Web Control

0 Blocked, 7 Warned

Print Control

0 Blocked, 221 Watermark Allowed, 0 Allowed

PC Healthcheck

PC Healthcheck

2 Secured Devices, 13 Weak Devices

Backup

Backup

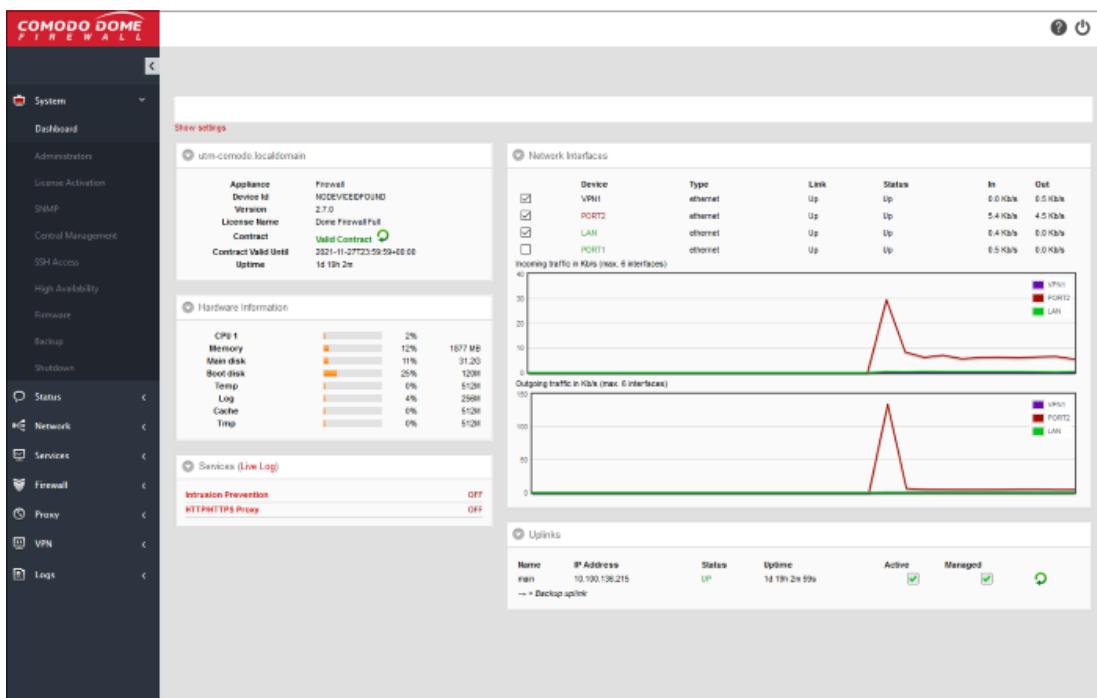
200GB Storage Used, 153K Files Backed Up, 10 Devices Backed Up, 4 Devices Not Backed Up

Copyright Exosphere, Inc. All rights reserved.

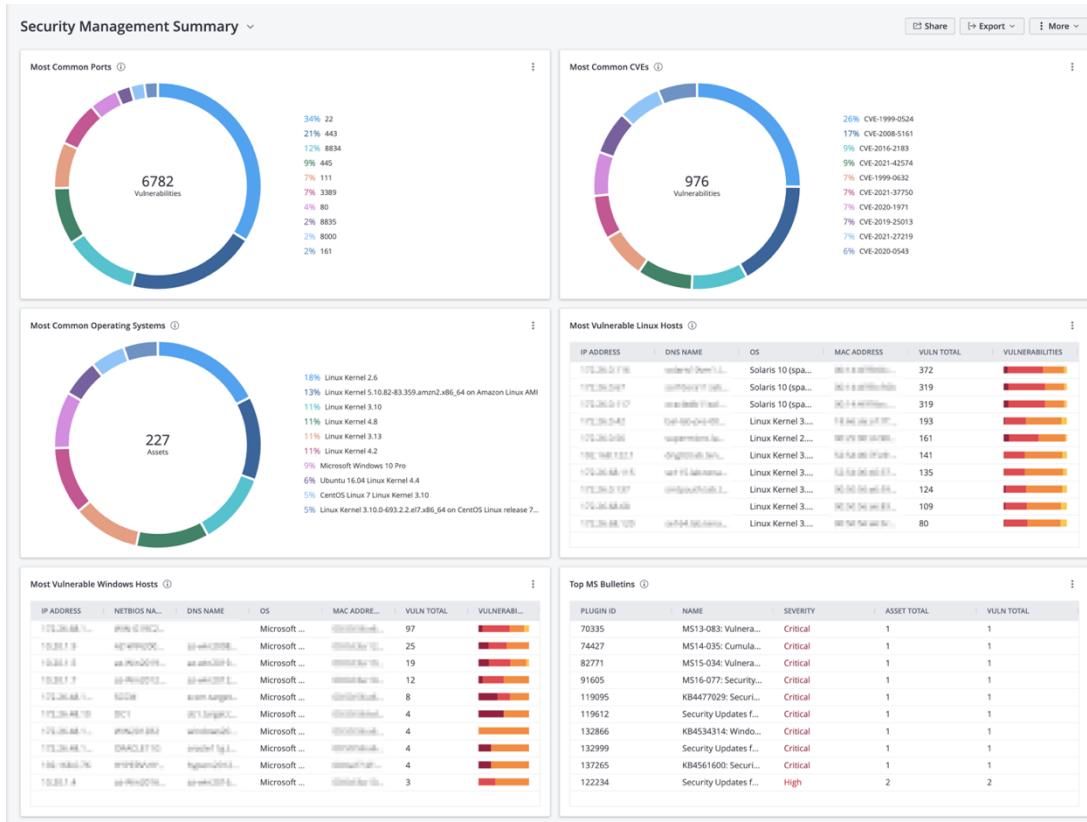
Şekil 3.4. Exosphere



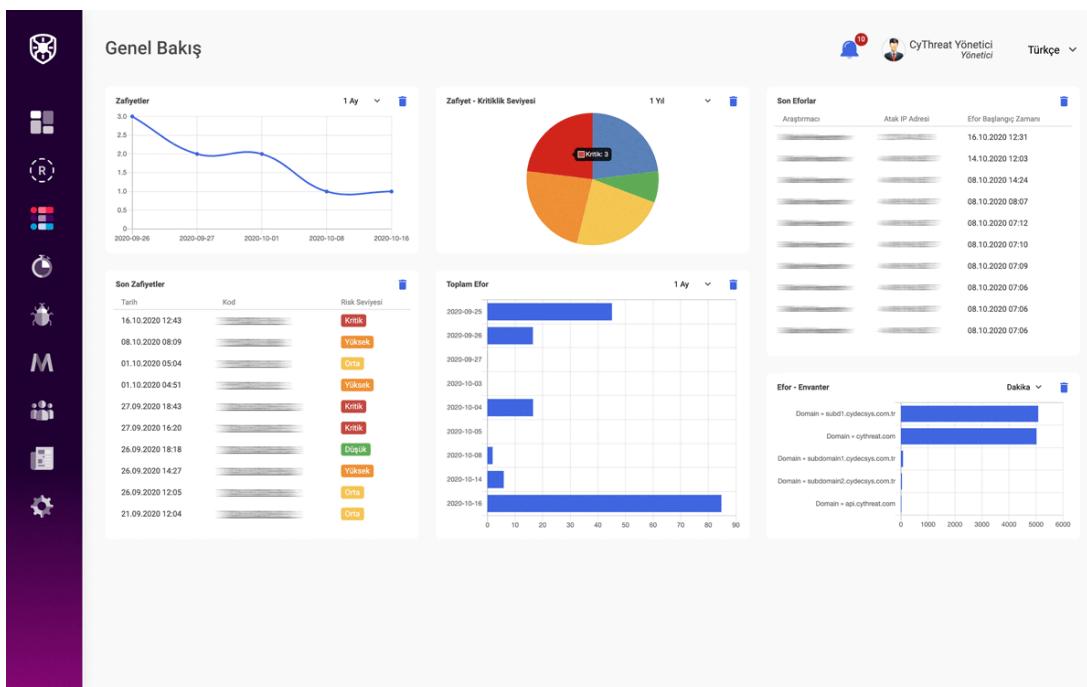
Şekil 3.5. AlienVault



Şekil 3.6. Comodo Dome Firewall



Şekil 3.7. Security Management Summary Dashboard by Carole Fennelly



Şekil 3.8. STMBugShield

BÖLÜM 4. MODÜLLER

UTM arayüzünde kontrolünün yapılmak üzere olan modüller ile alakalı bilgiler ve örneklemeler mevcuttur.

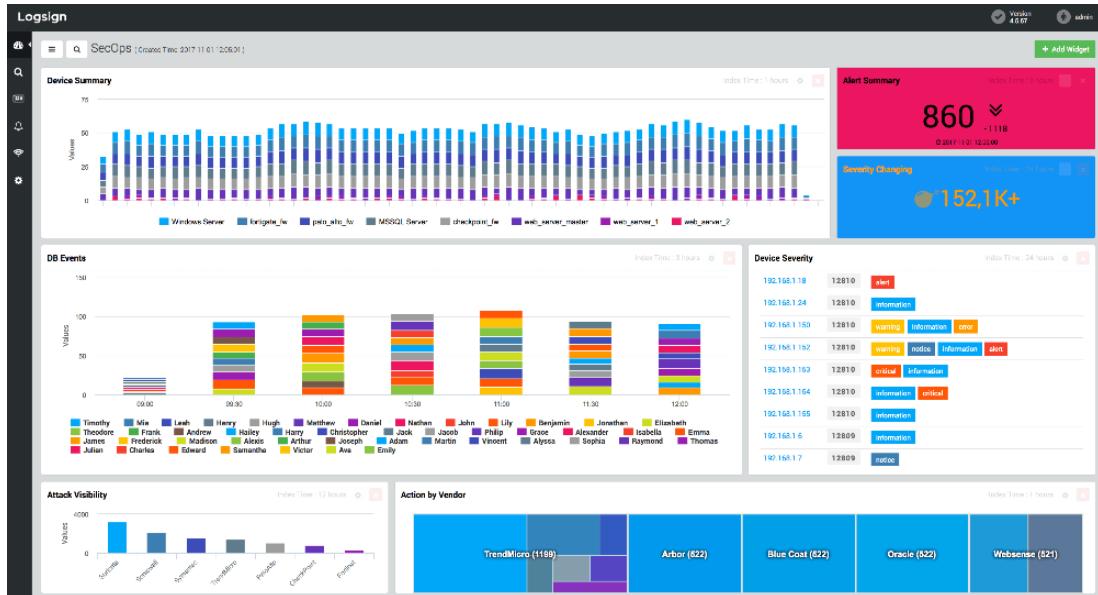
4.1. Log Yönetimi

Sunucular, güvenlik duvarları ve diğer BT ekipmanları dahil olmak üzere şirketinizin sistemlerinde ve ağlarında meydana gelen tüm olayları ayrıntılandıran dosyalardır. Günümüzde güvenlik noktasında önemi artan konuların başında gelmektedir. Log Yönetimi hem yasal zorunluluklar nedeni ile hem de uluslararası standartlar tarafından (ISO 27001) önerilmektedir. Eğer log yönetimi yapabiliyorsak örnek olarak aşağıdaki sorulara cevap verebilme durumumuz vardır [5].

- Belirli zaman aralıklarında kimler oturum açtı?
- Sistem yöneticileri takip ediliyor mu?
- Bilgisayar adı (hostname), IP adresi, MAC adresi değişikliği oldu mu?
- Kimler hangi IP adresini aldı?
- Kimler hangi saatle VPN bağlantısı kurdu?
- P2P program kullanan var mı?
- Domain admin hesabına kullanıcı eklendi mi? Vb.

4.1.1. LogSign

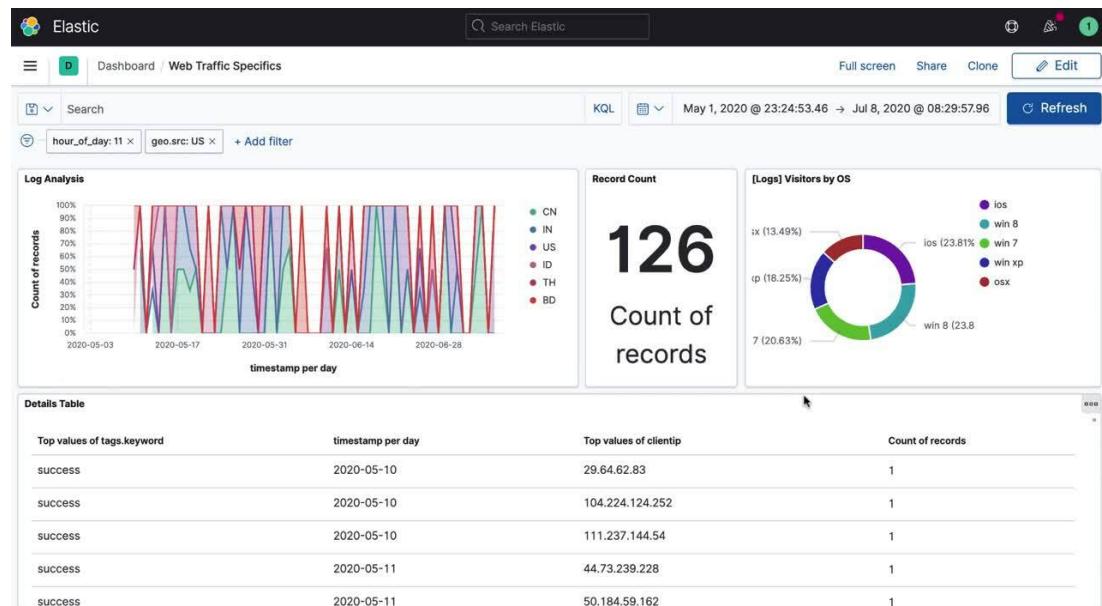
Belirlenen bir network üzerinde raporlamayı arzu ettiğiniz faaliyetlere ait logların yönetilebildiği ileri seviye SIEM güvenlik çözümüdür.



Şekil 4.1. LogSign

4.1.2. ELK

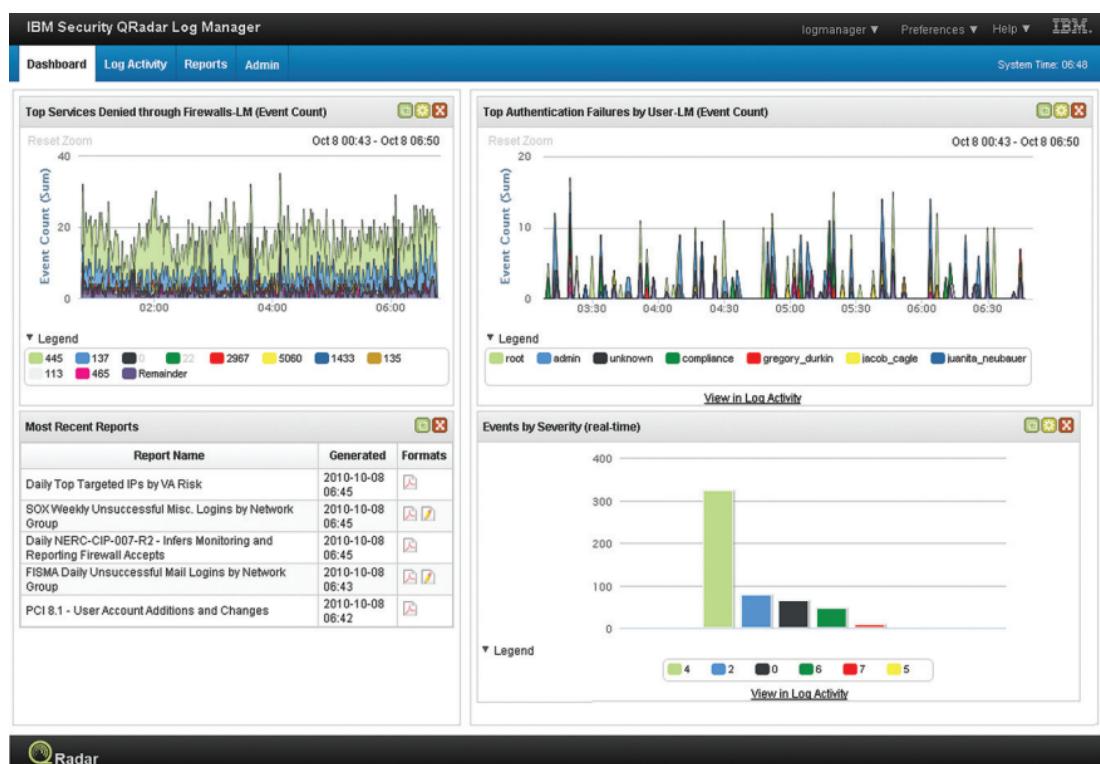
ELK (ElasticSearch, Logstash, Kibana) açık kaynak kodlu 3 ürünün bir araya gelerek oluşturduğu bir koleksiyondur. ELK büyük veriler içerisinde arama, analiz ve görselleştirme işlemleri gerçekleştirmek için kullanılır.



Şekil 4.2. ELK

4.1.3. IBM QRadar

IBM QRadar Log Manager, QRadar Sense Analytics motorunu kullanarak kuruluşların kendilerini tehditlere, saldırılara ve güvenlik ihlallerine karşı korumasına yardımcı olmak için Ağ güvenliği log olaylarını toplar, analiz eder, depolar ve bunlarla ilgili raporlama yapar. Sense Analytics, aygıtlardaki, sunuculardaki, işletim sistemlerindeki, uygulamalardaki, uç noktalardaki işlenmemiş olayları, eyleme dönüştürülebilir istihbarat verilerine dönüştürür.



Şekil 4.3. IBM QRadar

4.2. Zararlı Yazılım

Zararlı yazılım, kullanıcıların programlanabilme özelliğine sahip cihazlarına, web sitelerine veya ağlarına zarar veren veya yetkisiz erişim sağlayan herhangi bir yazılımdır. Siber suçlular genellikle bunu, mali kazanç içi kurbanlardan veri elde ederek baskı yapmak üzere kullanır [6].

4.2.1. Virüs

Bulaştığı bilgisayarda dosyaları silme, sabit diskin boşaltma ve verileri bozma gibi kötü amaçlı faaliyetler gerçekleştiren programlardır.

4.2.2. Solucan (Worm)

Kendini bir ağaç çoğaltmak ve diğer bilgisayarlara yaymak için tasarlanmış bir tür kötü amaçlı yazılımdır.

4.2.3. Truva Atı (Trojan)

Meşru bir program kılığına girmiş ancak yüklenikten sonra kötü amaçlı etkinlik gerçekleştiren zararlı yazılım çeşididir.

4.2.4. Casus Yazılım (Spyware)

Virüs bulaşmış bilgisayardan bilgi çalan ve bunları uzak bir konumdaki kötü amaçlı kişilere gönderen yazılımlardır.

4.2.5. Keylogger

Klavyede yaptığınız her tıklamayı izleyen, kaydeden ve kötü amaçlı kişilere gönderen bir tür casus yazılımdır.

4.2.6. Fidye Yazılımı (Ransomware)

Bilgisayarlarınızdaki dosyaları şifreleyerek erişilemez hale getiren bir kötü amaçlı yazılımdır.

4.2.7. Reklam Yazılımı (Adware)

İnternete bağlandığında reklamları görüntülemek için kullanıcının izni olmadan yüklenen yazılımlardır.

4.2.8. Tarayıcı Korsanları

Sayfaların içeriğini değiştirmek, sizi başka konumlara yönlendirmek veya diğer kötü amaçlı işlemleri gerçekleştirmek için tarayıcınızın davranışını değiştiren bir kötü amaçlı yazılım türüdür.

4.2.9. Rootkit

Virüs bulaşmış bir cihaza yönetim erişimi sağlayan ve onu farklı cihaz ve yazılımlara zarar vermek için kullanan kötü amaçlı yazılımdır.

4.3. Anomaly Detection

Anomaly Detection, en basit anlamıyla bir veride beklenmedik durumların veya kalıpların bulunmasını sağlayan bir tekniktir. Bu beklenmedik durumlara literatürde outliers (aykırı değerler), exceptions (istisnai durumlar) veya anomaliler denilmektedir. Anomali tespiti, izinsiz giriş tespiti, dolandırıcılık tespiti, arıza tespiti, sistem sağlığının izlenmesi, sensör ağlarında olay tespiti, ekosistem bozukluklarının tespiti ve makine görüşü kullanarak görüntülerde kusur tespiti gibi çeşitli alanlarda uygulanabilmektedir. Anomali tespiti için Makine Öğrenmesi Temelli Anomali Tespiti, K-Nearest Neighbour ve Density Based Algoritmaları, Kümeleme Tabanlı Algoritmalar, İstatistiksel Metodlarla Anomali Tespiti bulunmaktadır. Anomalilerin 3 adet tipi bulunmaktadır [7].

4.3.1. Point Anomalies

Bireysel bir veri örneği eğer diğer normal verilerden uzaktaysa bu bir anomali veridir. Burada bu anomali tespitine point Anomali denmesinin sebebi anomali tespitinin belli bir niteliğe (attribute) bağlı olmasıdır.

4.3.2. Contextual Anomalies

Belirli bir verimiz bazı durumlarda anomaliye işaret ederken diğer durumlarda normal bir veriye işaret ediyorsa, yani özel bir bağlamda (specific context) anomali davranış sergiliyorsa bu context anomaliye bir örnektir.

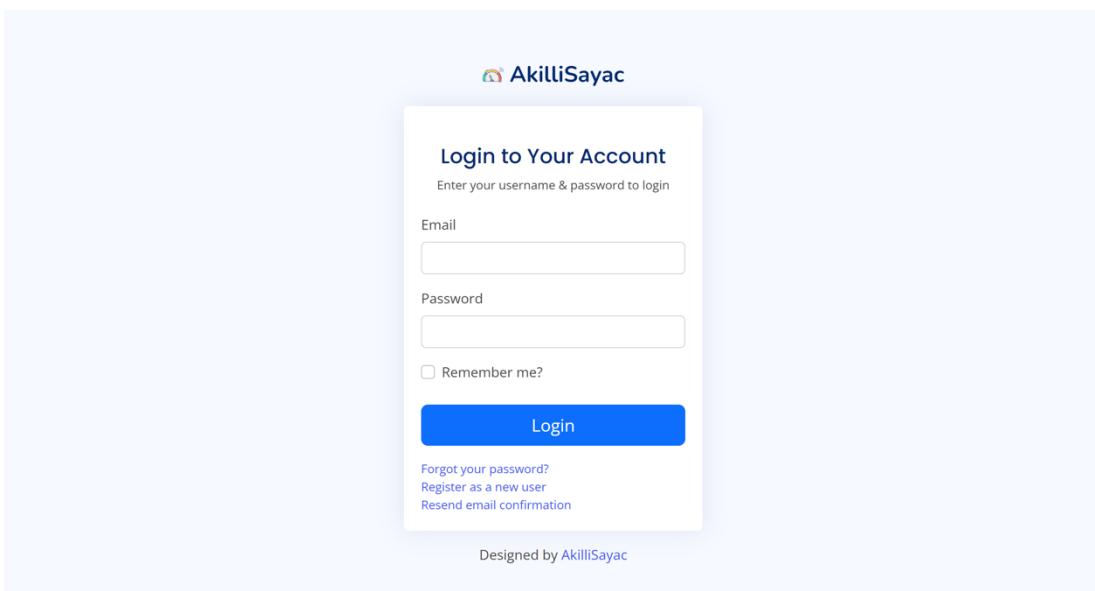
4.3.3. Collective Anomalies

Birbiriyle ilişkili olan veriler tüm verisetinde anomali davranış oluşturuyorsa bu bir collective anomaliye örnektir. Burada ilişkili olan bazı veriler bir araya geldiğinde anomali oluşturabilirken, bu veriler bireysel olarak verisetinde anomali davranış göstermiyor olabilir.

BÖLÜM 5. TASARIM

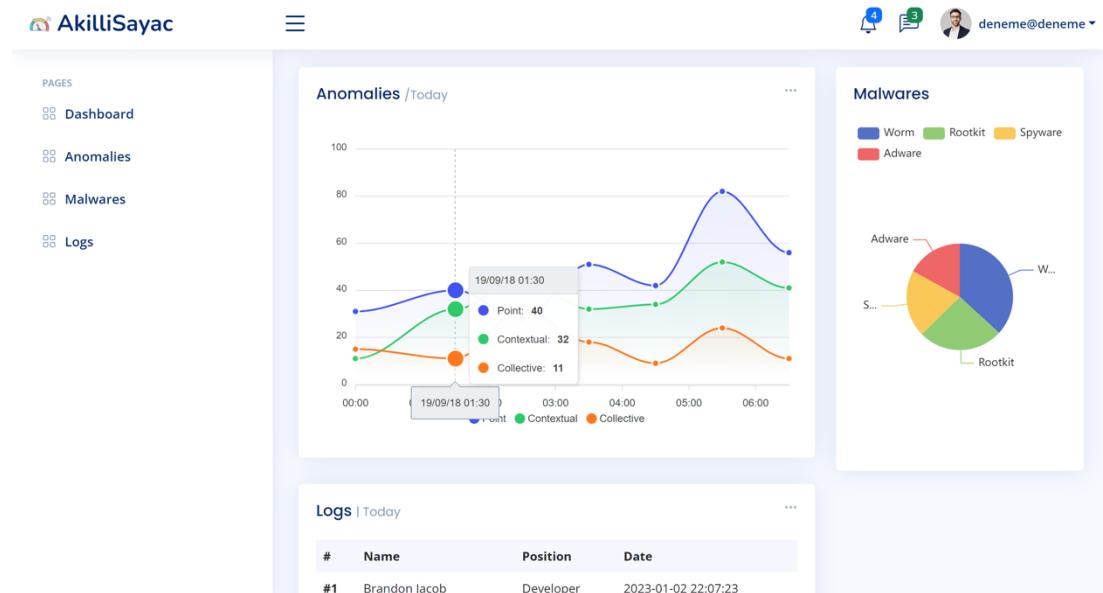
Tasarım çalışmamızda Güvenli Yazılım Geliştirme Kılavuzu, UTM Arayüzü ve entegre edilecek modüller ile ilgili gerekli araştırmaları tamamlayarak yapacağımız web uygulaması ile ilgili bilgi sahibi olduk ve bitirme çalışmasında tamamlayacak olduğumuzda bu projenin web uygulamasına bir başlangıç yaptık. İlgili arayüzün genel itibarı ile tamamlanacağı tasarımlar eklerde mevcut olup, üzerinde iyileştirmeler, değiştirmeler ve/veya yenilikler bitirme çalışmasında yapılacak, tamamlanacak ve gerekli olduğu şekilde güvenli yazılım geliştirme standartlarına uygun olarak güvenli bir ASP.NET Core 7.0 UTM arayüzü sunulacaktır.

Web uygulamasına giriş yapıldığında öncelikle bizi kullanıcı giriş ekranı karşılaşacak ve giriş yapmayan kullanıcı sisteme erişim sağlayamacaktır. Aynı zamanda dışarıdan kayıt olma işlemi yapılamayacak, veritabanı üzerinde Süper Admin olarak bir kullanıcı kaydı yapılacak ve yalnızca Süper Admin rolüne sahip olan kişi veya kişiler tarafından sisteme giriş yaptıktan sonra sistem üzerinde Admin rolüne sahip, yeni kullanıcı oluşturamayan fakat sisteme giriş yapabilen kullanıcılar oluşturulabilecektir.

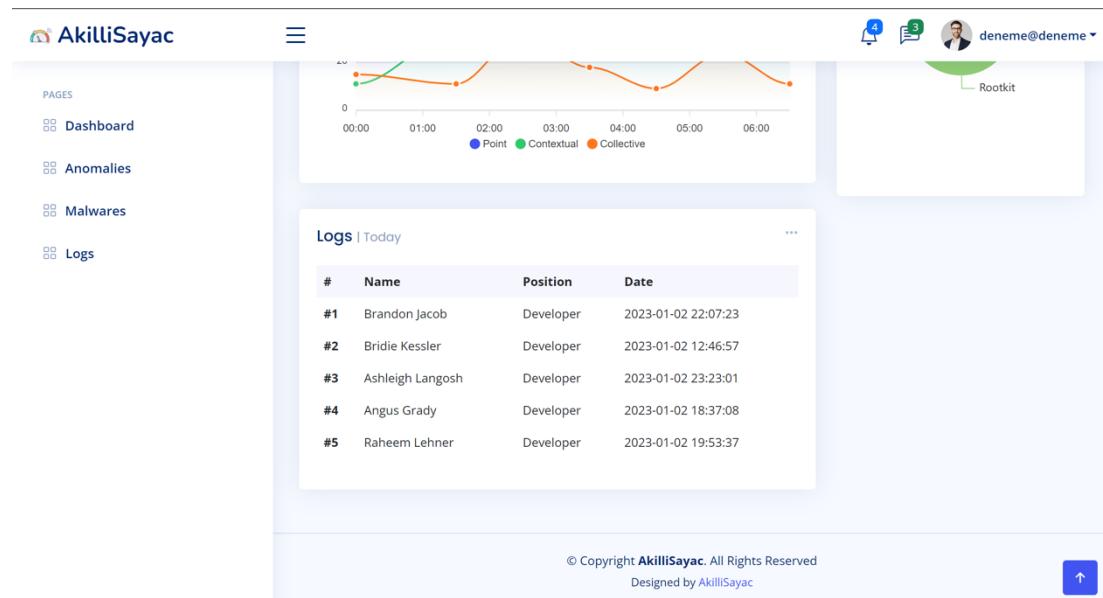


Şekil 5.1. Login Ekrani

Sisteme giriş sağlandıktan sonra arayüze erişim sağlanacak ve bilgi akışı sağlanacak. Yan panelden ise modüller ile ilgili detaylı bilgi alabilmek için sayfalara geçiş sağlanabilecektir.

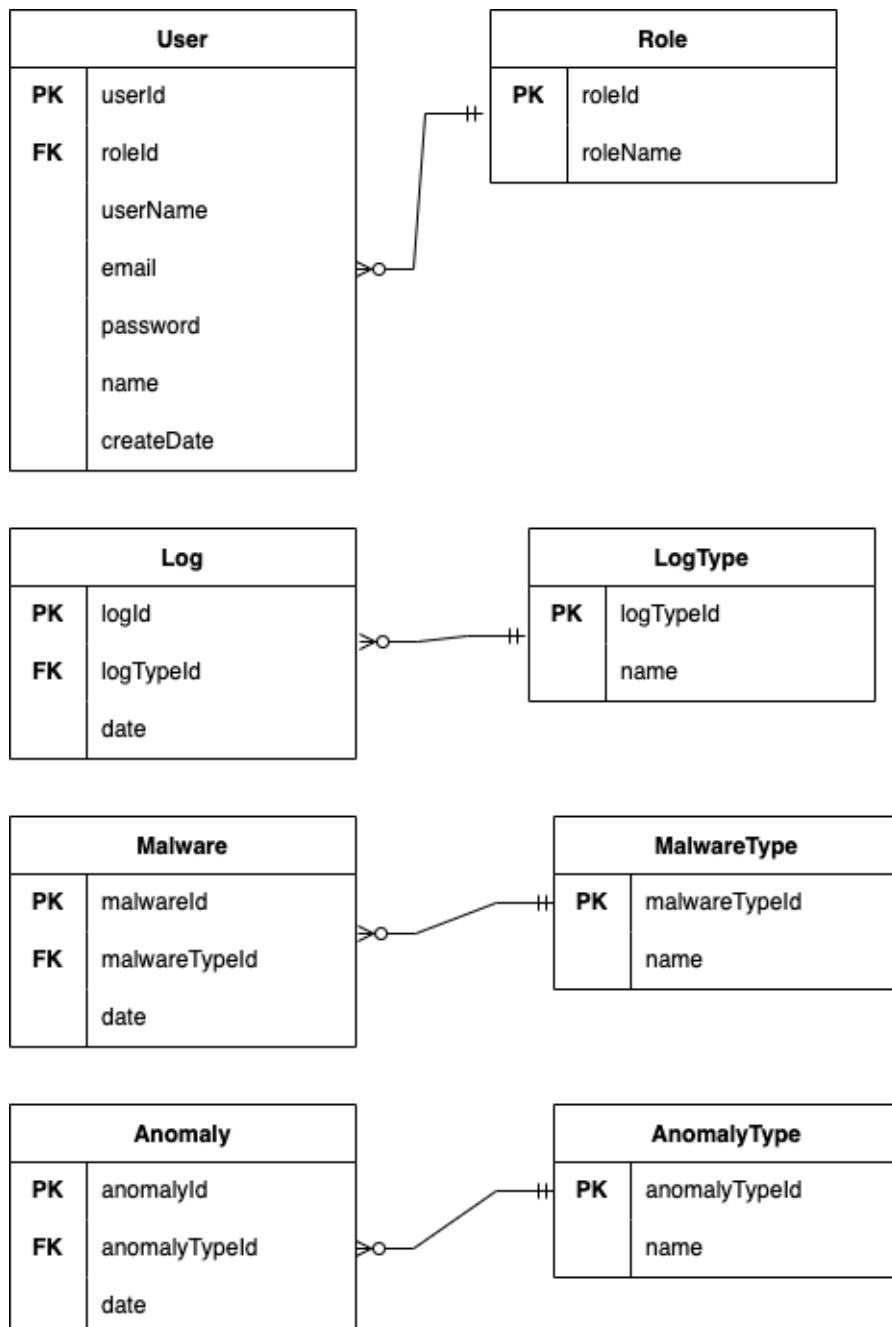


Şekil 5.2. Arayüz Görünümü



Şekil 5.3. Arayüz Görünümü Devamı

Şu anki aşamada muhtemel fakat düzenlenecek, geliştirilecek veritabanı ER diyagramı tasarıımı da gerçekleştirilmiştir.



Şekil 5.4. ERD

BÖLÜM 6. SONUÇLAR VE ÖNERİLER

Çalışmanın sonucunda ileriki süreçte web uygulamasının gerçekleşeceği akıllı sayaç, uzaktan takip web uygulamanın gerçekleştiriminde bilgilerine sahip olmamız gereken Güvenli Yazılım Geliştirme Standartları ve dikkat edilmesi gerekenler, UTM Arayüzü ile ilgili bilgiler ve örnek arayüzler, kullanılacak modüller ile ilgili bilgiler ve örnekleri incelenerek ve araştırılarak bilgi sahibi olunmuş, web uygulamasının geliştirilerek tamamlanması kalmıştır. Web uygulamasının gerçekleştirimi esnasında ve tamamlanana kadarki süreçte araştırmalardan faydalananarak ve sürüm denetim sistemi git de kullanılarak eksiksiz ve güvenli bir uygulama yapmak amaçtır.

KAYNAKLAR

- [1] NIST Secure Software Development Framework
- [2] TUBITAK Güvenli Yazılım Geliştirme Kılavuzu
- [3] TUBITAK SGEP Web Uygulama Güvenliği
- [4] UTM Whitepaper, 2012.
<https://fedtechmagazine.com/sites/default/files/utm-whitepaper.pdf>
- [5] Log Yönetimi ve SIEM Nedir?, 2021.
<https://tr.linkedin.com/pulse/log-yonetimi-ve-siem-nedir-dincer-oksuzbakan>
- [6] Zararlı Yazılım Nedir? Bilinen Zararlı Yazılımlar Nelerdir?, 2022.
<https://berqnet.com/blog/zararli-yazilim>
- [7] Anomaly Detection (Anomali Tespiti) Nedir?, 2018.
<https://medium.com/yazilim-bilimi/anomaly-detection-anomali-tespiti-nedir-989a956df7a7>

ÖZGEÇMİŞ

Barış Yılmaz, 29.01.1998'de İstanbul'da doğdu. İlk, orta ve lise eğitimini Sarıyer'de tamamladı. 2016 yılında Sarıyer Hüseyin Kalkavan Mesleki ve Teknik Anadolu Lisesi'nden mezun oldu. 2016 yılında İstanbul Topkapı Üniversitesi İnternet ve Ağ Teknolojileri Önlisans Bölümü'nü kazandı. 2017 yılında Ced Teknoloji Dan. Bilişim Destek Hiz. Ltd. Şirketinde network stajyeri olarak donanım stajını tamamladı ve 2018 yılında İnternet ve Ağ Teknolojileri bölümünden mezun oldu. 2019 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Lisans Bölümü'nü Dikey Geçiş Sınavı ile kazandı. 2022 yılında 4. sınıf öğrencisi olarak SAÜ Bilgisayar Mühendisliği Bölümü'nde eğitimini halen sürdürmektedir.

Cemal Aslan, 23.05.1999'da Giresun'da doğdu. İlk, orta ve lise eğitimini Gölcük'de tamamladı. 2017 yılında Gölcük İhsaniye Anadolu Lisesi'nden mezun oldu. 2018 yılında Sakarya Üniversitesi İnşaat Mühendisliği Bölümü'nü kazandı. 2019 yılında Merkezi Yatay Geçiş Sistemi ile Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'ne geçiş yaptı. 2022 yılında 4. sınıf öğrencisi olarak SAÜ Bilgisayar Mühendisliği Bölümü'nde eğitimini halen sürdürmektedir.

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU: Akıllı Şebekeler İçin Güvenli Yazılım Geliştirme Temellerine Uygun Bir UTM Arayüz Tasarımı

ÖĞRENCİLER (Öğrenci No/AD/SOYAD):

G191210303 / BARIŞ / YILMAZ

G191210387 / CEMAL / ASLAN

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma kluvusa uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımin/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımin gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişkili modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilibilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metodlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımin sizme testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN : DR. ÖĞR. ÜYESİ MUSA BALTA

DANIŞMAN İMZASI: