YMT311 Bilgi Sistemleri ve Güvenliği

Yazılım Güvenliği

Bölüm - 7

Prof.Dr. Resul DAŞ
Fırat Üniversitesi
Yazılım Mühendisliği Bölümü

Konu Başlıkları

- Yazılım ve Güvenlik
- Yazılım Geliştirmede Tehdit Modelleme
- Tehdit Modellemenin Faydaları
- Akış Diyagramı
- Risk Hesaplaması DREAD Kavramı
- Örnek Saldırı Seneryasu
- Tehditlerin Azaltılması

Yazılım Güvenliğine Genel Bakış

- **Yazılım güvenliği** genel olarak; yazılımların **tersine mühendislik** yöntemleri ve araçları ile *(debugger, disassembler, vb.)* algoritmalarının ortaya çıkartılması veya değiştirilmesini engellemeyi amaçlayan yöntemler bütünüdür.
- Yazılım güvenliğinin ele aldığı temel sorunlar, kodların açığa çıkmasını ve değiştirilmesini *(debugging, tracing ya da disassembly ile)* ve tersine mühendislik araçlarını *(debugger, disassembler, vb.)* engellemek şeklinde özetlenebilir.
- Tersine mühendislik için kullanılan araçları da kısaca tanıyacak olursak:
  - **Debugger:** Derlenmiş bir programın, çalışma anında assembly kodlarına dönüştürülmesi ve üzerinde değişiklik yapılmasını sağlayan araçlardır. En yaygın tersine mühendislik aracıdır. Çok çeşitlidir tersine mühendislerin en sevdiği araçtır.
  - **Disassembler:** Derlenmiş bir programın, çalıştırılmadan çözümlenmesini *(assembly kodlarına dönüştürme)* sağlayan tersine mühendislik aracıdır. Uygulama algoritmalarının ve denetim mekanizmalarının kavranarak çözümlenmesi amacıyla sıkça kullanılmaktadır.
- Özetleyecek olursak; yazılım güvenliğini ortaya çıkaran sorun tersine mühendislik araç ve yöntemlerinin kötüye kullanılmasıdır. Buna karşı yöntem geliştirmek çok fazla bilgi ve tecrübe gerektirdiğinden sıradan bir yazılımcı ya da programcılar tarafından engellenmesi de mümkün değildir.

Yazılım Güvenliğinde Etkileşim

- Gelişen bilgisayar ve yazılım teknolojileri yazılım güvenliğini daha da karmaşık hale getirmiştir.
- Dün, işletim sistemleri ve programlarımız sadece donanım ile etkileşim halinde iken bugün işletim sistemleri, işletim sistemi bileşenleri, kullanıcı bileşenleri, 3. parti bileşenler vs. ile etkileşimlidir.
- Etkileşim beraberinde güvenlik sorunlarını da getirmektedir.
- Böylesi karmaşık problemlerin çözümlenmesi için problemde soyutlama yapılarak, saldırıya en fazla maruz kalacak nesne *(base object)* tespit edilerek korunması sağlanır.
- Bu yaklaşım ise çoğu zaman etkileşimli bileşenlerin korunmasını göz ardı etmeyi gerektirir.

Yazılım ve Güvenlik

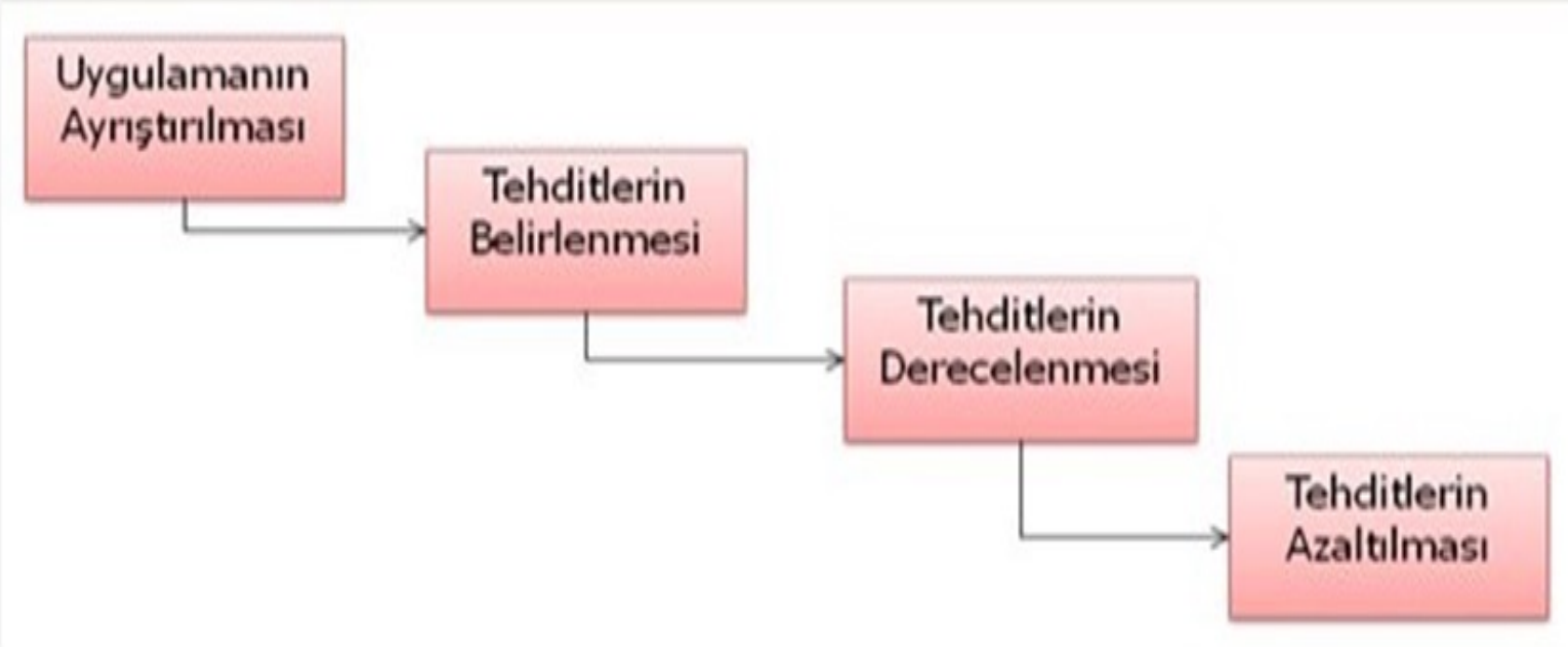
- Güvenliğin En Zayıf Halkası Çalışabilirler *(Programlar)*
- **Code:** Uygulamaların kodlarının bulunduğu bölümdür. Programların çalışabilir kodları burada bulunur. En fazla saldırıya maruz kalan bölümdür. Üzerinde değişiklik yapılması çok da kolay değildir. Temel assembly bilgisinden daha fazlasını gerektirir.
- **Import:** Uygulamaların diğer bileşenleriyle ve işletim sistemleriyle bütünleşme noktalarıdır. Programların çalışmasına doğrudan etkisi vardır. Ulaşılması ve değiştirilmesi oldukça kolay olduğundan saldırganlar tarafından sıkça kullanılmaktadır. Bölümün bir özelliği de, uygulamanın çalışmasına yönelik bilgiler bulundurmasıdır. Buradaki bilgiler normal kullanıcılar ya da program geliştiriciler için önemli gibi görünmesede, saldırgan için değerli bilgiler içerir. Bu bakımdan da saldırganlar için yüksek önceliğe sahiptir. Genel olarak bu bölüme yapılan saldırılar:
  - Bütünleşme adresleri üzerinde değişiklikler,
  - Saldırgan tarafından hazırlanan bileşenlerin uygulama etkileşimine dahil edilmesi,
  - Uygulama kodlarının ve verilerinin daha belleğe yüklenme sırasında değiştirilmesi ya da yönlendirilmesi,
  - Bütünleşmesi beklenen bileşenler yerine saldırgan tarafından hazırlanan bileşenleri ikame edilmesi, vb. yöntemler sayılabilir.
- **Resource:** Uygulama iç verilerinin tutulduğu ve ihtiyaç duyulduğunda karşılandığı bölümüdür. Programların çalışmasına dolaylı etkisi vardır. Ulaşılması ve değiştirilmesi oldukça kolaydır, ancak doğrudan çalışabilir olmadığından saldırganlar için daha düşük önceliğe sahiptir. Uygulama iç verileri *(resimler, sesler, ikonlar, vb.)* burada bulunduğundan başkaları tarafından da kolayca kopya edilebilir. Herhangi bir program üzerinden kopya edilen resimler, sesler ya da ikonlar değersiz gibi görünsede ülkemizde de patente konu olduğundan dikkatle korunması gerekmekir. Bu bölüm üzerinden saldırı yapmak daha fazla tecrübe gerektirdiğinden uzman saldırganlar tarafından tercih edilir.

Yazılım Geliştirmede Tehdit Modelleme

- Tehdit modelleme, organizasyonların yüksek düzeyli güvenlik risklerini anlamalarına yardımcı olan, güvenlik tabanlı bir analizdir.
- Tehdit modellemede amaç,
  - tehditlerin saptanması,
  - hangi tehditlerin azaltılmasının gerektiği
  - tehdit azaltma işleminde hangi yöntemlerin uygulanacağının belirlenmesidir.
- Risklerin azaltılması için tehdit modelleme kapsamında uygulanacak olan tehdit değerlendirmesi güvenli sistemler inşa etmek için olmazsa olmaz bir süreçtir.
- Bir organizasyonun ya da bir kurumun herhangi bir uygulama geliştirirken neden tehdit modellemeye gitmesi gerektiği şu maddelerle sıralanabilir:

Tehdit Modellemenin Faydaları

- **Uygulamanın daha iyi anlaşılması:** Uygulamanızın özelliklerinin analizi için zaman harcamak zorunda kalmanız, uygulamanızın ve parçalarının nasıl çalıştığı konusunda size daha geniş bir bakış açısı sağlayacaktır.
- **Tehditlerin belirlenmesi:** Herhangi bir yazılımsal süreç başlamadan, başka bir deyişle uygulama kodlanmaya başlamadan, sisteminizi ilgilendiren tehditlerin belirlenmesini sağlayacaktır.
- **Kod hatalarının kolayca belirlenmesi:** Kod hatalarının birçoğu tehdit modellemede ortaya çıkacaktır.
- **Proje takımının yeni üyelerinin uygulamaya uyumu:** Yeni işe başlayan birinin yüzde yüz performansla çalışmaya başlaması ve proje takımına uyum sağlaması için her zaman belirli bir sürenin geçmesi gerekmektedir. Tehdit modellemenin bütün uygulamayı gösterecek sistemli ve yapısal bir modelleme olması, yeni işe başlayan çalışanların uygulamayı hızlı bir şekilde öğrenmesini sağlayacaktır.
- **Testçiler için faydaları:** Testçilere hangi tehditlere göre test araçları hazırlamaları gerektiği hususunda katkı sağlayacaktır.
- **Diğer proje takımları için faydaları:** Sizin ürününüzle alakalı proje geliştiren diğer ürün geliştirme takımları, sizin oluşturduğunuz tehdit modellemeyi incelemelidirler. Bu sayede, diğer takımlar yeni bir tehdit modelleme oluşturmak zorunda kalmayacaktır ve uygulamanın tümünü görmek manasında proje hız kazanacaktır.



Şekil-1 Tehdit Modelleme Döngüsü

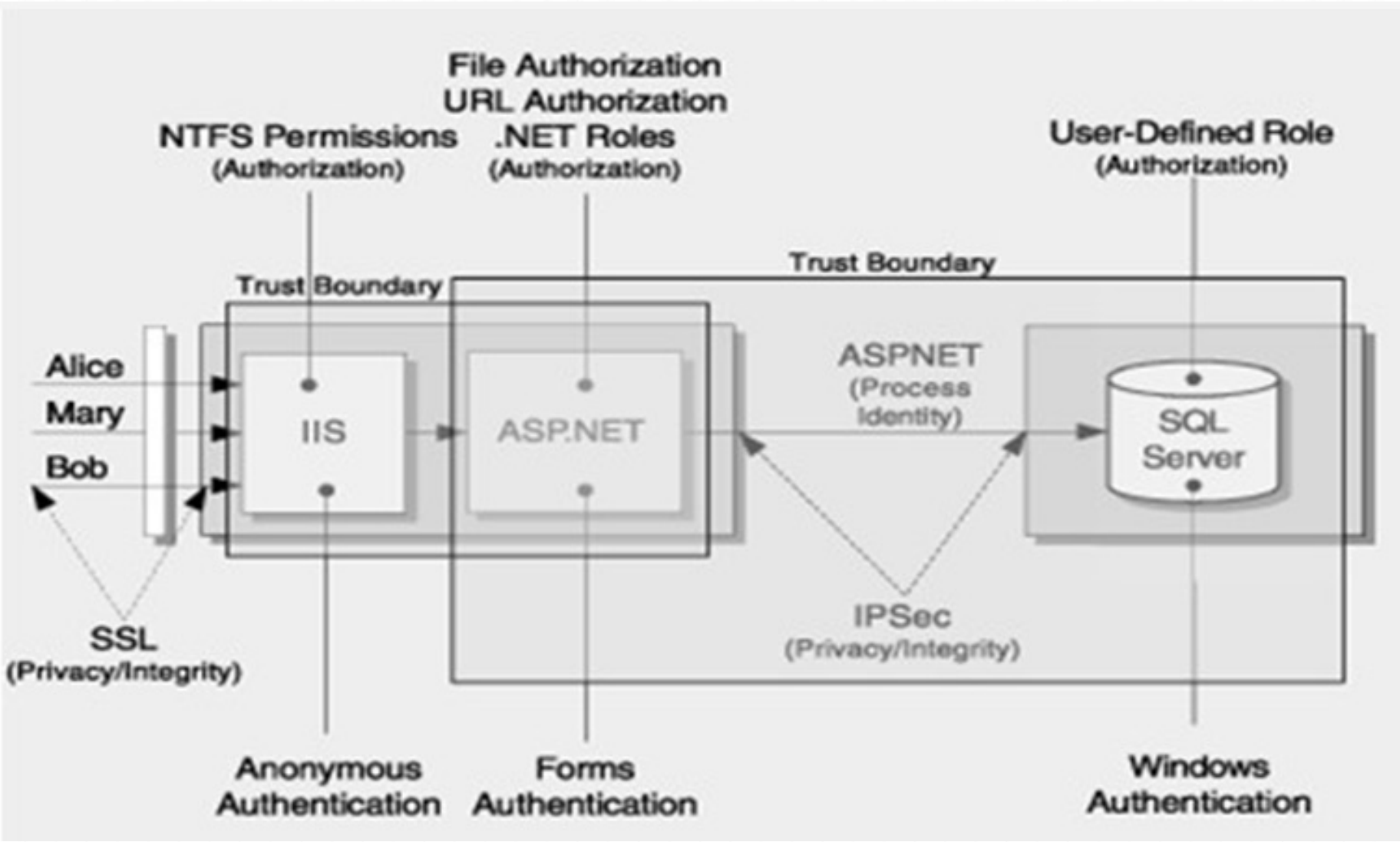
Uygulamanın Ayrıştırılması

- Açıklık tabanlı güvenlik profilinin oluşturulması için uygulama alt parçalara ayrılır.
- Güvenlik çemberi, akış diagramları, giriş noktaları, ayrıcalıklı kod parçacıklarının belirlenmesi ve güvenlik profilinin oluşturulması bu adımda gerçekleştirilir.
- Uygulamanız hakkında ne kadar detaylı bilgiye sahipseniz, tehditleri bulma olasılığınız o kadar yüksek olacaktır.

Uygulama Ayrıştırma		
Güvenlik Profili		Güvenlik Çemberi
Girdi Geçerleme	Oturum Yönetimi	Akış Diagramları
Kimlik Denetleme	Kriptografi	Giriş Noktaları
Yetkilendirme	Parametre Değşitirme	Ayrıcalıklı Kod
Yapılandırma Yönetimi	Hata Yönetimi	
Kritik Veri	Denetleme ve Kayıt	

Güvenlik Çemberinin Belirlenmesi

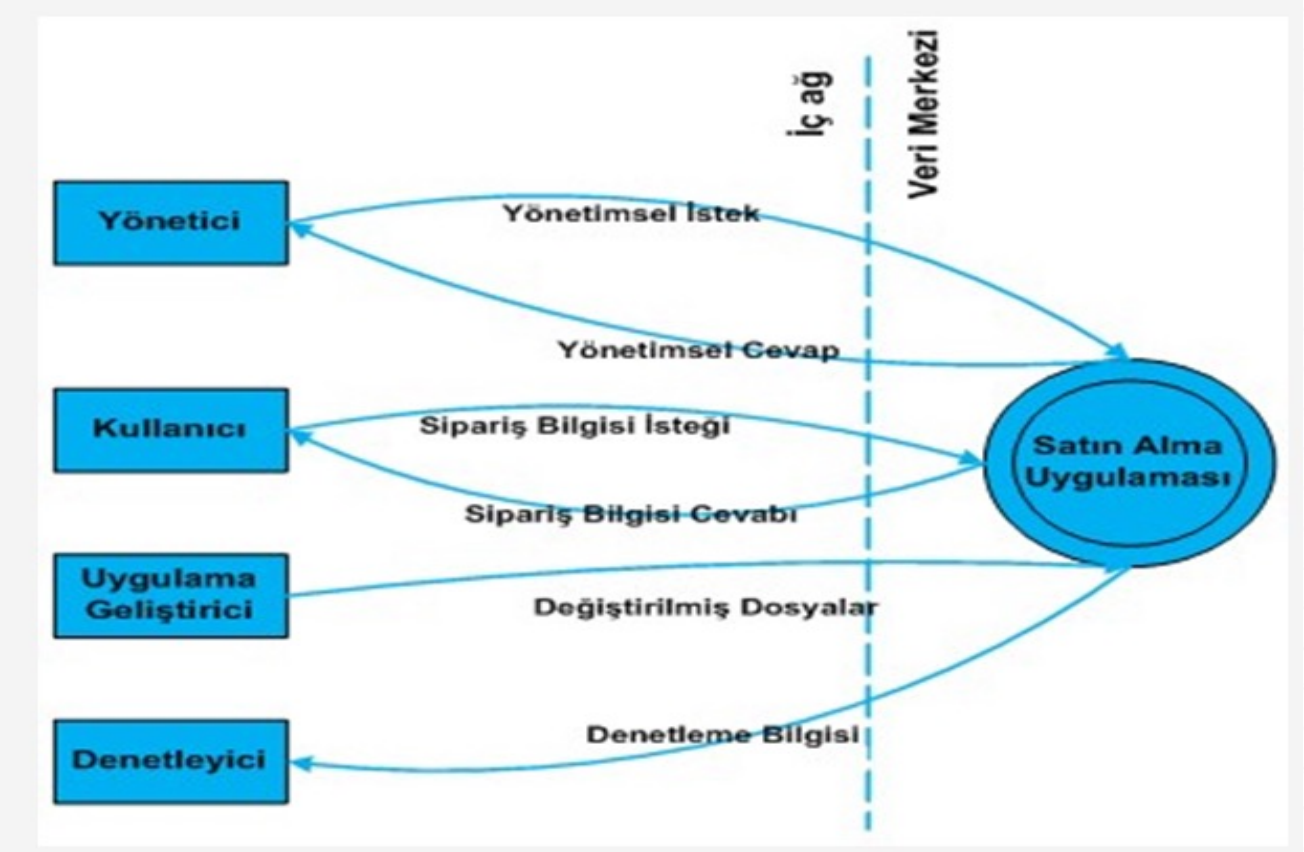
- Bütün varlıklarınızı çevreleyen güvenlik çemberlerinin belirlenmesi bu süreçte yapılır.
- Her bir alt sistem için, veri akışı veya kullanıcı girdisinin güvenliğinin belirlenmesi ve bu veri akışlarının ve kullanıcı girdilerinin nasıl yetkilendirildiği ve kimlik denetlemeye tabi tutulduğu belirlenir.
- Bununla beraber, dışardan çağırdığınız kod parçacığının güvenli olup olmadığını da incelenir.
- Güvenlik çemberinden genel kasıt şudur: O çemberin içindeki bütün giriş noktaları güvenli bir şekilde korunuyordur ve o çemberden geçen bütün veri girdi geçerlemeye tabi tutulmuştur.
- Ayrıca sunucu güvenlik ilişkilerinin de belirlenmesi gerekmektedir.
- Bir sunucu yetkilendirme ve kimlik denetleme gibi süreçleri başka bir sunucudan mı alıyor yoksa kendi mi bu mekanizmaları sağlıyor gibi soruların cevaplanması gerekmektedir.
- Aşağıdaki web uygulaması, ASP.NET Web uygulaması process account'ı güvenli olarak kabul ediyor ve veri tabanı sunucusuna bu güvenli hesap üzerinden ulaşıyor. Veri tabanı sunucu ise, uygulamaya yetkilendirme ve kimlik denetleme sürecinde güveniyor ve yalnızca doğrulanmış veriyi yetkilendirilmiş kullanıcıya dönüyor.


File Authorization
URL Authorization
NTFS Permissions
(Authorization)
.NET Roles
(Authorization)
User-Defined Role
(Authorization)
Trust Boundary
Trust Boundary
Alice
Mary
Bob
IIS
ASP.NET
ASPNET
(Process
Identity)
SQL
Server
SSL
(Privacy/Integrity)
IPSec
(Privacy/Integrity)
Anonymous
Authentication
Forms
Authentication
Windows
Authentication

Akış Diagramlarının Belirlenmesi

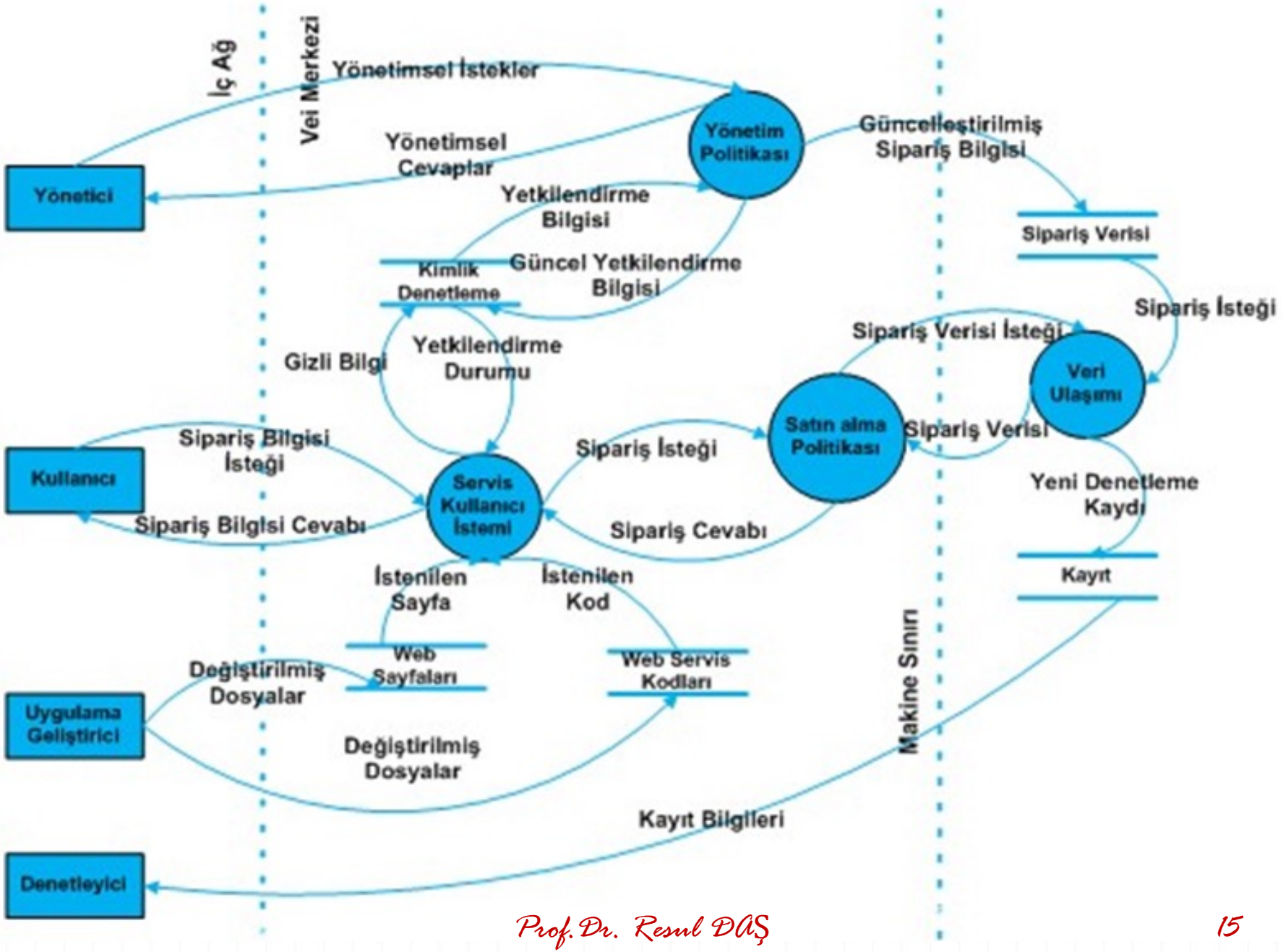
- Uygulama öncelikle bütünü kapsayacak şekilde, akış diagramı halinde gösterilir.
- Sonra uygulama parçalara bölünür ve her parça da büyüklüğüne göre parçalara bölünmeye devam eder ve akış diagramı halinde gösterilir.
- Örnek bir uygulama olarak basit bir satın alma uygulamasını ele alabiliriz.
- Bu uygulamada, yöneticilerin, kullanıcıların, uygulamayı geliştiricilerin ve denetleyicilerin olduğunu varsayalım.
- Yönetici uygulama yönetimiyle ilgili işlerden sorumlu kimse, kullanıcı sipariş bilgilerine bakabilen kimse, uygulama geliştirici uygulamayı geliştiren kimse ve denetleyici denetleme bilgilerini okuyan kimse olarak görev paylaşımında bulunmuş olsunlar.
- Bu uygulamanın öncelikle bütün uygulamayı temsil eden içerik diagramı aşağıda gösterilmiştir.

Akış Diyagramı



İçerik Diagramı

- İçerik diagramı hazırlanırken şu hususlara dikkat edilmelidir:
  - Olabildiğince yüzeysel bilgileri içermelidir. İçsel ve detaylı fonksiyonların incelenmesinden kaçınılmalıdır. Önemli olan kapsamın belirlenmesidir, detaylı fonksiyonların belirlenmesi değil.
  - Sistemin cevap vermek zorunda olduğu istekler bu diagramda belirlenir. Mesela, bir satın alma uygulaması, sipariş bilgisinin görüntülenmesi isteğiyle karşı karşıya kalabilir.
  - Uygulamanın bu isteklere nasıl cevap vereceği de bu diagramda gösterilir.
  - Herbir istekle ve cevapla alakalı olan veri kaynakları belirlenir.
  - Her cevabın muhatabı bu diagramda belirlenir.


İç Ağ
Veri Merkezi
Yönetimsel İstekler
Yönetimsel Cevaplar
Yönetici
Yönetim Politikası
Güncelleştirilmiş Sipariş Bilgisi
Yetkilendirme Bilgisi
Güncel Yetkilendirme Bilgisi
Kimlik Denetleme
Sipariş Verisi
Sipariş İsteği
Sipariş Verisi İsteği
Veri Ulaşımı
Gizli Bilgi
Yetkilendirme Durumu
Sipariş Bilgisi İsteği
Kullanıcı
Satın alma Politikası
Sipariş Verisi
Sipariş İsteği
Servis Kullanıcı İsteni
Sipariş Bilgisi Cevabı
Sipariş Cevabı
Yeni Denetleme Kaydı
Kayıt
İstenilen Sayfa
İstenilen Kod
Web Sayfaları
Web Servis Kodları
Değiştirilmiş Dosyalar
Uygulama Geliştirici
Değiştirilmiş Dosyalar
Makine Sınırı
Kayıt Bilgileri
Denetleyici

İçerik Diyagramı

- **Giriş Noktalarının Belirlenmesi**
  - Şunu hiçbir zaman unutmamak gerekir ki, sisteminizin giriş noktaları aynı zamanda saldırganlar için de giriş noktası manasındadır.
  - Öyleyse bu noktaların en iyi şekilde korunması gerekmektedir.
  - Örneğin, giriş noktalarınız HTTP isteklerini dinleyen web uygulaması içeriyor olabilir.
  - Normalde bu uygulama, son kullanıcıların kullanımına açılmış olan giriş noktalarıdır.
  - Ne var ki saldırganlar da bu noktaları kullanacaklardır.
  - Sisteminizdeki bütün giriş noktalarını bilmek zorundasınız. Her bir giriş noktası için, yetkilendirme ve kimlik denetleme süreçleri en iyi şekilde belirlenmelidir.
  - Mantıksal giriş noktaları şunları kapsayabilir: Web sayfaları, web servisleri için servis arayüzleri, mesaj kuyrukları vb... Fiziksel giriş noktaları ise portlar ve soketlerdir.
- **Ayrıcalıklı Kod Parçalarının Belirlenmesi**
  - Ayrıcalıklı kod parçaları, hassasiyeti yüksek veri kaynaklarına ulaştıkları ve ayrıcalıklı işlemler yapabildikleri için, dikkatlice ele alınmalıdırlar.
  - Hassasiyeti yüksek veri kaynağı olarak şu kaynaklar sıralanabilir:
  - DNS sunucları, dizin servisleri, çevresel değişkenler, olay günlükleri, dosya sistemleri, mesaj kuyrukları, performans sayaçları, yazıcılar, register sunucuları, soketler, Wen sunucuları vb.
  - Ayrıcalıklı kod parçasının güvensiz ya da potansiyel kötü niyetli bir yazılım tarafından kullanılmayacağına emin olunmalıdır.

İçerik Diyagramı

- Güvenlik Profilinin Oluşturulması için cevaplanacak sorular
- **Girdi Geçerleme**
  - Uygulamanız kapsamındaki bütün girdiler geçerlendi mi?
  - Saldırgan uygulamanıza kötü niyetli veri enjekte edebilir mi?
  - Veri ayrı güvenlik çemberlerinden geçerken, geçerlemeye tabi tutuluyor mu?
  - Veri tabanındaki veri güvenilir mi?
- **Kimlik Denetleme**
  - Şifre ve kullanıcı gibi gizli veriler ağ üzerinden geçerken güvenliği sağlanıyor mu?
  - Sıkı kullanıcı hesabı politikaları kullanılıyor mu?
  - Kullanıcılar güvenlik düzeyi yüksek şifre almaya zorlanıyorlar mı?
  - Kullanıcı şifreleri için şifre doğrulama uygulamaları kullanılıyor mu?
- **Yetkilendirme**
  - Giriş noktaları için hangi önlemler alındı?
  - Veri tabanında yetkilendirme nasıl şart koşuluyor?
  - Bağlantı kopmasında ya da hata anında uygulama güvenliği sağlanıyor mu?
- **Yapılandırma Yönetimi**
  - Uygulamanız hangi yönetimsel arayüzleri içeriyor?
  - Bu arayüzlerin güvenliği nasıl sağlanıyor?
  - Uzaktan yönetici yetkilendirmesi nasıl yapılıyor?
  - Hangi yapılandırma bilgileri saklanıyor ve bunların güvenliği nasıl sağlanıyor?
- **Kritik Veri**
  - Uygulama kapsamında hangi kritik bilgiler işlenmektedir?
  - Bu gizli bilgiler olduğu yerde ve ağ üzerinde nasıl korunuyor?
  - Nasıl bir şifreleme tekniği kullanılıyor ve şifreleme anahtarları nerede tutuluyor?

Güvenlik Profilinin Oluşturulması

- **Oturum Yönetimi**
  - Oturum tanımlama bilgileri nasıl tutuluyor?
  - Oturum bilgilerinin çalınmaması için ne gibi tedbirler alınıyor?
  - Oturum bilgileri ağ üzerinden geçerken güvenliği nasıl sağlanıyor?
- **Kriptografi**
  - Hangi şifreleme algoritmaları ve yöntemleri kullanılmaktadır?
  - Şifreleme anahtarları ne kadar süredir kullanılmakta ve bunların güvenliği nasıl sağlanmakta?
  - Uygulamanın kullandığı kendi şifreleme tekniği var mı?,
  - Şifreleme anahtarları ne kadar sürede bir değiştirilmektedir?
- **Parametre Değiştirme**
  - Uygulama oynanmış parametreyi tespit edebiliyor mu?
  - Form alanlarındaki, HTTP başlıklarındaki, oturum tanımlama bilgilerindeki parametreleri geçerliyor mu?
- **Hata Yönetimi**
  - Uygulama hata durumlarının nasıl üstesinden geliyor?
  - Hata bilgileri kullanıcıya iletilliyor mu?
  - Bu hata bilgilerinde sistemin kritik bilgilerini içeren uyarılar bulunuyor mu?
- **Denetleme ve Kayıt**
  - Uygulamanız bütün sunucu ve katmanlarda denetleme ve kayıt mekanizmasına sahip mi?
  - Kayıt dosyalarının güvenliği nasıl sağlanıyor?

Tehditlerin Belirlenmesi

- Uygulama ayrıştırıldıktan sonra, her bir bileşen tehdit modelleme için potansiyel tehdit olarak ele alınır.
- Burada amaç her şeyin nasıl çalıştığını görmekten ziyade, uygulamanın bileşenlerini anlamak ve bu bileşenler arasında veri alışverişinin nasıl yapıldığını görmektir.
- Bileşenler genel olarak tehdit hedefleri olarak adlandırılır.
- Belirlenen tehditlerin azaltılması için öncelikle tehditlerin sınıflandırılması gerekmektedir.
- Bunun için STRIDE sınıflandırma modeli kullanılabilir.
- STRIDE modeli aşağıda belirtilmiştir:

Tehditlerin Belirlenmesi

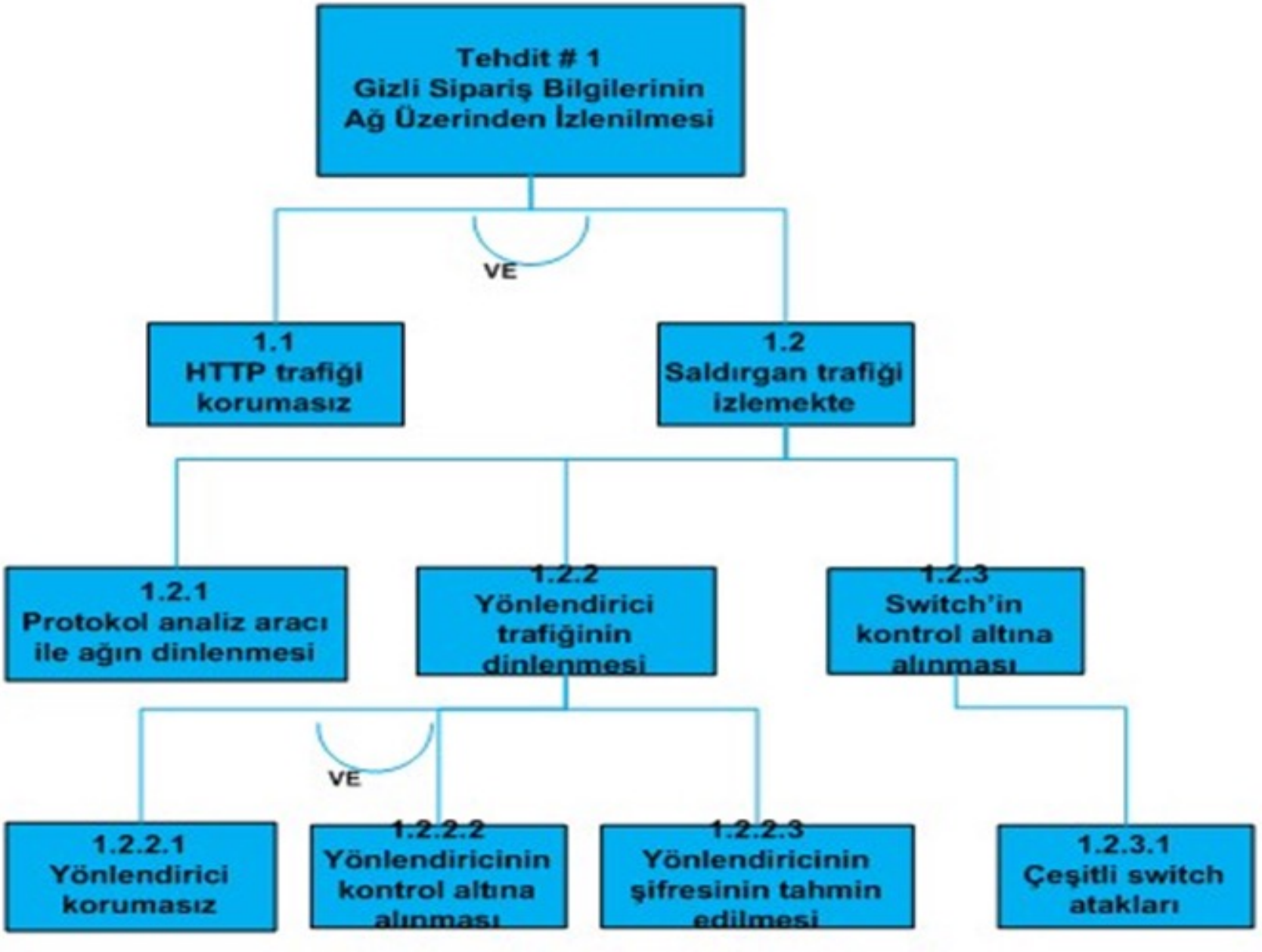
- **Kimlik Yanıltması (Spoofing Identity):** Yanıltma tehditleri saldırganların başka bir kullanıcı gibi davranmaları ya da geçerli bir sunucu gibi davranarak diğer sunucuları yanıltma amacı güder. Kimlik yanıltmaya örnek olarak, yasa dışı olarak başka bir kullanıcının şifre ve kullanıcı ismi gibi gizli bilgilerine ulaşmayı ve bu bilgileri kullanmayı gösterebiliriz.
- **Verinin Değiştirilmesi (Tampering with Data):** Bu işlem verinin kötü niyetli bir şekilde değiştirilmesi manasına gelir. İnternet üzerinden akan verinin değiştirilmesi buna örnek olarak gösterilebilir.
- **Inkar (Repudiation):** Bu tip tehditler, bazen saldırganın gerçekleştirdiği ama bunu inkâr edebileceği tehditler olarak görülür. Inkar tipi tehditler kredi kartı gibi işlemlerde kendini gösterir. Kullanıcı bazı alımlarda bulunur ve sonra yapmadığını iddia eder. Başka bir örnek de e-posta örneği olabilir. Size gelen bir maili, karşı taraf ben atmadım diye inkâra başvurabilir.
- **Bilginin Açığa Çıkarılması (Information Disclosure):** Bu tip tehditler, bazı bilgilerin onları görmemesi gereken kullanıcılar tarafından görünmesini kapsar. Örnek olarak, bir kullanıcının okumaması gereken bir dosyaya erişim hakkı olması ya da bir saldırganın ağ trafiğini dinleyip bazı bilgileri elde etmesidir.
- **Hizmet Dışı Bırakma (Denial of Service):** Hizmet dışı bırakma atakları sistemlerin kullanılmaz hale gelmesini sağlayan ataklardır. Web sunucusunun kullanılmaz hale gelmesi bu tip ataklara örnek olarak gösterilebilir. Sistemlerin güvenirliliği ve sürekliliği sağlamaları adına bu tip ataklara önlem almaları gerekmektedir.
- **Yetki Kazanma (Elevation of Privilege):** Bu tür ataklarda, yetkisiz bir kullanıcı sistemi kullanmak üzere yetki kazanır ve bütün sisteme zarar vermek ya da kontrolü altına almak için sisteme giriş yapar.

Örnek Saldırı

- Bu sınıflandırmaya dayalı olarak, örnek satın alma uygulamasında örnek bir tehdit üzerinde durabiliriz.
- Bu tehdit şekilde gösterilmiştir. Burada dikkat edilmesi gereken husus, bir kullanıcının başka bir kullanıcıya ait sipariş bilgilerini görmemesi gerektiğidir.
- Öyleyse, bilginin çalınmasına ya da başka kullanıcılar tarafından görüntülenmesine engel olunmalıdır.
- Bu tehdit bilginin açığa çıkarılması tehdit grubuna örnek bir tehdittir.
- Bir saldırganın bu veriyi görmesinin birçok yolu olabilir.
- Fakat en basit haliyle, saldırgan ağ trafiğini, protokolleri analiz eden bir araçla dinliyor olabilir; ya da ağda bir yönlendiriciyi kontrol altına almış olabilir.

Tehdit Ağaçları

- Tehdit ağaçlarının arkasındaki ana fikir bütün sistem tehdit hedeflerinden bir araya getirilmiş bir sistemler bütünüdür ve her tehdit hedefi bazı açıklıklara sahip olabilir.
- Bu açıklıklar da bütün sistemin saldırganlar tarafından kullanılması manasına gelebilir.
- Tehdit ağaçları, bir saldırganın sistemi ele geçirebilmek için izleyebileceği yolları çizen yapısal ağaçlardır.
- Uygulama ayrıştırması işlemi gerçekleştikten sonra, her bir bileşen için tehditlerin belirlenmesi süreci başlar.
- Tehditler belirlendikten sonra ise bu tehditlerin nasıl ortaya çıkabileceği tehdit ağaçlarıyla belirlenir.
- Yukarıda verilen tehdit örneği için, belirlenen tehdit ağacı aşağıdaki gibi olabilir.
- Tehdit ağacının en başındaki kutu, tehdit kutusu olarak adlandırılır ve altındaki kutular tehditin gerçeklenmesi için gereken adımları temsil eder.
- Bu örnek tehditte, tehditin gerçellenmesi için, HTTP trafiği korumasız olmalı ve saldırgan trafiği dinliyor olmalı.
- Saldırganın trafiği dinlemesi için, ağ trafiğini dinliyor olması ya da yönlendirici üzerinden geçen veriyi dinliyor olması gerekir.
- Yönlendirici dinleme senaryosunun gerçek olması için, ya hedef yönlendirici korumasız olmalı ve kontrol altına alınması gerekiyor ya da yönlendiricinin şifresinin bilinmesi gerekiyor.


Tehdit # 1
Gizli Sipariş Bilgilerinin
Ağ Üzerinden İzlenilmesi
VE
1.1
HTTP trafiği
korumasız
1.2
Saldırgan trafiği
izlemekte
1.2.1
Protokol analiz aracı
ile ağın dinlenmesi
1.2.2
Yönlendirici
trafiğinin
dinlenmesi
1.2.3
Switch'in
kontrol altına
alınması
VE
1.2.2.1
Yönlendirici
korumasız
1.2.2.2
Yönlendiricinin
kontrol altına
alınması
1.2.2.3
Yönlendiricinin
şifresinin tahmin
edilmesi
1.2.3.1
Çeşitli switch
atakları

Tehditlerin Derecelendirilmesi

- Tehdit ağaçları oluşturulduktan ve tehditler anlaşıldıktan sonra, belirlenen bu tehditlerin derecelendirmesi gerekmektedir.
- Derecelendirmeli amaç, öncelikle hangi tehdit üzerinde duracağımıza karar vermek ve bir sıralama oluşturmaktır.
- Uygulamamız için en büyük tehditin ilk önce giderilmesi mantıklı bir yaklaşım olacaktır.
- Derecelendirme yaparken bazı derecelendirme yöntemleri kullanılabilir.
- Bu derecelendirme yöntemlerinden DREAD adlı yöntemi örnek satın alma uygulamamız için kullanacağız.

DREAD Modeli

- **Zarar Potansiyeli (Damage Potential):** Tehdidin gerçekleşmesi durumunda, zarar ne kadar büyüklükte olacaktır. En kötü durum 10 ile gösterilebilecekken, en hafif durum 0 ile gösterilebilir. Yetki tehditleri genellikle 10 ile gösterilir. Tıbbi, mali, ya da askeri veri içeren uygulamalar içi zarar potansiyeli yüksek kabul edilir.
- **Uygulanabilirlik (Reproducibility):** Potansiyal bir saldırının başarıya ulaşma şansını belirlemeye çalışır. Belirli bir tehditin, saldırgan için başarılı bir saldırı olma kolaylığının incelenmesini ön görür.
- **Sömürülebilirlik (Exploitability):** Bu tehdit kapsamında bir atağı gerçellemek için ne kadar tecrübe ve çaba gerekmektedir. Örneğin, acemi bir kullanıcı, normal ev bilgisayarı ile tehdite bir atak uyguluyabiliyorsa bu ciddi bir durumdur ve bu tehdit sömürülebilirlik manasında 10 alabilir.
- **Etkilenen Kullanıcılar (Affected User):** Eğer bu tehdit bir saldırıya dönüştürülür ve saldırı başarılı olursa ne kadar kullanıcının bundan zarar göreceği araştırılır. Mesela, bir sunucuyla alakalı bir tehdit, birçok kişiyi ilgilendireceğinden yüksek önem arz eder.
- **Keşfedilebilirlik (Discoverability):** Açıklığın kolay ya da zor bulunabileceğinin ya da keşfedilebileceğinin değerlendirmesi yapılır.

DREAD Hesabı

- Bu değerlendirme kriterine göre, yukarıda verdiğimiz örnek tehdidin değerlendirmesi şu şekilde olabilir:

- *Tehdit #1:* Kasıtlı kullanıcı başka bir kullanıcının sipariş bilgilerini ağ üzerinden dinler.
  - **8** Zarar Potansiyali: Başkasının gizli sipariş bilgilerini okuyabilmek ciddi bir sorun.
  - **10** Uygulanabilirlik: Tamamen uygulanabilir.
  - **7** Sömürülebilirlik: Ağ üzerinde yer edinmeli ve ağı dinlemeli.
  - **10** Etkilenen Kullanıcılar: Yöneticiler dâhil herkes etkilenebilir.
  - **10** Keşfedilebilirlik: Bu açıklığın kolayca bulunacağını varsayalım.

DREAD ile ölçülen risk oranı = (8+10+7+10+10)/5 = **9**

Tehditlerin Azaltılması

- Tehditler belirlendikten ve önem sırasına göre sıralandıktan sonra her tehdit için ne yapılması gerektiği kararlaştırılmalı. Bu karar dört farklı şekilde olabilir:
- **Herhangi bir tedbir almama:** Bu yöntem genellikle doğru olmayan bir yöntemdir. En sonunda uygulamanın içerisindeki bu açıklık başınızı ağrıtabilir ve önlem almak zorunda kalabilirsiniz. Daha sonradan alacağınız önlem çok daha pahalıya size mal olabilir.
- **Kullanıcıyı uyar:** Diğer bir yol ise, kullanıcının bu hususta bilgilendirilmesi ve uygulamanın o özelliğini kullanıp kullanmayacağının kendisine bırakılmasıdır. Buna örnek olarak Microsoft Internet Bilgi Servisi (IIS) verilebilir. Bu sistem, yöneticiyi eğer SSL/TLS kullanmazsa kullanıcı bilgilerinin şifresiz bir şekilde ağda gezeceğini belirtir. Bu yöntem de birçok durumda problemli bir yöntemdir. Kullanıcıların çoğu nasıl seçim yapacaklarını bilmezler. Aslında çoğu zaman, kullanıcının karşısına çıkarılan seçenek, kullanıcı için anlaşılması zor bir seçenektir. Diğer bir durum ise, kullanıcılar genellikle hazır ayarları kabul ederler ya da uyarıları görmezden gelirler.
- **Problemi ortadan kaldır:** Problemi çözmek için yeterli zamanınız yoksa kısaca tehdit altında olduğunu düşündüğünüz özelliğe sahip parçayı sistemden çıkarmanız yeterlidir.
- **Problemi çöz:** En kesin ve gerçekçi çözüm budur. Teknolojiyi kullanarak, problemi azaltma ya da tamamen çözme yoluna gidilmelidir. Bu yöntem aynı zamanda en zor yöntemdir. Tasarımcılar, geliştiriciler, testçiler ve güvenlik ekibi için daha fazla iş yükü ve daha fazla maliyet demektir.

Güvenli Yazılım Geliştirme ve En Tehlikeli 25 Hata

Sıra	Puan	ID	İsim
[1]	93.8	CWE-89	SQL Komutlarında kullanılan özel elemanların uygunsuz nötrleştirilmesi ('SQL Injection')
[2]	83.3	CWE-78	İşletim sistemi komutlarında kullanılan özel elemanların uygunsuz nötrleştirilmesi ('OS Command Injection')
[3]	79.0	CWE-120	Girdi boyutunu kontrol etmeden ara bellek kopyalama ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Web sayfası oluşturulurken girdilerin uygunsuz nötrleştirilmesi ('Cross-site Scripting')
[5]	76.9	CWE-306	Kimlik Doğrulama gerektirmeyen kritik fonksiyonlar
[6]	76.8	CWE-862	Yetkilendirme eksikliği
[7]	75.0	CWE-798	Kaynak kod içine gömülmüş(hard-coded) kimlik doğrulama
[8]	75.0	CWE-311	Hassas verilerde şifreleme eksikliği
[9]	74.0	CWE-434	Sınırlandırılmamış tehlikeli türde dosya yüklenmesi
[10]	73.8	CWE-807	Güvenilmeyen girdilere dayalı güvenlik kararları
[11]	73.1	CWE-250	Gereksiz haklarla uygulama çalıştırma
[12]	70.1	CWE-352	Siteler arası istek sahteciliği (CSRF)

Güvenli Yazılım Geliştirme ve En Tehlikeli 25 Hata

[13]	69.3	CWE-22	Kısıtlanmış dizine erişime sebep olan uygunsuz dosya yolu sınırlaması ('Path Traversal')
[14]	68.5	CWE-494	Bütünlük kontrolu yapılmadan kod indirilmesi
[15]	67.8	CWE-863	Hatalı yetkilendirme
[16]	66.0	CWE-829	Güvenilmeyen kontrol alanından fonksiyonellik dahil edilmesi
[17]	65.5	CWE-732	Kritik kaynağa hatalı izin tahsisi
[18]	64.6	CWE-676	Muhtemelen tehlikeli fonksiyonların kullanılması
[19]	64.1	CWE-327	Kırılmış veya riskli şifreleme algoritmalarının kullanılması
[20]	62.4	CWE-131	Ara bellek boyutunun hatalı hesaplanması
[21]	61.5	CWE-307	Kimlik doğrulama teşebbüslerinin uygunsuz sınırlandırılması
[22]	61.1	CWE-601	Güvenilmeyen siteye URL yönlendirilmesi ('Open Redirect')
[23]	61.0	CWE-134	Kontrol edilmeyen metin formatı
[24]	60.3	CWE-190	Tamsayı Taşması veya Sarımı (Integer overflow or wraparound)
[25]	59.9	CWE-759	Tuzlanmamış kriptografik özet kullanılması

Risk Azaltma Konuları

ID	Name
M1	Tüm girdileriniz üzerinde kontrol mekanizması kurup devam ettiriniz.
M2	Tüm çıktılarınız üzerinde kontrol mekanizması kurup devam ettiriniz.
M3	Uygulama ortamınızı izole ediniz.
M4	Harici bileşenlerin suistimal edilebileceğini ve kaynak kodunuzun herkes tarafından okunabileceğini varsayın.
M5	Kendi yöntemlerinizi geliştirmek yerine, sektör tarafından kabul görmüş güvenlik ögelerini kullanın.
GP1	*(genel)* Açıklıklardan kaçınmanızı kolaylaştıracak kütüphane ve çerçeveleri kullanın.
GP2	*(genel)* Tüm yazılım geliştirme döngünüze güvenliği entegre edin.
GP3	*(genel)* Kapsamlı bir şekilde açıklıkları bulmak ve önlemek için geniş çeşitlilikte metodlar uygulayın.
GP4	*(genel)* İzole ortam kullanan istemcilere müsade edin.

Sonuç

- Alınması gereken en temel önlemler, risklere karşı sürekli uyanık olmak, bu çalışmada açıklanan saldırı tekniklerine karşı uyanık olmak, yeni gelişmeler ışığında gerekli güncellemeleri yaparak saldırılardan etkilenme olasılığını en aza indirgemek olarak belirtilebilir. Güvenliğin statik değil dinamik bir sürece sahip olduğu, koruma ve sağlamlaştırma ile başladığını, bir hazırlık işlemine ihtiyaç duyulduğu, saldırıların tespit edilmesinden sonra hızlıca müdahale edilmesi gerektiği ve sistemde her zaman iyileştirme yapılması gerektiği unutulmamalıdır.

Sorular