MUH441 Bilişimde Güvenlik – 1

Prof. Dr. Hasan Hüseyin BALIK

(11. Hafta)

İçerik

- 3.Yazılım Güvenliği ve Güvenilir sistemler
  - 3.1.Arabellek Taşması
  - 3.2.Yazılım Güvenliği
  - 3.3.İşletim Sistemi Güvenliği
  - 3.4. Bulut Güvenliği
  - 3.5. IoT Güvenliği

3.2.Yazılım Güvenliği

3.2.İçerik

- Yazılım Güvenliği Sorunları
- Program Girdisini İşleme
- Güvenli Program Kodu Yazma
- İşletim Sistemi ve Diğer Programlarla Etkileşim
- Program Çıktısını İşleme

Yazılım Hatası Kategorisi: Bileşenler Arasında Güvenli Olmayan Etkileşim

- Bir SQL Komutunda kullanılan Özel Öğelerin Uygunsuz Nötrleştirilmesi ("SQL Enjeksiyonu")
- İşletim Sistemi Komutunda Kullanılan Özel Unsurların Uygun Olmayan Nötrleştirilmesi ("İşletim Sistemi Komut Enjeksiyonu")
- Web Sayfası Oluşturma Sırasında Girdinin Uygun Olmayan Nötrleştirilmesi ("Siteler Arası Komut Dosyası Çalıştırma")
- Tehlikeli Türde Dosyanın Sınırsız Yüklenmesi
- Siteler Arası İstek Sahteciliği (CSRF)
- Güvenilmeyen Siteye URL Yönlendirmesi ("Açık Yönlendirme")

Yazılım Hatası Kategorisi: Riskli Kaynak Yönetimi

- Girdi Boyutunu Kontrol Etmeden Tampon Kopyalama ("Klasik Tampon Taşması")
- Kısıtlanmış dizine yol adının uygun olmayan şekilde sınırlandırılması ("Yol Geçişi")
- Bütünlük Kontrolü Olmadan Kodun İndirilmesi
- Güvenilmeyen Kontrol Alanından İşlevselliğin Dahil Edilmesi
- Potansiyel Olarak Tehlikeli İşlev Kullanımı
- Tampon Boyutunun Yanlış Hesaplanması
- Kontrolsüz Biçim Dizesi
- Tamsayı Taşması veya Sarmalama

Yazılım Hatası Kategorisi: Gözenekli Savunmalar

- Kritik İşlev için Eksik Kimlik Doğrulaması
- Eksik Yetkilendirme
- Sabit Kodlanmış Kimlik Bilgilerinin Kullanımı
- Hassas Verilerin Eksik Şifrelenmesi
- Bir Güvenlik Kararında Güvenilmeyen Girdilere Güvenme
- Gereksiz Ayrıcalıklarla Yürütme
- Yanlış Yetkilendirme
- Kritik Kaynak için Yanlış İzin Ataması
- Bozuk veya Riskli Bir Şifreleme Algoritmasının Kullanımı
- Aşırı Kimlik Doğrulama Girişimlerinin Uygunsuz Olarak Kısıtlanması
- Tuzsuz Tek Yönlü Hash Kullanımı

CWE/SANS En Tehlikeli 25 Yazılım Hatası (2011)

Sıra	Zaafiyet	Değ. (2021)
1	Belirlenen Alanın Dışına Yazma	-
2	Web Sayfası Oluşturma Sırasında Girdinin Uygun Olmayan Nötrleştirilmesi (“Siteler Arası Komut Dosyası Çalıştırma”)	-
3	Bir SQL Komutunda kullanılan Özel Öğelerin Uygunsuz Nötrleştirilmesi ("SQL Enjeksiyonu")	+3
4	Hatalı Giriş Doğrulaması	-
5	Belirlenen alanın Dışını Okuma	-2
6	İşletim Sistemi Komutunda Kullanılan Özel Unsurların Uygun Olmayan Nötrleştirilmesi ("İşletim Sistemi Komut Enjeksiyonu")	-1
7	Serbest kadıktan sonra kullanma	-
8	Kısıtlanmış dizine yol adının uygun olmayan şekilde sınırlandırılması ("Yol Geçişi")	-
9	Siteler Arası İstek Sahteciliği (CSRF)	-
10	Tehlikeli Türde Dosyanın Sınırsız Yüklenmesi	-
11	NULL Pointer dereferansı	+4
12	Güvenilmeyen Verilerin Seri Halinden Çıkarma	+1
13	Tamsayı Taşması veya Sarmalama	-1
14	Yanlış Kimlik Doğrulama	-
15	Hard Kodlanmış Kimlik Bilgilerinin Kullanımı	+1
16	Eksik Yetkilendirme	+2
17	Bir Komutta Kullanılan Özel Unsurların Uygunsuz Nötrleştirilmesi ("Komut Enjeksiyonu")	+8
18	Kritik İşlev için Eksik Kimlik Doğrulaması	-7
19	Bir Bellek Bufferının Sınırları İçerisindeki İşlemlerin Uygun Olmayan Kısıtlaması	-2
20	Yanlış Varsayılan İzinler	-1
21	Sunucu Tarafı İstek Sahtekarlığı (SSRF)	+3
22	Uygun Olmayan Senkronizasyonla Paylaşılan Kaynak Kullanarak Eşzamanlı Yürütme ("Yarış Durumu")	+11
23	Kontrolsüz Kaynak Tüketimi	+4
24	XML Harici Varlık Referansının Uygun Olmayan Kısıtlaması	-1
25	Kod Üretiminin Uygun Olmayan Kontrolü ("Kod Enjeksiyonu")	+3

CWE En Tehlikeli 25 Hata (2022)

Güvenlik Kusurları

- Kritik Web uygulaması güvenlik kusurları, güvenli olmayan yazılım koduyla ilgili beş tanedir. Bunlar
  - Doğrulanmamış giriş
  - Siteler arası komut dosyası çalıştırma
  - Arabellek taşması
  - Enjeksiyon kusurları
  - Uygun olmayan hata işleme
- Bu kusurlar, programlardaki verilerin ve hata kodlarının yetersiz kontrolü ve doğrulanmasının bir sonucu olarak ortaya çıkar.
- Bu sorunların farkında olmak, daha güvenli program kodu yazmak için kritik bir ilk adımdır.
- Yazılım geliştiricilerin bu bilinen endişe alanlarını ele alma ihtiyacına vurgu yapılmalıdır.

Yazılım Güvenlik Açıklarını Azaltma

- NIST raporu NISTIR 8151, yazılım güvenlik açıklarının sayısını azaltmak için bir dizi yaklaşım sunar
- Önerileri şunlardır:
  - Yazılım özelleştirmek ve oluşturmak için geliştirilmiş yöntemler kullanarak güvenlik açıklarını oluşmadan önce durdur
  - Daha iyi ve daha verimli test teknikleri kullanarak güvenlik açıklarını istismar edilmeden önce bul
  - Daha dayanıklı mimariler oluşturarak güvenlik açıklarının etkisini azalt

Yazılım Güvenliği, Kalite ve Güvenilirlik

- Yazılım kalitesi ve güvenilirliği:
  - Bazı teorik olarak rastgele, beklenmedik girdiler, sistem etkileşimi veya yanlış kod kullanımının bir sonucu olarak programın kazara başarısız olmasıyla ilgili endişeler
  - Bir programdaki mümkün olduğu kadar çok hatayı belirleyip ortadan kaldırmak için yapılandırılmış tasarım ve test kullanarak iyileştirme
  - Endişe, kaç hata olduğu değil, ne sıklıkta tetiklendikleridir
- Yazılım güvenliği:
  - Saldırgan, özellikle saldırgan tarafından istismar edilebilecek başarısızlıkla sonuçlanan hataları hedef alarak olasılık dağılımını seçer.
  - Genellikle beklenenden önemli ölçüde farklı olan girdiler tarafından tetiklenir
  - Yaygın test yaklaşımlarıyla tanımlanması olası değildir

Defansif Programlama

- Güvenli programlama olarak da anılır
- Saldırı altındayken bile çalışmaya devam edecek şekilde yazılım tasarlama ve uygulama
- Programın yürütülmesine, ortama ve işlediği veri türüne ilişkin tüm hususlara dikkat gerektirir
- Yazılım, bazı saldırılardan kaynaklanan hatalı koşulları tespit edebilir
- Temel kural, asla bir şey varsaymamak, tüm varsayımları kontrol etmek ve olası hata durumlarını ele almaktır.

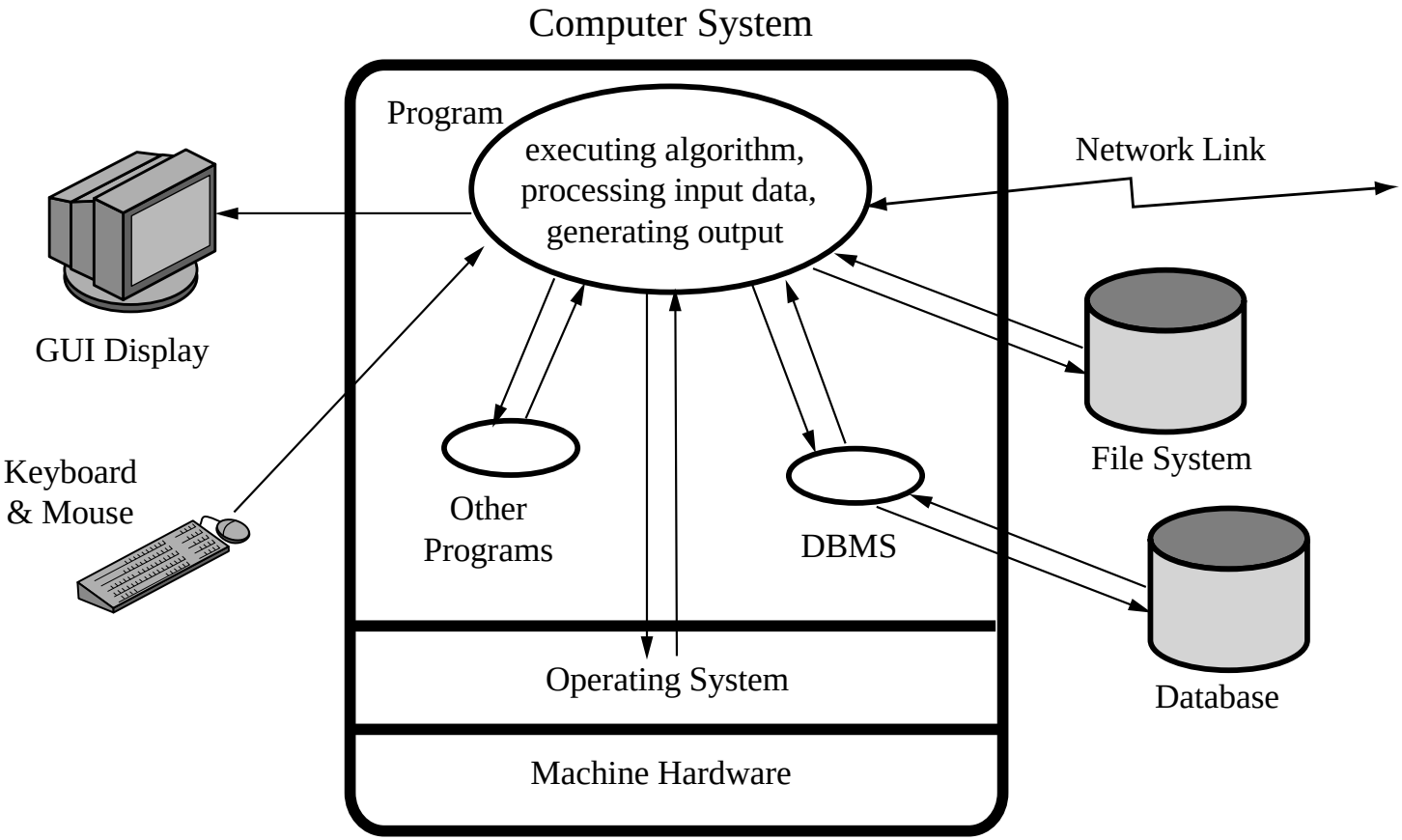


Figure 11.1 Abstract View of Program

Defansif Programlama

- Programcılar genellikle bir programın alacağı girdilerin türü ve yürütüldüğü ortam hakkında varsayımlarda bulunurlar.
  - Varsayımların program tarafından doğrulanması gerekir ve tüm olası hatalar zarif ve güvenli bir şekilde ele alınmalıdır
- Geleneksel programlama uygulamalarına göre değişen bir zihniyet gerektirir
  - Programcılar, çökmelerin nasıl meydana gelebileceğini ve programlarında meydana gelme şanslarını azaltmak için gereken adımları anlamalıdır.
- Pazar avantajını en üst düzeye çıkarmak için geliştirme sürelerini olabildiğince kısa tutmaya yönelik iş baskılarıyla çelişir

Tasarımla Güvenlik

- Güvenlik ve güvenilirlik, çoğu mühendislik disiplininde ortak tasarım hedefleridir.
- Yazılım geliştirme henüz bu olgunluk düzeyine ulaşmadı ve toplum, yazılımda diğer mühendislik disiplinlerinde olduğundan çok daha yüksek başarısızlık seviyelerine müsamaha gösteriyor
- Son yıllarda, güvenli yazılım geliştirme süreçlerini iyileştirmek için artan çabalar görülmektedir
- Kodda Mükemmellik için Yazılım Güvencesi Forumu (Software Assurance Forum for Excellence in Code - SAFECode)
  - Yazılım güvencesi için endüstrinin en iyi uygulamalarını özetleyen ve güvenli yazılım geliştirme için kanıtlanmış yöntemleri uygulamaya yönelik pratik tavsiyeler sağlayan yayınlar geliştir

Program Girdisini İşleme

Giriş Boyutu ve Arabellek Taşması

- Programcılar genellikle beklenen maksimum girdi boyutu hakkında varsayımlarda bulunurlar.
  - Ayrılan arabellek boyutu kontrol edilmez ise
  - Arabellek taşmasıyla sonuçlanır
- Test, güvenlik açığını belirlemeyebilir
  - Test girdilerinin, taşmayı tetikleyecek kadar büyük girdiler içermesi olası değildir.
- Güvenli kodlama, tüm girdileri tehlikeli olarak ele alır

Program Girdisinin Yorumlanması

- Program girdisi ikili veya metin olabilir
  - İkili girdinin yorumlaması kodlamaya bağlıdır ve genellikle uygulamaya özeldir
- Kullanılan karakter setlerinin çeşitliliği giderek artmaktadır
  - Hangi setin kullanıldığını ve hangi karakterlerin okunduğunu belirlemek için özen gösterilmelidir.
- Doğrulama başarısızlığı, istismar edilebilir bir güvenlik açığına neden olabilir
- 2014 Heartbleed OpenSSL hatası, ikili girdi değerinin geçerliliğini kontrol edememenin doğurduğu zafiyetin yakın tarihli bir örneğidir.

Enjeksiyon Saldırıları

- Özellikle program giriş verilerinin yanlışlıkla veya kasıtlı olarak programın yürütme akışını etkileyebileceği durumlarda, girdi verilerinin geçersiz işlenmesiyle ilgili kusurlardır

Çoğu zaman perl, PHP, python, sh gibi skrip dillerinde görülür

- Kodlama çabasından tasarruf etmek için mümkün olduğunda diğer programların ve sistem yardımcı programlarının yeniden kullanılmasını teşvik eder
- Bazı sistemlerde uygulama geliştirmek için kullanılır
- Genellikle HTML formlarından sağlanan verileri işlemek için Web CGI scripleri olarak kullanılır

Siteler Arası Komut Dosyası Çalıştırma (XSS) Saldırıları

Bir kullanıcı tarafından sağlanan girdinin daha sonra başka bir kullanıcıya verildiği saldırılardır

Komut dosyası içeren Web uygulamalarında yaygın olarak görülür

- Güvenlik açığı, komut dosyası kodunun HTML içeriğine dahil edilmesini içerir
- Komut dosyası kodunun diğer sayfalarla ilişkili verilere erişmesi gerekebilir
- Tarayıcılar güvenlik kontrolleri uygular ve aynı siteden gelen sayfalara veri erişimini kısıtlar

Bir sitedeki tüm içeriğin eşit derecede güvenilir olduğu ve dolayısıyla sitedeki diğer içerikle etkileşime girmesine izin verildiği varsayımından yararlanır

XSS yansıma güvenlik açığı

- Saldırgan, bir siteye sağlanan verilere kötü amaçlı komut dosyası içeriği ekler

Giriş Söz Dizimini (input syntax) Doğrulama

Verilerin sonraki kullanımdan önce veriler hakkında yapılan tüm varsayımlara uygun olduğundan emin olunmalıdır.

Girdi verileri istenenlerle karşılaştırıl-malıdır

Alternatif, girdi verilerini bilinen tehlikeli değerlerle karşılaştır-maktır

Yalnızca bilinen güvenli verileri kabul ederek programın güvenli kalması daha olasıdır

Alternatif Kodlamalar

Metni kodlamak için birden fazla araca sahip olabilir

Dünyanın dört bir yanındaki kullanıcıları desteklemek ve onlarla kendi dillerini kullanarak etkileşim kurmak için artan gereksinim vardır

Uluslararasılaştırma için Unicode kullanılır

- Karakterler için 16 bitlik değer kullanır
- UTF-8, 1-4 bayt dizileri olarak kodlar
- Birçok Unicode kod çözücü, herhangi bir geçerli eşdeğer diziyi kabul eder

Kanonikleştirme

- Girdi verilerini tek, standart, minimum bir gösterime dönüştürme
- Bu yapıldıktan sonra girdi verileri, kabul edilebilir girdi değerlerinin tek bir gösterimi ile karşılaştırılabilir

Sayısal Girdiyi Doğrulama

- Girdi verileri sayısal değerleri temsil ettiğinde ek endişe oluşur
- Sabit boyutlu değerde dahili olarak depolanır
  - 8, 16, 32, 64 bit tamsayılar
  - Kayan noktalı sayılar kullanımı işlemciye bağlıdır
  - Değerler işaretli (signed) veya işaretsiz (unsigned) olabilir
- Metin biçimini doğru yorumlamalı ve tutarlı bir şekilde işlemelidir
  - İşaretli (signed) veya işaretsiz (unsigned) sayılar arasında karşılaştırma sorunları vardır
  - Arabellek taşma kontrolünü engellemek için kullanılabilir

Girdi Fuzzing

1989 yılında Wisconsin Madison Üniversitesi'nde Profesör Barton Miller tarafından geliştirilmiştir

Rastgele oluşturulmuş verileri bir programa girdi olarak kullanan yazılım test tekniğidir

- Girdi aralığı çok geniştir
- Amaç, programın veya işlevin anormal girdileri doğru bir şekilde işleyip işlemediğini belirlemektir
- Basit, varsayımsız, ucuzdur
- Güvenilirliğin yanı sıra güvenlik konusunda da yardımcı olur

Bilinen sorunlu girdilerin sınıflarını oluşturmak için şablonları da kullanabilir

- Dezavantajı, diğer girdi biçimleri tarafından tetiklenen hataların gözden kaçmasıdır
- Girdilerin makul ölçüde kapsamlı bir şekilde kapsanması için yaklaşımların kombinasyonu gereklidir

Güvenli Program Kodu Yazma

- İkinci bileşen, gerekli sorunu çözmek için verilerin bir algoritma tarafından işlenmesidir.
- Yüksek seviyeli diller tipik olarak derlenir ve daha sonra doğrudan hedef işlemci tarafından yürütülen makine koduna dönüştürülür.

Güvenlik sorunları:
• Algoritmanın doğru uygulanması • Algoritma için doğru makine komutlarının kullanımı • Verilerin geçerli manipülasyonu

Doğru Algoritma Uygulaması

İyi program geliştirme tekniği sorunudur

Algoritma, tüm sorun varyantlarını doğru bir şekilde işlemeyebilir

Eksikliğin sonucu, ortaya çıkan programda istismar edilebilecek bir hatadır

Pek çok TCP/IP uygulaması tarafından kullanılan başlangıç sıra numaraları çok tahmin edilebilir

Sıra numarasının paketlerin tanımlayıcısı ve doğrulayıcısı olarak kombinasyonu ve paketlerin yeterince öngörülemez hale getirilmemesi, saldırının gerçekleşmesini sağlar

Başka bir varyant, programcıların test etmeye ve hata ayıklamaya yardımcı olmak için bir programa kasıtlı olarak ek kod eklemesidir

Çoğu zaman kod, bir programın üretim sürümünde kalır ve uygunsuz bir şekilde bilgi yayabilir

Kullanıcının güvenlik kontrollerini atlamasına ve normalde izin verilmeyecek eylemleri gerçekleştirmesine izin verebilir

Bu güvenlik açığı, Morris Internet Worm tarafından istismar edilmiştir

Makine Dilinin Algoritmaya Karşılık Gelmesini Sağlama

- Sorun çoğu programcı tarafından göz ardı edilir
  - Varsayım, derleyici veya yorumlayıcının, dil deyimlerini geçerli bir şekilde uygulayan kodu ürettiği veya yürüttüğüdür.
- Makine kodunu orijinal kaynakla karşılaştırmayı gerektirir
  - Yavaş ve zordur
- Çok yüksek güvence düzeyine sahip bilgisayar sistemlerinin geliştirilmesi, bu denetim düzeyinin gerekli olduğu tek alandır.
  - Güvence düzeyi EAL 7 için aranan kriterlerdir

Değerlendirme Güvence Düzeyi (EAL)

- EAL 1: İşlevsel olarak test edilmiştir
- EAL 2: Yapısal olarak test edilmiştir
- EAL 3: Metodik olarak test edildi ve kontrol edilmiştir
- EAL 4: Metodik olarak tasarlanmış, test edilmiş ve gözden geçirilmiştir
- EAL 5: Yarı resmi olarak tasarlanmış ve test edilmiştir
- EAL 6: Yarı resmi olarak tasarım doğrulanmış ve test edilmiştir
- EAL 7: Resmi olarak tasarım doğrulanmış ve test edilmiştir.

Doğru Veri Yorumlama

- Bilgisayarda bit/bayt olarak depolanan veriler
  - Word veya longword olarak gruplanmıştır
  - Kullanılmadan önce bellekte erişilir ve işlenir veya işlemci registerlerine kopyalanır
  - Verilerin yorumlanması yürütülen makine komutuna bağlıdır
- Farklı diller, değişkenlerdeki verilerin yorumlanmasını kısıtlamak ve doğrulamak için farklı imkanlar sunar
  - Bazı diller daha sınırlı ve daha güvenlidir
  - Diğer diller, verilerin daha liberal yorumlanmasına izin verir ve program kodunun yorumlanmayı açıkça değiştirmesine izin verir

Hafızanın Doğru Kullanımı

- Dinamik bellek tahsisi sorunu
  - Bilinmeyen miktarda veri
  - Gerektiğinde tahsis edilir, bittiğinde serbest bırakılır
  - Bellek sızıntısını değiştirmek için kullanılır
  - Yığındaki kullanılabilir bellekte tamamen tükenene kadar sürekli azalma
- Birçok eski dilde, dinamik bellek tahsisi için açık bir destek yoktur.
  - Belleği ayırmak ve serbest bırakmak için standart kütüphane rutinleri kullanılır
- Modern diller bu tahsisi otomatik yapar

Yarış koşulları

- Erişimlerin senkronizasyonu olmadan, paylaşılan değerlerin çakışan erişimi, kullanımı ve değiştirilmesi nedeniyle değerlerin bozulması veya değişikliklerin kaybolması mümkündür
- Çözümü, uygun senkronizasyon primitivlerinin doğru seçimini ve kullanımını gerektiren eşzamanlı kod yazarken ortaya çıkar
- Kilitlenme (deadlock)
  - İşlemler veya iş parçacıkları, diğeri tarafından tutulan bir kaynak için beklemesidir
  - Bir veya daha fazla programın sonlandırılması gerekiyor

İşletim Sistemi Etkileşimi

Programlar, bir işletim sisteminin kontrolü altındaki sistemlerde yürütülür

- Kaynaklara erişime aracılık eder ve bunları paylaşır
- Yürütme ortamı oluşturur
- Ortam değişkenlerini ve bağımsız değişkenleri içerir

Sistemlerin çoklu kullanıcı konsepti vardır

- Kaynaklar bir kullanıcıya aittir ve farklı kullanıcı çeşitli haklarla erişim izinlere sahiptir.
- Programların çeşitli kaynaklara erişmesi gerekir, ancak aşırı düzeyde erişim yetkisi tehlikelidir
- Birden çok programın ortak bir dosya gibi paylaşılan kaynaklara erişmesiyle ilgili endişeler vardır

Ortam Değişkenleri

Herbir process tarafından ebeveyninden miras alınan string değerlerinin toplanması	• Çalışan bir processin davranış biçimini etkileyebilir • Oluşturulduğunda belleğe dahil edilir
Program süreci tarafından herhangi bir zamanda değiştirilebilir	• Değişiklikler çocuklarına geçirilir
Güvenilmeyen program girdisinin başka bir kaynağı	
En yaygın kullanım, artırılmış ayrıcalıklar elde etmeye çalışan yerel bir kullanıcıdır	• Amaç, süper kullanıcı veya yönetici ayrıcalıkları veren bir programı yıkmaktır

Zafiyetli Derlenmiş Programlar

Programlar, PATH değişken manipülasyonuna karşı savunmasız olabilir

- "Güvenli" değerlere resetlenmelidir

Dinamik olarak bağlanırsa, LD_LIBRARY_PATH manipülasyonuna açık olabilir

- Uygun dinamik kütüphaneyi bulmak için kullanılır
- Ayrıcalıklı programları statik olarak bağlamalı veya bu değişkenin kullanımı engellenmelidir

Minimum Ayrıcalığın Kullanımı

Ayrıcalık arttırma

- Kusurlardan yararlanmak, saldırgana daha fazla ayrıcalık verebilir

Minimum ayrıcalık

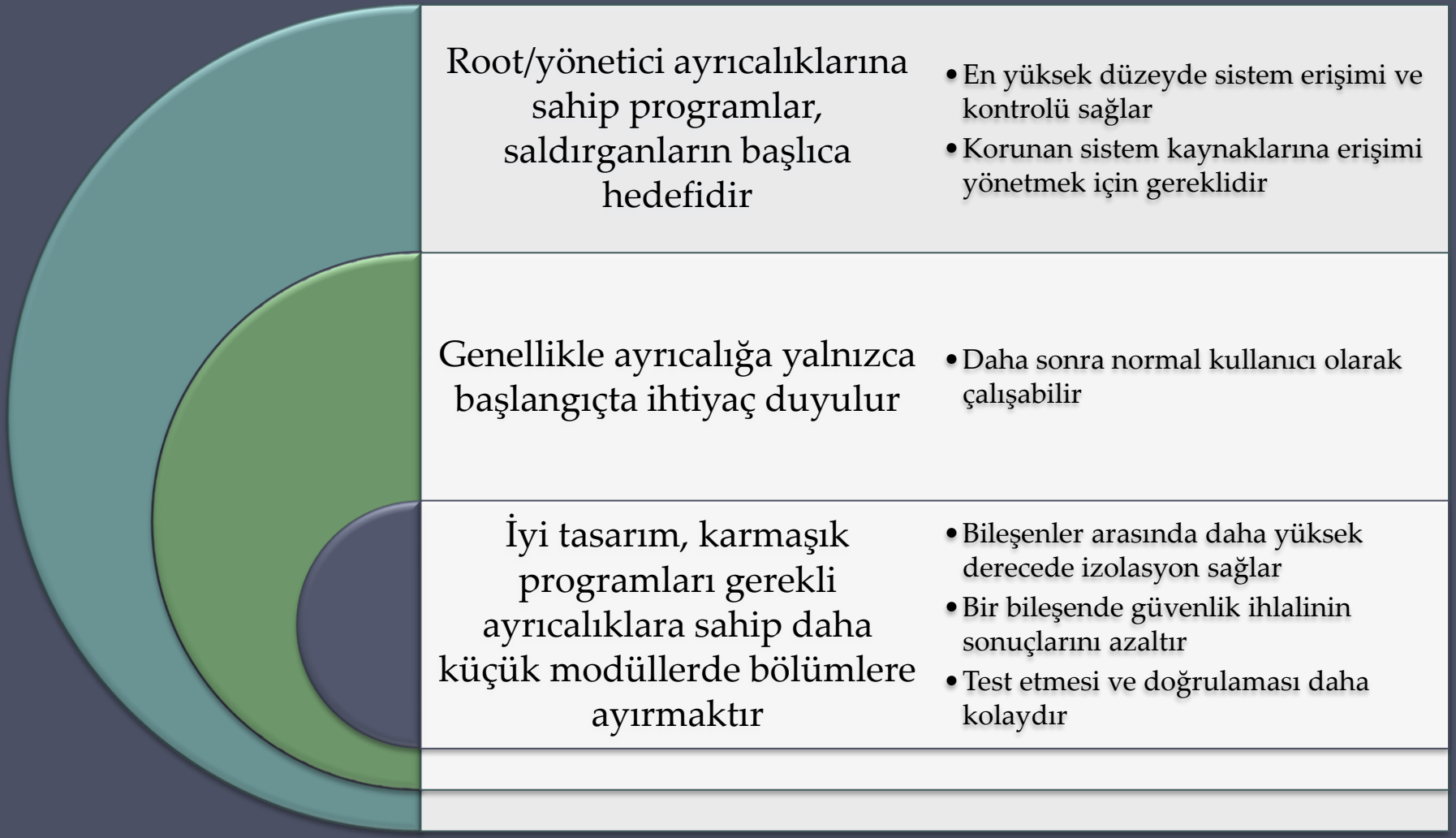
- İşlevlerini tamamlamak için gereken en az ayrıcalıkla programları çalıştırın

Gerekli uygun kullanıcı ve grup ayrıcalıklarını belirleyin

- Fazladan kullanıcı mı yoksa sadece grup ayrıcalıkları mı vereceğinize karar verin

Ayrıcalıklı programın yalnızca gerekli dosyaları ve dizinleri değiştirebildiğinden emin olun

Root/Yönetici Ayrıcalıkları



Root/yönetici ayrıcalıklarına sahip programlar, saldırganların başlıca hedefidir	• En yüksek düzeyde sistem erişimi ve kontrolü sağlar • Korunan sistem kaynaklarına erişimi yönetmek için gereklidir
Genellikle ayrıcalığa yalnızca başlangıçta ihtiyaç duyulur	• Daha sonra normal kullanıcı olarak çalışabilir
İyi tasarım, karmaşık programları gerekli ayrıcalıklara sahip daha küçük modüllerde bölümlere ayırmaktır	• Bileşenler arasında daha yüksek derecede izolasyon sağlar • Bir bileşende güvenlik ihlalinin sonuçlarını azaltır • Test etmesi ve doğrulaması daha kolaydır

Sistem Çağrıları ve Standart Kütüphane İşlevleri

Programlar, ortak işlemler için sistem çağrılarını ve standart kütüphane işlevlerini kullanır

Programcılar operasyonları hakkında varsayımda bulunurlar

- Yanlış davranış beklendiği gibi değilse
- Paylaşılan kaynaklara erişimi optimize eden sistemin bir sonucu olabilir
- Sistem kullanımını optimize etmek için arabelleğe alınan, yeniden sıralanan veya başka bir şekilde değiştirilen hizmetlere ilişkin isteklerin sonuçları
- Optimizasyonlar program hedefleriyle çelişebilir

Yarış Koşullarını Önleme

- Programların ortak bir sistem kaynağına erişmesi gerekebilir
- Uygun senkronizasyon mekanizmalarına ihtiyaç var
  - En yaygın teknik, paylaşılan dosya üzerinde bir kilit kullanmaktır.
- Kilit dosyası
  - Paylaşılan kaynağa erişim kazanmak için processin kilit dosyasını oluşturması ve sahip olması gerekir
- Endişeler
    - Bir program, kilit dosyasının varlığını göz ardı etmeyi ve paylaşılan kaynağa erişmeyi seçerse, sistem bunu engellemeyecektir
    - .Bu senkronizasyon biçimini kullanan tüm programlar işbirliği yapmalıdır
    - Uygulanma

Güvenli Temp Dosyalar

- Birçok program temp (geçici) dosyalar kullanır
- Genellikle ortak, paylaşılan sistem alanındadır
- Benzersiz olmalı, başkaları tarafından erişilmemelidir
- Genellikle işlem kimliğini kullanarak ad oluşturulur
  - Eşsiz ama öngörülebilir
  - Saldırgan tahmin edebilir ve program kontrolü ve oluşturma arasında kendi dosyasını oluşturmaya çalışabilir
- Güvenli geçici dosya oluşturma ve kullanma, rastgele adların kullanılmasını gerektirir

Diğer Programlarla Etkileşim

Programlar, diğer programların işlevlerini ve hizmetlerini kullanabilir

- Bu etkileşime özen gösterilmediği takdirde güvenlik açıkları ortaya çıkabilir
  - Bu tür sorunlar, kullanılan program ortaya çıkabilecek tüm güvenlik endişelerini yeterince tanımlamadığında özellikle endişe vericidir.
  - Programlara Web arayüzleri sağlama yönündeki mevcut eğilimle ortaya çıkar
  - Ortaya çıkabilecek herhangi bir güvenlik sorununu belirleme ve yönetme yükü yeni programlara düşer

Veri gizliliği/bütünlüğü sorunu

Etkileşimden kaynaklanan istisnaların ve hataların algılanması ve işlenmesi de güvenlik açısından önemlidir

Program Çıktısını İşleme

- Son bileşen program çıktısıdır
  - İleride kullanılmak üzere saklanabilir, ağ üzerinden gönderilebilir veya görüntülenebilir
  - İkili (binary) veya metin olabilir
- Çıktının beklenen biçime ve yoruma uygun olması program güvenliği açısından önemlidir
- Programlar, izin verilen çıktı içeriğinin ne olduğunu belirlemeli ve yalnızca geçerli çıktının görüntülenmesini sağlamak için güvenilmeyecek verileri filtrelemelidir.
- Karakter seti belirtilmelidir