SİBER GÜVENLİK UZMANLIĞI EĞİTİMİ

Bir CVE Gerçekten Sistemi Etkiliyor mu?

Linux Kernel, Mitigation ve PoC Doğrulama

Örnek Ders Notu

Sürüm eşleştirmesinden kanıt üretimine; kernel yapılandırması, aktif korumalar, PoC statik incelemesi ve izole laboratuvar disiplini.

Eğitmen: Can Koç
Altın Kariyer Akademi

Dokümanın Amacı

Bu örnek ders notu, bir CVE numarası veya internette bulunan PoC kodu üzerinden hızlı hüküm vermek yerine, güvenlik iddiasını çalışan sistem üzerinde doğrulama disiplinini kurar.

> **Temel hüküm**
>
> Bir CVE numarası, eşleşen kernel sürümü veya çalışan bir PoC tek başına güvenlik açığı kanıtı değildir. Gerçek doğrulama; dağıtım yamalarını, paket revizyonunu, build yapılandırmasını, mimariyi, exploit ön koşullarını, aktif mitigation'ları ve gözlemlenebilir teknik etkiyi birlikte incelemeyi gerektirir.

Bu doküman saldırı reçetesi değildir. PoC kodu güvenilir yazılım kabul edilmez; yalnızca izole laboratuvarda, kaynak kodu incelendikten ve davranış yüzeyi sınırlandıktan sonra araştırma materyali olarak değerlendirilir.

İçindekiler

1. Sürüm eşleşmesi neden yeterli değildir?
2. Doğrulama zinciri: çalışan sistemden teknik etkiye
3. Kernel mitigation'ları nasıl okunur?
4. Crash, vulnerability, exploit ve yetki yükseltme ayrımı
5. PoC çalıştırmadan önce statik inceleme
6. İzole laboratuvar ve gözlem planı
7. Baseline, hash ve binary doğrulama
8. Şüpheli process inceleme şablonu
9. Kanıt matrisi ve teknik raporlama
10. Uygulamalı mini laboratuvar ve değerlendirme

1. Sürüm Eşleşmesi Neden Yeterli Değildir?

Bir güvenlik tarayıcısı veya CVE veri tabanı, çalışan kernel sürümünü bilinen bir zafiyet kaydıyla eşleştirebilir. Bu sonuç yalnızca başlangıç filtresidir. “Sürüm eşleşti, sistem açıktır” hükmü teknik olarak eksiktir.

```
uname -r

# Örnek çıktı
6.x.x-generic
```

Dağıtımlar upstream kernel sürümünü değiştirmeden güvenlik düzeltmelerini geri taşıyabilir. Buna backport denir. Dolayısıyla aynı görünen upstream sürüm numarası, farklı dağıtım paketlerinde farklı güvenlik durumları taşıyabilir.

Kontrol	Neden gereklidir?
Dağıtım security advisory	İlgili açığın dağıtım tarafından kapatılıp kapatılmadığını gösterir.
Paket revision değeri	Aynı upstream sürüm içindeki dağıtım yama seviyesini ayırır.
Kernel build config	Açığın bağlı olduğu özelliklerin derlenip derlenmediğini gösterir.
Mimari	PoC veya zafiyet yalnız belirli CPU mimarilerinde geçerli olabilir.
Exploit ön koşulları	Namespace, capability, local access veya belirli cihaz/sürücü gereksinimlerini ayırır.
Aktif mitigation'lar	Bug tetiklense bile exploit zincirinin güvenilir yetki üretmesini engelleyebilir.
Gerçek çalışan kernel	Diskteki paketle boot edilmiş kernel aynı olmayabilir.

Paket ve yapılandırma için örnek kontroller

```
apt policy linux-image-$(uname -r)
dpkg -l | grep linux-image

grep -E 'CONFIG_(KASLR|SECCOMP|USER_NS)' \
  "/boot/config-$(uname -r)"
```

Ana fikir

Sürüm eşleştirmesi, vulnerability doğrulaması değildir. Doğru soru “bu CVE bu sürümü etkiliyor mu?” değil, “bu çalışan sistemde gerekli kod yolu, yapılandırma ve ön koşullar gerçekten mevcut mu?” sorusudur.

2. Doğrulama Zinciri: Çalışan Sistemden Teknik Etkiye

Sağlam bir doğrulama, tek bir komut veya tarayıcı çıktısı yerine bir kanıt zinciri kurar. Zincirin her halkası bir önceki varsayımı test eder.

```
CVE kaydı
  -> etkilenen kod yolu
  -> dağıtım yaması / paket revizyonu
  -> kernel config ve mimari
  -> erişim ve exploit ön koşulları
  -> aktif mitigation'lar
  -> kontrollü tetikleme / gözlem
  -> teknik etki
  -> tekrar üretilebilir kanıt
```

Aşama	Sorulacak soru	Kanıt örneği
Kapsam	Hangi kod veya subsystem etkileniyor?	Upstream commit, advisory, kaynak dosya/fonksiyon
Varlık	Etkilenen özellik bu kernelde derlenmiş mi?	Kernel config, module listesi, device node
Erişim	Saldırgan bu kod yoluna ulaşabiliyor mu?	UID, capability, namespace, syscall veya cihaz erişimi
Koruma	Aktif mitigation zinciri neyi engelliyor?	KASLR, SMEP/SMAP, seccomp, lockdown
Etki	Crash dışında ne kontrol edilebiliyor?	State değişimi, yetki sınırı, kalıcı değişiklik
Kanıt	Sonuç tekrar üretilebilir mi?	Log, trace, dump, hash, zaman çizelgesi

Kanıt seviyesi

Otomatik araç uyarısı gözlemdir. Kernel panic teknik etkidir. Kontrollü kernel state veya yetki sınırı değişimi exploit kanıtına yaklaşır. Kalıcılık ise ayrıca boot, service, module, filesystem veya policy katmanında gösterilmelidir.

3. Kernel Mitigation'ları Nasıl Okunur?

Modern kernel exploitation yalnızca bug bulma işi değildir. Bug ile güvenilir kontrol arasında savunma katmanları bulunur. Aynı PoC bir sistemde çalışırken başka bir sistemde config, patch veya mitigation farkı nedeniyle başarısız olabilir.

Mitigation	Koruduğu alan	Analiz sorusu
KASLR	Kernel adres yerleşimini rastgeleleştirir.	PoC sabit adres veya offset varsayıyor mu?
SMEP	Kernel modunda user-space kod yürütülmesini sınırlar.	Zincir user pointer üzerinden kod yürütmeye dayanıyor mu?
SMAP	Kernelin user-space verisine doğrudan erişimini sınırlar.	Veri akışı access kontrolünü nasıl geçiyor?
NX	Veri sayfalarında kod yürütmeyi sınırlar.	Exploit yeni kod mu çalıştırıyor, mevcut kodu mu zincirliyor?
Stack canary	Stack taşmalarını erken saptamaya çalışır.	Hata canary kontrolünden önce kontrol üretiyor mu?
CFI	Dolaylı kontrol akışını sınırlar.	Hedef çağrı/return zinciri izinli tipte mi?

Mitigation	Koruduğu alan	Analiz sorusu
Hardened usercopy	Kernel-user kopyalarında sınır kontrolünü sertleştirir.	Kopyalama boyutu ve nesne sınırı uyumlu mu?
Seccomp	Process syscall yüzeyini daraltır.	Gerekli syscall hedef process için izinli mi?
Lockdown / module signature	Kernel bütünlüğü ve module yüklemeyi sınırlar.	PoC module, debug veya düşük seviyeli yazma gerektiriyor mu?

Durum kontrolleri

```
cat /proc/sys/kernel/randomize_va_space

grep -oE 'smep|smap' /proc/cpuinfo | sort -u

cat /sys/kernel/security/lockdown 2>/dev/null

grep '^Seccomp' /proc/$$/status
```

> **Dikkat**
>
> Mitigation listesi görmek, sistemin “güvenli” olduğunu kanıtlamaz. Aynı şekilde tek bir mitigation'ın kapalı olması da exploit'in çalışacağını kanıtlamaz. Etkilenen kod yolu ile savunma zinciri birlikte okunur.

4. Crash, Vulnerability, Exploit ve Yetki Yükseltme Ayrımı

Güvenlik analizinde en sık yapılan hata, gözlemlenen her çöküşü exploit kabul etmektir. Bu kavramlar aynı zincirin farklı seviyeleridir.

Seviye	Ne gösterir?	Ne göstermez?
Hata / bug	Beklenmeyen yazılım davranışı.	Güvenlik etkisini tek başına göstermez.
Crash	Kod yolunun kontrolsüz duruma girdiğini.	Kontrollü veri veya kontrol akışı elde edildiğini göstermez.
Vulnerability	Hatanın bir güven sınırını etkileyebildiğini.	Her ortamda sömürülebilir olduğunu göstermez.
Exploit primitive	Read/write, UAF, bilgi sızıntısı veya kontrol akışı gibi kullanılabilir yetenek.	Tam yetki yükseltme veya kalıcılık sağlamayabilir.
Privilege escalation	Daha yüksek bir UID, capability veya kernel state elde edildiğini.	Boot zinciri veya yeniden başlatma sonrası kalıcılığı göstermez.
Persistence	Değişikliğin yeniden başlatma veya servis yaşam döngüsü sonrasında sürdüğünü.	Elde edilen ilk erişimin nasıl oluştuğunu tek başına açıklamaz.

```
Crash != exploit
Exploit != privilege escalation
Root != kernel control
Kernel control != secure-world control
Runtime access != boot-chain persistence
```

Her aşamada yetkinin nerede kaldığı açıkça yazılmalıdır: kullanıcı process'i, ayrıcalıklı service, kernel, hypervisor, TEE veya boot katmanı.

5. PoC Çalıştırmadan Önce Statik İnceleme

İnternetten bulunan PoC doğrudan derlenmez. Önce okunur. Exploit kodu araştırma materyalidir; güvenilir yazılım değildir.

Kaynak kodunda aranacak davranışlar

```
grep -RInE \
  'system\(|execve\(|popen\(|socket\(|connect\(|curl|wget|chmod|chown|setuid|/tmp|/etc/passwd|/etc/shadow' \
  .
```

Yüzey	Kontrol soruları
Derleme	Hangi kütüphaneler bağlanıyor? Warnings kapatılmış mı? Mimari sabit mi? Hard-coded offset var mı?
Ağ	Harici IP veya domain var mı? Payload indiriyor mu? Reverse shell veya callback davranışı var mı?
Filesystem	/tmp altına ne bırakıyor? SUID dosya, cron, service veya policy değişikliği oluşturuyor mu?
Kernel	Module yüklüyor mu? Device node açıyor mu? ioctl/syscall yüzeyi ne?
Yetki	setuid, capability, namespace veya sudo varsayımı var mı?
Temizlik	Başarılı/başarısız çalışmada hangi dosya, process veya socket kalıyor?

Kontrollü gözlem

Kaynağı okumadan dinamik analiz yapılmaz. Dinamik analiz yapılacaksa ağ, shared folder, clipboard ve kalıcı disk yüzeyi sınırlandırılır; sistem snapshot veya disposable disk üzerinden geri döndürülebilir olmalıdır.

6. İzole Laboratuvar ve Gözlem Planı

Laboratuvar güvenliği yalnız "sanal makinede çalıştırdım" demek değildir. Host, hypervisor ve guest arasındaki tüm veri geçişleri düşünülür.

- VM network adapter kapalı veya yalnız kontrollü host-only segmentte olmalı.
- Snapshot alınmalı ve geri dönüş noktası doğrulanmalı.
- Shared folder, clipboard, drag-and-drop ve USB passthrough kapatılmalı.
- Disposable veya kopyalanmış disk kullanılmalı.
- Test öncesi process, socket, service, capability ve SUID baseline alınmalı.

- Çalışma boyunca terminal kaydı, strace ve sistem logları tutulmalı.
- Test sonrasında snapshot geri yüklenmeli; yalnız raporlanmış kanıtlar dışarı çıkarılmalı.

Örnek gözlem komutları

```
strace -f -o trace.log ./program

ps -eo pid,ppid,user,stat,comm,args --forest > processes.txt
ss -lntup > sockets.txt
systemctl --type=service --state=running > services.txt
getcap -r / 2>/dev/null > capabilities.txt
find / -xdev -perm -4000 -type f 2>/dev/null > suid.txt
```

Gözlem	Amaç
strace	System call, dosya, process ve ağ davranışını zaman sırasıyla görmek.
journal / dmesg	Kernel, service ve güvenlik subsystem mesajlarını toplamak.
ps / proc	Yeni process, PPID ve executable ilişkisini kurmak.
ss	Yeni listening veya outbound socketleri görmek.
hash / package verify	Binary veya paket bütünlüğündeki değişikliği doğrulamak.

7. Baseline, Hash ve Binary Doğrulama

Şüpheli davranışı anlayabilmek için önce normal durumu bilmek gerekir. Baseline karar vermez; karar verebilmek için bağlam sağlar.

```
diff -u old_processes.txt new_processes.txt
diff -u old_sockets.txt new_sockets.txt
diff -u old_services.txt new_services.txt
```

Bir farkın nedeni saldırgan aktivitesi olabileceği gibi paket güncellemesi, yeni kullanıcı uygulaması, service restart, kernel update, donanım değişimi veya normal geçici process de olabilir.

Binary doğrulama

```
sha256sum /usr/bin/ssh
dpkg -S /usr/bin/ssh
dpkg -V openssh-client
```

Bulgu	Tek başına ne anlama gelmez?	Ek kanıt
Hash değişti	Otomatik olarak malware değildir.	Paket metadata, imza, repository ve harici baseline
Yeni process görüldü	Otomatik olarak saldırı değildir.	PPID, executable, UID, start time, socket ve service ilişkisi
Yeni listening port	Arka kapı olduğu kesin değildir.	Process, paket sahipliği, config ve beklenen servis rolü
Kernel log hatası	Exploit gerçekleştiğini kanıtlamaz.	Tekrar üretim, stack trace, state değişimi ve privilege etkisi

Daha güçlü doğrulama

Güvenilir repository metadata, paket imzası, read-only ölçüm kaynağı, harici baseline ve birden fazla bağımsız kanıt birlikte kullanılır. Yerel sistem compromise olmuşsa yalnız yerel veriye güvenilmez.

8. Şüpheli Process İnceleme Şablonu

Bir process adı tek başına kimlik değildir. İsim, executable, UID, parent, namespace, capability, açık dosya ve socket bilgisi birlikte okunur.

```
PID=<PID>

ps -p "$PID" -o pid,ppid,user,euser,group,egroup,lstart,stat,comm,args
readlink -f "/proc/$PID/exe"
tr '\0' ' ' < "/proc/$PID/cmdline"; echo
cat "/proc/$PID/status"
ls -l "/proc/$PID/fd"
cat "/proc/$PID/maps"
ss -ntup | grep "pid=$PID,"
```

Alan	Analiz sorusu
PID / PPID	Process hangi yaşam zincirinden doğdu? Parent beklenen process mi?
UID / EUID / groups	Gerçek ve etkili yetki nedir? Supplementary group ayrıcalık sağlıyor mu?
exe / cmdline / comm	Görünen ad ile gerçek executable aynı mı? Binary silinmiş veya değişmiş mi?
fd / maps	Hangi dosya, socket, shared library ve memory mapping açık?
capabilities / seccomp	Process root olmadan hangi ayrıcalıkları taşıyor? Syscall yüzeyi sınırlandırılmış mı?
namespace / cgroup	Process hangi container veya servis sınırında? Host görünümüyle guest görünümü aynı mı?

9. Kanıt Matrisi ve Teknik Raporlama

Bir araç çıktısı tek başına güvenlik bulgusu kabul edilmez. Her bulgu sistem bağlamı, saldırı yüzeyi, ön koşul, privilege seviyesi, teknik etki, kanıt, tekrar üretilebilirlik ve mitigation bilgisiyle raporlanır.

Rapor alanı	Yazılması gereken
Sistem bağlamı	Dağıtım, kernel, mimari, paket revision, aktif config ve ilgili subsystem
Saldırı yüzeyi	Local/remote erişim, syscall, device, file, service veya IPC giriş noktası
Ön koşullar	UID, capability, namespace, belirli module veya config gereksinimleri
Teknik etki	Crash, information leak, read/write primitive, privilege boundary veya persistence
Kanıt	Log, trace, dump, hash, process/socket ilişkisi ve zaman çizelgesi
Tekrar üretilebilirlik	Adımlar, test sayısı, başarısızlık oranı ve çevresel koşullar

Rapor alanı	Yazılması gereken
Mitigation	Aktif korumalar, neden yeterli/yetersiz olduğu ve iyileştirme önerisi

Kısa rapor örneği

```
Bulgu: CVE eşleşmesi görüldü, ancak vulnerability doğrulanmadı.

Gerekçe:
- Dağıtım paketi ilgili düzeltmeyi backport etmiş.
- Etkilenen CONFIG seçeneği kapalı.
- PoC sabit offset ve kapalı bir user namespace varsayıyor.
- Kontrollü testte yalnız beklenen hata kodu üretildi; crash veya state değişimi yok.

Sonuç: Tarayıcı uyarısı false positive / bağlam dışı eşleşme olarak sınıflandırıldı.
```

10. Uygulamalı Mini Laboratuvar

Bu laboratuvar gerçek exploit çalıştırmaz. Amaç, güvenlik iddiasını doğrulamak için gerekli sistem bilgisini ve kanıt disiplinini kurmaktır.

Görev 1 - Çalışan kernel ve paket bağlamı

```
uname -a
uname -r
apt policy linux-image-$(uname -r)
cat /etc/os-release
```

Kaydedilecekler: çalışan kernel, dağıtım sürümü, paket revision ve boot edilen kernel ile kurulu paketlerin uyumu.

Görev 2 - Koruma yüzeyi

```
grep -E 'CONFIG_(RANDOMIZE_BASE|SECCOMP|USER_NS|MODULE_SIG|STRICT_DEVMEM)' \
  "/boot/config-$(uname -r)"

cat /proc/sys/kernel/randomize_va_space
grep '^Seccomp' /proc/$$/status
```

Kaydedilecekler: açık/kapalı config değerleri ve bunların olası PoC ön koşullarına etkisi.

Görev 3 - Baseline

```
mkdir -p baseline
ps -eo pid,ppid,user,stat,comm,args --forest > baseline/processes.txt
ss -lntup > baseline/sockets.txt
systemctl --type=service --state=running > baseline/services.txt
```

Görev 4 - Güvenli örnek program gözlemi

Yalnızca eğitim için hazırlanmış, zararsız bir program `strace -f` altında çalıştırılır. Programın açtığı dosyalar, oluşturduğu child process ve yaptığı system call'lar kaydedilir. İnternetten indirilen exploit kullanılmaz.

```
strace -f -o trace.log ./egitim-programi

# Sonrasında
grep -E "openat|execve|clone|connect|write" trace.log
```

Görev 5 - Kanıt tablosu

İddia	Kanıt	Hüküm
Kernel sürümü CVE ile eşleşiyor	uname + advisory + package revision	Başlangıç filtresi
Etkilenen özellik mevcut	CONFIG / module / device kontrolü	Kod yolu mümkün
PoC ön koşulu sağlanıyor	UID, namespace, capability, architecture	Test edilebilirlik
Teknik etki oluştu	Trace, log, crash dump veya state değişimi	Vulnerability kanıtı
Yetki sınırı aşıldı	UID/capability/kernel state karşılaştırması	Exploit etkisi

Sık Yapılan Hatalı Çıkarımlar

Hatalı ifade	Daha doğru yaklaşım
“Kernel sürümü eşleşti, sistem kesin açık.”	Dağıtım yaması, package revision, config ve ön koşullar doğrulanır.
“PoC crash üretti, exploit çalıştı.”	Crash ile kontrollü state veya privilege etkisi ayrılır.
“Root oldum, kernel tamamen kontrolümde.”	User-space root, kernel ve secure-world sınırları ayrı değerlendirilir.
“Hash değişti, malware var.”	Paket güncellemesi, yerel patch ve güvenilir harici baseline kontrol edilir.
“KASLR açık, sistem güvenli.”	Mitigation tek başına hüküm değildir; tam zincir ve bypass varsayımları incelenir.
“PoC GitHub’da, güvenilirdir.”	Kaynak, build sistemi, ağ ve filesystem davranışı statik olarak incelenir.

Mini Değerlendirme

- Upstream kernel sürümü neden tek başına vulnerability kanıtı değildir?
- Backport edilen güvenlik düzeltmesi hangi iki yerde doğrulanabilir?
- Bir PoC'nin doğru kaynak kod içermesi neden exploit zincirinin çalışacağını garanti etmez?
- Crash ile privilege escalation arasındaki teknik fark nedir?
- KASLR, SMEP, SMAP ve seccomp hangi farklı yüzeyleri sınırlar?
- PoC statik incelemesinde ağ ve filesystem davranışı neden kontrol edilir?
- Baseline farkı neden otomatik saldırı hükmü değildir?
- Bir binary hash değişikliğini doğrulamak için hangi bağımsız kanıtlar kullanılabilir?
- Şüpheli process için comm, cmdline ve exe neden birlikte okunur?
- Bir bulgunun tekrar üretilebilirliği raporda nasıl gösterilir?

Dersin ana fikri

Güvenlik analizi, araç çıktısını hükme çevirmek değildir. İddianın hangi sistemde, hangi kod yolunda, hangi ön koşullarla ve hangi teknik etkiyle gerçekleştiğini kanıtlamaktır. CVE numarası başlangıçtır; mühendislik, doğrulama zincirinde başlar.

Bu örnek doküman, Siber Güvenlik Uzmanlığı Eğitimi kapsamında kullanılan sistem seviyesi analiz yaklaşımını göstermek amacıyla hazırlanmıştır.