**Server Side Request Forgery(SSRF) – uputstvo i objašnjenje implementiranih izazova**

U ovom dokumentu su opisani implementirani CTF (Captire the Flag) izazovi. Implementacija izazova se sastoji od jednog ili nekoliko web servera na kojima se nalazi ključ(flag) koji je potrebno pronaći primenom neke tehnike u okviru SSRF kategorije napada.

SSRF napadi se mogu izvršiti u situaciji kada server koji je na strani korisnika dohvata podatke sa drugih servera i prosleđuje ih korisniku. Ako korisnik može da izmeni URL sa kog server dohvata sadržaj, može da natera server da dohvati podatke sa neželjene lokacije. Uobičajni tok napada jeste da korisnik dohvati podatke sa internog servera organizacije sa kojim ne bi smeo da komunicira. Takođe moguće je naterati server da komunicira sa bilo kojim resursom na internetu, čime se mogu otkriti kredencijali servera.

Ovakva situacija je česta u organizacijama gde postoji jedan server sa kojim korisnici mogu da interaguju, a koji dohvata podatke sa ostalih servera u organizaciji i prosleđuje ih nazad korisniku. U zavisnosti od unosa korisnika se mogu kontaktirati različiti serveri u okviru organizacije. Ako korisnik pošalje adekvatne podatke može naterati server da dohvati podatke sa drugih servera u okviru organizacije kojem ne bi trebalo da ima pristup i da dobije pristup poverljivim podacima.

U nastavku je opisana implementacija nekoliko izazova koji imaju za cilj da demonstriraju ranjivosti tipa SSRF.

1. **Blacklist tip napada**

Ovaj izazov se sastoji od dva servera: main\_app.py i hidden\_app.py. Korisnici mogu da pristupaju main\_app.py serveru koji je pokrenut na <http://localhost:5000>. Korisnici ne mogu da pristupaju hidden\_app.py serveru koji je pokrenut na http:. Na ovom serveru se nalazi flag koji je rešenje izazova, a u stvari predstavlja poverljiv podatak.

Korisnici na početnoj stranici pomoću padajuće liste mogu da biraju na koju će stranicu otići. Ovo je implementirano tako što korisnik u padajućoj listi, koja je deo forme, bira tekstualnu vrednost URL (npr. <http://localhost:5000/post1>) i šalje POST zahtev na server. Server prihvata ovaj POST zahtev i na osnovu odabrane vrednosti dohvata stranicu i prikazuje je korisniku.

Da bi se server zaštitio od malicioznih korisnika implementirano je blacklist filtriranje URL koji je korisnik poslao. Ako URL sadrži string localhost ili ako je vrednost prvog bajta ip adrese 127, server ne dohvata stranicu na toj adresi i korisniku vraća poruku o grešci. Ako URL sadrži string ‘admin’ korisniku se opet prikazuje poruka o grešci.

Ovaj tip zaštite se može zaobići tako što se umesto localhost serveru pošalje localhosT. Provera admin stringa se može zaobići tako što se koristi URL Encoding. Karakteri u okviru URL se mogu enkodovati pomoću znaka % i njihovog heksadecimalnog ascii koda, na primer a se enkoduje kao %61. Tako se admin može napisati kao %61dmin. Pomoću inspect element korisnik može umesto linka ka post1 uneti <http://localhosT:5001/%61dmin> i dohvatiti sadržaj sa stranice administratora.

1. **Whitelist tip napada**

Ovaj izazov ima istu strukturu istu kao i prethodni, korisnik u padajućem meniju bira stranicu koju želi da učita. Razlika je što se za zaštitu od malicioznih URL koristi whitelist filtriranje. Dozvoljeno je dohvatiti sadržaj sa URL koji sadrže string localhost:5000. Ako URL ne sadrži ovaj string korisniku se ispisuje poruka o grešci.

Ovaj tip zaštite se može zaobići tako što se koriste neki od oblika validnog URL, na primer:

* Pomoću karaktera @ se mogu postaviti kredencijali pre imena malicioznog hosta:
* Pomoću karaktera # se mogu postaviti url fragmenti:
* Očekivani deo se može poslati kao query parametar na maliciozni sajt:
* Može se koristiti URL encoding ili double URL encoding

Kao rešenje ovog primera se može koristiti bilo koja od ovih tehnika. Jedno rešenje je da korisnik pomoću inspect element izmeni stranicu tako da se zahteva sadržaj stranice koja se nalazi na adresi <http://localhost:5001/admin?localhost:5000/post1>.

Whitelist filtriranje URL na serveru se zaobilazi korišćenjem veoma složenog formata koji validan URL može da ima. URL parseri će korektno isparsirati ovaj URL, ali će on proći filter jer sadrži određeni string koji filter proverava.

1. **File URL**

Ovaj izazov se sastoji od jednog servera main\_app.py, koji ima sličnu strukturu kao i prethodni primeri. Korisnik u padajućem meniju bira stranicu koju želi da dohvati, odabrani URL se šalje serveru koji na osnovu njega dohvata stranicu i prikazuje je korisniku.

URL parseri parsiraju kao validne URL koji ukazuju na fajlove koji se nalaze na lokalnoj mašini. Korisnik može da od servera zahteva da dohvati URL koji ukazuje ne file i da na taj način dohvati osetljive podatke. Recimo ako korisnik zna da je server pokrenut na nekoj distribuciji linuxa može da dohvati /etc/shadow fajl.

U okviru ovog primera je kreiran fajl my\_secrets.txt fajl koji sadrži osetljive podatke korisnika. Korisnik treba da izmeni stranicu pomoću inspect element tako da zahteva sadržaj na URL: 

1. **DNS Rebinding napad**

Ovaj izazov sastoji se od jednog servera app.py, pokrenutog na localhost:5000. Korisnik unosi URL sajta koji želi da poseti, koji se potom prikazuje kroz iframe, ukoliko je URL dozvoljen. Cilj je otkriti flag pristupom na lokaciju <http://127.0.0.1/flag>.

Jednostavan unos ovog URL-a neće dohvatiti flag, jer je aplikacija zaštićena [**advocate**](https://snyk.io/advisor/python/advocate) bibliotekom. Prilikom poziva, biblioteka proverava da li URL vodi ka lokalnoj mašini, i tek kada provera prođe, šalje zahtev za dohvatanje stranice.

Pošto se zahtevi šalju dva puta, ideja je da se odgovor sa nekog domena promeni između dva poziva i tako zaobiđe ova SSRF zaštita. Za rebind može se iskoristiti online rebinding servis, poput <https://lock.cmpxchg8b.com/rebinder.html>. Potrebno je dostaviti dve IP adrese, gde je jedna 127.0.0.1, a druga neka proizvoljna, koja će odvesti ka sajtu koji prolazi proveru.  
 Kada se desi da prvi zahtev ode ka ispravnoj adresi, a drugi ka localhostu, biće prikazan skriveni flag. URL koji vodi do uspešnog napada je u formatu [**http://7f000001.4a7d4464.rbndr.us/flag**](http://7f000001.4a7d4464.rbndr.us/flag).

1. **Port scan**

Cilj izazova je demonstracija kako napadač može otkriti skrivene servise koji se ne nalaze na standardnim portovima. Na serveru je postavljen web servis koji prikazuje *flag*, ali nije poznato na kom portu se servis nalazi. Napadač koristi Python skriptu koja pokušava da se poveže na različite portove i proverava da li na njima postoji HTTP odgovor. Kada pronađe port na kojem servis odgovara, može da otvori taj port u pretraživaču ili skriptom i dođe do *flag*-a.

Ovakvi napadi se u praksi nazivaju *port scanning* ili *port enumeration*, i često prethode ozbiljnijim napadima. Iako se ovde koristi jednostavna skripta, napredniji alati poput nmap mogu brzo otkriti veliki broj otvorenih portova na cilju.

Napad pokazuje važnost kontrole pristupa i filtriranja saobraćaja ka internim ili nestandardnim servisima.

1. **Blacklist tip napada sa duplim enkodiranjem**

Ovaj izazov se sastoji od dva servera: main\_app.py i hidden\_app.py. Server main\_app.py je javno dostupan i korisnici mu mogu pristupiti putem <http://localhost:5000>. Server hidden\_app.py nalazi se na http: i on nije direktno dostupan korisnicima. Na njemu se nalazi poverljiv podatak koji prestavalja rešenje izazova.

Korisnički interfejs na početnoj stranici omogućava izbor ciljne stranice preko padajućeg menija, u okviru forme. Nakon što korisnik odabere jednu od tekstualnih vrednosti URL-a (npr. <http://localhost:5000/post1>) i pošalje POST zahtev na server, server na osnovu odabrane vrednosti prepoznaje URL i prikazuje mu željenu stranicu.

Da bi se server zaštitio od malicioznih korisnika implementirano je blacklist filtriranje URL koji je korisnik poslao. Ako URL sadrži string localhost ili ako je vrednost prvog bajta ip adrese 127, server ne dohvata stranicu na toj adresi i korisniku vraća poruku o grešci. Ako URL sadrži string ‘admin’ korisniku se opet prikazuje poruka o grešci. Implementirana je i zaštita od pokušaja zaobilaženja filtera korišćenjem URL enkodovanja. Server će dodatno dekodirati putanju (npr. %61dmin postaje admin) i proverava da li se enkodovana vrednost poklapa sa nedozvoljenim izrazima. Na taj način su onemogućeni pokušaji skrivanja nedozvoljenih izraza pomoću jednostrukog URL enkodiranja.

Ovakav tip zaštite može se zaobići korišćenjem dvostrukog URL kodiranja. Karakter %, koji se koristi pri jednostrukom kodiranju, može se dodatno kodirati kao %25, pa tako jednostruko enkodirani %61dmin postaje dvostruko enkodirani %2561dmin. Pomoću opcije *Inspect element*, korisnik može izmeniti vrednost forme i umesto linka ka post1 uneti <http://localhosT:5001/%2561dmin>, čime zaobilazi zaštitu i uspeva da dohvati sadržaj sa stranice administratora.

1. **Blind Referer Izazov**

Ovaj izazov se sastoji od dva servera: main\_app.py i hidden\_app.py. Server main\_app.py je javno dostupan i korisnici mu mogu pristupiti putem <http://localhost:5000>. Server hidden\_app.py nalazi se na http: i on nije direktno dostupan korisnicima.

Korisnički interfejs na početnoj stranici omogućava izbor ciljne stranice preko padajućeg menija, u okviru forme. Nakon što korisnik odabere jednu od tekstualnih vrednosti URL-a (npr. <http://localhost:5000/post1>) i pošalje POST zahtev na server, server na osnovu odabrane vrednosti prepoznaje URL i prikazuje mu željenu stranicu.

Napad se izvršava pomoću alata Postman, gde napadač u zaglavlje zahteva dodaje Referer polje sa URL-om stranice koju će server da poseti: <http://localhost:5001/admin>. Server proverava Referer zaglavlje u kome se nalazi URL sa kog je stigao zahtev. Menjanjem ovog polja, napadač može da natera skriveni server da poseti neku stranicu i da na taj način sazna bezbednosne informacije o serveru: tokene, lozinke, podatke o operativnom sistemu... Takođe, u telo zahteva je potrebno dodati URL putanju sa koje se zahteva sadržaj: <http://localhost:5000/post1>.

Ovaj CTF demonstrira Blind SSRF ranjivost, gde se napadaču ne prikazuje nikakva informacija o tome koji server je posećen. Napadač može poznavajući osobine HTTP da umetne neka polja, kao što je Referer zaglavlje, i tako da natera server da poseti određeni resurs. Napadač može da natera server da poseti bilo kakvu malicioznu stranicu.

1. **Redirect**

U ovom izazovu korisnik ima mogućnost da pošalje URL koji aplikacija koristi da izvrši HTTP zahtev. Postoji neki jednostavan filtar koji blokira direktan poziv localhost.  
 Cilj napadača je da zaobiđe ove filtere i dođe do skrivenog admin endpointa koji se nalazi na http://127.0.0.1:5001/admin, gde se nalazi **flag**.  
 Zaobilaženje filtera je moguće pomoću **open redirect** mehanizma — aplikacija šalje zahtev ka URL-u koji vrši preusmerenje ka zabranjenoj adresi, čime filter biva zaobiđen.

1. **File URL napad sa kodiranjem**

U ovom izazovu postoji jedan server (main\_app.py) kojem korisnici pristupaju putem adrese <http://localhost:5000>. Korisnički interfejs na početnoj stranici omogućava izbor ciljne stranice preko padajućeg menija u okviru forme. Nakon što korisnik odabere jednu od tekstualnih vrednosti URL-a (npr. <http://localhost:5000/post1>) i pošalje POST zahtev na server, server na osnovu odabrane vrednosti prepoznaje URL i prikazuje željenu stranicu.

Na serveru je implementirana zaštita od SSRF napada putem file:/// protokola tako što se reč file blacklistuje u prosleđenim URL adresama. Ova zaštita se može zaobići pomoću kodiranih karaktera. Napadač može da izmeni HTML element pomoću *Inspect element* opcije u pregledaču i podmetne vrednost URL-a sa kodiranom rečju file, čime se zaobilazi blacklist mehanizam odbrane:

%66ile:///<path-to-file>\passwords.txt

1. **Todo**

**Zaključak**

U ovom dokumentu su predstavljeni izazovi koji demonstriraju propuste u okviru kategorije SSRF napada. Ova napadi demonstriraju da nekorektna obrada podataka koje unosi korisnik može da dovede do sigurnosnih propusta u organizaciji. Posledice ovakvih napada su pristup internim servisima organizacije, zaobilaženje firewall zaštite i ostalih restrikcija i prikupljanje osetljivih podataka.

Zaštita od korisničkog unosa se zasniva na filtriranju korisničkog unosa, odnosno odbacivanju zahtevanih URL koji se smatraju kao maliciozni. Glavni problem sa ovim je to što je sintaksa validno URL izuzetno složena, tako da je lako prevariti URL parsere u svim programskim jezicima. Pored filtriranja postoje i druge tehnike, kao što je DNS rebinding, kojima se dodatno može zaobići filtriranje URL.

Sa složenijim strukturama organizacija i uvođenjem proxy servera ovaj tip napada je sve više zastupljen i postaje sve bitniji u razmatranju bezbednosti servera.