## 1. Amaç

Bu politikanın amacı, Üniversitenin bilgi güvenliği hedefleri doğrultusunda, yazılım geliştirme süreçlerinde bilgi güvenliğini sağlayarak, güvenlik zafiyetlerini minimize eden, sürdürülebilir, güvenli ve güvenilir yazılımlar geliştirmek için tüm geliştirme süreçlerinde uyulması gereken prensipler ve standartları belirlemektir.

## 2. Kapsam

Bu politika, gerek iç gerekse dış kaynaklardan temin edilen, KTÜN iş süreçleri ile ilgili yazılım geliştirme faaliyetlerinde yer alan tüm ekip üyelerini, geliştirilen veya bakımı yapılan tüm yazılımları ve bu süreçlerde kullanılan donanım, yazılım, veri ve sistemleri kapsar.

## 3. Sorumlular

Konya Teknik Üniversitesi için yazılım geliştiren iç ve dış kaynaklar ile bu kaynaklardan hizmet alan KTÜN birim yöneticileri sorumludur.

**1.**

## 4. Genel Prensipler

**4.1.** Tüm yazılım projelerinde güvenlik, tasarım aşamasından itibaren dikkate alınmalıdır ("Security by Design" ilkesi).

**4.2.** Güvenlik, yazılım tamamlandıktan sonra değil, geliştirme süreçlerinin her aşamasında ele alınmalıdır. Her bir aşamada güvenlik değerlendirmeleri yapılmalıdır. Güvenli Yazılım Geliştirme Yaşam Döngüsü (Secure Software Development Lifecycle - Secure SDLC) yöntemi uygulanmalıdır.

**4.3.** Güvenli tasarım ilkeleri (örneğin, en az ayrıcalık prensibi, veri doğrulama, giriş kontrolü) uygulanmalıdır.

**4.4.** Yazılım mimarisi, izinsiz erişim veya kötüye kullanım risklerini en aza indirecek şekilde oluşturulmalıdır.

**4.5.** Yazılım, hassas verilere erişim için kimlik doğrulama ve yetkilendirme mekanizmalarını içermelidir.

**4.6.** Yazılımın farklı katmanlarında (ağ, uygulama, veritabanı) güvenlik kontrolleri sağlanmalıdır. Tek bir güvenlik mekanizmasına güvenmek yerine, birden fazla koruma katmanı oluşturulmalıdır. Her katmanda, kimlik doğrulama, şifreleme, erişim kontrolü gibi bağımsız güvenlik önlemleri uygulanmalıdır. (**Savunma Derinliği (Defense in Depth) ilkesi**)

**4.7.** Güvenlik gereksinimleri, projeye özgü olarak belirlenmelidir.

## 5. Güvenlik Gereksinimlerinin Belirlenmesi

**5.1.** Yazılım projeleri başlatılmadan önce, sistemin maruz kalabileceği tehditler analiz edilmeli ve riskler değerlendirilmelidir (Ör: STRIDE modeli).

**5.2.** ISO/IEC 27001 ve aşağıdaki gibi Güvenli SDLC ile İlgili uluslararası standartlar ve çerçeveler dikkate alınmalıdır:

- **OWASP SAMM (Software Assurance Maturity Model):** Güvenli yazılım geliştirme için bir olgunluk modeli.

- **ISO/IEC 27034-1:** Uygulama güvenliği yönetimi için uluslararası standart.
- **NIST Secure Software Development Framework (SSDF):** Yazılım güvenliği için geliştirilmiş bir çerçeve.

**5.3.** Güvenlik gereksinimleri, çözüm mimarileriyle uyumlu olmalıdır.

## 6. Kodlama Standartları

**6.1.** Kodlama yaparken, güvenlik standartlarını dikkate alınmalıdır. (OWASP Secure Coding Practices Rehberi, CERT Secure Coding Standards vb.).

**6.2.** Güvenli kodlama teknikleri uygulanmalı ve sık yapılan hatalar (örneğin, SQL enjeksiyon, XSS, açık oturum yönetimi) önlenmelidir.

**6.3.** Kod incelemeleri (code review) güvenlik odaklı olarak yapılmalıdır. Geliştiriciler, yazdıkları kodun başka bir ekip üyesi tarafından güvenlik açıkları açısından incelenmesini sağlamalıdır. Kod inceleme araçları ve otomatik statik kod analiz araçları kullanılabilir.

**6.4. Kodlar, aşağıdaki temel ilkeler çerçevesinde geliştirilmelidir:**

**6.4.1.** Yazılımlar, modüler planlanmalı, modüller arası ilişkilerde yapısallık göz önünde bulundurulmalı ve programcı müdahalesi asgari seviyede olacak şekilde parametrik hazırlanmalıdır. Sisteme yeni modüllerin ilavesi, modüllerin değiştirilmesi ya da silinmesi durumda sistemin bütünü etkilenmemelidir.

**6.4.2.** Tüm kullanıcı girdileri kontrol edilerek girdilerin doğrulanması ve temizlenmesi sağlanmalıdır. Örneğin SQL enjeksiyonu, çapraz site betikleme (XSS) gibi saldırılara karşı özel giriş doğrulama kuralları kullanılmalıdır.

**6.4.3.** Hassas verilerin şifrelenmesi. Tüm kritik veriler (şifreler, kimlik bilgileri) şifrelenerek saklanmalı ve iletilmelidir (Veritabanı bağlantıları ve Web uygulamalarında SSL/TLS kullanımı dahil).

**6.4.4.** Veritabanı için oluşturulan kullanıcı yetkileri sadece ilgili uygulamaya ait veritabanına erişecek şekilde yapılandırılmalıdır.

**6.4.5.** Koda gömülü parolalar kullanılmamasına özen gösterilmelidir. Örneğin API anahtarlarının kod içinde değil, şifreli bir ortam değişkeninde saklanması sağlanmalıdır.

**6.4.6.** Hata mesajlarının sınırlı tutulması ve ayrıntı verilmemesi. Örneğin "Bağlantı başarısız oldu" yerine "Hata oluştu, lütfen destek ile iletişime geçin" mesajının gösterilmesi gibi.

**6.4.7.** Yetkilendirme ve kimlik doğrulama kontrollerinin doğru uygulanması.

## 7. Geliştirme Ortam Güvenliği

**7.1.** Yazılım geliştirme işlemi fiziksel olarak korunaklı ve sadece yetkili personelin erişebileceği ortamlarda gerçekleştirilmelidir.

**7.2.** Yazılım kaynak kodlarına sadece ilgili yazılım geliştirme uzmanlarının erişimi olmalıdır.

**7.3.** Test, geliştirme ve operasyonel ortam birbirinden ayrılmalıdır.

**7.4.** Yazılım kaynak kodları SVN veya benzeri ortamlarda yedeklenmeli ve kodlarda yapılan değişikler takip edilmelidir.

**7.5.** **Yazılım Sürüm Yönetimi (Software Version Management) uygulanarak** yazılım geliştirme süreçlerinde sürüm değişikliklerinin detaylı bir şekilde ele alınması sağlanmalıdır. Bu kapsamda yazılım sürümlerinin **Semantic Versioning (SemVer)** gibi bir standarda uygun şekilde adlandırılması (Ör: Ana.Sürüm.Minor, 2.1.0); her sürüm için değişikliklerin açıklandığı bir "release note" dokümantasyonu hazırlanması gibi işlevler sağlanmalıdır.

## 8. Yazılım Testleri ve Doğrulama

**8.1.** Tüm yazılımlar yayına alınmadan önce statik kod analizi, dinamik uygulama testi ve penetrasyon testi gibi güvenlik testlerine tabi tutulmalıdır.

**8.2.** Bulunan zafiyetler, risk derecesine göre hızla giderilmeli ve yeniden test edilmelidir.

**8.3.** Testlerde tespit edilen güvenlik açıkları kapatılmadan yazılım dağıtıma alınmamalıdır.

**8.4.** Test verisi dikkatlice seçilmeli, korunmalı ve kontrol edilmelidir. Gerektiğinde anonimleştirme çalışmaları yapıldıktan sonra kullanılmalıdır.

## 9. Güncellemeler ve Yama Yönetimi

**9.1.** Yazılımın yaşam döngüsü boyunca ortaya çıkan güvenlik zafiyetleri takip edilmeli ve en kısa sürede yamalar uygulanmalıdır.

**9.2.** Yazılım tedarikçilerinden gelen güncellemeler düzenli olarak incelenmeli ve uygulanmalıdır.

**9.3.** Veritabanı, uygulama sunucusu ve web sunucusu gibi kullanılan yazılımların güvenlik yamaları en üst seviyede olmalıdır. Tüm üçüncü taraf bileşenler ve kütüphaneler güncel ve güvenli olmalıdır.

## 10. Erişim Kontrolleri

**10.1.** Yazılım geliştirme süreçlerinde rol bazlı erişim kontrolü (RBAC) uygulanmalıdır.

**10.2.** Geliştirme ortamları, test ortamları ve canlı ortamlar arasında ayrım sağlanmalıdır.

**10.3.** Yazılım geliştiricilerin sadece iş gereksinimleri çerçevesinde gerekli olan verilere erişim izni olmalıdır.

## 11. Loglama ve İzleme

**11.1.** Yazılım, kullanıcı ve sistem etkinliklerini kaydedecek şekilde tasarlanmalıdır.

**11.2.** Loglar, yetkisiz erişimi önleyecek şekilde güvenli bir ortamda saklanmalıdır.

**11.3.** Güvenlik olaylarının izlenmesi ve analizi için uygun sistemler kullanılmalıdır.

## 12. Hassas Verilerin Korunması

**12.1.** Hassas veriler yazılım geliştirme ve test ortamlarında kullanılmamalıdır.

**12.2.** Gerektiğinde, hassas verilerin maskelenmesi veya anonimleştirilmesi sağlanmalıdır.

**12.3.** Veri aktarımlarında şifreleme protokollerinin kullanılması zorunludur.

**12.4.** Web, uygulama ve veritabanı sunucularının sistem bileşenleri hakkındaki kritik bilgileri (sunucu adı ve sürümü, kullanılan program sürümü vb.) gizlenmelidir.

**12.5.** Uygulamalar, geliştirme ortamından test ortamına aktarılırken gereksiz olan dosyalar (örneğin test kodlar, demo programlar, yedek dosyalar) silinmeli, gerek yoksa kaynak

kod aktarılmamalı ve aktarılacak olan kaynak kodlardaki yorum satırları silinmelidir. Aktarım esnasında dosyalarda istenmeyen bir değişikliğin olmaması garanti edilmelidir.

**12.6.** Uygulamaların üzerinde bulunduğu sunucular, servis verdikleri dizinlerin içeriklerini listelememelidir.

**12.7.** Gerekmedikçe POST/GET dışındaki HTTP metotlarına izin verilmemelidir.

**12.8.** GET ve POST isteklerindeki HTTP parametreleri değiştirilerek üçüncü şahısların bilgilerine yetkisiz olarak erişilmemelidir.

**12.9.** Ana sistem için gereksiz olan dosyalara (örneğin yedekleme, arşiv, test, geliştirme için kullanılan dosyalar) erişim engellenmeli ve sistemdeki gereksiz uygulamalar (örneğin ön tanımlı sunucu sayfaları, demo uygulamalar) kaldırılmalıdır.

**12.10.** Uygulamayı çalıştıran sistem kullanıcısının, hizmet verilen dizin dışındaki yetkileri kaldırılmalıdır.

**12.11.** Sunucu üzerinde bulunan ve web tabanlı istatistik sağlayan uygulamalara erişim herkese açık olmamalıdır.

**12.12.** Yetki hakkının artık gerekmediği durumlarda (örneğin kurumdan ayrılma, projede rol değiştirme gibi) en kısa sürede ilgili haklar iptal edilmelidir.

**12.13.** Bütün başarılı ve başarısız login işlemleri ve kaynaklara erişim denemeleri kayıt altına alınmalıdır.

**12.14.** Kriptografi için yeterli rastgeleliği sağlayan kriptografik tekniklerin kullanım desteklenmeli ve anahtar yönetimi bulunmalıdır.

## 13. Farkındalık ve Eğitim

**13.1.** Yazılım geliştiricileri ve ekip üyeleri için, güvenli kodlama prensipleri ve güvenlik tehditleri konusunda düzenli eğitimler verilmelidir.

**13.2.** Eğitimler, sektördeki en iyi uygulamalar ve yeni tehditlere uyum sağlayacak şekilde güncellenmelidir.

**13.3.** Güvenlik tehditlerine karşı farkındalık oluşturacak içerikler ve rehberler paylaşılmalıdır.

**13.4.** Yazılım projelerinde, **MITRE tarafından desteklenen ve açık kaynak topluluk projesi olarak yayımlanan CWE (Common Weakness Enumeration)** listesinde tanımlanan zayıflıklar dikkate alınmalı ve bu zayıflıklara karşı gerekli önlemler alınmalıdır.

**13.5.** CWE listesinde yer alan yaygın güvenlik açıkları düzenli olarak gözden geçirilmeli ve yazılım geliştirme ekibi bu listeye karşı farkındalık sahibi olmalıdır.

**13.6.** Yazılımda kullanılacak harici materyaller için fikri mülkiyet haklarına göre materyallerin kullanımı ve patentli yazılım ürünlerinin kullanımı üzerindeki yasal, düzenleyici ve anlaşmalarla doğan gereksinimlere uyum sağlanmalıdır.

**13.7.** Bilgi sistemlerinin birbirine bağlantısı ile ilişkili bilgiyi korumak için politikalar ve prosedürler geliştirilmeli ve gerçekleştirilmeli, bilgi sızması fırsatları önlenmelidir.

## 14. Dış Kaynak ve Tedarikçi Yönetimi

**14.1.** Hizmet alınacak tedarikçilerin bilgi güvenliği standartlarına (ISO 27001, SOC 2, vb.) uygunluğu değerlendirilmelidir. Ayrıca, tedarikçilerin geçmiş performansları ve güvenlik ihlalleri incelenmelidir.

**14.2.** Dış tedarikçilerden alınan yazılımlar, güvenlik standartlarına uygunluk açısından değerlendirilmelidir. Alınan yazılımlar dağıtılmadan önce penetrasyon testlerine tabi tutulmalıdır.

**14.3.** Tedarikçilerden alınan yazılımların güvenlik testleri tamamlanmadan kullanıma alınmamalıdır.

**14.4.** Tedarikçilerle yapılan sözleşmelerde, güvenlik gereklilikleri(Ör: veri güvenliği, erişim kontrolleri, hizmet sürekliliği vb.) ve sorumluluklar açıkça belirtilmelidir. Örneğin sözleşmeye, "Tedarikçi, tüm hassas verilerin şifrelenerek saklanmasını sağlayacaktır" maddesinin eklenmesi gibi. Ayrıca Üniversitenin, tedarikçilerin güvenlik kontrollerini periyodik olarak denetleme hakkı sözleşmede yer almalıdır. Örneğin "Üniversite, tedarikçinin veri merkezi altyapısını yılda bir kez denetleyebilir" maddesi gibi.

## 15. Yaptırım

Bu politikanın ihlal edilmesi durumunda KTÜN BİLGİ GÜVENLİĞİ POLİTİKASI'nda belirtilen "POLİTİKANIN İHLALİ VE YAPTIRIMLAR" başlığı altındaki maddeler geçerlidir.

## 16. Yürürlük

Bu politika, BGYS komisyonu tarafından onaylandıktan sonra Konya Teknik Üniversitesi Bilgi İşlem Daire Başkanlığının web sayfasında yayımlanarak duyurusu yapıldığı tarihte yürürlüğe girer.

## 17. Yürütme

Bu politika, Bilgi İşlem Daire Başkanlığı tarafından yürütülür.