Tigress ve Çeşitlendirme

CEN429 Güvenli Programlama — Hafta 14

Dr. Öğr. Üyesi Uğur CORUH · 18.12.2026

Bugünün planı (3 saat)

Saat	Bölüm	Konu
1	0–1	Temel kavramlar · kaynaktan kaynağa gizleme · Tigress · lisans · temel akış
2	2	Dönüşüm aileleri · 9. hafta eşlemesi · dönüşüm hattı · adım adım örnek
3	3–4	Çeşitlendirme · ölçme · sınıf içi akış · derleme hattı (S15) · proje

Kısa tarihçe — kaynaktan kaynağa gizleme ve çeşitlendirme

- **1993** — Cohen: **çeşitlendirme** fikri (aynı işlev, farklı ikili)
- **1997** — Collberg vd. gizleme taksonomisi (9. haftanın temeli)
- **2013** — **Obfuscator-LLVM**: derleyici tabanlı gizleme
- **2010'lar** — **Tigress**: C için kaynaktan kaynağa + sanallaştırma + çeşitlendirme
- **2016–17** — Banescu vd. Tigress+KLEE ile **dayanıklılığı ölçer**

> Ana kural: **dayanıklılık ↔ maliyet**; koruma varlığın değeriyle orantılı seçilir.

Bu hafta nereye oturuyor?

- **9. hafta:** gizleme kurallarını **el ile** öğrendik
- **11. hafta:** anahtar için whitebox
- **Bu hafta (14):** aynı kuralları **araçla, otomatik** ve **çeşitlendirilmiş** uygulamak → Tigress

Üçü bir bütün; bugün otomasyon adımı.

Öğrenme çıktısı

Bu hafta **ÖÇ.3** (ikili uygulama korumaları) üstünedir.

Sonunda yapabileceğiniz:

- Kaynaktan kaynağa gizlemeyi açıklamak
- Bir **dönüşüm hattı** kurmayı tarif etmek
- Gizlemeyi **çeşitlendirip ölçmek**
- Gizlemeyi **derleme hattına** koymak (S15)

Baştan bir hatırlatma

9. haftanın ana kuralı bu hafta da geçerli:

> Gizleme kırılamazlık vermez, **maliyet** yükseltir.

Tigress bir **sihir değil**; el ile yaptığımızı otomatikleştirir ve **çeşitlendirmeyi** kolaylaştırır.

0. Temel kavramlar (sıfırdan)

Neden bu bölüm?

Bugün "kaynaktan kaynağa", "dönüşüm", "tohum", "derleme hattı" gibi terimler geçecek.

Önce hepsini **tek tek** tanımlayalım.

Hatırlatma · kaynak, derleyici, ikili

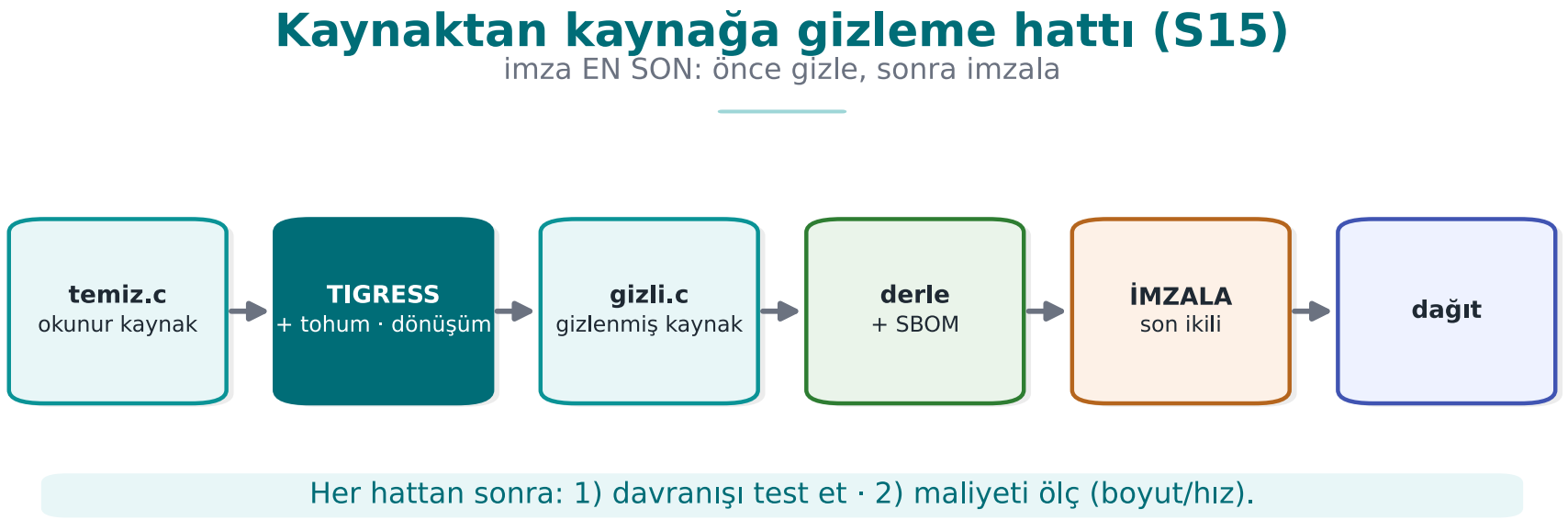
- **Kaynak kod:** insanın yazdığı program metni (C dosyası).
- **Derleyici:** kaynağı makine koduna çeviren program (gcc/clang).
- **İkili dosya:** çalıştırılan sonuç.

Hatırlatma · gizleme

- **Kod gizleme (obfuscation):** davranışı değiştirmeden kodu **anlaşılması zor** hale getirmek.
- Amaç: saldırgan için maliyeti artırmak (9. hafta).

Kaynaktan kaynağa (source-to-source)

- Girdi: C kaynağı. Çıktı: **yine C kaynağı** — ama gizlenmiş.
- Sonra normal derleyicinizle derlenir.



Her hattan sonra: 1) davranışı test et · 2) maliyeti ölç (boyut/hız).

Dönüşüm (transform) nedir?

- **Dönüşüm:** kaynağa uygulanan tek bir gizleme işlemi (ör. düzleştirme).
- Araç birçok dönüşüm sunar; siz hangisini, nereye seçersiniz.

Dönüşüm hattı (pipeline)

- **Hat:** birden çok dönüşümün **sırayla** uygulanması.
- Her dönüşüm bir öncekinin çıktısına uygulanır.
- Sıra önemlidir.

Tohum (seed) nedir?

- **Tohum:** rastgeleliği yöneten bir başlangıç sayısı.
- Aynı dönüşüm + farklı tohum = **farklı** gizlenmiş çıktı.
- Çeşitlendirmenin anahtarı budur.

Çeşitlendirme (diversification)

- **Çeşitlendirme:** aynı kaynaktan **davranışça eş, yapıca farklı** ikili dosyalar üretmek.
- Bir kopyaya yazılan saldırı diğerinde çalışmaz (9. hafta Kural 2).

CLI (komut satırı) nedir?

- **CLI (Command-Line Interface):** komutları yazarak çalıştırdığınız arayüz.
- Tigress bir CLI aracıdır: `tigress --Transform=... dosya.c`.

Birim testi (unit test)

- **Birim testi:** bir fonksiyonun doğru çalıştığını otomatik denetleyen küçük test.
- Gizlemeden **sonra** aynı testler geçmeli (davranış korunmalı).

CFG hatırlatma

- **CFG (Control Flow Graph):** temel blokları düğüm, geçişleri kenar yapan şema.
- Gizlemenin **gücünü** düğüm/kenar sayısıyla ölçeriz (9. hafta).

Sembolik yürütme (kısaca)

- **Sembolik yürütme:** program yollarını matematiksel kısıt olarak çözen otomatik analiz (ör. KLEE).
- Gizlemeye karşı bir **deobfuscation** yöntemidir; dayanıklılığı bununla sınarız.

Şimdi hazırız

Terimler:

kaynaktan kaynağa · dönüşüm · hat · tohum · çeşitlendirme · CLI · birim testi · CFG · sembolik yürütme

Şimdi: Tigress nedir ve nasıl çalışır?

1. Kaynaktan kaynağa gizleme ve Tigress

El ile gizlemenin üç sorunu

9. haftada kuralları **el ile** uyguladık. Öğreticiydi ama:
10. **Hataya açık** — davranışı bozabilir
11. **Bakımı zor** — kaynak okunamaz olur
12. **Çeşitlendirilemez** — her kopyayı elle farklılaştıramazsınız

Çözüm: araca yaptır

- Gizlemeyi bir **araç** yapsın.
- Siz **okunur** kaynağı koruyun.
- Gizleme, derleme hattında **otomatik** bir adım olsun.

Kaynaktan kaynağa fikri

Bakım maliyeti **okunur kaynakta** kalır; dağıtılan kaynak gizli.

Dönüşüm hattı — şema

Tigress nedir?

- Arizona Üniversitesi'nden Collberg ve ekibinin geliştirdiği **kaynaktan kaynağa C gizleyici/çeşitlendirici**.
- ix. haftadaki el ile kuralların **otomatik** karşılığı.

Tigress · özellikler

- Çok platform: Linux, macOS, Windows, Android
- Mimariler: Intel, ARM, WebAssembly
- Derleyiciler: GCC, Clang, MSVC
- Güncel sürüm: **v4**

⚠️ Lisans (önemli)

- **Kâr amacı gütmeyen** (akademik/araştırma) kullanım: **ücretsiz**.
- **Ticari** kullanım: Arizona Üniversitesi'nden **lisans** gerekir.
- Kaynak kodu açık değil; araştırmacılar şifreli kaynağa erişim isteyebilir.

⚠️ Derste kullanım kuralı

- Derste **Tigress ikilisi ya da eski sürüm dağıtılmaz.**
- Öğrenciler güncel sürümü **resmî siteden** (`tigress.wtf`) kendileri indirir.
- Lisans koşulunu **orada** doğrular.
- Yalnız **kendi** kodunuza, kendi makinenizde uygularsınız.

Neden araç, el ile değil? (1)

El ile (9. hafta)	Araçla (Tigress)
Öğreticidir	Ölçeklenir
Hataya açık	Davranışı korur (test edilmiş)
Kaynak okunamaz	Okunur kaynak sizde

Neden araç, el ile değil? (2)

El ile	Araçla
Çeşitlendirme elle olmaz	Tohumla otomatik
Tek tip, tanınabilir	Dönüşüm+tohum ile değişken

Ama araç da sihir değil

- Tigress de kırılamazlık **vermez**, maliyet yükseltir.
- İyi bilinen bir aracın kalıpları zamanla tanınabilir.
- Bu yüzden **çeşitlendirme** ve **katmanlı savunma** (RASP, sunucu) yine şart.

Temel akış · tek dönüşüm

```
# girdi: temiz.c → çıktı: gizli.c
tigress --Transform=Flatten --Functions=erisim_ver \
        --out=gizli.c temiz.c
cc -o program gizli.c
```

Temel akış · üç fikir

1. `--Transform=...` hangi dönüşüm (Flatten = düzleştirme, 9. hafta K-04)
2. `--Functions=...` hangi fonksiyonlara (yalnız hassas olanlara — maliyet kuralı)
3. Çıktı yine C; **kendi derleyicinizle** derlersiniz

Neden `--Functions` ile hedef seçiyoruz?

- Gizleme her fonksiyona uygulanırsa program **çok yavaşlar/şişer**.
- Yalnız hassas fonksiyonları (lisans, anahtar türetme, bütünlük) hedefleriz.
- Bu, 9. haftadaki "maliyet kuralının" araçtaki hâli.

Bölüm 1 — kısa sınama

1. Kaynaktan kaynağa gizleme nedir?
2. El ile gizlemeye göre üç avantaj?
3. Tigress derste neden dağıtılmaz?

Bölüm 1 — cevaplar

1. Kaynak koda dönüşüm uygulanıp **yeni kaynak (yine C)** üretilir, sonra normal derlenir. Tigress C alır, gizlenmiş C verir.
2. **Tekrarlanabilir/otomatik** (her build), **tutarlı ve ölçekli** (çok fonksiyon), **orijinal kaynak temiz kalır** + tohumla çeşitlendirme.
3. **Lisans/kullanım koşulları**; herkes kendi indirir. Demolar Tigress yoksa **temiz türev/fallback** ile çalışır.

2. Dönüşüm aileleri ve hat

Dönüşümler = 9. hafta kuralları

Tigress dönüşümleri, 9. haftadaki gizleme ailelerine karşılık gelir.

Grup grup görelim; her birini bir K-kuralına bağlayacağız.

Dönüşüm ↔ kural eşlemesi — şema

Tigress dönüşümleri ↔ 9. hafta kuralları

araç yeni bir şey icat etmiyor — aynı kuralları otomatikleştiriyor

Flatten	K-04 kontrol akışı düzleştirme
AddOpaque / InitOpaque	K-01 opak yüklem · K-03 ölü dal
EncodeArithmetic	K-02 aritmetik kodlama
EncodeLiterals	K-07 sabit / dize gizleme
Virtualize	K-10 sanallaştırma (en pahalı)

Hangi dönüşümü seçtiğinizi S9'da kuralla eşleyerek gerekçelendirin.

Kontrol akışı dönüşümleri

- **Flatten:** kontrol akışını düzleştirir → **K-04**
- **InitOpaque / AddOpaque / UpdateOpaque:** opak yüklem + sahte dal → **K-01, K-03**

Veri dönüşümleri

- **EncodeArithmetic:** aritmetiği denk karmaşık ifadeyle (MBA) → **K-02**
- **EncodeLiterals:** sabit ve dizgeleri kodlar → **K-07, K-08**
- **EncodeData:** değişken gösterimini kodlar → **K-09**

Fonksiyon dönüşümleri

- **Split / Merge:** fonksiyonları böler/birleştirir → **K-06**
- **Virtualize:** fonksiyonu özel VM bayt koduna → **K-10**
- **Jit:** kodu çalışma anında üretir → **K-12 (dinamik)**

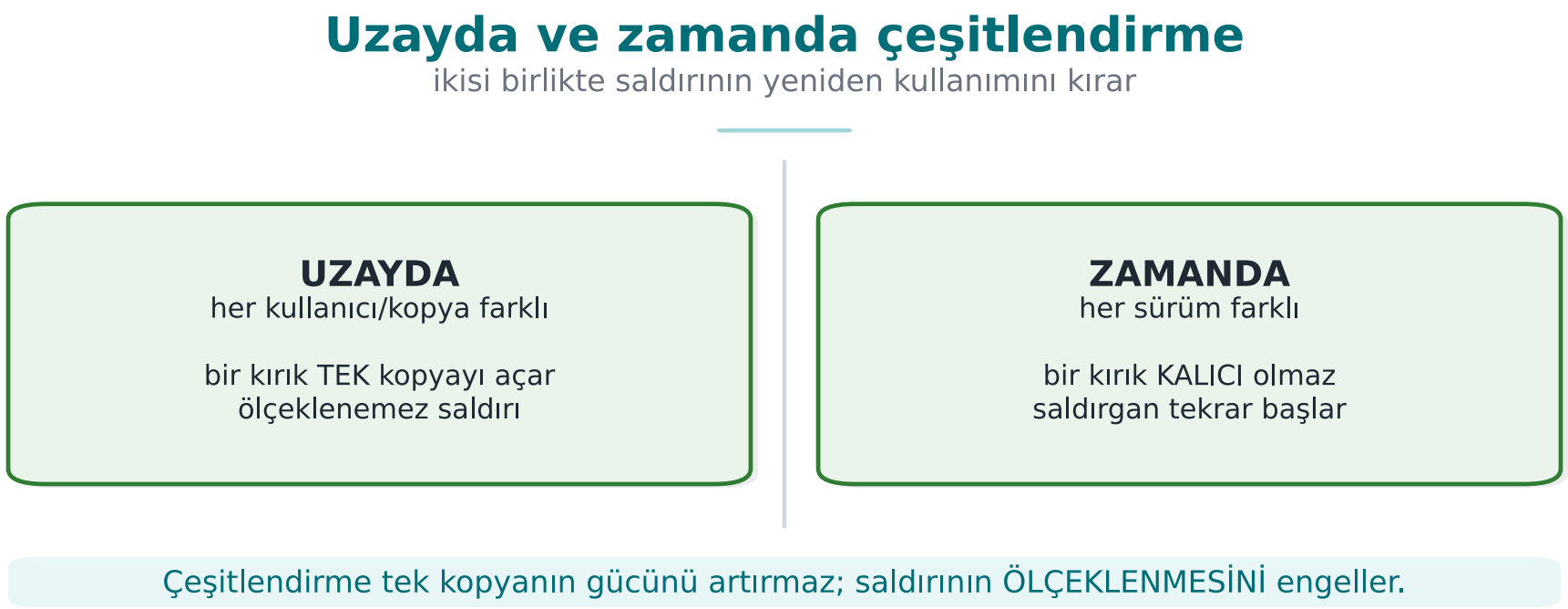
Analiz-önleme dönüşümleri

- **AntiBranchAnalysis / AntiAliasAnalysis / AntiTaintAnalysis:** statik analiz tekniklerini zorlaştırır → önleyici aile

Çeşitlendirme dönüşümleri

- **RandomFuns:** rastgele sahte fonksiyonlar
- **RndArgs:** sahte parametreler
- Tohumla birleşince her yapı farklı → çeşitlendirme, **K-06**

Uzayda/zamanda çeşitlendirme — şema



Dönüşüm ↔ kural · tablo

Tigress	Ne yapar	9. hafta
Flatten	Düzleştirme	K-04
AddOpaque	Opak yüklem/sahte	K-01, K-03
EncodeArithmetic	Aritmetik kodlama	K-02
EncodeLiterals	Sabit/dize	K-07, K-08
Virtualize	Sanallaştırma	K-10
RandomFuns/RndArgs	Çeşitlendirme	K-06

Aileler ve maliyet

- **Ucuz:** EncodeArithmetic, EncodeLiterals
- **Orta:** Flatten, opak yüklemler
- **Pahalı:** Virtualize (onlarca kat yavaşlama)

Bu yüzden Virtualize yalnız **küçük, kritik** fonksiyonlara; `--Functions` ile hedeflenir.

Dönüşüm hattı

"Tek teknik değil, birlikte"

9. haftanın en önemli kuralı.

Tigress'te bu, dönüşümleri **sırayla** uygulamak demektir.

Kavramsal hat

```
tigress \
  --Transform=EncodeLiterals   --Functions=erisim_ver \
  --Transform=EncodeArithmetic --Functions=erisim_ver \
  --Transform=Flatten          --Functions=erisim_ver \
  --Transform=AddOpaque        --Functions=erisim_ver \
  --out=gizli.c temiz.c
```

Hat ne yapıyor?

Tek denetimi (`erisim_ver`):

1. Sabit/dize kodla
2. Aritmetiğini kodla
3. Düzleştir
4. Opak yüklemlerle güçlendir

= 9. haftada K-04'ün "güçlendirilmiş düzleştirme" dediği şeyin otomatik hâli.

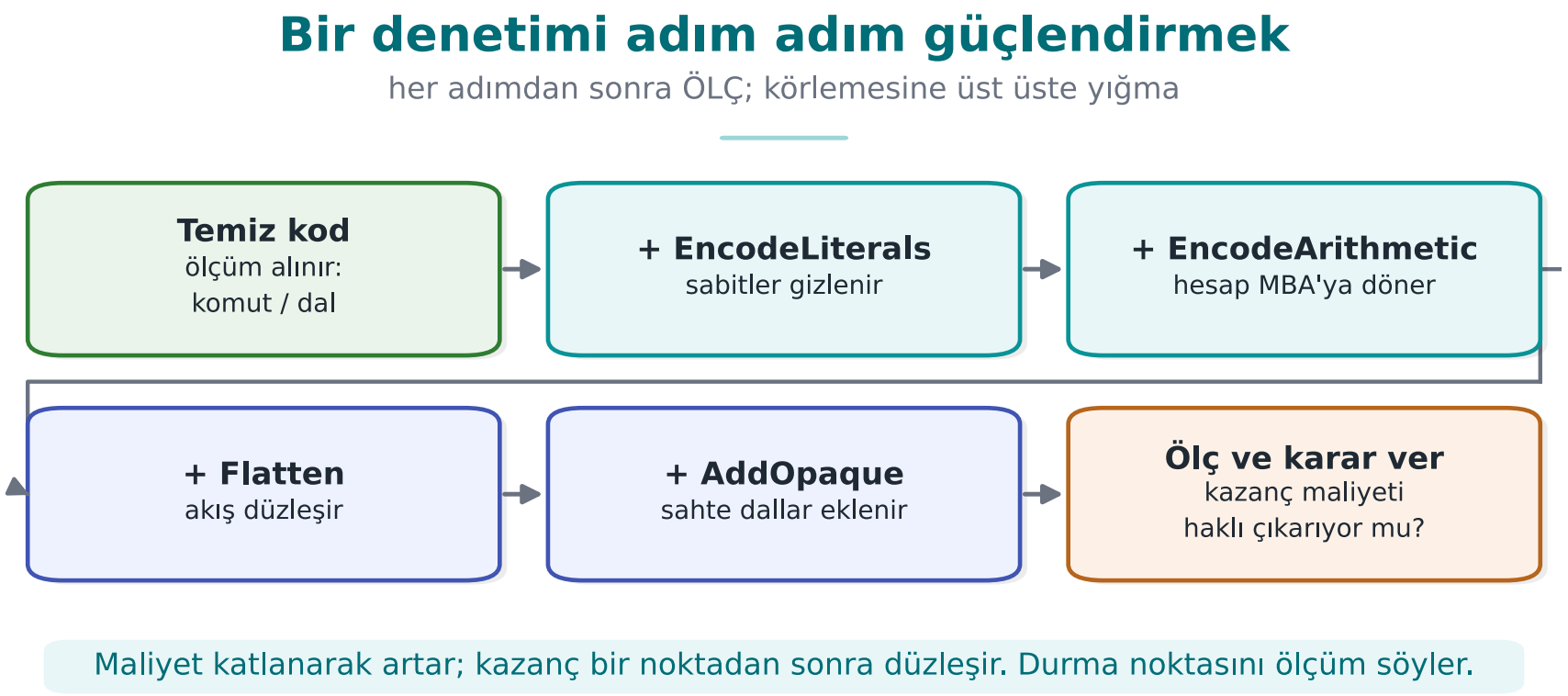
⚠️ Sıra ve test kuralı

- Dönüşüm sırası sonucu **etkiler**; kimi sıra başarımı gereksiz bozar.
- **Kural:** her hattan sonra **birim testlerini** çalıştır (davranış korunmuş mu?) ve boyut/hız ölç.
- Gizleme davranışı **asla** değiştirmemeli; değiştiriyorsa hat yanlış.

Bu, 12. haftadaki "S16 sonuçları"na bağlanır.

Adım adım örnek

Adım adım güçlendirme — şema



Maliyet katlanarak artar; kazanç bir noktadan sonra düzleşir. Durma noktasını ölçüm söyler.

Başlangıç · temiz.c

```c
int erisim_ver(const char *jeton) {
    if (jeton_gecerli(jeton)) return IZIN;  /* tek dal */
    return RED;
}
```

Korumasız: tek dal, `strings`, tek bayt yamasıyla atlanır.

Adım 0 · saldırgan

- `strings` → `IZIN` / `RED`
- Tersine derle → tek `if`
- Başarısızlık dalını başarıya çevir

Süre: dakikalar.

Katman katman

Adım	Dönüşüm	Ne değişir	Maliyet
0	—	Tek bayt yamasıyla atlanır	—
1	EncodeLiterals	IZIN/RED düz görünmez	Çok düşük
2	EncodeArithmetic	Karşılaştırma karmaşık	Düşük
3	Flatten	Tek dal görünmez	Orta
4	AddOpaque	Sahte dallar	Orta
5	Virtualize (gerekirse)	VM bayt kodu	Yüksek

Çıkarım

- Her adım saldırıyı biraz daha **pahalı** yapar.
- Her adım bir **maliyet** ekler.
- Nerede duracağınız: **varlık değeri** + **ölçtüğünüz maliyet**.

Adım 5 (Virtualize) çoğu denetim için gereksiz.

Bu tablo → S9'un ta kendisi

Bu "adım / dönüşüm / ne değişir / maliyet" tablosu,

projede **S9 koruma tablonuzun** doğrudan karşılığıdır.

Bölüm 2 — kısa sınama

1. Flatten, EncodeArithmetic, Virtualize hangi K-kurallarına karşılık?
2. Neden `--Functions` ile hedef seçilir?
3. Her hattan sonra **mutlaka** yapılacak iki şey?

Bölüm 2 — cevaplar

1. **Flatten** = kontrol akışı düzleştirme; **EncodeArithmetic** = aritmetik/veri gizleme; **Virtualize** = sanallaştırma (en güçlü, en pahalı).
2. Her fonksiyonu gizlemek maliyeti/performansı **patlatır**; yalnız **hassas** fonksiyonları seçmek dayanıklılığı oraya yoğunlaştırır.
3. (1) **Davranışı test et** (birim testi geçmeli), (2) **maliyeti ölç** (boyut/hız/komut sayısı, önce-sonra).

3. Çeşitlendirme ve ölçme

Kural 2'yi hatırla

> Bir kopyada işe yarayan saldırı, bütün kopyalarda işe yaramasın.

Saldırgan bir kopyayı kırıp yamayı **herkese** dağıtabiliyorsa, bir kırık her yeri açar.

Çeşitlendirme nedir?

Aynı kaynaktan **davranışça eş, yapıca farklı** ikili dosyalar.

Tigress bunu bir **tohum** ile yapar:

- Aynı dönüşümler + farklı tohum → opak yüklem, sahte dal, durum değerleri **değişir**.

İki tohum, iki farklı ikili

```
tigress --Seed=1001 --Transform=Flatten --Transform=AddOpaque \
        --Functions=erisim_ver --out=gizli_a.c temiz.c
tigress --Seed=2002 --Transform=Flatten --Transform=AddOpaque \
        --Functions=erisim_ver --out=gizli_b.c temiz.c
```

`gizli_a` ve `gizli_b` aynı işi yapar; makine kodları **farklı**.

İki tür çeşitlendirme

- **Uzayda:** farklı kullanıcı/cihaza farklı tohumlu kopya → bir kopyanın saldırısı diğerinde çalışmaz.
- **Zamanda:** her sürüm yeni tohum → eski saldırı yeni sürümde bozulur.

10. haftadaki anahtar/sürüm yenilemeyle birlikte çalışır.

Çeşitlendirme neyi engeller?

- Gizlemenin **gücünü** artırmaz (bir kopyayı kırmak yine mümkün).
- Ama **ölçeklenmesini** engeller.

`RandomFuns`, `RndArgs` de tohumla birleşince yapıyı değiştirir.

Çeşitlendirme etkinliği · ölçüt

- Aynı kaynağı iki tohumla derle.
- Korunan fonksiyonların **bayt farkını** ölç.
- Fark ne kadar yüksekse, bir saldırının diğerinde çalışma olasılığı o kadar düşük.

Ölçütü S9'a yaz.

Gizlemeyi ölçmek

Tigress'in öğretim değeri

9. haftada gizlemeyi dört boyutta (güç, dayanıklılık, gizlilik, maliyet) ölçmeyi öğrendik.

Tigress bu ölçümleri **somut** yapmanızı sağlar: aynı programı gizle, önce/sonra ölç.

Maliyet ölçümü (kolay, zorunlu)

Her hat için ölç ve S9/S15'e yaz:

Ölçüt	Nasıl
İkili boyut	`size` / `ls -l`
Çalışma süresi	Aynı girdiyle N kez, ortalama
CFG karmaşıklığı	Tersine derleyicide temel blok sayısı

Maliyet ölçümü · komut

```
size ./program_temiz ./program_gizli   # boyut farkı
# süre: aynı girdiyle N kez çalıştır, ortalamayı karşılaştır
```

Basit ama **kanıt** üretir.

Maliyet · örnek tablo

Ölçüt	Önce	Sonra
Boyut	100 KB	130 KB
Süre	1.0×	1.8×
Temel blok (hedef fn.)	5	40

Rakamlar örnek; siz **kendi** ölçtüğünüzü yazarsınız.

Dayanıklılık ölçümü (ileri, kavram)

- Dayanıklılık = otomatik araca (sembolik yürütme) direnç.
- ix. hafta: "iddia değil, ölçülen bir nicelik".
- Tigress bu araştırmanın **merkezinde**.

Banescu vd. · Tigress + KLEE

Tigress dönüşümlerinin **sembolik yürütmeye** ne kadar dayandığını ölçen çalışma. Bulgular:

- Tek dönüşüm (yalnız Flatten) → çoğu zaman geri açılır.
- Dönüşümleri **birleştirmek** + durum uzayını büyütmek → çözücüyü üstel zorlar.
- Ama **maliyet** de artar.

Dayanıklılık ↔ maliyet — şema

Ders için çıkarım

> Dayanıklılık ve maliyet **ödünleşir** (9. haftadaki gibi).

"Güçlü" deme; **"şu araca karşı şu boyutta problemi şu sürede çözdürmedi"** de.

Ölçüm kuralı (S9/S15)

Bir gizleme kararını ölçüyle savun:

> "Flatten + AddOpaque hattı boyutu %30 büyüttü, işlemi 1.8× yavaşlattı; hedef fonksiyonun temel blok sayısı 8 katına çıktı."

Ölçülmemiş gizleme = değerlendiricide (12. hafta) **iddia**.

Bölüm 3 — kısa sınama

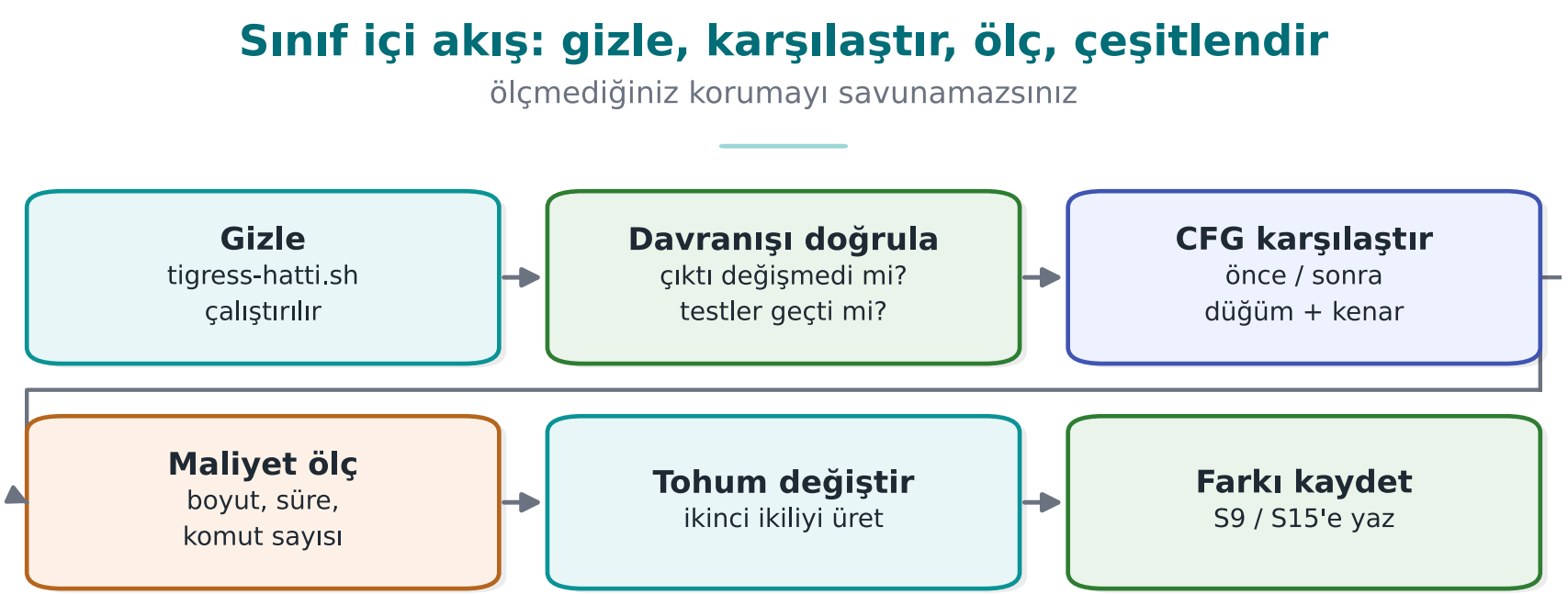
1. Tohum çeşitlendirmeyi nasıl sağlar?
2. Uzayda ve zamanda çeşitlendirme farkı?
3. Banescu vd.'nin ana kuralı? (dayanıklılık ↔ maliyet)

Bölüm 3 — cevaplar

1. Aynı kaynağı farklı **tohum**la gizleyince araç farklı rastgele seçimler yapar → her build **farklı ikili** (aynı davranış).
2. **Uzayda:** farklı kopyalar/kullanıcılar farklı (bir kırık herkesi kırmaz). **Zamanda:** sürümler arası değişir (kırık kalıcı olmaz).
3. Daha güçlü gizleme daha çok **dayanıklılık** ama daha çok **maliyet**; koruma varlığın **değeriyle orantılı** seçilir.

4. Sınıf içi akış (adım adım)

Sınıf içi akış — şema



Davranış doğrulaması atlanırsa gizleme sessizce programı bozar — en sık yapılan hata budur.

Ne yapacağız?

Kendi küçük, sentetik programınızı:

gizle → davranışı doğrula → karşılaştır → ölç → çeşitlendir → raporla

Adım adım gidelim.

Adım 1 · Hazırlık

- Küçük, sentetik bir program yaz (ör. `erisim_ver`).
- Birim testleriyle **doğru çalıştığını** göster.
- Bu senin "temiz" temel çizgin.

Adım 2 · Gizle

```
tigress --Transform=EncodeLiterals --Transform=EncodeArithmetic \
        --Transform=Flatten --Transform=AddOpaque \
        --Functions=erisim_ver --out=gizli.c temiz.c
cc -o program_gizli gizli.c
```

Bir dönüşüm hattı uygula.

Adım 3 · Davranışı doğrula

- Aynı birim testlerini **gizli** sürümde çalıştır.
- Sonuç **birebir aynı** olmalı.
- Farklıysa: hat yanlış → düzelt.

> Gizleme davranışı asla değiştirmez.

Adım 4 · Karşılaştır

- Tersine derleyicide (Ghidra/objdump) temiz ve gizli sürümü aç.
- Kontrol akışını karşılaştır.
- Temel blok sayısındaki **artışı** not et.

Adım 5 · Ölç

```
size ./program_temiz ./program_gizli
# süre: aynı girdiyle N kez çalıştır, ortalamayı karşılaştır
```

Boyut ve süre farkını yaz.

Adım 6 · Çeşitlendir

- Aynı hattı **iki farklı tohumla** çalıştır.
- İki ikilinin farklı olduğunu göster (bayt farkı).

```
tigress --Seed=1001 ... --out=gizli_a.c temiz.c
tigress --Seed=2002 ... --out=gizli_b.c temiz.c
```

Adım 7 · Raporla

Bulguları bir tabloya dök:

Dönüşüm	Boyut	Süre	Blok	Tohum farkı
…	…	…	…	…

Bu tablo doğrudan **S9/S15**'e girer.

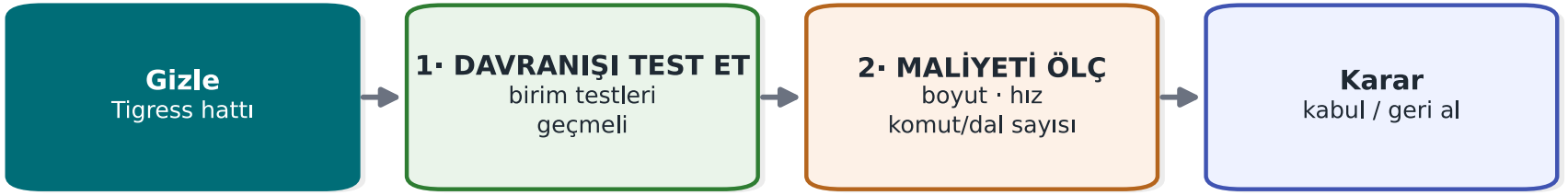
⚠️ Etik ve güvenli kullanım

- Tigress yalnız **kendi** kodunuza.
- Örnek program bilgisayara **zarar vermez**: sistem ayarı değiştirmez, ağa dokunmaz.
- Bütün değerler **sentetik**.

Sınıf içi · özet akış

Her hattan sonra mutlaka iki şey

ölçmeden 'güçlendirdim' denemez



Örnek ölçüm: erisim_ver 28 komut/3 dal → 51 komut/6 dal.

Adım adım akış · neden bu sıra?

- Önce **davranış** (testler) — bozulmadığından emin ol.
- Sonra **kanıt** (CFG, ölçüm) — kazancı göster.
- Sonra **çeşitlendirme** — ölçeklenmeyi kır.

Ölçmeden "güçlü" deme.

Bölüm 4 — kısa sınama

1. Gizledikten sonra ilk **mutlaka** yapılacak şey?
2. Çeşitlendirmeyi nasıl gösterirsin?
3. Rapor tablosu hangi proje bölümlerine gider?

Bölüm 4 — cevaplar

1. Davranışın **değişmediğini test et** (birim testi geçmeli); gizleme işlevi bozmamalı.
2. İki farklı tohumla üret; ikilileri karşılaştır (**hash/boyut/komut farklı**) ama **aynı girdi→aynı çıktı** (diff/objdump).
3. **S9** (kod sağlamlaştırma) ve **S15** (derleme/dağıtım hattı); önce/sonra ölçüm kanıt olarak.

5. Derleme hattı (S15) ve proje

Gizleme "en sonda elle" değil

Gizleme, derleme **hattının** bir adımı olmalı.

Vize sonrası projenizin **S15** bölümü tam bunu belgeler.

Derleme hattı · şema

Sıra bozulursa imza geçersizleşir; TOE kimliği (12. hafta) buradan çıkar.

Hat kuralı 1 · yalnız hassas fonksiyonlar

- `--Functions` ile yalnız hassas fonksiyonları gizle.
- Gerekçesini yaz (hangi fonksiyon, neden).
- Maliyet kuralı (9. hafta).

Hat kuralı 2 · her sürüm çeşitlendirilir

- Her sürüm bir **tohumla** çeşitlendirilir.
- Tohum ve sürüm kimliği **kaydedilir**.
- Zamanda çeşitlendirme (bölüm 3).

Hat kuralı 3 · imzalama gizlemeden sonra

- Önce gizle, **sonra** imzala.
- Sürüm kimliği ve özet değerleri (12. haftadaki TOE) **tutarlı** olsun.
- Neden sonra? İmza, dağıtılan **son** ikiliyi kapsamalı.

Hat kuralı 4 · CI'da otomatik

- Hat, sürekli entegrasyonda (CI) çalışır.
- Her sürümde: birim testleri + boyut/hız ölçümü **otomatik**.
- Davranış bozulursa yapı **durur**.

9, 11, 14 birlikte

- **9:** gizleme kuralları (kod)
- **11:** whitebox (anahtar)
- **14:** otomasyon + çeşitlendirme (bu hafta)

Ortak kural: koruma = **gecikme**; güç katmanlardan ve ölçümden.

Proje · S9 ileri + S15 hat — 1

1. Dönüşüm hattı (S9): hangi fonksiyonlar, hangi dönüşümler, hangi sırayla, **neden**.

Proje · S9 + S15 — 2

2. Ölçüm tablosu: boyut, süre, (varsa) blok sayısı — önce/sonra.

Proje · S9 + S15 — 3

3. Çeşitlendirme: iki tohumla üretim + fark ölçütü; uygulamayacaksanız **gerekçe**.

Proje · S9 + S15 — 4, 5

4. S15 hattı: gizleme + imzalama + sürüm kimliği adımlarını **diyagramla** belgele.

5. Davranış kanıtı: gizli sürümün birim testlerini geçtiğini göster (S16 bağı).

Değerlendirici gözüyle

- "Çok dönüşüm kullandım" **değil**.
- **Her dönüşümün gerekçesi ve ölçüsü**.
- Ölçülmemiş gizleme = iddia (12. hafta).

Bölüm 5 — kısa sınama

1. İmzalama neden gizlemeden **sonra**?
2. "Yalnız hassas fonksiyonlar" kuralının gerekçesi?
3. S15 hattının adımları?

Bölüm 5 — cevaplar

1. İmza **son ikiliyi** korur; önce imzalayıp sonra gizlersen ikili değişir → imza geçersiz. Sıra: derle → gizle → **imzala**.
2. Bütünü gizlemek performans/boyutu patlatır ve çoğu kod hassas değil; korumayı **kritik** fonksiyonlara yoğunlaştır → maliyet düşük, etki yüksek.
3. Kaynak → (statik analiz) → derle → **gizle** → çeşitlendir (tohum) → paketle/SBOM → **imzala** → dağıt (hepsi CI'da kayıtlı).

Çözümlü kendini sınama

Soru 1

Kaynaktan kaynağa gizleme nedir? El ile gizlemeye göre üç avantajı?

Cevap: Araç C kaynağı alıp gizli C kaynağı üretir. Avantajlar: davranışı korur (hataya kapalı), okunur kaynak sizde kalır, tohumla çeşitlendirme kolay.

Soru 2

Tigress'in dersteki yeri; hangi lisans, neden derste dağıtılmaz?

Cevap: 9. hafta kurallarının otomatik hâli. Kâr amacı gütmeyen kullanım ücretsiz, ticari için Arizona lisansı. Kaynak kapalı → ikilisi derste dağıtılmaz; öğrenci resmî siteden indirir.

Soru 3

Şu dönüşümleri 9. hafta kurallarıyla eşle: Flatten, EncodeArithmetic, EncodeLiterals, Virtualize, AddOpaque.

Cevap: Flatten→K-04, EncodeArithmetic→K-02, EncodeLiterals→K-07/K-08, Virtualize→K-10, AddOpaque→K-01/K-03.

Soru 4

Bir dönüşüm hattı neden tek dönüşümden güçlü? Sıra neden önemli?

Cevap: Katmanlar birbirini güçlendirir (9. hafta "birlikte"). Sıra sonucu ve başarımı etkiler; kimi sıra gereksiz yavaşlatır ya da zayıf sonuç verir.

Soru 5

Her dönüşüm hattından sonra hangi iki şeyi mutlaka yaparsın?

Cevap: (1) Birim testlerini çalıştır (davranış korunmuş mu?), (2) boyut/hız ölç (maliyet).

Soru 6

Virtualize neden yalnız küçük ve kritik fonksiyonlara uygulanır?

Cevap: Maliyeti çok yüksek (onlarca kat yavaşlama, boyut). Getirisi ancak en değerli, küçük fonksiyonlarda maliyeti haklı çıkarır.

Soru 7

Tohum (`--Seed`) çeşitlendirmeyi nasıl sağlar? Uzayda/zamanda fark?

Cevap: Farklı tohum → farklı opak yüklem/sahte dal/durum. Uzayda: farklı kopyalar aynı anda. Zamanda: her sürüm yeni tohum.

Soru 8

Çeşitlendirme gizlemenin gücünü mü, ölçeklenmesini mi engeller?

Cevap: Gücünü artırmaz (bir kopya yine kırılabilir); **ölçeklenmesini** engeller — bir kırık her yeri açmaz.

Soru 9

Banescu vd. çalışmasının ders için ana kuralı?

Cevap: Dayanıklılık ölçülen bir niceliktir. Dönüşümleri birleştirmek sembolik yürütmeyi üstel zorlar; ama maliyet de artar → **dayanıklılık ↔ maliyet ödünleşir**.

Soru 10

Gizlemeyi derleme hattına koyarken imzalama neden gizlemeden sonra?

Cevap: İmza, dağıtılan **son** ikiliyi kapsamalı. Önce gizlenir, sonra imzalanır; sürüm kimliği/özet tutarlı olur.

Soru 11

S15'te "yalnız hassas fonksiyonlar gizlenir" kuralının gerekçesi?

Cevap: Her fonksiyonu gizlemek programı çok yavaşlatır/şişirir. Maliyeti yalnız değerli fonksiyonlarda üstlenmek mantıklı.

Soru 12

9, 11 ve 14. haftaların ortak kuralını bir cümleyle yaz.

Cevap: Koruma kırılamazlık değil **gecikme** sağlar; gücü **katmanların birleşiminden**, **çeşitlendirmeden** ve **ölçülmüş** olmaktan gelir.

Kapanış · üç hafta

- **9:** gizleme kuralları (el ile)
- **11:** whitebox (anahtar)
- **14:** otomasyon + çeşitlendirme

Hepsi aynı çerçeve: maliyet + katman + ölçüm.

Ek A · İşlenmiş ölçüm

Senaryo

`erisim_ver` fonksiyonunu iki hatla gizleyip ölçelim:

- **Hat A:** EncodeLiterals + EncodeArithmetic (hafif)
- **Hat B:** Hat A + Flatten + AddOpaque (ağır)

Önce · temel çizgi

```
program_temiz:
  boyut: 100 KB
  işlem süresi: 1.00× (referans)
  erisim_ver temel blok: 5
```

Hat A · ölçüm (örnek)

```
boyut: 103 KB   (+%3)
süre:  1.05×
blok:  7
strings hassas dize: 0  (öncesi 3)
```

Ucuz; dize gizleme büyük getiri.

Hat B · ölçüm (örnek)

```
boyut: 131 KB   (+%31)
süre:  1.80×
blok:  40
```

Güç arttı (blok 5→40); maliyet de arttı.

Karşılaştırma tablosu

Ölçüt	Temiz	Hat A	Hat B
Boyut	100 KB	103 KB	131 KB
Süre	1.00×	1.05×	1.80×
Blok	5	7	40
Hassas dize	3	0	0

Karar

- Varlık değeri **düşük** → Hat A yeter.
- Varlık değeri **yüksek** → Hat B (maliyeti kabul et) + çeşitlendirme.

> Karar sayılara dayanır, hisse değil.

Çeşitlendirme ölçümü (örnek)

- Hat B'yi tohum 1001 ve 2002 ile üret.
- Korunan fonksiyonun bayt farkı: **%92**.
- Yorum: bir kopyaya yazılan yama diğerinde büyük olasılıkla çalışmaz.

Ek B · Mini vaka

Vaka · kurgu

Sentetik bir uygulamada bir **lisans denetimi** ve bir **anahtar türetme** fonksiyonu var.

İlkisini de mi ağır gizleyeceğiz?

Vaka · karar

- **Lisans denetimi:** orta değer → Hat B (Flatten + AddOpaque).
- **Anahtar türetme:** yüksek değer → Hat B + **Virtualize** (küçük olduğu için maliyet kabul edilebilir).
- Gerisi: gizlenmez (maliyet gereksiz).

Vaka · çeşitlendirme + hat

- Her sürüm farklı tohum.
- Derleme hattı: gizle → imzala → sürüm kimliği.
- CI'da birim testleri + ölçüm otomatik.

Vaka · S9/S15 çıktısı

Fonksiyon	Hat	Boyut/süre	Gerekçe
lisans_dogrula	B	+%31 / 1.8×	orta değer
anahtar_turet	B+Virtualize	+%60 / 3×	yüksek değer, küçük fn.

Vaka · kalan risk

- Yeterince kararlı saldırgan tek kopyada çözebilir.
- Telafi: çeşitlendirme + kısa ömürlü anahtar + sunucu denetimi.
- **Açıkça yazılır.**

Ek C · Alternatif ve sınırlar

O-LLVM (derleyici tabanlı)

- **Obfuscator-LLVM:** dönüşümleri **derleme sırasında** uygular.
- Kaynaktan kaynağa değil; LLVM ara temsili üstünde.
- ix. hafta K-11. Tigress'e bir alternatif.

Tigress mi, O-LLVM mi?

	Tigress	O-LLVM
Nerede	Kaynak (C→C)	Derleyici (IR)
Görünürlük	Gizli kaynağı görebilirsin	Derleme adımı
Dönüşüm zenginliği	Çok (Virtualize dahil)	Daha sınırlı

Ortak sınır

- İkisi de iyi bilinir → kalıpları tanınabilir.
- İkisi de **çeşitlendirme** ve **katmanlı savunma** ister.
- İkisi de kırılamazlık **vermez**.

Terimler sözlüğü

Terim	Anlam
Kaynaktan kaynağa	C girip gizli C çıkaran gizleme
Dönüşüm	Tek gizleme işlemi
Hat	Dönüşümlerin sıralı bütünü
Tohum	Rastgeleliği yöneten sayı (çeşitlendirme)
Sembolik yürütme	Yol-çözen otomatik analiz (KLEE)

Kaynaklar

- **Tigress** resmî sitesi/çalışma sayfaları (`tigress.wtf`) — dönüşümler, sözdizimi, v4, lisans
- Collberg & Nagra, *Surreptitious Software* — taksonomi, ölçme
- Banescu vd. — Tigress + KLEE dayanıklılık ölçümü
- Obfuscator-LLVM — derleyici tabanlı alternatif

Özet: bu haftanın tek cümlesi

> Tigress, 9. haftanın gizleme kurallarını **otomatik** ve **çeşitlendirilmiş** uygular; ama kural aynı: gizleme
> **kırılamazlık değil gecikme**; birleştir, çeşitlendir, **ölç**.

Sonraki hafta

15. hafta — Final proje gösterimleri (RAP2)

Dönem içeriği tamamlandı. Final raporunda bu haftanın hattı (S15) ve ölçümleri (S9) beklenir. 16. hafta: Quiz-2.

Ek D · Dönüşüm dönüşüm

Neden tek tek?

Bir hat kurarken **hangi dönüşümü neden** seçtiğinizi bilmelisiniz.

Her dönüşümü: ne yapar · ne zaman · maliyet.

Flatten

- **Ne yapar:** kontrol akışını tek switch'e taşır (K-04).
- **Ne zaman:** algoritma yapısını gizlemek için; çoğu hatta.
- **Maliyet:** orta.

AddOpaque / InitOpaque

- **Ne yapar:** opak yüklemler ve sahte dallar ekler (K-01, K-03).
- **Ne zaman:** Flatten'ı güçlendirmek için.
- **Maliyet:** orta.

EncodeArithmetic

- **Ne yapar:** aritmetiği denk karmaşık ifadeyle değiştirir (K-02).
- **Ne zaman:** hesap/karşılaştırma içeren fonksiyonlarda.
- **Maliyet:** düşük.

EncodeLiterals

- **Ne yapar:** sabit ve dizgeleri kodlar (K-07, K-08).
- **Ne zaman:** neredeyse her zaman (ucuz, yüksek getiri).
- **Maliyet:** çok düşük.

EncodeData

- **Ne yapar:** değişkenlerin gösterimini kodlar (K-09).
- **Ne zaman:** hassas değişkenleri gizlemek için.
- **Maliyet:** düşük–orta.

Split / Merge

- **Ne yapar:** fonksiyonları böler/birleştirir; yapıyı bulanıklaştırır (K-06).
- **Ne zaman:** fonksiyon sınırlarını gizlemek için.
- **Maliyet:** düşük–orta.

Virtualize

- **Ne yapar:** fonksiyonu özel VM bayt koduna çevirir (K-10).
- **Ne zaman:** yalnız en kritik, **küçük** fonksiyonlar.
- **Maliyet: yüksek** (onlarca kat yavaşlama).

Jit

- **Ne yapar:** kodu çalışma anında üretir (dinamik, K-12).
- **Ne zaman:** çok seçici; ileri kullanım.
- **Maliyet:** yüksek; OS korumalarıyla dikkat.

AntiBranchAnalysis / AntiAliasAnalysis / AntiTaintAnalysis

- **Ne yapar:** ilgili statik analiz tekniğini zorlaştırır (önleyici aile).
- **Ne zaman:** belirli analiz tehdidine karşı.
- **Maliyet:** değişken.

RandomFuns / RndArgs

- **Ne yapar:** rastgele sahte fonksiyon/parametre ekler.
- **Ne zaman:** çeşitlendirmeyi güçlendirmek için (tohumla).
- **Maliyet:** düşük–orta.

Hangi dönüşüm önce?

Tipik sıra (kavramsal):

1. EncodeLiterals (ucuz temizlik)
2. EncodeArithmetic
3. Flatten
4. AddOpaque
5. (gerekirse) Virtualize
6. RandomFuns/RndArgs (çeşitlendirme)

Ek E · Gizleme kararı akışı

"Bu fonksiyonu gizlemeli miyim?" — 1

Soru 1: Bu fonksiyon hassas mı? (lisans, anahtar, bütünlük, denetim)

- **Hayır** → gizleme; maliyeti boşa harcama.
- **Evet** → devam.

"Bu fonksiyonu gizlemeli miyim?" — 2

Soru 2: Değeri yüksek mi?

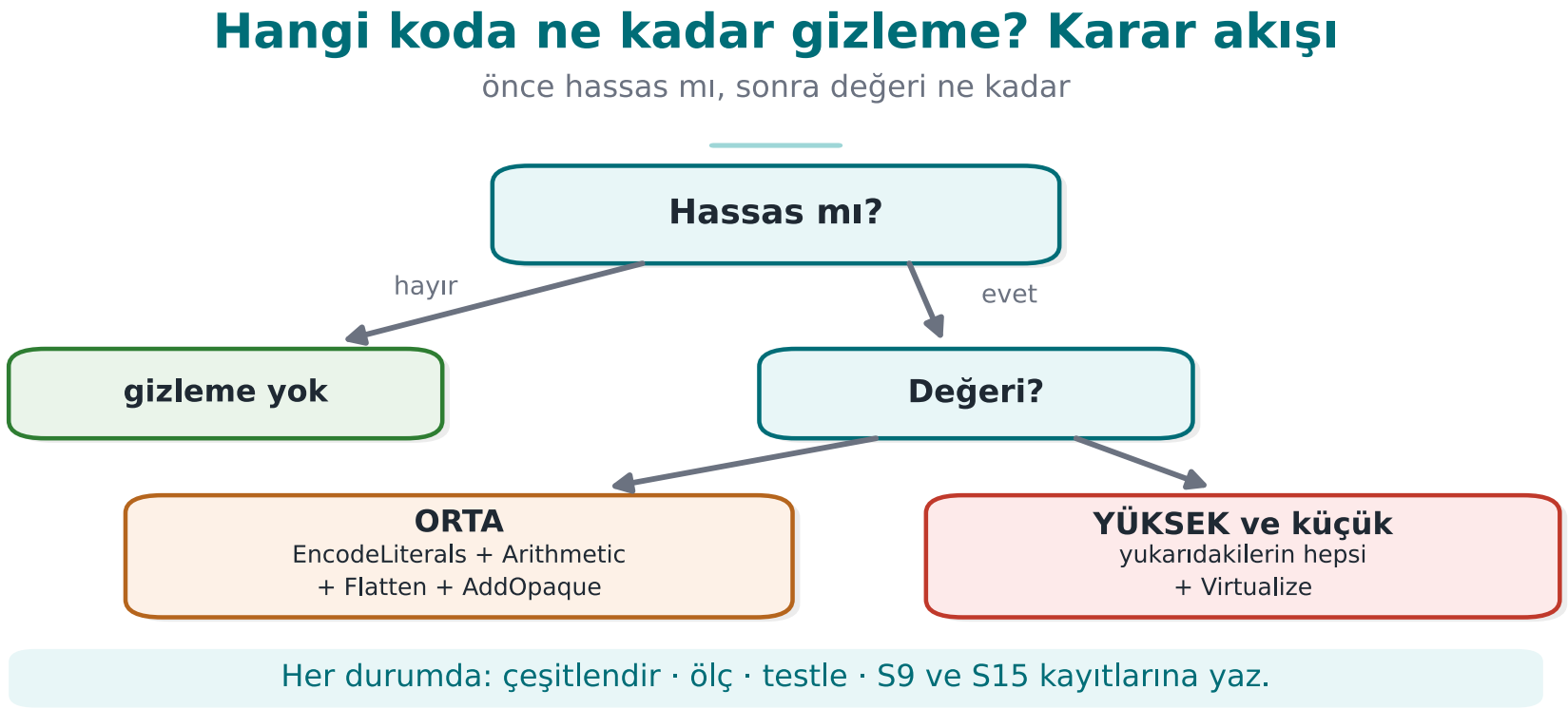
- **Orta** → EncodeLiterals + EncodeArithmetic + Flatten + AddOpaque.
- **Yüksek ve küçük** → yukarısı + Virtualize.

"Bu fonksiyonu gizlemeli miyim?" — 3

Soru 3: Her durumda:

- Çeşitlendir (tohum)
- Ölç (boyut, süre, blok)
- Birim testleriyle davranışı doğrula
- S9/S15'e yaz

Karar akışı · tek bakış



Her durumda: çeşitlendir · ölç · testle · S9 ve S15 kayıtlarına yaz.

Ek F · Sık yapılan hatalar

Hata · her şeyi gizlemek

- Tüm fonksiyonları gizlemek programı çok yavaşlatır/şişirir.
- Yalnız hassas ve değerli fonksiyonlar.

Hata · testsiz gizleme

- Gizlemeden sonra birim testleri çalıştırılmazsa, bozuk davranış fark edilmez.
- Her hattan sonra test **zorunlu**.

Hata · ölçmeden "güçlü" demek

- Ölçüm yoksa değerlendirici (12. hafta) bunu **iddia** sayar.
- Boyut/süre/blok sayılarını yaz.

Hata · çeşitlendirmeyi atlamak

- Tek tip gizleme → bir kırık her kopyayı açar.
- Her sürüm/dağıtım farklı tohum.

Hata · imzayı gizlemeden önce yapmak

- İmza dağıtılan **son** ikiliyi kapsamalı.
- Önce gizle, sonra imzala.

Hata · eski/dağıtılan Tigress kullanmak

- Derste ikili dağıtılmaz; güncel sürümü resmî siteden indir.
- Lisans koşulunu doğrula.

Ek · kapanış notu

Bu ekler size **kendi hattınızı** kurmanın somut kalıbını verdi:

- dönüşüm dönüşüm seçim
- karar akışı
- kaçınılacak hatalar

> Doğru araç + doğru ölçüm + çeşitlendirme = savunulabilir bir S9/S15.

Ek G · İkinci işlenmiş hat

Yeni örnek · bütünlük denetimi

```
int dosya_saglam(const uint8_t *veri, size_t n,
                 const uint8_t *beklenen) {
    uint8_t ozet[32];
    ozet_hesapla(veri, n, ozet);        /* SHA-256 benzeri */
    return sabit_zamanli_esit(ozet, beklenen, 32);
}
```

Sürüm bütünlüğünü denetleyen sentetik bir fonksiyon.

Neden bunu gizleyelim?

- Saldırgan bu denetimi bulup **atlatmak** ister (kurcalanmış dosyayı geçirmek için).
- Denetimin dönüş noktası ve karşılaştırması hedef.
- ix. hafta: opak boolean + rastgele çıkış burada kritik.

Hat seçimi

```
tigress \
  --Transform=EncodeLiterals \
  --Transform=EncodeArithmetic \
  --Transform=Flatten \
  --Transform=AddOpaque \
  --Functions=dosya_saglam \
  --out=gizli.c temiz.c
```

Adım adım · ne olur?

- EncodeLiterals: `32` ve varsa dizgeler düz görünmez.
- EncodeArithmetic: karşılaştırma karmaşıklaşır.
- Flatten: tek dönüş noktası görünmez.
- AddOpaque: sahte dallar; "başarı" dalı tek yerde değil.

Davranış doğrulama (zorunlu)

- Doğru özetli dosya → **geçer**.
- Bozuk özetli dosya → **reddedilir**.
- Gizli sürümde de aynı sonuç → hat doğru.

CFG önce/sonra (kavram)

Denetimin nerede geçtiği/kaldığı akıştan okunmaz.

⚠️ Sabit zamanlılık korunmalı

- `sabit_zamanli_esit` gizlense de **sabit zamanlı** kalmalı.
- Gizleme uğruna erken çıkış eklenirse yan kanal açılır (3. hafta).
- Dönüşümlerin bunu bozmadığını **test et**.

Ölçüm (örnek)

Ölçüt	Önce	Sonra
Boyut	100 KB	128 KB
Süre	1.0×	1.7×
Blok (dosya_saglam)	4	33

Çeşitlendirme

- İki tohumla üret → iki farklı ikili.
- "Bütünlük denetimini atla" yaması bir kopyada işe yarasa bile diğerinde yaramaz.

İkinci örnek · çıkarım

- Bütünlük denetimi gibi **atlatma hedefi** fonksiyonlarda gizleme değerlidir.
- Ama asıl güç: gizleme + RASP (6. hafta) + sunucu denetimi.
- Gizleme tek başına atlatmayı **geciktirir**, engellemez.

Ek H · Hızlı başvuru

Dönüşüm seçim kılavuzu

İstediğin	Dönüşüm
Dizge/sabit gizle	EncodeLiterals
Hesabı gizle	EncodeArithmetic
Yapıyı gizle	Flatten
Sahte/opak ekle	AddOpaque
Makine kodunu gizle	Virtualize (pahalı)
Çeşitlendir	--Seed, RandomFuns

Kontrol listesi

- [ ] Yalnız hassas fonksiyonlar (`--Functions`)
- [ ] Hat sırası mantıklı (ucuzdan pahalıya)
- [ ] Birim testleri geçiyor (davranış)
- [ ] Boyut/süre/blok ölçüldü
- [ ] İki tohumla çeşitlendirildi
- [ ] İmza gizlemeden sonra
- [ ] S9/S15'e yazıldı

Son söz (14. hafta)

> Araç güçlüdür ama **karar sizindir**: neyi, neden, hangi maliyetle gizliyorsunuz?

Ölç, çeşitlendir, katmanla. Gizleme kırılamazlık değil, **kazanılan zamandır**.