

FreeBSD Denetim (Audit) Sistemi*

Metin KAYA

EnderUNIX Üyesi

Endersys Yazılım Mühendisi

metin at enderunix.org

metin.kaya at endersys.com.tr

<http://www.enderunix.org>

<http://www.endersys.com.tr>

30 Nisan 2008 Pazar EEST 23:23:34

* Robert N. M. Watson ve Wayne Salamon 'un <http://www.trustedbsd.org/20060303-ukuug2006lisa-audit.pdf> adresli makalesinden birebir çeviridir.

ÖZET

Bu makale, TrustedBSD projesi tarafından FreeBSD işletim sistemine eklenen güvenlik olaylarının denetlendiği mekanizmanın gerçekleşmesi üzerinedir. Denetim, işletim sistemi güvenliğinde ve işlemlerinde çok önemlidir; ancak her iki konuda da çok fazla karmaşıklık meydana getirir. Makalede denetim mekanizmasının gereksinimleri, FreeBSD çekirdeğinde nasıl gerçekleştirildiği, OpenSolaris BSM 'den alınan dosya biçimi, denetim izlerinin incelenme biçimi, OpenBSM kütüphanesi ve araçları üzerinde durulacaktır. Denetim kayıtlarının içeriğinin yanında bu kayıtların sırası ve yazılması da çok önemlidir.

1. Giriş

Bu makale FreeBSD [12] işletim sistemi için TrustedBSD projesi tarafından geliştirilen güvenlik denetim sisteminden bahseder. Güvenlik denetim sistemi, güvenlikle ilgili olayların güvenilir ve detaylı biçimde kayıt edilmesi demektir ve güvenliğin hassas olduğu sistemlerde çok önemlidir.

Güvenlik denetiminin işlemleri, standart ihtiyaçları, tasarım ve gerçekleştirilmesi, denetim kayıtlarının yönetimi ve bu kayıtların biçimini tanımlamalıyız. Denetim mekanizmasının gerçekleştirilmesi FreeBSD 'nin hem kullanıcı hem de çekirdek uzayındaki kodlarının değiştirilmesini, çekirdek ve uygulamalara ilişkin denetim verilerinin toplanması ve yönetilmesini kapsar. Ticari UNIX sistemlerde, güvenlik denetim mekanizması *Common Criteria* standartları uyarınca zaten mevcuttur; ancak çok sayıda tasarım hatası içermektedir ve buradaki örnek çalışmayla bu hataların giderilmesine yardım etmeyi umuyoruz.

FreeBSD işletim sistemindeki güvenlik denetim mekanizması, Apple Computer 'ın Mac OS X [6] işletim sisteminden alınmıştır. Apple tarafından BSD lisansı altında çıkarılan Darwin [1] çekirdeği FreeBSD 'ye katkıda bulunmaktadır. Hem FreeBSD hem de Darwin 'deki güvenlik denetim mekanizması, çok sayıda uygulamanın kullandığı ve gayr-i resmi standart haline gelen Solaris 'in BSM projesinden büyük oranda etkilenmiştir.

2. Güvenlik Denetim Kayıtlarının Tutulması İçin Gerekenler

Modern ve güvenilir bir işletim sistemi, güvenliğin temel bileşenlerini içerir. Bu bileşenlerden biri olan güvenlik denetim mekanizması ise sistem yöneticilerine, kimliği doğrulanmış kullanıcıların, sistem güvenliğini etkileyen tüm hareketlerini izleme olanağı verir. İşletim sistemi açısından "kullanıcı", kullanıcı tarafından çalıştırılan süreçler demektir. Kimliği doğrulanmış kullanıcıları sadece *uid* ile takip etmek yetersiz olacaktır: *setuid* programlarla normal kullanıcı gibi görünmeye devam edebilirler.

Güvenlik denetim mekanizması tasarımındaki en önemli konulardan biri hangi olayların denetleneceğidir. CAPP [7] koruma profiline göre aşağıdaki sınıflara giren olaylar denetlenmelidir:

Kontrol Altındaki İşlemler: Kontrol altındaki dosyalara, ağ kaynaklarına ve diğer nesnelere erişimle ilgilidir. İzin, sahiplik ve ayrıcalık kontrolleri gibi tüm erişim kontrol kararları bu bağlamdadır.

Doğrulama: Başarılı ve başarısız doğrulama işlemleri, güvenlik servislerinin kullanımı v.s. UNIX kullanıcı uzayındaki sıradan yapılardır.

Güvenlik Yönetimi: Kullanıcı haklarında, denetim mekanizmasında ve UNIX */etc* dizinindeki yapılandırma dosyalarında yapılan değişikliklerdir.

Saldırı girişimi tespit sistemleri, ölümcül felaket sonrası analizleri ve sistemin genel işleyişiyle ilgili kayıtlar güvenlik denetim mekanizmasının ilgilenmesi gereken noktalar. Hem standartlar hem de pratik uygulamalar açısından bir anahtar gereksinim, denetim mekanizmasının çok daha detaylı bilgi sunması mümkün iken bunun kısıtlanabilmesidir.

Dikkat edilmesi gereken noktalar genelde kimliği doğrulanmış kullanıcılar, sahiplik, izin, tip ve metot gibi nesne özellikleri ve erişim kontrol kararlarının sonuçları ve doğası gereği yapılması gerekenlerdir. Takip edilen güvenlik olaylarının detayının artması denetim kayıtlarının büyük disk alanı tutmasına neden olabilir. Bu nedenle sistem yöneticileri izlenen olayların çeşitlerini – örneğin; sisteme girmeye çalışan herkesin değil sadece girebilmiş olanların izlenmesi – ve detay seviyesini gözden geçirebilir. Ön seçimler, nelerin izlenip izlenmeyeceğine karar verildiğinde belli olurken son seçimler denetim kayıtları incelendikten sonra ortaya çıkar.

Denetleme sistemi kontrol edilen nesnelere erişmek isteyen süreçlerle ilgili kayıtları tutar. Çekirdek olayları, erişim kontrollerine aracılık eden sistem çağrılarını; ancak bunun dışında çok sayıda işlemin de kaydı tutulur: sistemin kapatılması, *root* hakkının kullanılması ve denetim sistemiyle ilgili yapılandırma dosyalarının değiştirilmesi.

Denetim kayıtlarını çekirdek sahiplenir ve bu kayıtları tek yazma yetkisi çekirdektedir. Ama güvenlikle ilgili kayıtları tutmak için pek çok yazılım gerçekleştirildi. Mevcut *syslog* [5] özelliğinin aksine, güvenilmeyen süreçlerin *audit* kayıtlarına yazma yetkisi yoktur. Bu nedenle süreçler, kaydedilmesini istedikleri denetim kayıtlarını çekirdeğe iletir. UNIX 'in klasik erişim kontrol mekanizması hem denetim kayıtlarını hem de yapılandırmasını korumak için kullanılıyor.

Kaydı tutulabilir olayla ile denetim kaydının oluşturulması arasında anahtar bir ilişki söz konusudur. Standartlarda bu konu açıkça ifade edilmiştir: eğer bir işlem kaydedilebilirse ve bu işlemin gerçekleşmesine izin verildiyse bu olay kaydedilmelidir. Bu durum büyük bir riski beraberinde getiriyor – eğer diskte denetim kaydını yazacak yer kalmadıysa işlemin gerçekleşmesine izin verilmeli mi? 2 sakıncalı seçenek var: sistem diskte yer kalmadığı için kendini tamamen kapatır veya bu işlemin kaydı düzgün tutulmadan gerçekleşmesine izin verilir. Standartlarda denetim kaydının yazılmasında bir sorun olduğunda sistem durdurulmalıdır; ancak bunun değiştirilebilir bir davranış olması gerekir. Harici bir sorun olduğunda, örneğin elektrik kesintisi, denetim mekanizmasının tanımlı bir üst limiti kayıp kayıtlar için tanımlayabilmesi de gerekir. Bu limit genellikle yazılmayı bekleyen asenkron kayıtların sayısına eşit olarak tanımlanır.

2.1. Genel Kriter (Common Criteria - CC) ve Kontrol Edilen Erişim Koruma Şekilleri (Controlled Access Protection Profiles - CAPP)

CAPP [7], bilgi güvencesi belgeleri olan CC [2] 'nin bir parçasıdır. CC 'nin amacı, güvenlik özellikleri etkin bilgi teknolojileri ürünleri geliştirmek ve değerlendirmektir. CC 'nin hedefinde gizlilik, bütünlük ve sistem tarafından oluşturulan bilgilerin erişilebilirliği vardır. Bu nedenle insan hareketleri ve davranışları üzerinde yoğunlaşmıştır. FreeBSD denetim mekanizması da CC 'nin belirlediği kriterler altında sistemdeki kritik izleme gereksinimlerini karşılamayı hedefler.

Bir koruma şekli, sistemin yerine getirmesi gereken güvenlik gereksinimlerinin tasarlanmasıdır. CC çatısı altında çok sayıda koruma şekli tanımlanmıştır. CAPP özellikle kullanıcı verilerine aracılık yapan isteğe bağlı erişim kontrolleriyle ilgilenir – bu yaklaşık olarak TCSEC (Trusted Computer System Evaluation Criteria) 'in C2 sınıfına karşılık düşer.

CAPP mevcut özelliklerin yanında sisteme giriş, sistemden çıkış, nesne-konu etkileşimi, denetim mekanizmasının yönetim özellikleri ve bunun gibi pek çok işlemin denetlenebilmesini gerektirir. Her ne kadar bu makalenin yazıldığı zaman resmi bir değerlendirme yapılmamış olsa da FreeBSD denetim mekanizması bu gereksinimlerin hepsini karşılamaktadır.

3. Denetim Mekanizmasının Gerçeklenmesi

FreeBSD denetim mekanizması aşağıdaki bileşenlerden oluşmaktadır:

- **sys/security/audit**: güvenilir denetim kayıtları kuyruğu ve denetleyici sistem çağruları
- **contrib/openbsm**: BSM kütüphanesi, belgeleri ve denetim araçları
- **etc/security**: yapılandırma dosyaları
- **usr.sbin/auditd**: denetim servisi

Denetim mekanizmasının kayıtları değiştirilemez ve güvenilir olmak zorunda olduğundan kayıt kuyruğu ve denetlenebilen olayların yakalanması çekirdekte gerçekleştirilir. Çekirdekte üretilen denetim kayıtlarına ek olarak güvenilir süreçlerin, denetim kayıtlarını *audit()* sistem çağrısıyla çekirdeğe bildirmeleri gerekmektedir. Denetim kayıtlarının gözden geçirilmesi ve yönetilmesi denetim izlerine ve yapılandırmada belirtilen izinlere göre kullanıcı uzayındaki uygulamalarla yapılır.

3.1. BSM Denetim İzi Biçimi

Başlık	Sözcük	Sözcük	Konu	Dönüş	İz
--------	--------	--------	------	-------	----

Şekil 1: Denetim Kayıt Biçimi

Sun Microsystems, BSM (Basic Security Module) [11] kayıt biçimini tanımladı. Denetim kayıt biçiminin bu şekilde tasarlanmasının altında yatan neden mevcut işleme ve yönetim araçlarının kullanımına izin veren gayr-ı resmi denetim izi standardını oluşturmaktır. Ayrıca, bu projede tanımlanan denetim kaydı biçimi, işlemci mimarisinden bağımsız olacak şekilde tasarlandı ve Solaris 'te şu an bulunmayan özelliklerin FreeBSD 'de kolayca geliştirilmesini sağladı.

Bir BSM günlüğü, bir ya da daha fazla sayıda denetim kaydından oluşur. Her bir kayıt; başlık, konu, dönüş ve iz gibi veri elemanlarını taşıyan bir dizi sözcüktür. Tip ve uzunluk bilgileri her kayıta yer aldığından ayrıştırıcılar (parser) tanımadıkları kayıtları es geçebilirler. Denetim kaydı biçimi Şekil 1 'de gösterilmiştir.

Denetim kaydındaki başlık kısmı; toplam kayıt uzunluğu, olay tipi ve zaman gibi denetim olayı hakkındaki genel bilgileri içerir. Konu alanında kullanıcının kimlik numaraları, süreç numarası ve süreçle ilgili diğer bilgiler yer alır. Dönüş kısmında ise sistem çağrısının başarılı ya da başarısız olduğunu gösteren sonuç bilgisi vardır. İz kısmı, denetim kaydını sonlandırır ve tüm kaydın toplam boyutunu gösterir.

Kullanılan nesnenin yapısına bağı olarak geliştirilmiş çok sayıda denetim kayıt biçimi vardır. Örneğin; dosya nesnesi 2 kısımla tanımlanır: *path* kısmı ve dosya nesnesinin sahip ve izin bilgilerinin tutulduğu *attribute* kısmı. Şekil 2 'de bir denetim kaydı tam olarak gösterilmiştir.

```
header,188,1,open(2) - read,write,creat,0,Wed Oct 19 19:50:51 2005, + 290 msec
argument,3,0x180,mode
argument,2,0xa02,flags
path,/usr/home/wsalamon/audit3/tools/regression/audit/test/file/temp2.duFV
subject,666,root,wheel,root,wheel,500,777,99,0.0.0.66
return,success,23
trailer,188
```

Şekil 2: Örnek Bir Denetim Kaydı

3.2. Denetim Olayları ve Sınıfları

FreeBSD çekirdeğinde denetim olayları, sistem çağruları ile ilişilendirilmiştir. Hangi sistem çağrısının denetlendiği çeşitli faktörlere bağlıdır. Eğer sistem çağrısı korumalı bir nesneye erişmek istiyorsa, ayrıcalıklı haklar gerektiriyorsa veya denetim yapılandırması değiştirildiyse bir denetim kaydı oluşturulabilir. Pratikte sistem çağrılarının çoğu başka süreçlerle etkileşimde bulunduğundan ve erişim kontrolleri gerektiren bir sistem yolunu kullanmak istediğinden bir önceki cümlede anlatılan kategoriye girmektedir.

Olay sınıfları, denetim ön seçiminin ilgili olduğu diğer denetim olayları ile yapılandırılmasına izin verir. Denetim olayları 0 ya da daha fazla sayıda sınıfa atanmıştır: örnek sınıflar süreç, ağ ve ilgili dosya erişimlerini içermektedir. Olayların hangi sınıflarla ilişkili olduğu çekirdekteki dahili bir tabloda tutulur ve bu tablo *auditon()* sistem çağrısıyla değiştirilebilir. Başlangıçtaki olay-sınıf ilişkisi, denetim işlemi başlatıldığında bir yapılandırma dosyasından çekirdeğe yüklenir.

Her kullanıcı adı, denetlenmek üzere bir olay sınıfı kümesi ile ilişkilendirilebilir. İlgili kullanıcının süreçlerini maskeleyen 2 denetim sınıfı vardır: başarılı olayları seçen maske ve başarısız olayları seçen maske. Sistem yöneticisi, denetim olay maskelerini kullanarak her kullanıcının olaylarını daha iyi denetleyebilir. Bu maskeler süreç kontrol bloğunun parçası olarak saklanır ve kullanıcı sisteme girdiğinde ayarlanır.

4. FreeBSD Değişiklikleri

McAfee Research, Apple ile anlaşmalı olarak Mac OS X 10.3 için Darwin-7.X çekirdeğine denetim mekanizması desteği ekledi. Çekirdek olay denetimi, kullanıcı uzayı programları ve *login* gibi üzerinde değişiklik yapılan programlar da dahil olmak üzere yazılan kodun tamamı BSD açık kod lisansı ile lisanslandı. Her ne kadar ilk gerçekleştirilmeden sonra çok büyük değişiklikler yapılmış olsa da, Darwin kodu FreeBSD denetim mekanizmasının esas yapısını teşkil eder. Yapılan değişikliklerden bir kısmı Darwin geliştiricilerine geri bildirilmektedir.

FreeBSD için yazılan denetim mekanizmasının kaynak kodunu yönetmek üzere TrustedBSD projesi altında 2 yeni proje başlatıldı. Bir proje çekirdekteki denetim mekanizmasının işlevselliğini ve kullanıcı uzayındaki programlarını yönetir. Diğer proje, OpenBSM [8], ise

genel denetim tanımlarını ve denetim kayıtlarını okumak/yazmak için kullanılan kütüphaneleri yönetir.

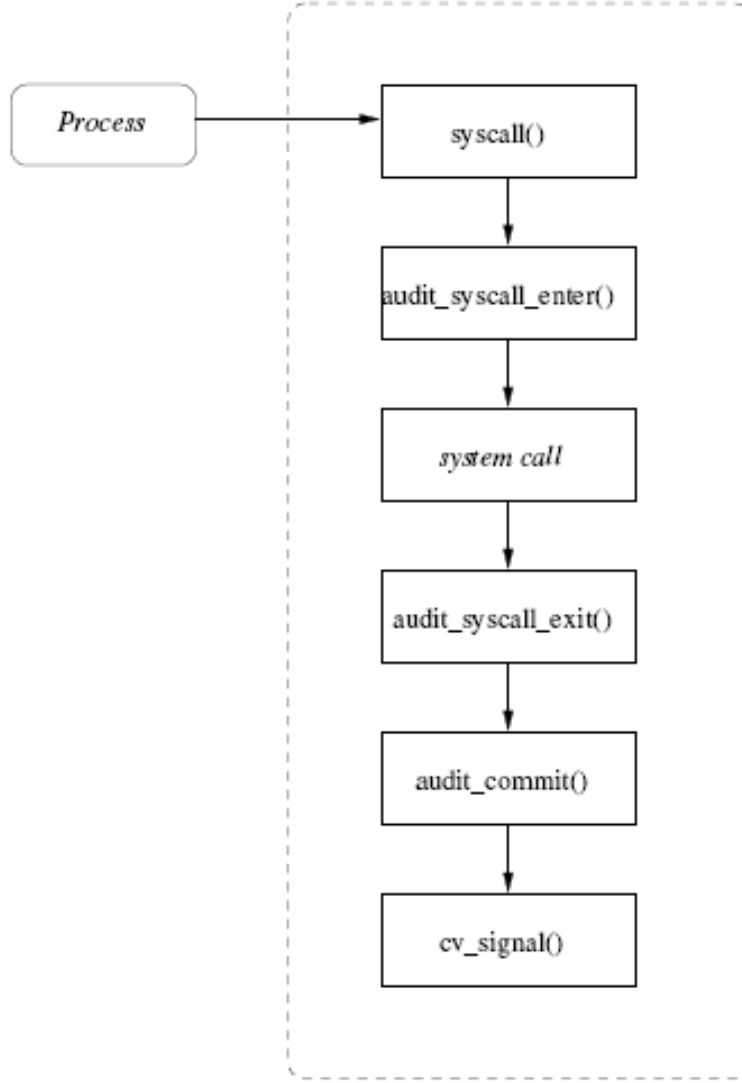
Bu bölümde denetim mekanizmasının çekirdekteki kaynak kodundan söz edilecektir. Burada denetim için kullanılan araçlar ve uygulamalar 5. bölümde açıklanacaktır. OpenBSM projesinden ve ilgili kodlardan 6. kısımda bahsedilecektir.

4.1. Çekirdek Denetim Süreci

FreeBSD çekirdeğinde yapılan değişiklikler 3 grupta incelenebilir: sistem çağrılarının denetimi, denetim kayıtlarının iç yönetimi ve denetim kayıtlarının dosya sistemine kullanıcı uzayındaki bilgilendirmelerle teslim edilmesi.

Çekirdekteki denetim kayıtlarının alacağı yerlerin merkez noktası sistem çağrıları tarafından belirlenir. Bunun nedeni ise kontrol edilen tüm nesnelere sistem çağrıları ile erişilir ve erişimlerden önce ilgili kontrollerin yapılması gereklidir. Bu olaylar için oluşturulan denetim kayıtları sürecin sahibi (konu) ve davranışlarıyla (sahiplik, v.s.) ilgili bilgileri içerecektir. Denetim olayı tanıtıcıları sistem çağrıları tablosundaki sistem çağrılarına atanmıştır. Aynı servis üzerinde çalışılıyorsa bazı sistem çağrıları aynı olay tanıtıcılarını paylaşabilir. Doğal (native) sistem çağrılarında atama işlemi *syscalls.master* 'da yapılır; ABI emülasyonunda ise ABI 'nin gerçekleştirmesindeki sistem çağrıları tablosunda yapılır.

Şekil 3 'te denetim kaydının çekirdekte nasıl oluşturulduğu görülmektedir. Sistem çağrısı girdisinde (*audit_syscall_enter()*), sürecin denetim maskeleri veya ön tanımlı maskler ile denetim kaydının oluşturulup oluşturulmayacağına karar verilir. Bu durum; denetim kuyruğu üst sınıra ulaştığında veya denetim kaydını diske yazabilmek için yeterli kaynak bulunmadığında denetim kaydı tutulacak olan sürecin çalışmasını durdurabilmek için büyük bir fırsattır. Eğer denetim kaydı tutulacaksa denetim kaydının çekirdek biçimi sürece eklenir ve konu bilgisi yazılır.

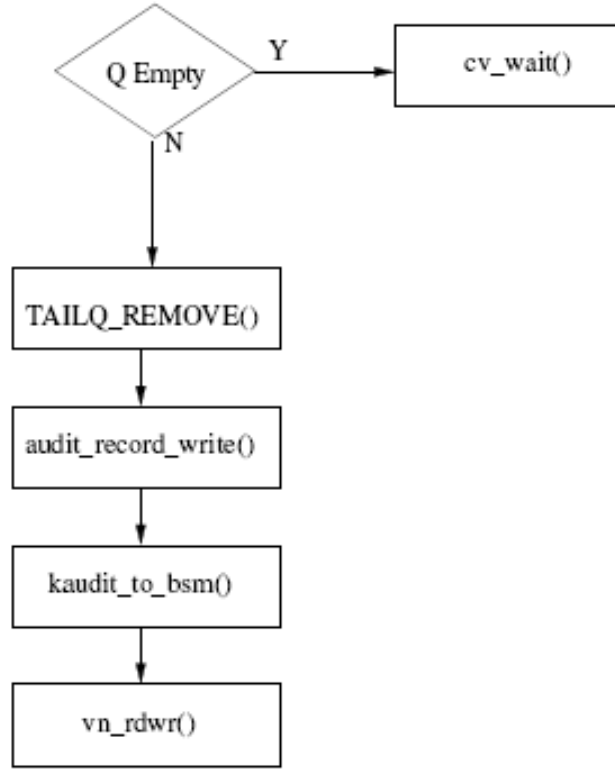


Şekil 3: Denetim Kaydı Oluşumunun Çekirdek Akışı

Sistem çağrısı denetlenmek istendiğinde parametleri ve nesne bilgisi yakalanıp çekirdek denetim kaydında saklanır. Nesne bilgisi çoğu kez sistem çağrısının başlangıcında alınır. Yol ismi (path name) aramalarında nesne bilgisi (*vnode*) yolun çekirdeğe kopyalandığı yer olan merkezileştirilmiş arama kodunda (*namei()*, *lookup()*) yakalanır.

Sistem çağrısı sonlanırken (*audit_syscall_exit()*) diğer ön seçimler, dönüş değerine uygulanır ve kayıt ya kayıt kuyruğuna teslim edilir ya da silinir. Daha sonra ise *audit_worker* çekirdek ipliğine (thread), kaydı asenkron olarak işlemesi için sinyal verilir. *audit()* sistem çağrısı için 2 denetim kaydı oluşturulur: sistem çağrısının kendisini tanımlayan denetim kaydı ve kullanıcının sağladığı denetim kaydı.

Denetim kayıt kuyruğu, Şekil 4 'te gösterilen *audit_worker* isimli çekirdek ipliği tarafından işlenir.



Şekil 4: audit_worker 'ın Çalışma Şekli

Denetim kayıtları, dosya sisteminde yeterli miktarda boş alan olup olmadığını ve günlüklerin döndürülebilirliğini (rotation) kontrol eden *audit_record_write()* tarafından silinir. Herhangi bir sınır değere ulaşıldığında *auditd* tetiklenerek denetim kaydının uygun şekilde yönetilmesi sağlanır. Gerekirse denetim mekanizması askıya alınır ve çekirdek böyle yapılandırıldıysa, kaydı diske yazamayacağı için panikleyecektir. Sistem, panik durumuna geçmeden önce yeni bir denetim yapmanın mümkün olmadığı durumda sistemin sonlanmasını önlemek amacıyla olay kuyruğunda beklemede olan tüm kayıtlar diske yazılır. Daha detaylı kurtarma davranışı mümkün olduğu müddetçe; muhtemel kurtarma işlemleri, denetim olmaksızın çalıştırılmasına izin verilmeyen ayrıcalıklı eylemleri kullanır. Özellikle yapılandırılmadıysa ön tanımlı olan, sistemdeki boş alan tükendiğinde denetim kaydını iptal etmektir.

Dosya kontrolleri başarılı olursa *kaudit_to_bsm()*, denetim verilerini çekirdeğin iç veri yapılarından denetim izine yazılabilir biçimde olan BSM kaydına dönüştürür. Dönüşüm işlemi sırasında olay tipi, nesne bilgisi ve ilgili diğer veriler de dahil olmak üzere olayı tanımlayan BSM sözcükleri oluşturulur. Son olarak da süreci tanımlayan konu kısmı, dönüş kodu kısmı ve izleme kısmı oluşturulur. Şekil 2 'de 2 argüman ve birer de başlık, dönüş, yol ve izleme kısmı gösterilmiştir. Tüm kısımlar oluşturulduktan sonra tüm kayıt *vn_rdwr()* ile dosya sistemine yazılır.

4.2. Denetimle İlgili Sistem Çağruları

auditon() sistem çağrısı çekirdeğin devreye alma/devreden çıkarma gibi denetim yapılandırmasını ve global değerlerin ayarlanmasını kontrol eder. Ayrıca, denetim maskelerini

işler, olay-sınıf ilişkisini kurar ve denetim günlük dosyasının azami değerini düzenler. Bu sistem çağrısı ayrıca denetimde başarısız olduğunda gösterilecek davranışın ne olacağını belirler: kaydı sil, devam et veya panikle.

Maskeleri ve diğer süreç bilgilerini elde etmek, yeni değer atamak amacıyla pek çok sistem çağrısı yazıldı. *auditctl()* sistem çağrısı, denetim günlük dosyasının adını düzenlemek için kullanılırken *audit()* sistem çağrısı günlükteki denetim kayıtlarını teslim etmede kullanılır.

4.3. Çekirdek ile Kullanıcı Uzayı Arasındaki İletişim

Denetim servisinden çekirdeğe sistem çağrılılarıyla bağlantı kurulurken, çekirdekten denetim servisine */dev/audit* özel dosyası üzerinden erişilir. Bu özel dosya yazılabilir değildir ve sadece çekirdekten servise günlük dosyası çevrimi gibi olayları haber vermek için kullanılır. Bir uygulama, denetim servisine bir sinyal (örneğin; denetimi sonlandırma) göndermek istediğinde bu haber çekirdek üzerinden yönlendirilir: uygulama *auditon()* sistem çağrısını kullanılır, çekirdek de gerekli haberi gönderir.

Günlük dosyasına tek yazma hakkı çekirdekte olduğundan, çekirdek günlük dosyasını takip eder ve denetim servisini ne yapması gerektiği konusunda bilgilendirir. Günlük dosyasının durumuna göre denetim servisine gönderilebilecek haberler şunlardır:

- **OPEN NEW:** Mevcut denetim günlük dosyasının boyutu üst sınıra ulaştı demektir. Bu dosyanın boyutu parametrelerle yapılandırılabilir. Dosya boyutu üst sınırı *hafif* bir kısıttır ve denetim işlemine devam edilir.
- **LOW SPACE:** Günlük dosyasının üzerinde bulunduğu dosya sistemindeki boş alan azaldı demektir. En az boş alan miktarı da yapılandırılabilen bir değerdir. Bu kısıt da hafiftir ve denetim işlemi devam ettirilir.
- **NO SPACE:** Günlük dosyasının üzerinde bulunduğu dosya sistemindeki boş alan tükenmiştir. Bu *sert* bir limittir: ya sistem panikler ya da denetim servisi durdurulur.

5. Kullanıcı Uzayı Bileşenleri

Denetim ihtiyaçlarını tamamen karşılayabilmek için FreeBSD 'de kullanıcı uzayında çeşitli uygulamalar geliştirildi. Esas uygulama, denetim yapılandırmasını ve denetim takip dosyasını yönetmekle yükümlü olan denetim servisi. Denetimi seyrelten ve güvenlik bileşenleri olarak kullanılan çeşitli yönetim araçları mevcuttur.

5.1. *auditd* Servisi

Denetim servisi, *auditd*, sistem açılışının hemen başlarında kullanıcıların sisteme girişlerine öncülük etmek için çalışmaya başlar. *auditd* 'nin yapması gereken ilk iş denetim günlüklerini düzenlemektir. Öte yandan bu servisin yapması gereken denetimi başlatma/sonlandırma gibi görevleri de vardır. *auditd* servisi CAPP 'nin belirlediği kurallar çerçevesinde her

açılış/kapanış/günlük döndürme işlemlerinde denetim kayıtlarını sisteme teslim eder. Bölüm 4.3 'te bahsedilenlere ek olarak *auditd* servisine aşağıdaki haberler de verilir:

- **READ FILE:** *auditd* servisi yapılandırma dosyalarını okuyup gerekli değişiklikleri yapacaktır.
- **CLOSE AND DIE:** Denetim dosyasını kapat, denetimi devre dışı bırak ve sonlan.

5.1.1. Denetimin Başlaması ve Sonlanması

Denetimin başlatılması ve sonlandırılması *auditd* servisinin görevidir. Başlama süresinde çekirdekte olay-sınıf ilişkisi düzenlenir. Daha sonra *auditctl()* sistem çağrısıyla çekirdeğe denetim günlük dosyasının adı belirtilir. Bu çağrı aynı zamanda denetimin başlamasını da sağlar.

Kapanış esnasında *auditd* servisi çekirdeğe denetimin devre dışı bırakılmasını söyler ve mevcut günlük dosyasını o anki zaman ile isimlendirir.

5.1.2. Denetim Günlüğü Yönetimi

Günlük dosyası döndürüldüğünde yeni dosyanın ismi çekirdeğe haber verilir ve önceki dosya yeniden isimlendirilir. Dosya isimleri, denetimin başlangıç ve bitiş zamanını gösterir ve bu sayede denetim yapılan zaman aralıkları kolayca anlaşılır.

auditd servisi diskteki mevcut günlük dosyasının yönetiminden de sorumludur; ancak bu dosyanın takibi bölüm 4.3 'te anlatıldığı üzere çekirdek tarafından yapılır.

5.2. Denetim Kontrol Dosyaları

/etc/security dizininde bulunan birkaç dosya ile denetim sistemi yapılandırılır. Bu dosyalar denetim araçları ve BSM kütüphaneleri tarafından kullanılır. Çekirdek bu dosyaları doğrudan kullanmaz: *auditd* servisi *auditon()* sistem çağrısıyla çekirdeği mevcut yapılandırmadan haberdar edecektir. Yapılandırma dosyaları şunlardır:

- **audit_class:** Denetim olaylarının sınıflarını tanımlar.
- **audit_control:** Denetim günlüğü dizinleri, ön tanımlı denetim maskeleri ve günlüğün üzerinde bulunduğu dosya sistemindeki en az boş alanın eşik değeri burada belirtilir.
- **audit_event:** Denetim olayları ve sınıf atamaları.
- **audit_user:** Kullanıcılara özel denetim maskeleri.
- **audit_warn:** Denetim servisinden bir kullanıcı çıktığında çağrılan betiktir (script). Genellikle uyarıyı günlüğe ekler; ancak sistem yöneticisine e-posta gönderecek şekilde de yapılandırılabilir.

Denetim yapılandırma dosyaları son derece hassastır ve yalnızca sistem yöneticisi tarafından değiştirilebilir. *audit_user* ve *audit_control* dosyaları, CAPP kuralları gereği normal

kullanıcılar tarafından, hangi olayların denetlendiğinin anlaşılması için, okunamaz. Denetim yapılandırması hakkında yalnızca çekirdeğin ayrıcalıklı süreçleri bilgi alabilir.

5.3. Denetim İzleme Araçları

Denetim günlükleri yalnızca Darwin kaynak kodundan alınan araçlarla sorgulanabilir:

- **auditreduce**: Denetim günlüğünden, belirtilen kullanıcı/grup numarası, tarih, olay ve diğer kriterlere göre kayıt seçer.
- **praudit**: Denetim kayıtlarını insanların okuyabileceği biçimde ekrana basar.

Sistem yöneticileri bu araçlarla istedikleri kayıtları seçip normal metin biçiminde saklayabilir. Yalnız; şu an için OpenSolaris XML dosya biçimi desteklenmemektedir.

6. OpenBSM Projesi

OpenBSM, TrustedBSD projesinin denetim mekanizmasının kullanıcı uzayındaki bileşenlerinden oluşan bir pakettir:

- BSM çekirdeği ve kullanıcı uzayı başlık dosyaları
- BSM API 'nin bazı eklentilerle gerçekleştirilmiş hali – *libbsm*
- BSM dosya biçimi ve API 'sinin belgelendirilmesi
- Örnek */etc/security* yapılandırma dosyaları
- *praudit* ve *auditreduce* komut satırı araçları
- Test bileşenleri

OpenBSM, BSD lisansı altında FreeBSD ve Darwin dahil olmak üzere çok sayıda platformda çalışabilen bir denetim mekanizması gerçekleştirmeyi amaçlamaktadır. Apple BSM kodunda yapılan öncelikli değişimler çok sayıda gereksiz kodun atılması, kodun etkin kullanılması, denetim sisteminin en yüksek değerli sekizlinin (byte) nerde olduğundan bağımsızlaştırılması ve Solaris 'in son sürümlerindeki 64 bitlik BSM kayıtlarının desteklenmesidir.

7. Geleceğe Dair Planlar

Gelecekte geliştirilmesi beklenen çok sayıda denetim konusu vardır. Bunlardan biri ise MAC (Mandatory Access Control) sistemi ile denetim mekanizmasının bütünleştirilmesidir. Ayrıca bütünlüğün sağlanması adına tüm sistem çağrılarının denetim kapsamına alınması da yapılacak işler arasındadır.

Test ve performans analizlerinin yapılması gerekmektedir: Darwin kodundan bir test takımı FreeBSD 'de kullanılmak üzere alındı ve bu test araçları gelişmeye devam etmektedir.

Güvenlikle ilgili *ssh* ve diğer bazı uygulamalar hala denetim desteği gerektirmektedir. Denetim servisinin kendisi de hata toleransı, günlük döndürme ve aşırı uç olaylarda yapılacaklar konusunda test edilmelidir.

8. Sonuç

Bu makalede FreeBSD işletim sistemi için gerçekleştirilmiş denetim mekanizması, CAPP, güvenlik standartları, taşınabilir API 'ler ve kayıt biçimleri ele alındı. Denetim mekanizmasının koduna BSD lisansı altında TrustedBSD projesinin bir parçası olarak erişilebilir.

9. Teşekkürler

Yazarlar; Apple Computer 'dan Kevin Van Vechten ve Ron Dumont 'a OpenBSM projesini desteklediklerinden ve Apple BSM kodunu BSD lisansı ile açtıklarından ötürü teşekkür eder. Ayrıca TrustedBSD projesinden Tom Rhodes ve diğer geliştiricilere belgelendirme ve hata bildirimini konularındaki katkılarından ötürü teşekkür ederiz.

10. Kaynaklar

- [1] Apple Geliştiricileri Bağlantısı: Darwin, <http://developer.apple.com/darwin/>.
- [2] The Common Criteria Evaluation and Validation Scheme, <http://niap.nist.gov/cc-scheme/pp>.
- [3] M. K. McKusick ve G. V. Neville-Neil, The Design and Implementation of FreeBSD Operating System, Addison-Wesley, 2005.
- [4] FreeBSD, <http://www.freebsd.org>.
- [5] FreeBSD Sistem Günlüğü API 'si, <http://www.freebsd.org/cgi/man.cgi?query=syslog&format=html>.
- [6] Apple Mac OS X, <http://www.apple.com/macosx/>.
- [7] National Security Agency / Information Systems Security Organization, Controlled Access Protection Profile: http://niap.nist.gov/cc-scheme/pp/pp/PP_CAPP_V1.pdf.
- [8] OpenBSM, <http://www.openbsm.org/>.
- [9] OpenDarwin, <http://www.opendarwin.org/>.
- [10] OpenSolaris, <http://www.opensolaris.org/>.
- [11] Sun Microsystems, Inc., System Administration Guide: Security Services Part No: 816-4557, Sun Microsystems, 2005: <http://docs.sun.com/app/docs/prod/solaris.10>
- [12] TrustedBSD, <http://www.trustedbsd.org/>.