Güvenli Yazılım Geliştirme Modelleri ve Ortak Kriterler Standardı

Erkut Beydağlı[1] Mehmet Kara[2] Hayretdin Bahşi[3] Erdem Alparslan[4]

[1,2,3,4] TÜBİTAK-UEKAE, Kocaeli

[1]e-posta: beydagli@uekae.tubitak.gov.tr [2] e-posta: mkara@uekae.tubitak.gov.tr

[3] e-posta: bahsi @uekae.tubitak.gov.tr [4]e-posta: ealparslan@uekae.tubitak.gov.tr

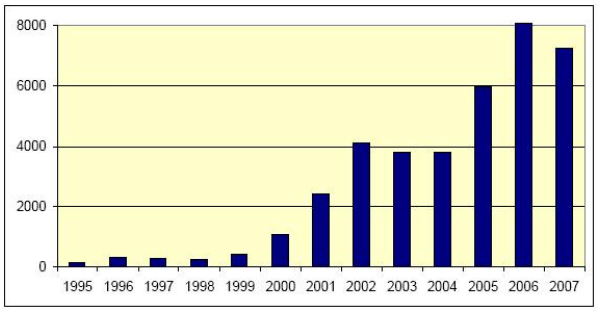
Özetçe

Bu makalede, iş uygulamalarının bilgisayar ortamında yaygın olarak kullanılması ve internet kullanımının artmasıyla birlikte büyük önem kazanan güvenli yazılım geliştirme konusunda geçmişten günümüze ortaya konan güvenli yazılım geliştirme modelleri, standartları ve anaçatıları (Framework) incelenmiştir. Hali hazırda güvenli yazılım geliştirmek için kullanılan modellerin Ortak Kriterler standardının (Common Criteria/ISO 15408) sunduğu güvenlik yaklaşımı ile karşılaştırılmaları yapılmış ve Ortak Kriterler standardının güvenli yazılım üretmeye etkisi irdelenmiştir.

1. Giriş

Bilgisayarların ilk ortaya çıkmasıyla birlikte yazılım geliştirme süreci de başlamıştır. Bu süreç 1940'lı yıllara kadar gitmektedir. İlk yıllarda geliştirilen yazılımlarda görülen en büyük eksiklik yazılım projelerinin zamanında tamamlanamaması ve istenilen kalitede (dokümantasyon, fonksiyonellik, harcanan fazla iş gücü) olmamasıdır. Bunlara ek olarak teknolojinin hızla değişmesi, birçok yazılımın hız ve fonksiyonellik eksikliklerinden dolayı devre dışı kalmasına ve işin yeniden projelendirilmesine neden olmuştur. 1980'li yıllarda Fred P. Brooks "No Silver Bullet" [1] makalesi ile bu konuda tek parametre ile başarı sağlamanın mümkün olmadığın ifade etmiştir. Bu problemlerin tespit edilmesinden sonra yeni diller (C++, Java vb.), yeni araçlar, yeni metodolojiler ve yeni disiplinler geliştirilmeye çalışılmıştır. İlk önceleri yazılım projelerinin zamanında, istenilen iş gücü ile belli bir disiplin içerisinde geliştirilmesi için çalışmalar yapılmıştır. İnternetin yaygın olarak kullanılmaya başlanması ile yazılımlar içerdikleri güvenlik zayıflıkları/açıklıkları nedeniyle kötü niyetli veya bilinçsiz kullanıcılar tarafından kötüye kullanılmaya başlanmış ve e-ticaret, e-devlet, e-sağlık gibi günlük hayatımızdaki işlerin yazılımlar aracılığı ile güvenli olarak sağlanması riskli olmaya başlamıştır. Şekil 1'de CC CERT (Coordination Center Computer Emergency Response Team) tarafından yayınlanan yıllara göre açıklıkların sayısı görülmektedir.

Şekil 1'den görüleceği gibi son yıllarda yazılımlarda görülen açıklıklar önemli oranda artmaktadır. Bu açıklıkların artma nedenlerinin başında internetin yaygın olarak kullanılması ve bilgisayarın iş uygulamaları da dahil olmak üzere kullanım oranının çok artması gelmektedir Diğer önemli sebep yazılımların güvenliği dikkate alınmadan sadece fonksiyonellik göz önüne alınarak geliştirilmesidir. Bu durumda yazılımlar birçok açıklık içermektedir. Ayrıca yazılımın ortaya çıkmasından sonra tespit edilen açıklıkların kapatılması için çok fazla iş gücü ve zaman harcanmaktadır. Çünkü bir güvenlik açıklığı yazılım geliştirme evresinde ne kadar erken tespit edilebilirse, o açıklığın kapatılması maliyeti o kadar az olacaktır.



Şekil 1 Yıllara göre açıklıklar

Yazılım geliştirme sürecinin yaygın olarak kullanılmaya başlandığı ilk yıllarda kaliteli ve olgun yazılım üretmek için, son yıllarda ise özellikle güvenli yazılım geliştirmek için çok sayıda model ve anaçatı üzerinde çalışılmıştır. Bu süreç CMM (Capability Maturity Model) ile başlamış daha sonrasında CMMI (Capability Maturity Model Integration), FAA–iCMM (Federal Aviation Administration integrated Capability Maturity Model), Trusted CMM, SSE-CMM (Security System Engineering CMM), Microsoft Security Development Lifecycle, OpenSAMM (Software Assurance Maturity Model) modelleri gibi modeller geliştirilmiştir.

Yazılım/Donanım ve sistem güvenliği değerlendirmesi konusundaki ilk çalışmalar, Orange Book olarak da bilinen TCSEC (Trusted Computer System Evaluation Criteria) standardının 1983 yılında Amerika Birleşik Devletleri Savunma Bakanlığı (USA Department of Defense) tarafından yayınlanması ile başlamıştır. 1980'li yıllarda Avrupa'da; İngiltere, Almanya, Fransa ve Hollanda kendi güvenlik test metodolojilerini oluşturmuşlardır. Daha sonra bu ülkeler değerlendirme standartları arasındaki farkları ortadan kaldırmak ve bir yerde yapılan değerlendirmenin her yerde geçerli olmasını sağlayabilmek için 1991 yılında ITSEC (Information Technology Security Evaluation Criteria) standardını oluşturmuşlardır. Kanada'da ITSEC ve TCSEC'den faydalanarak 1993 yılında milli değerlendirme standardı CTCPEC (Canadian Trusted Computer Product Evaluation Criteria)'i yayınlamıştır.

Avrupa, Kanada ve Amerika Birleşik Devletleri'nde üretilen yazılım/donanım ve sistemlerin farklı standartlara göre güvenlik değerlendirmelerinin gerçekleştirilmesi, uluslararası satılan ürünlere uygulanmış testlerin diğer ülkelerde anlaşılamamasına, yazılım/donanım ve sistem güvenliği konusundaki çalışmaların farklı ülkeler farklı şekilde

geliştirilmeye çalışılması sorunlara sebep olmuştur. Bu sorunların önüne geçilebilmesi için Kanada, Fransa, Almanya, İngiltere, Avustralya, Yeni Zelanda ve Amerika Birleşik Devletleri 1996 yılında bir araya gelerek Ortak Kriterler (Common Criteria) standardı sürüm 1.0'ı yayınlamışlardır. Ortak Kriterler Standardı ürün güvence değerlendirmesini EAL1 (Evaluation Assurance Level)'den EAL7'ye kadar 7 seviyede yapmaktadır. EAL1 seviyesinde fonksiyonellik, kullanıcı kılavuzları gibi üst seviye bilgiler kontrol edilmekle birlikte seviyeler arttıkça kontrol edilen bilgiler de (üst seviye tasarım, detaylı tasarım, kaynak kodu inceleme, açıklık analizleri, matematiksel olarak doğrulma vb.) artmaktadır. Hali hazırda standardın 3.1 sürümü kullanılmaktadır. Ortak Kriterler standardı 25 ülke tarafında kabul edilmiştir. Ortak Kriterler aynı zamanda ISO tarafından EN ISO/IEC 15408 standardı olarak yayınlanmıştır.

Bu çalışmada amaç, Ortak Kriterler standardı bir BT(Bilişim Teknolojileri) ürünü güvenlik değerlendirme standardı olmasına rağmen özellikle EAL4 (Evaluation Assurance Level) ve üzeri değerlendirmelerde yazılım geliştiricilerinin güvenli bir yazılım geliştirme döngüsü kurmasın gerektirdiğini ve bu konuda geliştiricilere standardın rehberlik ettiğini ortaya koymaktır. Bu hedefi desteklemek amacıyla güvenli yazılım geliştirmenin önemli unsurları belirlenmiş, bu unsurlar çerçevesinde Ortak Kriterler standardı diğer güvenli yazılım geliştirme anaçatı ve standardları ile karşılaştırılmıştır.

Makale giriş, yazılım geliştirme modelleri, Ortak Kriterlerin güvenli yazılım geliştirme sürecine yaklaşımı, modellerin güvenliğe bakışı ve sonuçlar olmak üzere beş bölümden oluşmaktadır.

Bölüm 2'de başlangıçtan günümüze yazlım geliştirme ve güvenli yazılım geliştirme çalışmaları güvenlik özelliklerini de içerecek şekilde incelenmiştir.

Bölüm 3'te Ortak Kriterler standardının ürün (yazılım-donanım) geliştirme sürecine katkıları irdelenmiştir.

Bölüm 4'te geliştirilen modellerin güvenli yazılım geliştirilmesine katkıları karşılaştırılmalı olarak incelenmiştir.

Bölüm 5'te ise Ortak Kriterler ve diğer modellerin sundukları güvenlik özelliklerinin karşılaştırılmalarına ait yorumlar verilmiştir.

2. Yazılım Geliştirme Modelleri

2.1. Yetenek Olgunluk Modeli (CMM - Capability Maturity Model)

CMM, Amerikan Savunma Bakanlığı'nın isteği üzerine Carnegie Mellon Üniversitesi'ne bağlı Yazılım Mühendisliği Enstitüsü tarafından 1986 yılında geliştirilmeye başlanmıştır. Yetenek Olgunluk Modeli (Capability Maturity Model) belirli mühendislik disiplinleri için referans olgunluk modeli sağlar. Bir organizasyon kendi pratik uygulamalarını muhtemel iyileştirmeleri sağlamak için model ile karşılaştırır. CMM özel süreçler (yazılım mühendisliği, sistem mühendisliği, güvenlik mühendisliği) için hedef aşamalar tanımlar. Fakat bu işlemler için rehber dokümanlar sağlamaz. Bu süreçler ne istendiğini tanımlar fakat nasıl olacağını tanımlamaz. "CMM tabanlı değerlendirmeler ürün değerlendirmesinin ya da sistem serifikasyonunun yerini alacağı anlamına gelmez. Daha çok organizasyon değerlendirmesi ile ilgilenir yani özel alanlardaki zayıflıkların iyileştirilmesine odaklanır"[2].

2.2. Tümleşik Yetenek Olgunluk Modeli (CMMI Capability Maturity Model Integration)

CMMI (Capability Maturity Model Integration) uzun vadede organizasyonun iş performansının olgunluğunun artırmasına yardım etmektedir. CMMI, organizasyonların işlemlerinin yeteneğini ve olgunluğunu değerlendirmek için servis geliştirme, bakım, satın alma mekanizmaları için en iyi pratikleri sunmaktadır. Bu model tarafında içerilen geliştirme alanları, sistem mühendisliği, yazılım mühendisliği, tümleşik ürün ve süreç geliştirme, tedarikçi kaynakları ve satın alma alanlarıdır. CMMI, CMM'in üzerine geliştirilmiş olup sekiz yıldan beri kullanılmaktadır. CMMI geniş bir kullanım oranına sahiptir. Mart 2009'da Software Engineering Institute 3446 organizasyon ve 21141 projenin CMMI kullandığını açıklamıştır[3].

CMMI'da süreç iyileştirme ve değerlendirme dört kategoride incelenir. Bunlar Proje Yönetimi, İşletme Yönetimi, Mühendislik ve Destektir. CMMI süreçleri içerisinde doğrudan güvenlikle ilgili bir süreç yoktur.

2.3. Federal Havacılık Yönetim Tümleşik Yetenek Olgunluk Modeli (FAA-iCMM Federal Aviation integrated Maturity Model)

FAA-iCMM federal havacılık yönetiminde yaygın olarak kullanılır. FAA-iCMM modeli, dış kaynak kullanımı ve kaynak yönetimini de içeren büyük yazılım sistemlerinde (enterprise) ilerlemeler yapmayı hedefleyen en iyi pratiklerden oluşan bir model sunar. Son sürümü tümleşik büyük yazılım yönetimi, bilgi yönetimi kullanım/geçiş/devre dışı bırakma ve işlem/destek süreçlerini içerir. FAA-iCMM ISO 9001:2000, EIA/IS 731, Malcom Baldridge National Quality Award ve President's Quality Award Criteria, CMMI-SE/SW/IPPD ve CMMI-A, ISO IEC TR 15504, ISO/IEC 12207 ve ISO/IEC CD 15288 standart ve modellerini bütünleştirir.

CMMI'da olduğu gibi FAA-iCMM genel en iyi pratikleri içerir herhangi bir alan için doğrudan güvenliği hedeflemez.

2.4. Güvenli CMM/Güvenli Yazılım Metodolojisi (Trusted CMM/Trusted Software Methodology (T-CMM, TSM))

90'lı yılların başında Strategic Defense Initiatives (SDI) "Güvenli Yazılım Geliştirme Metodolojisi" olarak adlandırılan bir model geliştirdi. Daha sonra bu model güvenli yazılım metodolojisi (Trusted Software Methodology (TSM)) olarak adlandırıldı. Bu model düşük seviyelerde bilmeden yapılan geliştirici hatalarına karşı yüksek seviyelerde ise bilerek yapılan zararlı yazılım ataklarına karşı direnç sağlayan seviyelerden oluşmaktadır. TSM daha sonra CMM ile birleştirilmiş (Harmonize) ve Trusted CMM ortaya çıkmıştır [4]. TCMM/TSM günümüzde yaygın olarak kullanılmamasına karşın ilerde yazılım geliştirme sürecinde bir kaynak olarak kullanılabilir.

2.5. Sistem Güvenlik Mühendisliği Yetenek Olgunluk Modeli (SSE-CMM -- Systems Security Engineering Capability Maturity Model)

SSE-CMM bir organizasyonun güvenlik mühendisliği yeteneğinin değerlendirilmesi ve geliştirilmesi için kullanılabilecek bir süreç modelidir. SSE-CMM, güvenlik mühendislik pratiklerini genel olarak kabul edilmiş mühendislik prensiplerine göre değerlendirip kabul edilebilir bir çerçeve ortaya koyar. Böyle bir çerçeve güvenlik mühendislik prensiplerinin uygulamalarında performansı ölçme ve iyileştirmeyi sağlar. SSE-CMM ISO/IEC 21827 standardı olarak yayınlanmış ve hali hazırda sürüm 3 yayınlanmıştır [5].

Güvenlik mühendisliği bu konuda genel olarak kabul edilmiş birkaç prensibe sahip olmasına karşın güvenliği değerlendirmek için gerekli bütüncül bir çerçeveden yoksundur. SSE-CMM, prensipleri uygulamalarının performansını ölçmeyi ve iyileştirmeyi amaçlayan çerçeveyi oluşturmayı amaçlar.

Model, proje ve organizasyon süreçleri ve güvenlik mühendisliği olmak üzere iki geniş alana sahiptir. Güvenlik mühendisliği mühendislik süreçleri, güvenlik süreçleri ve risk süreçleri içinde yer alır. Üç organizasyon içinde 22 süreç bulunmaktadır.

SSE-CMM The International Systems Security Engineering Association (ISSEA) tarafından sürdürülmektedir [6].

2.6. Microsoft Security Development Lifecycle (SDL)

Microsoft SDL güvenli geliştirme döngüsü, Microsoft geliştiricilerinin daha güvenli yazılımlar geliştirebilme ihtiyaçları ve bu ihtiyaçlara yönelik arayışlarından ortaya çıkmış bir anaçatıdır. Temelleri Ocak 2002'de yayınlanan Trustworthy Computing (TwC) yönergesine dayanır.

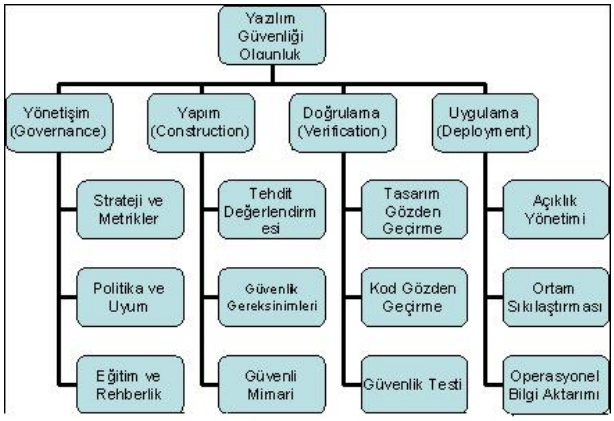
Bu model yazılım geliştirmenin başlangıcından itibaren güvenli yazılım geliştirme ve yaşam döngüsünü hedefler. Bu hedeflere ulaşmak için, eğitim ve farkındalık, proje başlangıcı, en iyi pratikleri tanımla ve uygula, risk analizi yap, risk analizi aracı, risk analizi, yazılım dokümantasyonu araçları ve müşteriler için en iyi patrikler, güvenli kodlama politikası, güvenli test politikası, güvenlik ekleme, son güvenlik kontrolü, güvenlik müdahale planlaması, ürün çıkarma güvenlik cevapları ve işletme olmak üzere on üç adımı kullanır[7].

Bu modelin temel dezavantajı doğrudan proje yöneticisinin liderliğine bağlı olmasıdır.

2.7. OpenSAMM Modeli

OWASP (Open Web Application Security Platform) tarafından desteklenen OpenSAMM projesi kapsamında yazılım garanti olgunluk modeli (software assurance maturiy model) ortaya konmuştur [8]. Bu çalışmada güvenli yazılım geliştirme amacıyla bir anaçatı oluşturulmaya çalışılmıştır. Anaçatı, büyüklükten bağımsız bir şekilde her organizasyonun kendine adapte edebileceği, normal yazılım geliştirme döngülerine uyarlayabileceği ve bir organizasyondaki yazılım güvenliği alanında gelişmeyi yönlendirebilecek bir yapıda oluşturulmuştur.

Anaçatı dört ana başlığa ayrılmıştır. Bu başlıklar, yönetişim (governance), yapım (construction), doğrulama (verification) ve uygulama (deployment) olarak belirlenmiştir. Bu ana başlıklar aslında normal yazılım geliştirme döngüsünün temel adımlarına karşılık gelmektedir. Her bir ana başlık altında üçer güvenlik eylemi yer almaktadır. Güvenlik eylemleri Şekil 2'de verilmiştir. Bu yapıda, güvenli bir yazılım geliştirmek için her bir temel yazılım geliştirme adımına karşılık yapılması gereken güvenlik çalışmaları güvenlik eylemi olarak değerlendirilmiştir. Her bir eylem altında da organizasyonun o eyleme konu alan alandaki olgunluk seviyesine göre hedefler (objectives) belirlenmiştir.



Şekil 2 OpenSAMM güvenlik eylemleri

Yönetişim başlığı altında, organizasyonda uygulanacak yazılım güvenliği programının stratejik yönünün belirlenmesi, organizasyona ait güvenlik çalışmalarının performansını ölçme yöntemlerinin ortaya konması, belirlenmiş kurum içi standartlara ve kurum dışı düzenlemelere uyum sağlanması ve çalışanların yazılım güvenliği konusunda eğitilmesi ile söz konusu çalışanlara rehberlik yapılması konuları ele alınmaktadır. Yapım başlığı, güvenli yazılım oluşturmak için yazılım gereksinimi ve tasarımı aşamalarında gerçekleştirilmesi gereken güvenlik eylemlerine değinmektedir. Yazılımın karşılaşacağı tehditlerin değerlendirilmesi, güvenlik ihtiyaçlarının belirlenmesi ve güvenli mimarinin oluşturulması "Yapım" başlığının altındaki güvenlik eylemleridir. Doğrulama başlığı, tasarım, yazılım kodlama ve yazılım testleri aşamasında gerçekleştirilmesi gereken güvenlik gözden geçirmelerini ve güvenlik testlerini kapsamaktadır. Tasarım gözden geçirmesi, kod analizi ve güvenlik testleri bu kapsamdaki güvenlik eylemleridir. Uygulama başlığı ise yazılımın canlı sisteme kurulması ve desteğinin verilmesi aşamalarını kapsamaktadır. İlgili güvenlik eylemleri, açıklık yönetimi, ortam sıkılaştırması ve operasyonel bilgi aktarımı olarak ele alınmıştır.

OpenSAMM, her bir güvenlik eyleminin organizasyondaki uygulanmasını nitel olarak ölçebilmek amacıyla COBIT (Control Objectives for Information and Related Technology)'teki [9] olgunluk modeline benzer bir model oluşturmuştur. Bu modelde, ilgili güvenlik eyleminin olgunluk seviyesi '0' ile '3' arasındaki bir değerle değerlendirilmektedir. '0', ilgili eylemin uygulanmaması anlamına gelmektedir. '1', eylemle ilgili organizasyonda başlangıç düzeyinde bir anlayış olduğunu ve eylemin

sistematik olmayan bir yapıda gerçekleştiğini, '2' söz konusu eylemin belirli bir etkinlikte ve verimlilikte yapıldığını, '3' ise eylemin kapsamlı bir şekilde uygulandığını göstermektedir. Bu olgunluk modeline göre belirli aralıklarla proje temelli ya da organizasyon temelli yapılabilecek denetimlerin sonuçları organizasyonun güvenlik eylemleri bazında ilerlemesini ortaya koyabilecektir.

3. Ortak Kriterlerin Güvenli Yazılım Geliştirme Sürecine Yaklaşımı

Ortak Kriterler (ISO 15408) standardı; Seçilen garanti paketinin (EAL-Evaluation Assurance Level) içeriğine bağlı olarak, tasarlanması planlanan BT ürününe (yazılım-donanım) tasarım başlangıcından itibaren geliştirici tarafından uygulanması gereken bir metodolojiyi şart koşmaktadır [10]. Ortak Kriterler standardının ortaya koyduğu metodolojiye uygun olarak tasarlanan BT ürününe, yine Ortak Kriterler standardına uygun olarak değerlendiriciler tarafından güvenlik ve güvenilirlik analizi uygulanır. Değerlendiriciler tarafından gerçeklenen analiz sürecinde; hedef BT ürününün tasarım sürecinin Ortak Kriterler standardına uygun olarak gerçeklenip gerçeklenmediği kontrol edilir ve müteakibinde fonksiyonel test ve açıklık analizi ile değerlendirme faaliyeti tamamlanır.
Ortak Kriterler standardının fonksiyonel test ve açıklık analizi değerlendirme adımlarına ek olarak; BT ürününün tasarım sürecine de bir disiplin ve metodoloji getirmesi, BT ürününün seçilen Ortak Kriterler garanti paketine bağlı olarak minimum zayıflıkla tasarlanmasını ve maksimum güvenlik ve güvenilirlik ile fonksiyonel test ve açıklık analizi değerlendirme aşamalarına girmesini sağlamaktadır. Ortak Kriterler standardının EAL 4 ve üzeri garanti seviyesi için BT ürününün tasarım sürecine getirdiği disiplin ve metodoloji aşağıda maddeler halinde listelenmiştir.

3.1. Risk Analizi Faaliyeti

ASE (Security Target) garanti sınıfı, tasarlanması planlanan BT ürünü için tasarım başlangıcında risk analizi faaliyetinin uygulanmasını zorunlu kılar. Ortak Kriterler standardına uygun olarak geliştirilmesi planlanan BT ürünlerinde uygulanması gereken risk analizi faaliyetinde, tasarımcı öncelikli olarak BT ürününü ve ilgili işletim ortamını BT ve BT olmayan bileşenleri ile birlikte tanımlar. BT ürününün kullanılacağı ortam ile ilgili varsayımlar ve BT ürününü kullanacak tüketicinin uyması gereken kurumsal güvenlik politikaları belirlenir ve belirlenen kapsamda BT ürünü tarafından korunması gereken ve elde edilmesi durumunda güvenlik veya güvenilirliğin tehlikeye düşeceği varlıklar (asset) listelenir. Tasarımcı listelenen her bir varlık için olası tehdidi veya tehditleri, tehdit ajanı ve tehdit senaryosu ile birlikte varsayım ve kurumsal güvenlik politikaları ile ters düşmeyecek biçimde ayrıntılandırır. Risk analizi faaliyetinin son adımı olarak her bir tehdit, varsayım ve kurumsal güvenlik politikası, tasarımcı tarafından belirlenecek olan BT ve BT olmayan güvenlik hedefleri/güvenlik gereksinimleri ile ilişkilendirilerek karşılanır.

Ortak Kriterler standardı tarafından şart koşulan ve tasarım başlangıcında BT ürününe uygulanan risk analizi faaliyeti ile BT ürünü için olası bir tehdidin tasarımcının gözünden kaçma olasılığı asgari seviyeye indirilmekte ve var olan tüm tehditlerin karşılanması için BT ürününün veya işletim ortamının bir güvenlik mekanizması sağlaması garanti edilmektedir.

3.2. Konfigürasyon Yönetimi

ALC_CMC (Configuration Management) garanti ailesi ile insan hatasının tasarım sürecine etkisi asgari seviyeye indirgenmektedir. BT ürününün karmaşıklığı ile doğrudan ilişkili olarak tasarım grubu içerisinde bulunan tasarımcı personel sayısı artmaktadır. Tasarımcı personel sayısının artışı, Konfigürasyon Yönetimi gereği olarak görevlerin ayrılığını gerektirir ve bunun sonucu olarak roller ve sorumlulukları belirlenir. Belirlenen roller ve sorumluluklara uygun olarak otomatik bir konfigürasyon yönetim aracı (CVS, SVN gibi.) ile BT ürününün güvenlik ve güvenilirliğini etkileyecek tüm dokümantasyon (tasarım dokümanları, test dokümanları, güvenlik hedefi dokümanı gibi.) konfigürasyon yönetim planına uygun olarak takip edilir. Otomatik konfigürasyon yönetim aracının kontrolü ile tasarımcı roller sorumlulukları dahilinde yapılması gerekenleri kontrollü olarak gerçekleştirirler. İnsan hatasından kaynaklanacak; Güncel olmayan kaynak kod parçasını kullanma, güncel olmayan tasarım dokümanını kullanma, BT ürününün bir modülünü yanlışlıkla silme, benzer görevi olan yeni bir BT ürünü modülü oluşturma, BT ürününün yanlış sürümünü üretme vb. problemler, tasarım sürecinde Ortak Kriterler standardının şart koştuğu Konfigürasyon Yönetimi ile asgari seviyeye indirgenerek, BT ürününün daha fazla güvenli ve güvenilir olması sağlanmaktadır.

3.3. Metodolojik Tasarım

ADV (Development) garanti sınıfı ile metodolojik bir tasarım geliştiriciye şart koşulmaktadır. Metodolojik tasarım ile BT ürünü öncelikli olarak alt sistemler halinde, daha sonra alt sistemler içerisinde modüller ile ve son olarak modüller içerisinde güvenlik fonksiyonları ile adım adım tanımlanır. Sonrasında; Alt sistemler, modüller ve güvenlik fonksiyonları arası tüm iç ara yüzler ve alt sistemlere, modüllere ve güvenlik fonksiyonlarına olan tüm dış ara yüzler ayrıntılı olarak açıklanır. Tüm güvenlik fonksiyonları ve ara yüzlerin davranışları ve hata durumunda vereceği uyarılar belirlenir. Ürünün maksimum güvenlik ve güvenilirlikte tasarlanabilmesi için üst seviye Ortak Kriterler garanti paketlerinde (EAL 5, EAL 6 ve EAL 7) tasarımın yarı-formal (semi-formal) veya formal olarak tasarlanabilmesi için yarı formal tasarım dillerinin (ör: UML) ve formal tasarım dillerinin (ör: Z model) uygulanması şart koşulmaktadır. Böylece informal tasarım sürecinde tasarıma yönelik yapılabilecek mantık hataları asgari seviyeye indirgenmektedir. Ortak Kriterler standardının ortaya koyduğu metodolojik tasarım ve üst seviye Ortak Kriterler standardı garanti paketlerinde uygulanmasını zorunlu kıldığı informal ve formal tasarım dilleri ile geliştirilecek BT ürününün maksimum güvenlik ve güvenilirlikte tasarlanmasını sağlamaktadır.

3.4. Yaşam Döngüsü Modeli

ALC_LCD (Life Cycle Definition) garanti ailesi ile Ortak Kriterler standardı, BT ürününe tasarım sürecinde bir yaşam döngüsü modeli (ör: waterfall model, spiral model) uygulanmasını şart koşmaktadır. Uygulanan yaşam döngüsünün BT ürününün maksimum güvenlikte ve güvenilirlikte tasarlanmasına katkıda bulunup bulunmadığı Ortak Kriterler standardına uygun gerçeklenen değerlendirme sürecinde kontrol edilmektedir. Tasarım sürecinde minimum hata yapılmasına ve BT ürünü ile ilgili var olan hataların tasarım sürecinde ve erken aşamalarda tespit edilmesine,

uygulanan yaşam döngüsü modelinin katkıda bulunduğu garanti edilir. Böylece daha kaliteli tasarım süreci ve bunun sonucu olarak ortaya daha güvenli ve güvenilir BT ürünlerinin çıkması sağlanır.

3.5. Geliştirme Araçları

ALC_TAT (Tools and Techniques) garanti ailesi ile BT ürününün geliştirilmesi sürecinde kullanılan geliştirme araçlarının kalitesi değerlendirilir. Tasarım ve fonksiyonel test amaçlı kullanılan geliştirme araçlarının işlevlerini doğru olarak yerine getirmemesi BT ürününün güvenlik ve güvenilirliğini direk olarak etkilemektedir. Ortak Kriterler standardı, tasarım sürecinde kullanılan geliştirme araçlarının bilinen/lisanslı veya doğrulaması gerçeklenmiş araçlar olmasını şart koşarak, BT ürününün güvenlik ve güvenilirliğini tasarım aşamasında maksimum seviyeye getirmektedir.

3.6. Geliştirme Ortamı

ALC_DVS (Development Security) garanti ailesi ile BT ürününün geliştirildiği ortamın güvenliği değerlendirilmektedir. Geliştirme ortamının güvenliğinden kaynaklanabilecek bir tehdidin tasarım sürecinde BT ürününü etkilememesi ve bunun sonucu olarak daha güvenli ve güvenilir BT ürünlerinin ortaya çıkmasını sağlayabilmek amacıyla, Ortak Kriterler Standardı değerlendirme sürecinde geliştirme ortamına yapılabilecek bir yetkisiz erişimin olup olamayacağı tespit edilmeye çalışılmaktadır. Tasarım ortamında BT ürününe güvenlik fonksiyonları pasifize edilmiş halde ulaşılabileceğinden, tasarım dokümanlarına erişilip BT ürünü ile ilgili maksimum bilgiye sahip olunabileceğinden, kaynak kodda güvenlik fonksiyonları pasifize edilebilecek şekilde modifikasyon yapılabileceğinden, vb. tasarım ortamının güvenliği BT ürünü için en zayıf halkalardan birisi olarak düşünülebilir. Bu sebeple; Ortak Kriterler standardının geliştirme ortamı için getirdiği kontroller, BT ürününün daha güvenli ve güvenilir olmasına katkı sağlamaktadır.

3.7. Hata Düzeltme

ALC_FLR (Flaw Remediation) garanti ailesi ile BT ürününün tasarımının tamamlanması sonrası sahada kullanımı sırasında tespit edilen güvenlik zayıflıklarının, güvenlik açıklıklarının ve hatalarının tümünün farkında olunması, doğru olarak analiz edilmesi ve doğru sürüm güncelleme faaliyetinin gerçeklenmesi garanti edilmeye çalışılır. Bu garanti ailesi, asıl tasarım süreci sonrası yapılabilecek ek güncellemelere yönelik tasarım sürecine kalite getirmektedir. BT ürününün her bir sürümüne, tüketiciler tarafından raporlanan problemlerin izlenmesi için bir prosedürün belirlenmesi, her bir problemin etkilerinin analiz edilmesi ve düzeltilmesi için ilgili metodun ve eylemlerin belirlenmesi, Ortak Kriterler standardı ile sağlanmaktadır.

3.8. Fonksiyonel Testler

ATE (Tests) garanti adımı ile BT ürününün tasarım sürecinin son adımı metodolojik olarak test edilmesi garanti edilir. Ortak Kriterler standardı ile BT ürününün tasarım dokümanlarında belirtilen her bir güvenlik fonksiyonunun, var olan iç ve dış ara yüzleri kapsamında tüm davranışları ile tasarım sürecinde tasarımcı tarafından test edilmesi sağlanır. Böylece hedeflenen BT ürünü işlevselliğinin tümüyle gerçeklendiği, tasarımcı tarafından gerçeklenen test adımı ile garanti edilir. BT ürünü testlerinin herhangi bir metodoloji uygulanmadan tasarımcı tarafından rasgele yapılması durumunda, her bir güvenlik fonksiyonunun tüm davranışları test edilemeyeceğinden ortaya çıkacak olan BT ürünü, bir kısım işlevselliklerinin gerçekleştirilemediği tespit edilemeden sahaya çıkacaktır. Bunu engellemek amacıyla; Ortak Kriterler standardı ile tasarım sürecinde metodolojik bir test yaklaşımı sunularak, ürünün daha güvenli ve güvenilir olarak tasarlanması sağlanmaktadır.

4. Modellerin Güvenliğe Bakışı

Bu bölümde, güvenli yazılım geliştirme döngüsü kapsamında uygulanabilecek güvenlik çalışmaları Ortak Kriterler, SSE-CMMI standartları ile Microsoft-SDL ve OpenSAMM modellerinin sundukları güvenlik özellikleri değerlendirilmiştir. Değerlendirmeler, Tablo 1'de verilmiştir.

Ortak Kriterler standardı, ürün güvenliği değerlendirme odaklı bir standarttır. Geliştiricilerden belirli kriterleri yerine getirmesini bekler. Bu bölümde, söz konusu standardın bir yazılım ürününün geliştirilme aşamasında geliştiriciden yerine getirmesini beklediği unsurlar dikkate alınarak değerlendirme yapılmıştır. SSE-CMMI, sistem geliştirme sırasında güvenliğin ele alınmasını irdeleyen bir model önermektedir. Sistem geliştirme içerisinde elbette yazılım da yer alabilir ama ortaya konan model doğrudan yazılım geliştirme odaklı değildir. Microsoft-SDL ve OpenSAMM doğrudan yazılım güvenliğini ele alan anaçatı dokümanlardır.

Yazılım geliştiricilerinin ve proje yöneticilerinin bilgi güvenliği konusunda bilinçlendirmiş olmasını ve gerekli eğitimleri almış olmasını SSE-CMMI, Microsoft-SDL ve OpenSAMM modelleri doğrudan gerekli sayar. Ortak Kriterler standardı ürün odaklı bir standart olduğundan doğrudan geliştirici organizasyonu odaklı bu parametreyi detaylı irdelemez. Öte yandan, Ortak Kriterler değerlendirmesi kapsamında, değerlendiriciler geliştirici ortamlarına saha ziyareti gerçekleştirmekte ve geliştirme ortamı güvenliğini denetlemektedirler. Bu denetimlerde denetleyiciler, bilinçlendirme ve eğitim çalışmaları ile ilgili kontroller de yapabilirler.

Geliştirme ortamının fiziksel güvenliği ve geliştirme ortamının ağ ve uygulama bazında erişim kontrolünü sağlama anlamına gelen mantıksal güvenliği, Ortak Kriterler ve SSE-CMMI'da ele alınmış, Microsoft-SDL ve OpenSAMM'da doğrudan irdelenmemiştir.

Güvenli yapılandırma yönetimi, yazılım geliştirme dokümanlarının güvenli ve kontrollü bir şekilde yönetilmesini, kaynak kodlara güvenli erişilmesinin sağlanmasını ifade etmektedir. Bu konu, Ortak Kriterler ve SSE-CMMI kapsamında ele alınmaktadır. Microsoft-SDL ve OpenSAMM güvenli yapılandırma yönetimini doğrudan adreslememektedir. Tehdit modellemesi, risk analizi, güvenlik gereksinimlerinin belirlenmesi, güvenli mimari belirleme ve güvenli tasarım konuları güvenli yazılım geliştirme açısından en önemli unsurlardır. Ele alınan tüm standardlar ve anaçatılar bu konuları ele almıştır.

Yazılım kaynak kodlarının güvenlik açısından gözden geçirilmesi yazılımlardaki güvenlik açıklıklarının tespit edilmesi adına yapılabilecek en önemli faaliyetlerden birisidir.

Kaynak kod analizi sırasında kullanmak üzere açık kaynak kodlu ve ticari otomatik araçlar da mevcuttur. Bu araçlar, söz konusu çalışmanın otomatize edilmesini sağlamada etkindirler ama çıktılarının uzman kişilerce incelenerek içerisindeki hatalı pozitif sonuçların (araç tarafından açıklık olarak tespit edilen ama gerçekte açıklık olmayan) elemine edilmesi gereklidir. Ayrıca bu araçlar tüm açıklıkları da bulamayabilirler. Dolayısıyla, otomatize araçlar kullanılsa bile kod analizi için önemli iş gücünü gereksinimi vardır. Ortak Kriterler standardı, geliştiricilerin kaynak kodu analizi yapmasını şart koşmamaktadır. Değerlendiricilerin gerçekleştireceği açıklık analizi sırasında da değerlendiriciler gerekli görürse kod analizi gerçekleştirirler. SSE-CMMI, sistem odaklı bir standard olması sebebiyle doğrudan kaynak kod analizini adreslememektedir. Microsoft-SDL ve Open-SAMM bu analize ayrı bir önem vermiş, otomatik araçların da kullanılmasının çalışma kapsamında yer alması gerektiğini ifade etmişlerdir.

Tablo 1'de yer alan "Açıklık analizi" başlığı, kara kutu veya beyaz kutu mantığı ile yazılımlarda olabilecek açıklıkların tespit edilmesi çalışmasına karşılık gelmektedir. Ortak Kriterler standardı, geliştiricilerin açıklık analizi yapmasını şart koşmaz ama bu çalışmanın değerlendiriciler tarafından gerçekleştirilmesini zorunlu kılar. Standard, önceki sürümlerinde açıklık analizinin geliştirici tarafında da yapılmasını zorunlu koşmaktaydı fakat çalışmanın gerektirdiği iş gücünün çok olması sebebiyle geliştiricilerin itirazı ve bu çalışmanın zaten değerlendirici tarafında gerçekleştirilmesi sebeplerine binaen standardın son sürümünde geliştirici tarafından açıklık analizi yapma şartı ortadan kaldırılmıştır. SSE-CMMI, Microsoft-SDL ve OpenSAMM açıklık analizi ile ilgili gereksinimleri içermektedir.

Güvenlik doğrulaması, yazılımın güvenlik gereksinimlerinin doğru tasarlanmasını ve tasarımın doğru uygulanmasını kontrol etmeyi kapsamaktadır. Bu amaçla yazılım güvenliği döngüsünün uygun noktalarında güvenlik gözden geçirme ve testlerinin gerçekleştirilmesi sağlanır. Tasarımın gözden geçirilmesi, birim güvenlik testleri, entegrasyon güvenlik testleri ve güvenlik kabul testleri güvenlik doğrulanmasının alt bileşenleridir. Ele alınan tüm standartlar ve anaçatılar güvenlik doğrulaması konusunu kapsamaktadır. Microsoft-SDL ve OpenSAMM güvenlik doğrulamasının nasıl gerçekleştirileceği ile ilgili daha detaylı açıklamaları ve yöntemleri içermektedir.

Açıklık yönetimi kavramı, yazılımın tüketicilere dağıtıldıktan sonra gerek geliştiriciler tarafından gerekse yazılım tüketicileri veya kullanıcılar tarafından tespit edilebilecek güvenlik açıklıklarının veya zayıflıklarının ilgili geliştirme birimine iletilmesi, ele alınması, yazılımın güncellenmesi süreçlerini kapsamaktadır. Bu süreçlerin yazılım geliştirme organizasyonlarında etkin bir şekilde işletilmesi bu kapsamda hedeflenmektedir. Ele alından tüm standardlar ve anaçatılar açıklık yönetimi konusunu düzenlemektedirler.

Güvenli Geliştirme Teknikleri Belirleme ve Uygulama başlığı, yazılım geliştiricilerinin kullanacağı güvenli kod geliştirme prosedürlerinin belirlenmesi, kullanılacak yazılım geliştirme teknolojisi ile ilgili en iyi güvenlik yöntemlerinin ortaya konması ve yazılım geliştirme çalışmaları sırasında bu prosedür ve yöntemlerin uygulanmasını kapsamaktadır. Ortak Kriterler standardı, ürüne odaklanması sebebiyle doğrudan bu uygulamaları gerektirmemektedir fakat değerlendiriciler geliştirme ortamının saha ziyareti sırasında bu tür kriterleri de değerlendirebilirler. SSE-CMMI, Microsoft-SDL ve OpenSAMM güvenli geliştirme teknikleri konusunda kriterler içermektedir.

Yazılımları operasyonel ortamlara kuracak ve bu ortamlarda desteklerini verecek sistem yöneticilerine ve kullanıcılarına kullanıcı kılavuzlarında ve yardım menülerinde yazılımın güvenli kurulması, yönetilmesi ve kullanılması ile ilgili bilgilerin yer verilmesi "operasyonel ortama bilgi aktarımı" başlığı altında ele alınmıştır. Yazılımların kurulacağı ortamda yazılımla etkileşen işletim sistemi, ağ ve diğer bileşenlerin güvenliğinin sağlanması konusu "çevre bileşenlerle güvenli entegrasyon" konusu altında irdelenmiştir. Her iki başlık ele alınan tüm standartlar ve anaçatılarda yer almaktadır.

Üretilen ürünlerin tüketicilere bütünlüğü garanti edilmiş bir şekilde iletilmesi ile ilgili prosedürlerin ve kontrol mekanizmaların belirlenmesi ve uygulanması güvenli teslim etme başlığı altında ele alınmıştır ve Microsoft-SDL dışındaki diğer standart ve anaçatılarda bu başlığa değinilmiştir.

Tablo 1: Yazılım geliştirme modellerinin ve Ortak Kriterlerin karşılaştırılması

	Ortak Kriterler	SSE-CMMI	Microsoft- SDL	OpenSAMM
Bilgi Güvenliği Bilinçlendirmesi ve Eğitimi	X	✓	✓	✓
Fiziksel ve Mantıksal Güvenlik	✓	✓	X	X
Güvenli Yapılandırma Yönetimi	✓	✓	X	X
Yasa, Politika ve Prosedür Uyumluluğu	X	✓	X	✓
Tehdit Modellemesi	✓	✓	✓	✓
Risk Analizi	✓	✓	✓	✓
Güvenlik Gereksinimleri Belirleme	✓	✓	✓	✓
Güvenli Mimari	✓	✓	✓	✓
Güvenli Tasarım	✓	✓	✓	✓
Kaynak Kod Analizi	X	X	✓	✓
Açıklık Analizi	X	✓	✓	✓
Güvenlik Doğrulaması	✓	✓	✓	✓
Açıklık Yönetimi	✓	✓	✓	✓
Güvenli Geliştirme Teknikleri Belirleme ve Uygulama	X	✓	✓	✓
Operasyonel Ortama Bilgi Aktarma	✓	✓	✓	✓
Çevre Bileşenlerle Güvenli Entegrasyon	✓	✓	✓	✓
Güvenli Teslim Etme	✓	✓	X	✓

5. Sonuçlar

Bilgi sistemlerinde kullanılan yazılımların, güvenli olarak geliştirilmemesinden dolayı yazılımlarda sürekli açıklıklar çıkmaktadır. Bu açıklıklar ve zayıflıklar kullanıcılar tarafından bilerek ya da bilmeden kötüye kullanılabilmektedir. Kullanıcılar bilgi sistemlerinde kullanacakları yazılım ya da sistemlerin de bu tür açıklıkların olmamasını istemektedir. Bu açıklıkları ve onların etkilerini azaltmanın en etkin yöntemi ise yazılımın güvenli olarak geliştirilmesidir.

Güvenli yazılım oluşturmak için hali hazırda kullanılan güvenli yazılım geliştirme standart, anaçatı veya modellerinin önemlileri bu makalede incelenmiştir. İnceleme sonucunda hiç biri güvenli yazılım geliştirmek için tüm kriterleri sağlamamaktadır. Temel hedefi BT ürünlerinin güvenlik fonksiyonlarının değerlendirilmesi olan Ortak Kriterler standardının özellikle yüksek güvenlik seviyeleri hedeflenen ürünlerde güvenli yazılım geliştiriciler için bir güvenli geliştirme rehberi de olabileceği görülmektedir.

Ortak Kriterler standardının 25 ülke tarafından doğrudan kabul edilmiş olması, değerlendirme metodolojisinin olması, uluslar arası standart olarak yayınlanmış olması onu güvenlik konusunda diğer sistemlere göre ön plana çıkarmaktadır. Özellikle geliştirme ortamının fiziksel ve mantıksal güvenliği konusunda diğer standartlar ve anaçatılara göre daha fazla güvenlik önlemi içermektedir.

Bu makalede Ortak Kriterlerin geliştiriciler tarafından bir rehber olarak alınması durumunda, sağlayacağı faydalar diğer metotlarla karşılaştırılmıştır. Ortak Kriterlerin güvenli yazılım geliştirmede etkin bir rehber olacağı ortaya çıkmıştır. Bu konuda gelecekte yapılacak çalışmalarda Ortak Kriterler kullanılarak hazırlanmış bir yazılım ürünün tarafsız bir güvenlik laboratuarı tarafından değerlendirilmesinin yazılım ürününe katacağı değer, irdelenebilir.

6. Kaynakça

[1] Brooks, F.P., "No Silver Bullet Essence and Accidents of Software Engineering", *Computer Magazine*, 1987

[2] Redwine, S. T. ve Davis, N., "Processes to Produce Secure Software." *Improving Security Across the Software Development Lifecycle* (National Cybersecurity Partnership Taskforce Report), Appendix B. http://www.cyberpartnership.org/init-soft.html (2004).

[3] CMMI yıllık raporu (2009) http://www.sei.cmu.edu/appraisal-program/profile/pdf/CMMI/2009MarCMMI.pdf

[4] Kitson, D. H. "A Tailoring of the CMM for the Trusted Software Domain." *Proceedings of the Seventh Annual Software Technology Conference*. Salt Lake City, Utah, April 9-14, 1995.

[5] System Security Engineering Capability Maturity Model V 3.0 http://www.sse-cmm.org/docs/ssecmmv3final.pdf

[6] http://www.issea.org/

[7] Microsoft Security Lifecycle (SDL) Version 3.2 http://msdn.microsoft.com/en-us/library/cc307748.aspx

[8] Software Assurance Maturity Model, A Guide to Building Security into Software Development Version. 1.0, http://www.opensamm.org

[9] COBIT 4.1, IT Governance Institute, http://www.itgi.org

[10] Common Criteria for Information Technology Security Evaluation Part 1-2-3. (Version 3.1) http://www.commoncriteriaportal.org