TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Hans Erik 185430IAIB

SERVERIPOOLSE PÄRINGU VÕLTSIMISE VIRTUAALLABORITE KURSUSE KAVA

Bakalaureusetöö

Juhendaja: Sten Mäses
PhD

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Hans Erik

28.05.2022

Annotatsioon

Käesoleva lõputöö eesmärgiks oli luua virtuaallaborite kursuse kava RangeForce'i õppeplatvormi jaoks, mille põhjal loodud kursus võimaldab õppuritel tutvuda serveripoolse päringu võltsimise küberturvariski ja selle kaitsetega. Kursuse kava on RangeForce'i tiimi poolt kasutatav sisemine dokument, mille põhjal RangeForce'i arendajad loovad kursused.

Virtuaallaborite kursuse eesmärk oli anda põgus ülevaade serveripoolse päringu võltsimise küberründemeetoditest ja käsitleda põhjalikult turvariski parandamist ja vältimist. Lõputöö eesmärgi saavutamiseks oli vaja süvitsi mõista serveripoolse päringu võltsimise turvariski, et sellest seejärel luua lühike, aga informatsioonirikas kursus, mille läbinud õppurid saavad enesekindlalt õpitud kaitsemeetmeid reaalelus rakendada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 22 leheküljel, 5 peatükki, 6 joonist.

Abstract

Server-Side Request Forgery Virtual Lab Course Plan

The goal of this thesis was to develop a virtual lab course plan for the online learning platform RangeForce. A course created according to this plan allows learners to familiarize themselves with the server-side request forgery vulnerability and the defences against it. A course plan is an internal document used by the RangeForce team to develop online virtual lab courses.

The goal of the virtual lab course was to give a brief overview of the bypasses and exploits related to server-side request forgery and more thoroughly discuss the defences against it. For the purposes of achieving this goal, it was imperative to understand and map the server-side request forgery vulnerability, in order to create a short but informative course. As a result of passing this course, a learner should be capable of defending an application against server-side request forgery attacks in a real-world situation.

The thesis is in Estonian and contains 22 pages of text, 5 chapters, 6 figures.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface,* rakendusliides
AWS	*Amazon Web Services,* Amazoni veebiteenused
HTML	*HyperText Markup Language,* hüperteksti märgistuskeel
HTTP	*Hypertext Transfer Protocol,* hüperteksti edastusprotokoll
IP	*Internet protocol,* internetiprotokoll
OWASP	*Open Web Application Security Project,* avatud veebirakenduse turvalisuse projekt
SDK	*Software Development Kit,* tarkvaraarenduskomplekt
SSH	*Secure Shell,* turvakest
SSR	*Server-side Request,* serveripoolne päring
SSRF	*Server-side Request Forgery,* serveripoolse päringu võltsimine
URI	*Uniform Resource Identifier,* ühtne ressursiidentifikaator
URL	*Uniform resource locator,* üldine infoallika asukohamääraja

Sisukord

Jooniste loetelu

1 Sissejuhatus

RangeForce on pilvepõhine küberoskuste platvorm, mis kasutab õpetamiseks reaalset IT infrastruktuuri, päris turvavahendeid ja päris küberohte [1]. Üks õppimismeetod, mida Rangeforce'i platvorm pakub, on virtuaallaborid. Need virtuaallaborid aitavad õppuril reaalelulisi küberturbe situatsioone, nii küberrünnakuid ja kaitsmist, läbi mängida ohutus keskkonnas.

Küberturve on kiiresti muutuv teema ning samaaegselt, kui veebiarendustehnoloogiad ja -võtted edasi liiguvad, tekkivad turvaohud, mille kohta peab arendaja teadlik olema. RangeForce'i poolt arendatud virtuaallaborite valik on lai ning katab ära suure osa avatud veebirakenduse turvalisuse projekti (OWASP – ingl *Open Web Application Security Project*) veebirakenduste turvariskide esikümne, aga veebiturvalisuse maailm on muutuv ning uued turvariskid või varasemalt vähem tähtsaks peetud ohud võivad esile tulla. Aastal 2021 uuendatud OWASPi veebirakenduste turvariskide esikümnesse kümnendal kohal on serveripoolse päringu võltsimine (SSRF – ingl *Server-Side Request Forgery*) [2].

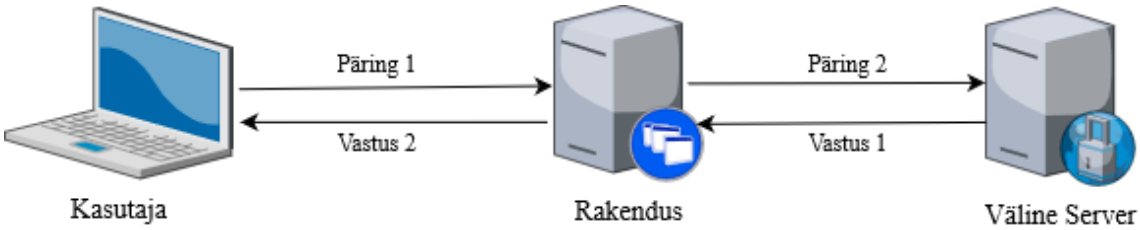
Antud lõputöö eesmärk on luua virtuaallaborite kursuse kava, mille põhjal loodud virtuaallaborid aitaks varem SSRF teemaga mitte kokku puutunud õppuril mõista turvaohu olemust, ründemeetodeid ja kõige tähtsamalt, et kuidas arendatavat veebirakendust kaitsda SSRF küberrünnakute eest.

Hetkeseisuga teeb Tallinna Tehnikaülikooli mitmes aines koostööd RangeForce'iga, et edendada üliõpilaste teadlikkust küberturbe põhiteemade ja mõistete kohta [3].

Töö esimeses osas antakse ülevaade kirjandusest. Töö teises osas selgitatakse ülesande püstituse protsessi. Töö kolmandas osas kirjeldatakse töö käiku ja esitatakse lõpptulemus.

2 Kirjanduse ülevaade

Serveripoolne päring (SSR – ingl *Server-side Request)* on kommunikatsioonimuster, milles osaleb kolm osapoolt: kasutaja, rakendus ja väline server. See muster on kujutatud joonisel 1. Kommunikatsiooniprotsess algab, kui kasutaja saadab hüperteksti edastusprotokolli (HTTP – ingl *Hypertext Transfer Protocol*) päringu (*päring 1)* rakendusele, mis sisaldab üldist infoallika asukohamäärajat (URL – ingl *uniform resource locator*). Rakendus kasutab saadud URL-i, et luua ühendus välise serveriga ja pärida (*Päring 2*) vastav ressurss (*Vastus 1*) väliselt serverilt. Kui päring saab vaste, siis rakendus tagastab (*Vastus 2*) päritud infromatsiooni kas muudetud või muutmata kujul kasutajale. SSR-ks kutsutakse päringut, mille rakendus teeb välisele serverile (*Päring 2*) [4].



Joonis 1. SSR kommunikatsioonimuster

2.1 Serveripoolse päringu võltsimine

Serveripoolse päringu võltsimine ehk SSRF tekkib, kui rakendus ei valideeri piisavalt kasutaja poolt antud sisendeid, olgu need kas URL-d või muud andmed, millest SSR luuakse [4]. Tagajärjena võib rakendus tagastada tundlikku informatsiooni. Samuti on SSRFi ära kasutades võimalik kaardistada rakenduse võrgustikku, saada ligi muul juhul ligipääsematule eraserverile ja äärmuslikel juhtudel ka kasutaja poolt varustatud koodi rakenduse peal käivitada [5].

2.2 OWASP

OWASP on tuntud mittetulunduslik sihtasutus, mille eesmärk on edendada tarkvara turvalisust. Aastal 2021 väljandatud OWASP veebirakenduste turvariskide esikümnes ilmus esimest korda SSRF. Selle turvariski esinemissagedus on ajalooliselt madal ja

testimiskate kõrge [6], aga sellegipoolest vääris see OWASPi tiimi hinnangul esikümnes kümnendat kohta. SSRF oli OWASPi poolt läbi viidud küsitluse andmetel number üks turvarisk [7], mille kohta küsitluse täitjad tundsid, et OWASP peaks rohkem tähelepanu pöörama.

2.3 OWASP Cheat Sheet Series

OWASP Cheat Sheet Series on OWASP-i poolt arendatav projekt, mille eesmärk on pakkuda tarkvaraarendajatele kokkuvõtlikku ülevaadet rakendusturvalisuse teemade kohta. OWASP Cheat Sheet Series pakub ülevaatliku kokkuvõtet SSRF kaitsmise teemal, kus on välja toodud erinevad kaitsemeetodid ning nende kasutusjuhendid. Virtuaallborite kursuse kava loomise käigus toetuti OWASP Cheat Sheet Series poolt pakutud juhenditele ja materjalidele [8].

2.4 HackTricks

HackTricks on Carlos Polopi poolt arendatud häkkimismetoodikate kogum. HackTricks on mitme küberturbe firma poolt toetatud projekt. Selle eesmärk on pakkuda teadmisi ja tööriistu kübersissetungi testijatele. HackTricksi kasutatakse rohkelt ka rakenduse küberkurjategijate vastu kaitsemises, kuna teadmised ründemeetoditest võivad suunata turvameetmete arendust. SSRF teemal pakub HackTricks sissejuhatavat tutvustust turvariski kohta ning laia valikut turvariski ära kasutamise ja kaitsemeetodite läbimise võtteid. Virtuaallaborite kursuse loomise käigus kasutati HackTricksi materjale, et kujundada kursuse ründamisele suunatud osa [9].

3 Ülesande püstitus

Lõputöö eesmärk oli luua küberturbe teemasid käsitlev virtuaallabor või -laborid. Eesmärgi saavutamiseks tehti koostööd RangeForce firmaga. Ülesande struktuur ja teema olid algselt lahtised ja esialgne teemavalik tehti enne RangeForce'i meeskonnaga kontakteerumist. Küberturbe valdkond on väga laialdane, seega teemasid oli palju, millest valida. Eesmärgiga, et vähendada valikute arvu, uuriti OWASPi veebiriskide esikümmet, mida oli aastal 2021 uuendatud. Uuendusel oli esikümnes niimõnigi uus teema ja esikümnest valimine tagas, et valik oli asjakohane. 2021 uuendatud OWASPi veebiriskide esikümnes esines kolm uut teemat [2]:

1. *A04:2021-Insecure Deisgn*
2. *A08:2021-Software and Data Integrity Failures*
3. *A10:2021-Server-Side Request Forgery (SSRF)*

Esimesed kaks, ebaturvaline disain ning tarkvara ja andmete terviklikkuse rikked, on laialdased teemad, mida on raske otseselt käsitleda, sest nad sisaldavad paljusid väiksemaid teemasid. Kolmas teema, serveripoolse päringu võltsimine, oli teisest kahest konkreetsem. Lõppkokkuvõttes jäi seesama konkreetsus otsustavaks faktoriks, et milline teema Rangeforce'i tiimiga kontakteerumisel välja pakutakse.

3.1 RangeForce

RangeForce on pilvepõhine küberoskuste platvorm, mis kasutab õpetamiseks reaalset IT infrastruktuuri, päris turvavahendeid ja päris küberohte. Tallinna Tehnikaülikool teeb koostööd RangeForce'iga näiteks aines Küberturbe Alused (ITI0216), et anda tudengitele ühe kursuse mahus teoreetiline ülevaade ja praktiline kogemus kõige tähtsamate küberturbe aspektide kohta.

RangeForce kui õppimisplatvorm aitab õppuritel omandada praktilist kogemust turvariskidega kokku puutumises ilma, et needsamad turvariskid õppurile reaalset ohtu osutaks. Nad saavutavad seda kasutades pilvepõhiseid virtuaalmasinaid, milles jookseb

tavaliselt Windowsi või Linuxi põhine operatsioonisüsteem, milles on üles seatud virtuaallabori keskkond.

Virtuaallaborite ülesehitus oleneb käsitletavast teemast. Näiteks veebirakenduse turvariske käsitlevad laborid sisaldavad tihti lokaalset veebirakendust ja vajaduse korral, kui õpetatakse turvariski parandamist, ka rakenduse koodi, millele õppur saab ligi ning võib muudatusi teha.

RangeForce'i virtuaallaborite kasutamine küberturbe kursustel on aidanud Tallinna Tehnikaülikooli õppejõududel aega ja ressursse kokku hoida. Toetudes RangeForce'i platvormile ei pea kursuste korraldajad ise laboreid looma ja majutama. Küberturbe maailm on muutuv ning kasutades RangeForce'i meeskonna poolt loodud virtuaallaboreid on kursustel käsitlevate teemade muutmine ja uuendamine lihtne [10].

3.2 Olemasolevad RangeForce'i laborid

Esimeseks sammuks ülesande püstitamise käigus oli uurida juba olemasolevaid virtuaallaboreid, mida RangeForce'il SSRF teemal pakkuda oli. RangeForce'il oli olemas kaks SSRF teemaga seonduvat laborit.

Esimene labor, „AWS Instance Metadata SSRF", oli viieteist minutiline lihtsa raskusastmega virtuaallabor, mis käsitles otseselt ühte SSRF-i ründevektoritest, Amazoni veebiteenuste (AWS – ingl *Amazon Web Services*) virtuaalmasina metaandmete kätte saamist.

Teine labor, „Spider Challenge", oli pikem, neljakümne viie minuti pikkune, väljakutse vormis labor. Sellise labori eesmärk on panna õppur proovile, et nad kasutaksid varem õpitut, et lahendada ülesanne. Selle labori sisu oli väga kaudselt seotud SSRF-ga, mida kasutati ainult ühes kohas pika labori vältel, ja oli samuti ründamisele suunatud.

3.3 Kursuse eesmärk

RangeForce'i kursuste üks eesmärkidest on õpetada veebiarendajatele, kuidas veebirakendusi turvata ja turvariskidega ümber käia. Säärast veebirakenduse kaitsemisele pühendatud laboreid SSRF-i teema jaoks veel olemas ei olnud. Esimesel kokkusaamisel RangeForce'i tiimiga avaldus nendepoolt samasugune tähelepanek.

Algselt oli kaalutluse all teha üks virtuaallabor otsast lõpuni, mille kõik osad oleks loodud lõputöö käigus. Kuid see plaan asendati sellega, et lõputöö käigus luua valmis terve virtuaallaborite kursuse kava, mis kataks ära kõik põhiteemad, mis hõlmavad veebirakenduse turvamist SSRF rünnakute vastu.

3.4 Ülesande skoop

Pandi paika lõputöö ülesande skoop. Lõputöö ülesanne oli luua virtuaallaborite kursuse kava, mida läbinud õppur oskab kaitsda veebirakendusi ja -liideseid SSRF rünnakute eest ning vajadusel informatsiooni juurde otsida. RangeForce'i tiim avaldas huvi just virtuaallaborite kava vastu, sest nende endi hinnangul on turvariski teoreetilise tausta kaardistamine ja siduvasse ning piisavalt kokkuvõtlikku ja sisutihedasse formaati raske panna. See vajab rohket kirjanduse uurimist ja iteratiivset arengut, mis võtab tihti nädalaid kui mitte kuid aega. RangeForce'i meeskonna hinnangul oli kursuse arendamine parem jätta arendusmeeskonna kätte, kuna nad on juba tuttavad arendusprotsessiga ja kasutatavate tehnoloogiatega. Seega jäi virtuaallaborite arendamine ülesande skoobist välja ja skoop laiendus terve virtuaallaborite kursuse kava loomiseni.

4 Töö käik

Esimeses kohtumises RangeForce'i tiimiga pandi paika millist lõpptulemust oodata on, milliseid teemasid käsitleda ning kuidas ülesande täitmisega edasi minna.

Kursuse eesmärk oli anda ülevaade SSRF rünnakute kohta, enimlevinud kaitsemeetodite kohta ning põgusalt üle käia meetoditest, mida ründajad kasutavad, et ebaturvaliselt ülesseatud kaitsemeetoditest mööda saada. Kursuse peamine eesmärk oli anda õppurile n.ö. „blue team" kogemus, ehk rakenduse kaitsemisele suunatud kogemus SSRF kohta.

Selleks, et luua kvaliteetne kursuse kava, oli vaja luua ülevaade olemasolevast kirjandusest SSRF kohta. Vaatamata sellele, et kursuse põhieesmärk oli õpetada rakendust kaitsma, oli SSRF probleemi ja olemasolevate kaitsetavade aru saamiseks vaja ka süveneda SSRF rünnakute sooritamise teooriasse.

Virtuaallaborite kursuse kava loomist ja kirjutamist sooritas lõputöö autor. RangeForce'i meeskond pakkus perioodiliselt tagasisidet arendava kava kohta ning näitematerjale, mille põhjal kava kujundust luua.

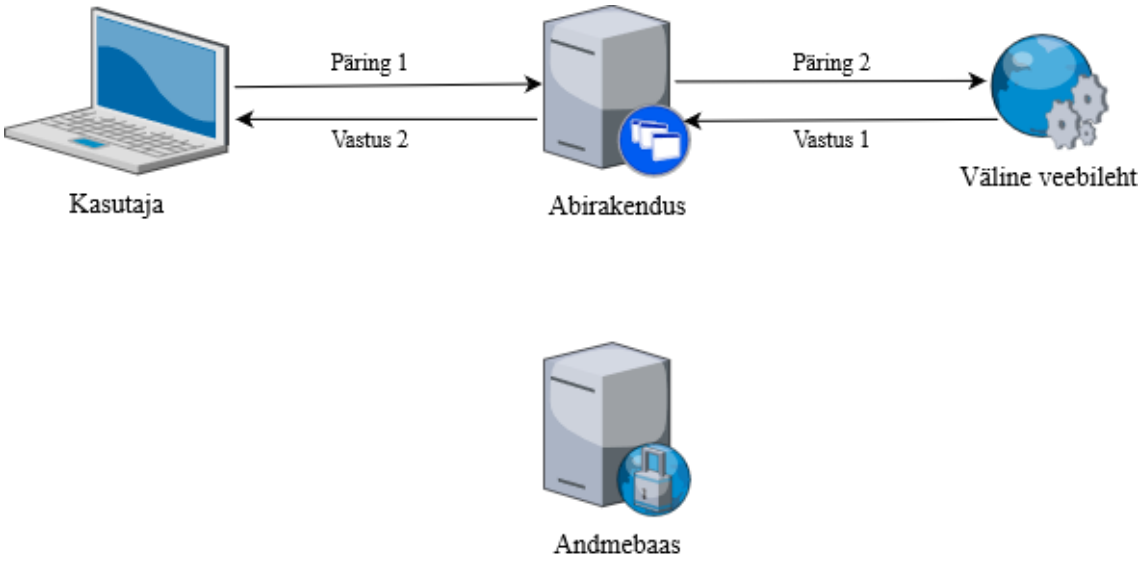
4.1 Abirakendus

Selleks, et proovida järgi erinevaid SSRF ründemeetodeid, loodi lõputöö käigus lihtne veebirakendus, mis on haavatav SSRF rünnakutele. Seda veebirakendust kasutati abitööriistana, et testida erinevaid SSRF rünnakuid.

Rakenduse *back-end* on kirjutatud Javas, Spring raamistikku kasutades, ja *front-end* kasutab Express.js raamistiku. TIOBE organisatsiooni poolt kogutud statistika näitab, et 2022. aasta aprillis oli Java kolmas kõige populaarsem programmeerimiskeel [11]. Lisaks sellele on lõputöö tegija kogemus Java *back-end* süsteemide arendamises võrreldamisi laiem kui teiste keeltega. Express.js valiti sellepärast, et see on kõige populaarsem Node.js raamistik [12]. Kuna rakenduse loogika leidis aset Java *back-endis* ja rakenduse loogika simuleerimine oli abirakenduse ainuke eesmärk, siis *front-endi* üles seadmine pidi olema võimalikult lihtne ja kiire, mida Express.js võimaldas.

Kuna abirakenduse eesmärk oli simuleerida SSRF rünnakute vastu turvamata rakendust, siis programmeerimiskeelte ja raamistike valikut mõjutas rohkelt raamistike ja programmeerimiskeelte populaarsus, mis tagas, et rakenduse funktsionaalsus ei erine suurel määral avalike veebirakendustega.

SSRF rünnakute omapära nõudis, et rakendus pärib kasutajalt funktsiooni täitmise käigus URL-i või internetiprotokolli (IP – ingl *Internet protocol*) aadressi, millele rakendus päringu teeb ja päringu vastuse kasutajale tagasi saadab. Abirakenduse tavafunktsionaalsus on kujutatud joonisel 2. Loodud rakendus päris kasutajalt URL-i (*päring 1*) ja tagastas kõik hüperteksti märgistuskeele (HTML – ingl *HyperText Markup Language*) *img* elemendid ja nende allikad (*vastus 2*), mis kasutaja poolt varustatud URL-i külastades (*päring 2*) lehe HTML koodist leiti (*vastus 1*). Lisaks sellele tagastas rakendus kasutajale kogu päringu tulemuse, mis on siinkohal HTML kood (*vastus 2*). See oli abirakenduse põhifunktsionaalsus. Rakendus ei sooritanud URL-i valideerimist ja oli seega haavatav SSRF rünnakutele.

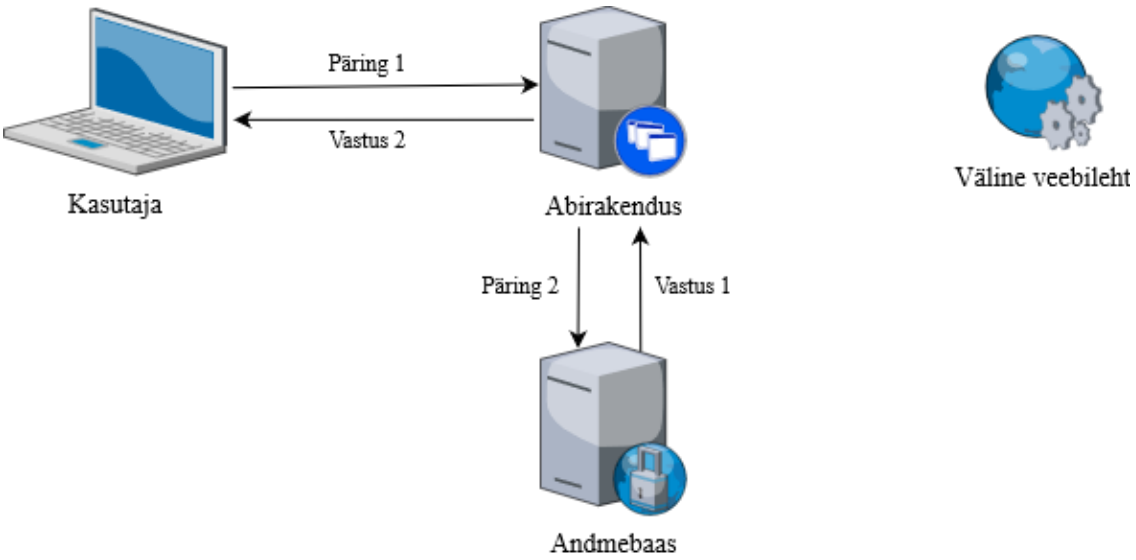


Joonis 2. Abirakenduse tavafunktsionaalsus.

Selleks, et paremini simuleerida SSRF rünnaku sooritamist loodi abirakenduses lihtne andmebaas, mis sisaldas tehisandmeid, mille eesmärk oli matkida tundlikku informatsiooni rakendustes, milleks oli siinkohal kasutajate isikuandmed. Isikuandmed sisaldasid välja mõeldud nimesid, aadresse ja kasutajate õigusi rakenduse skoobis. Rünnaku simuleerimises abirakenduse peal oli eesmärgiks seatud saada kätte loodud tehisandmed läbi rakenduse põhifunktsionaalsuse.

Seda sooritati sääraselt, et kasutaja andis rakendusele URL-i, mis viitab andmebaasi andmeid väljastavale rakendusliidese (API – ingl *Application Programming Interface*) lõpp-punktile. Tihti on säärased lõpp-punktid kasutajatele kättesaamatud ja on mõeldud rakendusesiseseks kasutamiseks, kuid kasutades ära abirakenduse põhifunktsionaalsust on võimalik kasutajal teha päring andmebaasile või muule sisemisele rakendusele läbi abirakenduse enda. Kui rakenduse turvameetmed ei ole piisavalt ranged, siis säärane rakenduselt rakendusele (S2S – ingl *service-to-service*) päring võib olla turvamata [13].

SSRF rünnakut abirakendusele on kujutatud joonisel 3. Abirakenduse näitel annab kasutaja rakendusele URL-i, mis suunab andmebaasi API lõpp-punktile (*päring 1*). Abirakendus teeb päringu API-le (*päring 2*), mis serveerib rakendusesisest andmebaasi. Andmebaasi API lõpp-punkti turvameetmed lubavad ainult lokaalsest võrgust päringud läbi. Kuna abirakendus ja andmebaas on samas lokaalses võrgus, siis andmebaasi API tagastab isikuandmed (*vastus 1*). Abirakendus ei leia ühtegi *img* elementi, kuid tagastab sellegipoolest andmebaasi poolt saadud vastuse (*vastus 2*). Lõpptulemusena on kasutaja saanud kätte isikuandmed rakendusesisesest andmebaasist.



Joonis 3. SSRF rünnak abirakendusele.

Abirakendust kasutati kursuse kava loomise käigus selle eesmärgiga, et kontrollida kasutatud kirjanduses väljatoodud informatsiooni õigsust. Kuna suur osa kava loomise käigus kasutatud materjalidest ei sisaldanud tõestusi, et näidatud ründemeetodid ja kaitsemeetodid töötasid, nagu kirjeldati, siis abirakenduse põhjal oli võimalik infromatsiooniallikaid verifitseerida. See tagas, et kursuse kava ei sisaldanud väärinformatsiooni.

4.2 Kursuse kava arenduskäik

Kursuse kava arendamise ja kirjutamise käigus valmis mitu versiooni. Iga versioon esitati RangeForce'i tiimile, et saada tagasiside ning hinnang kava kohta. Versioonide vahel jäi üldine kava struktuur ja temaatika samaks, kuid sisu muutus rohkelt. Kursuse kava arendamine vajas rohket teooriaga tutvumist, lahtimõtestamist ning koostööd RangeForce'i tiimiga. Arendamine jätkus seni, kuni kursuse kava oli RangeForce'i tiimi hinnangul valmis. Kursuse kava valmis olemine tähendas, et see edastati RangeForce'i arendajatele, kes saavad kava põhjal valmis arendanda moodulid, mis on kavas kirjeldatud. Moodul on kursuse kava kontekstis ühte teemat käsitlev virtuaallabor või teooriat sisaldav õppematerjal.

4.3 Kursuse kava esimene versioon

Kursuse kava esimene versioon kirjeldas ära terve kursuse plaani, mis sisaldas kuute moodulit ja mille läbimine võttis hinnanguliselt kuni kuus tundi aega. Kursuse kava esimene versioon on näidatud Lisas 2. See versioon oli võrreldes järgnevate versioonidega kõige mahukam. Moodulid olid disainitud sääraselt, et kataks teemasid tuntud küberturbe õppematerjalidest ja teadusartikklitest [8], [13]. Moodulite struktuuri loomisel lähtuti varasemast kogemusest RangeForce'i platvormil õppimisest. Kava kujunduses lähtuti RangeForce'i poolt pakutud kursuse kavast, mis käsitles pahavara analüüsimist [14]. Kava moodulid olid järgnevad:

- Sissejuhatus SSRF-i
- Sisendi keelamine SSRF-is
- Sisendi lubamine SSRF-is
- Teekide kasutamine sisendi valideerimises
- Tulemüüride kasutamine
- Validatsioonist mööda pääsemine URL ümbersuunamisega

Moodul nimega „sissejuhatus SSRF-i" oli kursuse kava esimeses versioonis baasinformatsiooni käsitlev moodul. See sisaldas kirjeldust SSRF-i olemuse kohta, selle poolt põhjustatavate ohtude kohta ning näiteid SSRF turvavea ärakasutamise kohta. Peale teooria osaga tutvumist anti õppurile ligipääs veebirakendusele, mis oli haavatav SSRF

rünnakule. Õppur pidi seejärel üles leidma haavatava koha ja seda ära kasutades kätte saama lipu. RangeForce'i virtuaallaborite kontekstis on lipp mingi suvaline numbritest või tähtedest koosnev jada, mille õppur esitab tõestusena, et on mooduli läbinud.

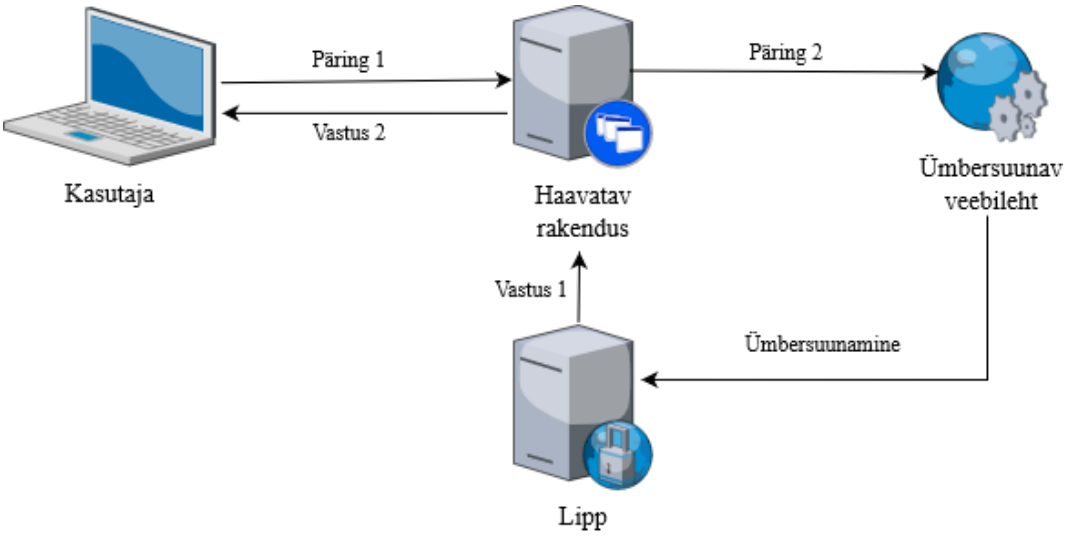
Moodulid „sisendi keelamine SSRF-is" ja „sisendi lubamine SSRF-is" käsitlevad levinumaid sisendi valideerimismeetodeid, mis on toodud välja OWASP Cheat Sheet Series projektis, millega SSRF rünnakute eest kaitstakse. Sisendi lubamine näeb ette, et kui rakenduse funktsionaalsuseks on vaja ainult teatud domeeniaadressi või IP-ga veebiliideseid ja rakendusi. Sel juhul tuleb sisendi valideerimises rakendada ainult kontroll, et kasutaja sisend vastab oodatud mustritele. Kui rakenduse funktsionaalsus näeb ette, et kasutaja võib rakendusele anda ükskõik missuguse sisendi, näiteks abirakenduse funktsionaalsus või webhookid, siis tuleb rakendada sisendi keelamist. Sisendi keelamine tähendab, et rakendus filtreerib kasutaja sisendit, keelates ära IP-d ja URL-id, mille lubamine võib SSRF haavatavuse tekitada. Need on tihti lokaalsed aadressid, sisevõrguaadressid, kui ka teatud ühtse ressursiidentifikaatoriga (URI – ingl *Uniform Resource Identifier*) aadressid. Moodul „siseni keelamine SSRF-is" nägi ette, et õppur tutvub mooduli teooria osaga, seejärel ründab SSRF rünnakule haavatavat veebirakendust ning lõpuks muudab haavatava veebirakenduse koodi, rakendades sisendi keelamist sääraselt, et lihtsamad rünnakud ära hoida. Moodul „sisendi lubamine SSRF-is" algas sellega, et õppur kasutas SSRF ründevõtteid, et sisendeid keelavast validatsioonist, mis oli meelega valesti üles seatud, mööda saada. Seejärel tutvus õppur mooduli teooria osaga ning lõpuks muutis rakenduse koodi, rakendades sisendi lubamise kaitset sääraselt, et moodulis õpitud ründevõtted ei ohustaks enam rakenduse turvalisust.

Moodul nimega „teekide kasutamine sisendi valideerimises" käsitles lühidalt programmeerimiskeeltega ja tarkvaraarenduskomplektidega (SDK – ingl *Software Development Kit*) kaasa tulevate teekide kasutamist, et valideerida URL-e ja IP-sid. Moodul nägi ette, et õppur kasutab eelnevalt õpitud ründemeetodeid, et sisendeid keelava rakenduse validatsioonist mööda pääseda. Peale seda, õppur tutvub mooduli teooria osaga ja seejärel muudab rünnatud rakenduse koodi, kasutades moodulijuhendi poolt ette antud teeke, et rakenduse valideerimist tugevndada.

Moodul „tulemüüride kasutamine" tutvustas õppurile võrgustike eraldamist (ingl *Network Segregation*) kaitsena SSRF rünnakute vastu. Moodul nägi ette, et SSRF rünnakuga haavatava veebirakenduse lokaalses võrgus leidub teine privaatne rakendus,

mis hoiab tundlikku informatsiooni. Moodulis on välja toodud, et haavataval veebirakendusel ei pea olema juurdepääs privaatsele rakendusele, kuid puuduliku võrgustike eraldamise tõttu on see juurdepääs olemas. Õppur pidi tutvuma mooduli teooria osaga, ründama haavatavat rakendust ning seejärel looma turvakesta (SSH – ingl *Secure Shell*) ühenduse privaatse rakendusega ja terminali kaudu üles seadma tulemüüri reeglid, mis piiraks haavatava rakenduse ligipääsu privaatsele rakendusele.

Moodul nimega „validatsioonist mööda pääsemine URL ümbersuunamisega“ käsitles ühte ründemeetodit, millega SSRF rünnakut teostada saab. Mooduli eesmärk oli õppurile õpetada, et kuidas ründed juhtuda võivad, et õppur oskaks paremini rakenduse kaitsemisemiseks kasutatavaid turvameetmeid mõista. SSRF rünnakut URL ümbersuunamisega on kujutatud joonisel 4. Ründemeetod näeb ette, et kasutaja varustab rakendusele URL-i (*päring 1*), mis viitab avalikule veebiaadressile. Kui rakenduse turvameetmed on valesti üles seatud, siis rakendust kontrollib ainult, et kas kasutaja poolt varustatud URL viitab lokaalsele IP aadressile. Kuna veebiaadress on avalik, siis rakendus peab seda valiidseks sisendiks. Seejärel teeb rakendus päringu antud URL-ile (*päring 2*), aga antud veebilehe kood suunab haavatava rakenduse ümber teisele URL-ile, mis on ründaja poolt määratud. Haavatav rakenduse päring lahendub uuele URL-ile (*ümbersuunamine*), mis võib sisaldada tundlikku informatsiooni. Kuna rakendus pidas kasutaja varustatud URL-i lubatavaks, siis tagastab tundliku informatsiooni kasutajale. Mooduli mastaabis oli see faili URI skeemiga aadress, mis viitas tekstifailile, mis omakorda sisaldas lippu. Rakendus loeb faili sisu (*vastus 1*) ning tagastab selle kasutajale (*vastus 2*).



Joonis 4. SSRF rünnak URL ümbersuunamisega.

4.4 Kursuse kava esimese versiooni vead ja kriitika

Kuigi kursuse kava oli teemade valiku ja struktuuri poolest väga sarnane viimase versiooniga, oli nende sisu väga erinev. Kursuse kava esimeses versioonis olid tehtud mitmed vead, mis RangeForce'i tiimiga kokkusaamisel ja arutamisel välja tulid.

Praktilised osad moodulites on kasulikud ja RangeForce'i tiimi hinnangul üks parimaid õppemeetodeid. Need võiks leiduda igas moodulis, mis neid võimaldab, aga kursuse kava esimeses versiooni oli mitmes moodulis rohkem kui üks praktiline osa. Rohkem kui üks praktiline osa või üks pikk praktiline osa ei ole soovitatud moodulisse, mis sisaldab ka teooria õppimist. Lisaks süvendab praktiliste osade liiasus järgmist probleemi.

Moodulite läbimisaeg, milleks alguses hinnati kuni kuus tundi, oli liiga konservatiivselt arvestatud. Moodulite loomisel oli läbimisaja määramisel lähtutud sellest, et kui kaua lõputöö koostaja umbes moodulit lahendaks. Säärane lähenemine oli vale, kuna õppur, kes SSRF kursust võtab, ei ole varem teemaga kokku puutunud ning iga mooduli etapp võtab kauem, kui isikul, kes varasemalt SSRF kohta teab. Kui kursuse kava läbimisaega õppuri vaatepunktist hinnata, võib kursuse läbimine võtta kümme tundi või rohkem. Samuti oli mitme mooduli läbimisaeg üle tunni aja, mis ei olnud RangeForce'i tiimi poolt soovitatav. Kuna RangeForce'i virtuaallaborid on pilvepõhised, kui õppur paneb labori kinni, siis labori virtuaalmasin kustutatakse. Kui õppur paneb labori poole pealt kinni, siis ei ole tal võimalust seda hiljem samast kohast jätkata.

Lisaks oldi kursuse kava loomisel kava eesmärgist kõrvale kaldutud. Iga moodul sisaldas mingil määral SSRF rünnaku sooritamist praktilises osas. Kuigi ründamise metoodikate mõistmine on kasulik rakenduse kaitsmisel, siis kursuse kava esimeses versioonis oli seda mahu poolest üleliia. Paradoksaalselt oli ründamismetoodikaid vähe arutatud. RangeForce'i soovitusel võiks olla üks moodul, mis arutab läbi ja toob näiteid paljudest levinud ründemeetoditest ning ülejäänud moodulid keskenduks kaitsemisele. Samuti soovitas RangeForce'i tiim sissejuhatuse moodulis rohkem arutada SSRF rünnakute ohte ja mõju.

Moodulid „URL-ide lubamine SSRF-is", „Teekide kasutamine URL-ide valideerimises" ja „Validatsioonist mööda pääsemine URL ümbersuunamisega" käsitlesid mooduli vältel mitut erinevat teemat. RangeForce'i tiimi hinnangul on õppuril kergem materjali omandada, kui üks moodul käsitleb ühte konkreetset teemat.

Kõikides moodulites on mainitud veebirakendus, mida õppur ründab või kaitseb, aga kavas ei ole piisavalt lahti selgitatud, et missugune see rakendus on. RangeForce'i kursuste kavades on tavaks eraldi sektsioonis ära kirjeldada n.ö. märklaud, mis oli siinkohal SSRF rünnakute poolt haavatav veebirakendus.

4.5 Kursuse kava teine versioon ja tagasiside

Kursuse kava teise versiooni jaoks oli lähenemine erinev esimesest versioonist. RangeForce'i tiimi soovitusel loodi täies mahus kursuse kava asemel kondikava. Kondikavas oli täpipunktidena kirja pandud teemad, mida peaks SSRF kursus katma ning seejärel oli teemad jaotatud moodulite vahele. Kursuse kava teine versioon on näidatud Lisas 3.

Kursuse kava esimese versiooni ja teise versiooni RangeForce'i tiimile esitamise vahel oli üks nädal aega. Väike ajavahemik oli valitud eesmärgiga, et kõigepealt panna paika parandatud kursuse kondikava, esitada see RangeForce'ile ning edasise tagasiside põhjal luua SSRF kursuse kava parandatud versioon.

Kuna SSRF valdkonnas on väga palju teemasid ja nüansse ning kursuse maht on piiratud, siis oli vaja otsustada, et milliseid teemasid käsitleda ja millised välja jätta. Kondikava loomine aitas erinevad SSRF-ga seotud teemad jaotada kategooriatesse laiali. Kõigepealt jaotati teemad ründamisega või kaitsmisega seotud jaotusesse. Seejärel järjestati teemad,

alustades nendest, mis tuli kindlasti kursuse jooksul ära katta. Järgmisena olid teemad, mis ei ole hädavajalikud ja lõpetati teemadega, mis ei ole väga tähtsad.

Luues parema ülevaate teemadest, mida kursus katta võiks, oli kergem panna moodulid järjekorda, mis kõige loogilisema õppimistee loob. RangeForce'i kursused on disainitud sääraselt, et õppurilt on eeldatud, et ta läbib moodulid järjekorras, kuid see ei ole sunnitud. Teemade parema jaotuse kaudu oli õppuril võimalik saada hea ülevaade SSRF kohta ainult kahte või kolme moodulit läbides ning ülejäänud moodulid aitasid täiendada varasemat materjali. Lisaks olid moodulites viited välistele materjalidele, kust õppur võib omal soovil kursuseväliselt juurde õppida.

Kava esimese ja teise versiooni ning nende tagasiside põhjal pandi paika üldine disainiplaan parandatud kava jaoks. Üks tähtsamaid punkte oli vähem ründamisele keskenduda. Ülesandepüstituses oli määratud, et kursuse põhieesmärk on õpetada kaitsma SSRF vastu. Seega viimasesse versiooni jäeti ründamise kohta üks moodul ja väga lihtne praktiline osa sissejuhatuses. Moodulite järjestus ja sisu oli loogiline ja sujuv, kui neid järjest teha, aga neid oli võimalik ka erinevas järjekorras läbida, kui sissejuhatav moodul on läbitud. Kui moodul käsitles mitut teemat, siis nendel teemadel oli loogiline sidusus.

4.6 Kursuse kava viimane versioon

SSRF kursuse kava parandatud versioon osutus selle viimaseks versiooniks. Kursuse kava viimane versioon on näidatud Lisas 4. Kava oli struktuuri ja teemade poolest võrdlemisi sarnane esimese versiooniga, aga moodulite sisu ja ülesehitus olid väga erinevad. Kasutadest eelnevalt loodud kondikava ja järgides RangeForce'i tiimi tagasisidet ja soovitusi, loodi järgnevad moodulid:

- Sissejuhatus SSRF-i
- Levinud kaitsed SSRF vastu
- SSRF kaitsetest mööda pääsemine
- SSRF kaitse: sisendi lubamine
- SSRF kaitse: sisendi keelamine
- Kaitsemise väljakutse

Võrreldes teiste moodulitega, sarnanes moodul „sissejuhatus SSRF-i“ kõige rohkem esimese kava versiooni samanimelise mooduliga. Mooduli eesmärk oli anda sissejuhatav ülevaade SSRF-i kohta ilma liigsesse detaili laskumata. Selgitati lahti, et mis SSRF on ja kuidas seda kasutatakse, et rünnakuid sooritada. Võrreldes esimese versiooniga oli mooduli teoreetiline osa mahukam kui praktilise osa. Moodul selgitas laiemini lahti SSRF rünnakute ohud ja mõjud ning tõi näiteid päris maailmast, kui SSRF haavatavust oli ära kasutatud ja sellega suurt kahju põhjustatud. Praktiline osa ei sisaldanud seekord SSRF sisenemispunkti identifitseerimist. Õppurile näidati ette sisenemispunkt ning tema ülesanne oli seda ära kasutada, et saada kätte lipp.

Moodul „levinud kaitsed SSRF vastu“ oli sissejuhatav moodul SSRF kaitsemeetodite valdkonda. Moodul käsitles kolme levinud katisemeetodit, mida rakenduse kihis kasutatakse. Kõigepealt selgitati lahtu sisendi lubamine ja sisendi keelamine. Mõlema teema kohta anti lühike sissejuhatav tehniline ülevaade. Seejärel selgitati lahti, et millistes olukordades tuleks kumbagi kaitsemeetodit rakendada. Sisendi keelamise teema juures oli lisapeatükk, mis andis ülevaate selle kasutamisega kaasnevatest ohtudest. Kolmanda kaitsemeetodina käsitleti teekide kasutamist sisendi valideerimises. Moodul selgitas lahti, et millised valideerimisvõimalused vaikimisi kaasa tulevad populaarsemates programmeerimiskeeltes. Seejärel arutati spetsiaalsete SSRF-vastaste teekide kasutamist, toodi nende kohta näiteid ja selgitati, et kuidas neid vajadusel otsida. Viimaseks arutati ka lühidalt võrgustike eraldamist kaitsemeetodina SSRF vastu ning anti viide edasiseks lugemiseks sel teemal. Moodul ei sisaldanud praktilist osa, vaid mooduli lõpus olid valikvastustega küsimused, millele vastates loeti moodul läbituks.

Võrreldes SSRF kursuse kava esimese versiooniga oli kolme mooduli teoreetiline osa pandud viimases versioonis kokku ühte suurde teooriamoodulisse. Moodul „levinud kaitsed SSRF vastu“ sisaldas täielikumat ülevaadet teemadest, mis kava esimeses versioonis olid kaetud moodulites „sisendi keelamine SSRF-is“, „sisendi lubamine SSRF-is“ ja „teekide kasutamine sisendi valideerimises“. Seda saavutati sääraselt, et jäeti ära mahukad praktilised osad, mis kursuse kava esimeses versioonis iga mooduli lõpus olid.

Moodul „SSRF kaitsetest mööda pääsemine“ tutvustas õppurile levinud võtteid, mida SSRF kaitsetest mööda pääsemiseks kasutatakse. Lisaks anti iga võtte kohta tehniline ülevaade ning selgitati, kuidas võtte eest rakendust kaitsda. Kuna kursuse eesmärk on õpetada arendajatele rakenduse kaitsmist, siis suur osa mooduli mahust oli pühendatud

just selle, et kuidas võtete vastu rakendust turvata. Mooduli lõpus oli lühike ja lihtne praktiline osa, kus õppur ründab RangeForce'i poolt loodud rakendust. Praktilise osa eesmärk ei olnud õppurit ründamises proovile panna, vaid kinnitada, et õppur luges teooria materjali läbi.

Kursuse kava viimases versioonis läheneti ründemeetodite õppimisele erinevalt, kui kava esimeses versioonis. Käsitleti rohkemaid ründevõtteid, kattes ära suure osa levinud ründevõtetest. Samas käsitleti teemasid rohkem teooria poolest ja praktilist käsitlust komplekssemad ründemeetodid ei leidnud.

Moodulid nimedega „SSRF kaitse: sisendi lubamine" ja „SSRF kaitse: sisendi keelamine" käsitlesid samanimelisi kaitsemeetodeid. See moodul arutab detailsemalt teemasid, mida moodul „levinud kaitsed SSRF vastu" tutvustas. Kuna õppurile on juba teemat tutvustatud, siis saavad need moodulid täies mahus kaitsemeetodi detailidesse süvitsi laskuda ja lahti mõtestada. Mõlemad moodulid õpetavad õppurile täpselt, et kuidas rakenduses katisemeetodit implementeerida ning milliste nüanssidega tuleb kumbagi kaitsemeetodit kasutades arvestada. Eriti tähtis oli moodulis „SSRF kaitse: sisendi keelamine" arutada, et miks seda kaitsemeetodit peetakse ebaturvalisemaks ning kuidas selle nõrkustega toime tulla. Mõlema mooduli lõpus on praktiline osa, kus õppurile antakse ligipääs veebirakenduse koodile, mille vastu simuleeritakse SSRF rünnakuid. Õppuri ülesanne on rakendada moodulis õpitud kaitsemeetodi sisendi valideerimises, et rakendus turavata.

Võrreldes kursuse kava esimese versiooniga ei sisalda viimase versiooni moodulid enam praktilist osa, kus õppur ründab rakendust ennem rakenduse turvamist. Seda peeti ebavajalikuks, kui ründamise jaoks oli eraldi moodul.

Moodul „kaitsemise väljakutse" on väljakutsemoodul. See tähendab, et mooduli põhieesmärk ei ole õpetada materjali, vaid lasta õppuril omandatud teadmisi proovile panna. See moodul eeldab, et õppur on läbinud teised kursuse moodulid. Võrreldes tavaliste õppemoodulitega, ei anta väljakutsemoodulis õppurile vihjeid, kui nad omal jõul moodulit läbida ei suuda. Kuna kursuse eesmärk on õpetada rakendust kaitsma SSRF rünnakute eest, siis on väljakutsemoodul on loodud sama eesmärgiga. Õppurile antakse ligipääs veebirakendusele, mille vastu simuleeritakse SSRF rünnakuid. Õppurile antakse ka ligipääs veebirakenduse logidele. Selleks, et moodul oleks loetud läbituks, peab

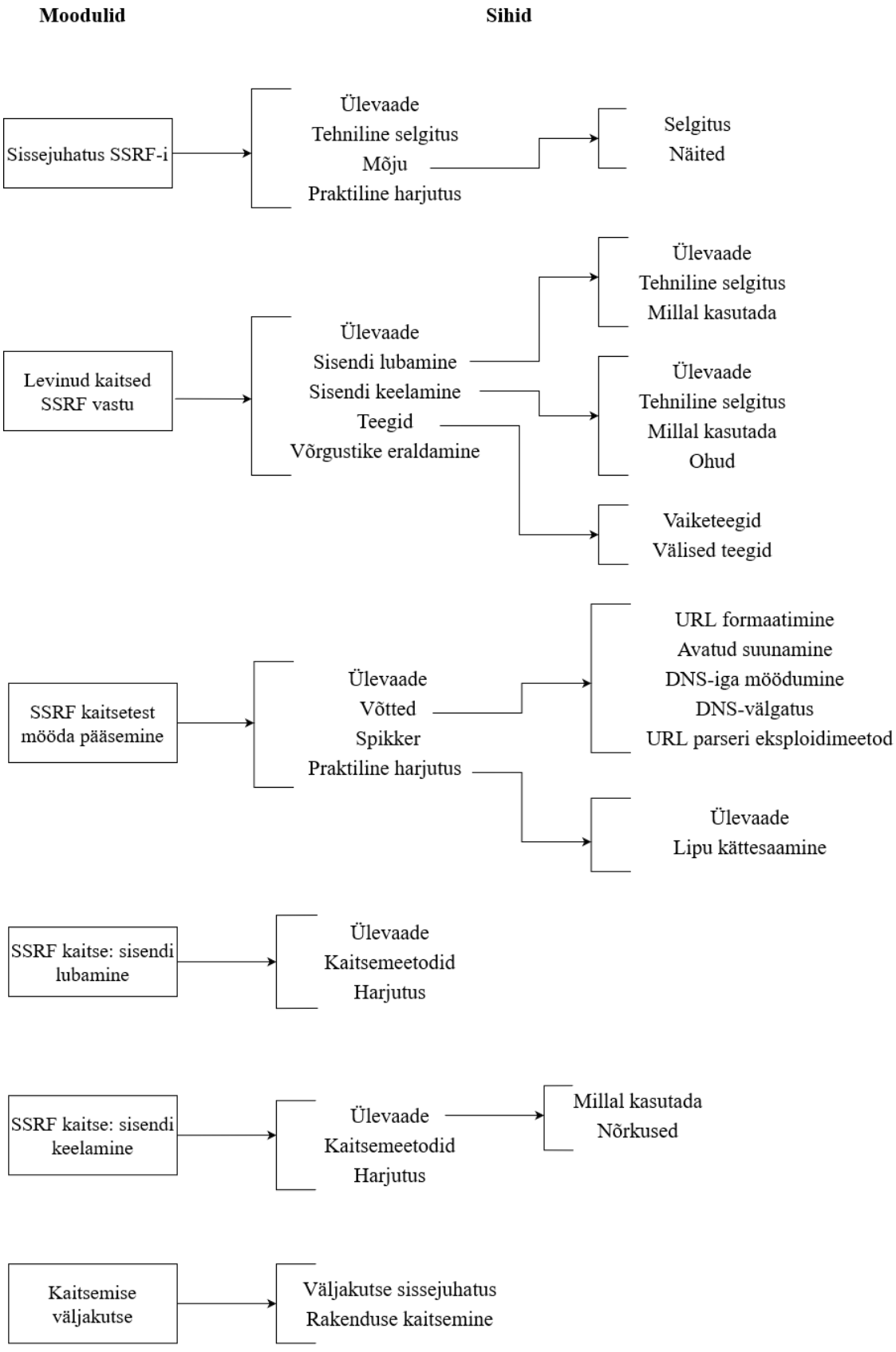
rakendus säilitama tavapärast tööd, aga peatama kõik ründed. Õppur peab tuvastama, et rakenduse funktsionaalsus nõuab, et kaitsemeetodiks peab valima sisendite keelamise. Logide abiga saab õppur tuvastada ründemeetodeid ja nende vastu kaitse luua.

Kursuse kava esimeses versioonis ei olnud väljakutsemoodulit. RangeForce'i meeskonna disainisoovituste järgi võiks kursus lõppeda säärase mooduliga. Väljakutsemooduli läbimine näitab, et õppur on omandanud kursuses õpetatava.

Moodulites oli ka õppurile pakutud viiteid välistele materjalidele, kust õppur saab soovi korral lisaks juurde õppida.

Moodulite järjestus kursuse ülesehituses on hoolikalt valitud. Iga moodul täiendab eelnevas moodulis õpitut. Moodulid „sissejuhatus SSRF-i" ja „levinud kaitsed SSRF vastu" tulevad ennem mooduleid „SSRF kaitse: sisendi lubamine" ja „SSRF kaitse: sisendi keelamine", sest esimesed moodulid annavad õppurile baasteadmised, mis on vajalikud viimaste moodulite mõistmiseks. Samuti on moodul „SSRF kaitsetest mööda pääsemine" kursuse esimeses pooles, et õppuril oleks põhjalik ülevaade ründemeetoditest ennem, kui kaitsemeetodeid süvitsi õpib. Sääraselt on õppuril kergem mõista kaitsmise nüansse ja iseloomu. Moodul „SSRF kaitse: sisendi lubamine" on ennem moodulit „SSRF kaitse: sisendi keelamine". Kuigi mõlemad kaitsemeetodid on põhimõtte poolest sarnased, siis sisendi keelamine on võrreldamisi keerukam kaitsemeetod, kui seda õigesti rakendada. Õpetades sisendi lubamise teemat varem on sisendi keelamise moodulis õppuri jaoks vähem uut informatsiooni. Seega on uue informatsiooni kogus kahe mooduli peale võrdsemini ära jaotatud. Väljakutsemoodul on kõige viimane, sest vajab kõikides eelnevates moodulites õptitut.

Moodulite järjestus ja moodulites sisalduvad alamülesanded ehk sihid on välja toodud Joonisel 5. Moodulite ja mooduli sihtide täpsem kirjeldus on välja toodud Lisas 4.

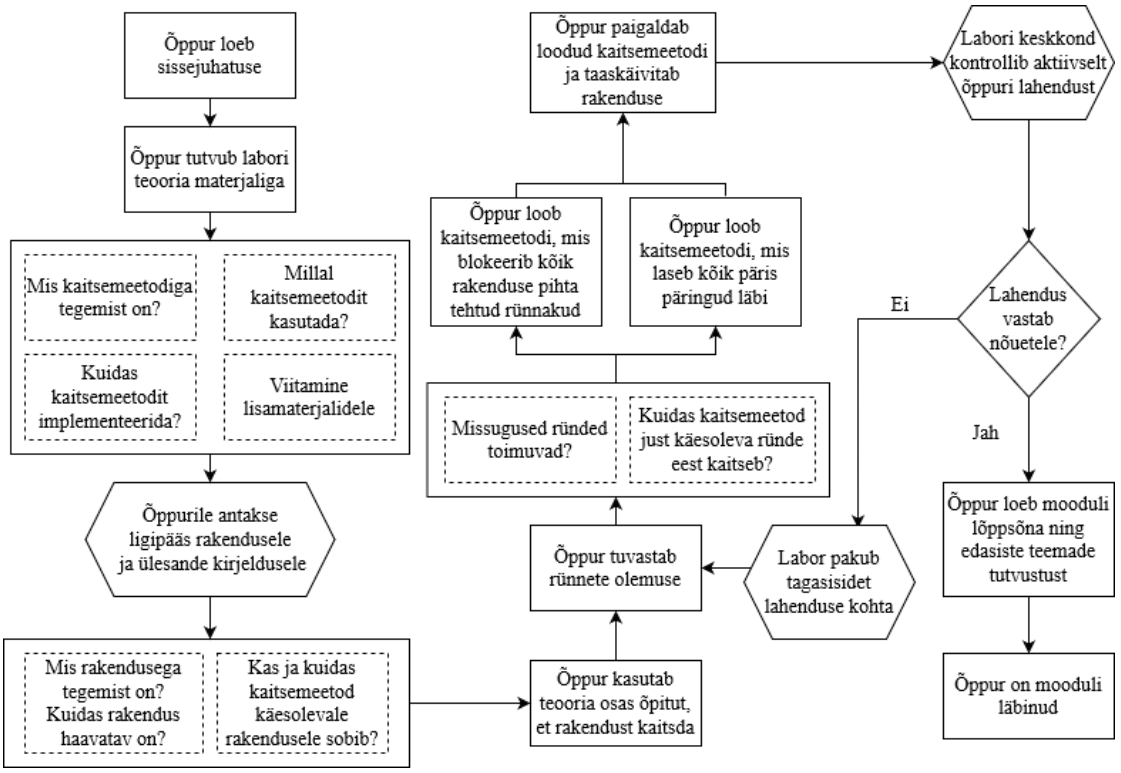


Joonis 5. Kursuse moodulite struktuur.

4.7 Kava loomisprotsess ja tähtsamad otsustuspunktid

Eelnevates peatükkides olid välja toodud otsustuskohad kava struktuuris ja teemavalikus. Lisaks toodi välja platvormist tulenevad piirangud ja RangeForce'i meeskonna nõuded ja soovitused kava loomisel. Sellegipoolest oli kava loomises rohkem nüansse ja mõttekohti.

Iga mooduli loomine oli mitmeastmeline protsess. Esimene samm oli visualiseerida mooduli sisu ja lahendamisprotsessi etapid. Visualiseerimist teostati paberi peale vooskeeme joonistades ning peas labori lahendamist läbi mängides. Joonisel 6 on toodud näide rakenduse kaitsemist käsitleva mooduli vooskeemist. Pideva joonega ristkülikutes on välja toodud õppuri tegevused mooduli etappides. Kuusnurksetes kastides on automaatsed tegevused, mida labor peab funktsionaalselt võimaldama. Katkendliku joonega kastides on välja toodud näited küsimustest, mis võivad õppuril igas etapis tekkida ning millele moodul peab vastused andma. Mida spetsiifilisem oli mooduli teema, seda rohkemate detailidega pidi arvestama. Selles sammus tuli kasuks varasem kogemus RangeForce'i platvormil laborite lahendamisega. Samas, mooduli sisu visualiseerimises pidi arvestama õppuri vaatepunktiga. Õppuril on tõenäoliselt minimaalsed varasemad teadmised käsitletava teema kohta. SSRF valdkonnaga rohkem kursis olles ning varasema õpetamiskogemuse puudumisel oli kerge kogemata mõni väike detail välja jätta, mis võis sellegipoolest õppuri õppimiskogemuse jaoks hädavajalik olla. Lisaks tuli lahenduskäigus vältida etappe, mis vajavad kõrvalisi ja teemaväliseid teadmisi. Kui sellist etappi ei saanud vältida, siis tuli kindlasti õppurile selgitada, et mida täpselt teha on vaja. Üks näide sellisest etapist on Lisas 4 viimases moodulis, nimega *„Defence Challenge"*, kus õppur peab terminali kaudu logisid lugema. Ideaalselt selliseid etappe ei olnud, aga mõne mooduli sisu nõudis nende sisaldamist.



Joonis 6. Mooduli vooskeemi näide

Järgmisena on vaja kirjutada detailne mooduli kirjeldus arendaja jaoks. Detailne kirjeldus peab eelnevas sammus visualiseeritud mooduli võimalikult täpselt ära kirjeldama. Detailses kirjelduses on hoolikas sõnade valik väga tähtis. Kirjutades tuleb arvestada, et halvasti valitud fraase võib arendaja valesti tõlgendada ning mooduli sisu tähendus võib muutuda. Üks viis fraaside tähenduse selgitamiseks oli viitamine materjalile, mida kasutati sisu loomises. Arendaja saab materjalidega tutvuda ja täpselt mõista, et mida on vaja luua. Üks näide selle võtte kasutamisest on Lisas 4 esimeses moodulis, nimega *„Introduction to SSRF“*, kus selle asemel, et pikka selgitust käsitsi välja kirjutada ja riskida mitut moodi mõistetava kirjelduse loomist, on viidatud artiklile, mis kirjeldab täpselt, et kuidas SSRF haavatavusi on võimalik ära kasutada.

Lisaks eelnevalt välja toodule on moodulite loomises vaja arvestada märklauaga. Märklaud tähendab RangeForce'i moodulite kontekstis objekti, millele mooduli sisu on suunatud. Käesoleva teema jaoks on märklaud veebirakendus, kuid muude teemade mastaabis võib see erinev olla. Moodulite loomisel peab arvestama sellega, et märklaud on ideaalselt kursusel läbivalt sama ning ainult väikesed detailid muutuvad. Kui märklaud muutub ilma põhjuseta, siis võib õppeprotsess õppuri jaoks segasem olla ning tekitada valearvamust, et märklaua muutumine on seotud käsitletava alamteemaga isegi kui see ei ole. Seega on moodulite loomisel vaja läheneda eesmärgiga, et märklaud saaks igas

moodulis sama olla. Lõputöö käigus loodud kursuse kava viimases versioonis oli praktilise osaga moodulites märklaud sama veebirakendus. Kursus oli disainitud ümber veebirakenduse, mis võttis kasutajalt sisendiks URL-i ja tegi SSR-i. Näiteks on Lisa 4 lõpus välja toodud märklaua kirjeldus, millele arendaja toetuda saab.

Lõppkokkuvõttes kõige ajamahukam osa lõputöö tegemisest oli õppimine. Kursuse kava loomise käigus puututi kokku mitmete teemade ja valdkondadega, millega autoril puudus varasem kogemus. Üks oli SSRF, mille mõistmine ja analüüsimine oli võrdlemisi suur osa lõputööst. Samas mainimata ei saa jääda, et RangeForce'i platvormi eesmärk on pakkuda õpetuslike materjale. Õpetamine vajab ülevaadet käsitlevast teemast ja võimet näha seda teemat õppuri vaatepunktist. Kuna lõputöö autoril puudus varasem õpetamise kogemus, siis kava moodulite visualiseerimise etapp võttis selle võrra kauem aega. Samuti puudus lõputöö autoril varasem kogemus RangeForce'i meeskonna tavade ja ootustega. Kursuse kava dokumendi struktuur põhineb RangeForce'i poolt pakutud näidisele [14], aga näidiseks toodud kursuse teema, pahavara analüüsimine, erines SSRF kaitsemisest suurel määral ning sel põhjusel oli SSRF kursuse kava dokumendi struktuuris vaja teha muudatusi, mille tegemine oli ajamahukas protsess. Lõputöö autori ainuke varasem kogemus RangeForce'i platvormiga oli õppurina laborite lahendamine, aga lahendatud laborid olid võrdlemisi vanad ning RangeForce'i arendusstandardid ja ootused olid üle aja muutunud. Varasema õpetamise kogemuse ja RangeForce'i kursuste loomise kogemuse puudumine oli suurim põhjus, miks kava esimene versioon ümber kirjutati. Lõpliku kursuse kava loomine nägi ette eelnevalt tehtud vigade ja ebatäpsuste analüüsimist ning nendest õppimist.

4.8 Kursuse kava viimase versiooni tagasiside

Tagasiside kursuse kava viimase versiooni kohta oli positiivne. Esitatud viimasel versioonil olid mõned väikesed vead, nagu ebatäpne sõnastus mõne mooduli kirjelduses või kava vormistuse täpsustamine.

Peale viimaste vigade parandamist oli kursuse kava RangeForce'i hinnangul hästi loodud ja valmis arendamisesse minemiseks. Arendamisesse minemine tähendab, et kursuse kava on valmis ning RangeForce'i tiimi arendajad saavad mooduleid selle põhjal looma hakata.

5 Kokkuvõte

Püstitatud ülesanne oli luua RangeForce'i platvormi jaoks sobilik virtuaallaborite kursuse kava, mis käsitleb veebirakenduste kaitsmist SSRF rünnakute eest. Ülesande lõpptulemus pidi olema kava, mis on valmis minema arendusse RangeForce'i arendustiimi poolt ja mis on RangeForce'i meeskonna hinnangul piisavalt teaberikas, et seda läbinud arendaja oskab veebirakendust SSRF rünnakute eest kaitsda.

Eesmärgi saavutamiseks kasutati OWASPi Cheat Sheet Series ja HackTricks materjale, mida omakorda kontrolliti abirakendusega. Tehti koostööd RangeForce'i meeskonnaga, kelle tagasiside põhjal kursuse kava puudujääke parandati ja iteratiivselt välja töödeldi.

Ülesanne loeti täidetuks, kui lõputöö käigus loodud kursuse kava ei vajanud muudetusi RangeForce'i meeskonna poolt, et arendusmeeskond seda arendama saaks hakata. SSRF kursuse kava viimase versiooni koos parandustega esitamisel kinnitas RangeForce, et loodud kava vastab ootustele.

Loodud kursuse kava ja selles sisalduvad moodulid saavad olla aluseks edasiseks arenguks RangeForce'i platvormil. Sissejuhatavaid ja ründamisele keskenduvaid mooduleid on võimalik kasutada ka muudes kursustes. Kuna RangeForce'i platvorm pakub ka võimalusi õppida küberründamist, siis on võimalik olemasolevaid mooduleid ja moodulites sisalduvaid osi kasutada ka ründamisele suunitletud SSRF kursuses.

Kasutatud kirjandus

[1] RangeForce, „RangeForce About,“ [Võrgumaterjal]. Available: https://www.rangeforce.com/about-rangeforce. [Kasutatud 21 Aprill 2022].

[2] OWASP, „OWASP Top Ten Project,“ 2021. [Võrgumaterjal]. Available: https://owasp.org/www-project-top-ten. [Kasutatud 21 Aprill 2022].

[3] Tallinna Tehnikaülikool, „Tallinna Tehnikaülikooli õppeinfosüsteem,“ [Võrgumaterjal]. Available: https://ois2.ttu.ee. [Kasutatud 21 Aprill 2022].

[4] G. Pellegrino, O. Catakoglu, D. Balzarotti ja C. Rossow, „Uses and Abuses of Server-Side Requests,“ 2016. [Võrgumaterjal]. Available: https://publications.cispa.saarland/1053/1/SSR_raid2016.pdf. [Kasutatud 21 Aprill 2022].

[5] V. Li, „Exploiting SSRFs,“ Medium, 3 Juuni 2019. [Võrgumaterjal]. Available: https://vickieli.medium.com/exploiting-ssrfs-b3a29dd7437. [Kasutatud 21 Aprill 2022].

[6] OWASP, „A10 Server Side Request Forgery (SSRF),“ [Võrgumaterjal]. Available: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/#factors. [Kasutatud 21 Aprill 2022].

[7] OWASP, „OWASP Top 10 - 2021,“ [Võrgumaterjal]. Available: https://owasp.org/Top10. [Kasutatud 21 Aprill 2022].

[8] OWASP, „OWASP Cheat Sheet Series,“ [Võrgumaterjal]. Available: https://cheatsheetseries.owasp.org. [Kasutatud 21 Aprill 2022].

[9] C. Polop, „HackTricks,“ [Võrgumaterjal]. Available: https://book.hacktricks.xyz. [Kasutatud 21 Aprill 2022].

[10] RangeForce, „How TalTech Redesigned its Cybersecurity Curriculum with RangeForce,“ [Võrgumaterjal]. Available: https://go.rangeforce.com/hubfs/Website%20Collateral/Case%20Studies%20(updated%2011.2021)/RangeForce%20Barclays%20Case%20Study%2011.2021.pdf. [Kasutatud 15 Mai 2022].

[11] TIOBE, „TIOBE Index for April 2022,“ 2022. [Võrgumaterjal]. Available: https://www.tiobe.com/tiobe-index. [Kasutatud 21 Aprill 2022].

[12] Node.js Foundation, „2018 Node.js User Survey Report,“ [Võrgumaterjal]. Available: https://nodejs.org/en/user-survey-report. [Kasutatud 21 Aprill 2022].

[13] B. Jabiyev, O. Mirzaei, A. Kharraz ja E. Kirda, „Preventing Server-Side Request Forgery Attacks,“ Märts 2021. [Võrgumaterjal]. Available: https://seclab.nu/static/publications/sac21-prevent-ssrf.pdf. [Kasutatud 21 Aprill 2022].

[14] RangeForce, „Malware Analysis Statically Examining Malware“.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Hans Erik

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „serveripoolse päringu võltsimise turvariski virtuaallaborite kursuse kava“, mille juhendaja on Sten Mäses.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

28.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Kursuse kava esimene versioon

Intro to SSRF

TODO: *Write longer description of what should be talked about in this module.*

Description:
This is the first module of the SSRF course. This module will contain a completely vulnerable web application (target). The goal of this module is to teach the basics of SSRF. After reading the introduction to SSRF, the student's goal is to discover the SSRF vulnerability in the web app and exploit it. The vulnerability should be as similar to the examples given in the introduction as possible to make it easier to recognize.

Activity:
Student will read the introductory explanation of SSRF and answer a few simple questions. Student is then directed to the web application. Student will use what they have learned to identify an SSRF vulnerability. The student will then use the vulnerability to retrieve a flag from the target's file system.

Keywords:
Learning outcomes:

- Knowledge of what an SSRF vulnerability is.
- Knowledge of the dangers of an SSRF vulnerability.
- Ability to identify an SSRF vulnerability.

Duration: 1800 – 2700
Difficulty: Foundational
Outline:

- Overview
    - Introduction to SSRF
    - Technical Overview
    - Questions
- Detection
    - Find the Vulnerability
- Exploit the Service
    - Retrieve the Flag

URL Deny Listing in SSRF

Description:
This is the second module of the SSRF course. This module will contain the same target from the first module (Intro to SSRF). The student's goal is to repeat the process of identifying the SSRF vulnerability and exploiting it from the previous module. They will retrieve a flag from the target's localhost. This flag is stored in an API endpoint that is not directly accessible. The URL to the endpoint should be given to the student. To simplify this process, this module should contain the materials from the first module.
After the student has repeated these steps. They are given access to the target's codebase. They are given an introduction to deny listing as a defence against SSRF. The student will then implement a deny list in input validation to prevent basic SSRF attacks while maintaining the usual functionality of the target. In the introduction to deny listing, it is **very important** to mention that this is not a secure solution to SSRF and can be circumvented with more sophisticated attack methods.

Activity:
Student will read the Deny Listing overview. Student is directed to the web application. The student will retrieve the flag using what they learned in the previous module. Give student access to the web application code. Instruct the student on how to fix the vulnerability using a deny list in input validation. The student will then implement a deny list.
Learning outcomes:

- Ability to identify an SSRF vulnerability.
- Knowledge of the deny listing solution to SSRF.
- Knowledge of the shortcomings and insecurity of the deny listing solution.

Duration: 2700
Difficulty: Foundational
Outline:

- Overview
    - Introduction to Deny Listing
    - Technical Overview
- Exploit the Service
    - Retrieve the Flag
- Fixing the Vulnerability
    - Apply the Fix
    - Deploy the Fix

URL Allow Listing in SSRF

TODO: *Is this module too large? Circumventing deny list here makes sense, but might be too much for one module. Clarify that circumventing the deny list should not be made too difficult, as it is a secondary part of the module? This is supposed to be blue/purple team.*

Description:
This is the third module in the SSRF course. This module will contain a web application that is using a deny list to validate URL inputs. Explain to the student why deny listing is a bad solution to SSRF vulnerabilities. Introduce to the student the multiple ways to access localhost[1] that can circumvent a deny list. The student's goal is to circumvent the deny list by trying the different methods. The student will capture a flag in an API endpoint, the path to which is given. It is up to the student to figure out the domain part.
The idea here would be to make the deny list include the first several options outlined in the reference, so that the student is encouraged try different ones.
The next goal would be to replace the deny list input validation with an allow list. Give an introduction to what an allow list is. Explain the differences between a deny list and allow list, both technical differences and security differences.
TODO: *Once target is decided, describe an application-specific allow list to implement.*

Activity:
Student is directed to the web application. Instruct the student on how to bypass the deny list and access localhost by using different methods. Instruct student to capture the flag in the API endpoint given to them. After that is done, give student access to the web application code. Instruct the student how to fix the vulnerability implementing an allow list.
TODO: *Same as above.*
Learning outcomes:

- Knowledge about the allow listing solution to SSRF.
- Knowledge of the shortcomings of the deny listing solution.
- Knowledge of the differences between allow listing and deny listing.
- Ability to implement allow listing in URL validation.

Duration: 3600
Difficulty: Intermediate
Outline:

- Circumventing a Deny List
    - Introduction
    - Methods of Accessing Localhost
    - Capture the Flag
- Overview of Allow Listing
    - Introduction to Allow Listing
    - Technical Overview
- Fix the Vulnerability
    - Apply the fix
    - Deploy the fix

[1] https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery/url-format-bypass

Using Libraries to Validate URLs in SSRF

Description:
This is the fourth module in the SSRF course. This module will contain a web application that is using deny listing to validate URL inputs. The student's initial goal is to circumvent the deny list using alternative URLs (as outlined in the previous module) to retrieve a flag. The URL of the API endpoint containing the flag should be given to the student. The student will then supplement the deny list by using a library such as, but not limited to, ipaddress[1] (python) or InetAddress[2] (java) or ip-address[3] (javascript) to verify whether a URL points to a local IP and block it.
Include the same materials from the previous module on how to circumvent a deny list. Materials should also list libraries for validating an URL in the most popular back-end languages, such as, but not limited to JavaScript, Python, PHP, and Java, Ruby.

Activity:
Student is directed to the web application. Instruct the student on different ways to circumvent a basic deny list. Instruct student to use one of these methods to commit to capture the flag. After that is done, give student access to the web application code. Instruct the student how to fix the vulnerability by checking whether a URL directs to a local IP.
Learning outcomes:

- Repeat of the shortcomings of deny listing to ensure that the student understands them.
- Knowledge of different libraries that can be used to verify URLs.

Duration: 2700
Difficulty: Fundamental
Outline:

- Circumventing a Deny List
    - Introduction to Module
    - Capture the Flag
- Overview of URL Validation Libraries
    - Overview
    - Technical Overview (of the library used in this module)
- Fix the Vulnerability
    - Apply the Fix
    - Deploy the Fix

[1] https://docs.python.org/3/library/ipaddress.html

[2] https://docs.oracle.com/javase/7/docs/api/java/net/InetAddress.html and https://commons.apache.org/proper/commons-validator/apidocs/org/apache/commons/validator/routines/InetAddressValidator.html

[3] https://www.npmjs.com/package/ip-address

Firewalls in SSRF

Description:
This module will contain two separate applications (the web application aka target and a private application) hosted on the same local network that are able to make requests to each other. The private application contains sensitive information and can only be accessed by applications on the same local network.
For the purposes of this module, even though the virtual lab and all applications therein are hosted on the same local network, firewall rules should be implemented to "simulate" the student's system not being on the same local network. To clarify, the student should have access to the web application, but until they retrieve the flag by exploiting the SSRF vulnerability, they should not have access to the private application.
TODO: *Clarify this part? Should the student know of this "simulation"? Should this be restructured?*
The other web application is the target outline above, vulnerable to SSRF. In the overview of this module, it should be explained to the student that this web application is not supposed to have access to the private application, and this is the misconfiguration they will exploit and afterwards fix.
The student's goal is to exploit the targets SSRF vulnerability to access information in the private service to retrieve a flag. The URL of the API endpoint should be given to the student. They then fix this vulnerability by adding a firewall rule to prevent the targets access to the private application. After that is successfully done, the module will be considered complete.

Activity:
Student will read the "Firewalls in SSRF" overview and acquaint themselves with the network schema. Student is directed to the web application. Student is to identify the SSRF vulnerability. Student is to access sensitive information (retrieve a flag). After that is done, give student access to both applications and instruct the student how to fix the vulnerability by adding a firewall rule on the server the private application is hosted.
Learning outcomes:

- Awareness of security threats related to firewall misconfiguration.
- Knowledge of how an SSRF vulnerability can be a threat to more than the host.
- Ability to implement a firewall restriction.

Duration: 3600
Difficulty: Intermediate
Outline:

- Overview
    - Introduction to Firewalls
    - Technical Overview
- Practical Example
    - Introduction to Target
    - Network Schema Explanation
    - Capture the Flag
- Creating the Firewall
    - Access the Private Application
    - Create the Firewall Rule

SSRF Bypass Using Open Redirection

Description:
The objective of this module is to discover and exploit an SSRF vulnerability using open redirection and then learn how to fix the vulnerability. This module contains a web application (target) that is protected by a thorough deny list. It should be protected from all of the different ways of URL format bypass outlined here[1]. The objective of this module is to not allow a direct SSRF attack however, still allow the student to supply a URL that uses open redirection to access localhost. As such, URL validation should take place before it is requested.
The student will read an introduction to the idea of open redirection and a technical overview of how to accomplish it. The student is instructed on how to set up a server with python, which redirects to a desired URL, for example as outlined here[2]. The student will use this to retrieve a flag from an URL given to them.
After the student has retrieved the flag, they are given an overview of how to defend against an SSRF bypass using open redirection. The student is given access to the web applications code and is instructed to validate a response after it is made. It is important to mention that an allow list could also be extended this way and should be the preferred method, as a deny list is not a safe solution.
Activity:
The student is given an overview and technical explanation about open redirection. The student is then given access to the web application and instructed to retrieve the flag by bypassing URL validation using open redirection. After the student has retrieved the flag, they are given a technical explanation on how to fix the vulnerability. The student is given access to the web application's code and instructed to fix the vulnerability.
Learning outcomes:

- Knowledge of what open redirection is.
- Knowledge of how an SSRF vulnerability can be exploited using open redirection.
- Ability to defend against an SSRF bypass using open redirection.

Duration: 3600
Difficulty: Intermediate
Outline:

- Overview
    - Introduction to Open Redirection
    - Technical Overview
- Bypassing a Deny List
    - Retrieve the Flag
- Fixing the Vulnerability
    - Defence Overview
    - Apply the Fix
    - Deploy the Fix

[1] https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery/url-format-bypass

[2] https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery/url-format-bypass#bypass-via-redirect

Lisa 3 – Kursuse kava teine versioon

Points to Cover

- Red Team
    - Identify Vulnerability
    - URL Formatting
    - Open Redirection
    - Practical
    - Advanced Topics (Maybe shouldn't be included):
        - Domain Confusion
        - URL Mapping
        - DNS Rebinding
        - Blind SSRF
- Blue Team
    - Allow Listing (+ Deny Listing)
    - Libraries
    - Practical
    - Advanced Topics (Maybe shouldn't be included):
        - Firewalls
        - Redirection Protection
- General
    - SSRF Overview (What is it?)
    - Impact of SSRF

Labs:

1. Intro to SSRF
    a. Overview
    b. Technical Explanation + Identification
    c. Impact
    d. Bypasses (mentioned by name)
    e. Practical Part (Very simple, make request to API, no techniques)
2. Common SSRF Defences [Blue Team]
    a. Overview
    b. Deny Listing
    c. Allow Listing
    d. Libraries
3. Bypassing SSRF Defences [Red Team]
    a. Overview
    b. Techniques
        i. IP formatting
        ii. URL parser quirks
        iii. DNS pinning
        iv. Open Redirection
    c. Cheatsheet
    d. Exercise
4. SSRF Defence: Deny Listing [Blue Team]
5. SSRF Defence: Allow Listing [Blue Team]
6. Blue Team Challenge

Lisa 4 – Kursuse kava viimane versioon

Introduction to SSRF

Description:
This is the first module of the SSRF course. The objective of this module is to give a short and simple introduction to SSRF without going into excessive detail. It will begin with a general description of what SSRF means. This will be followed by a more technical explanation. After the technical explanation, give examples of how an SSRF vulnerability can be exploited, for example as outlined here[1].
The last part of the introduction is a short and simple practical exercise. The student is directed to the target. The student is shown and explained the SSRF vulnerable part of the web app. They are then instructed to make a request to an API endpoint given to them to retrieve a flag. After that is done, the module is completed.
Activity:
Student will read the SSRF overview and description. The student will then read the technical explanation about SSRF. After that, the student will read about the impact of SSRF. When the student has read the theoretical part and start the practical exercise, they are directed to the target. The student is directed to the vulnerability and given the API endpoint that returns the flag. The student will then exploit the SSRF vulnerability to retrieve the flag.
Learning outcomes:

- Knowledge of what an SSRF vulnerability is.
- Knowledge of the dangers and impact of an SSRF vulnerability.
- Ability to identify an SSRF vulnerability.
- Knowledge about the fundamentals of how an SSRF exploit is performed.

Duration: 1800 – 2700
Difficulty: Foundational
Outline:

- Overview
- Technical Explanation
- Impact
    - Explanation
    - Example
- Practical Exercise

[1] https://vickieli.medium.com/exploiting-ssrfs-b3a29dd7437

Common SSRF Defences

Description:
This module will discuss commonly used methods of input validation to protect from SSRF. Before the student proceeds to the defence methods, they are given a quick reminder of what SSRF is. Then explain the two basic cases in which SSRF can happen. First being an application that makes requests only to a few trusted URLs and second being an application that can send requests to any URL.
After that overview, we discuss the first case. In such a scenario we can use the allow listing approach. Give a layman's explanation of what allow listing means in the context of SSRF. Then give a technical explanation with a few short code examples. Explain to the student that allow listing is the preferred form of URL validation and should be used in any cases where it can be used.
After discussing the first case, we discuss the second case. Give a layman's explanation of what deny listing means in the context of SSRF. Bring out the differences between allow listing and deny listing, as on the surface, they are rather similar. Give a technical explanation with some code examples. Explain to the student that deny listing should only be used in the case when allow listing cannot be used. This is because deny listing is vulnerable to a far larger number of bypasses than allow listing and maintaining a secure deny list is extremely difficult, if not impossible. A deny list alone is never enough to properly protect against SSRF.
After discussing these two basic cases, it is important to mention that allow/deny listing alone might not protect against more sophisticated SSRF attacks. Different libraries can be used to supplement these methods to provide a more secure solution. First mention that some common web development languages and frameworks have built-in libraries that can perform basic checks on URLs and IPs. Such as ipaddress in python, java.net in java. These can be used to create own algorithms to protect against SSRF, however, they do not provide much use out-of-the-box. For better solutions there are third-party libraries that require little configuration and are designed to specifically protect against SSRF, such as Advocate[1] for python or paranoid-request[2] for node. Mention that it is wise to look for these libraries when protecting against SSRF and keep them up to date.
The above-mentioned methods are input validation on the application layer. It should also be mentioned that proper network segregation and firewall setup is vital to protecting against SSRF. A publicly faced application should not be able to perform calls to all other internal applications. It should only have access to applications it needs for its function. More about it here[3].
In the end of this theory module, there should be a few simple multiple-choice questions to confirm that the student read the material. Here are a few examples: When can allow listing be used? What is the difference between deny listing and allow listing? Why is deny listing considered less secure than allow listing?

---

[1] https://github.com/JordanMilne/Advocate

[2] https://github.com/uber-archive/paranoid-request

[3]https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html#network-layer

Activity:
The student will read the material in the specified order. After reading the material the student will answer a few simple questions.
Learning outcomes:

- Knowledge of the deny listing and allow listing defences against SSRF.
- Knowledge of the different cases where deny listing or allow listing can be used.
- Knowledge about the dangers of the deny listing solution and why allow listing is preferred.
- Knowledge about the use of libraries in protecting against SSRF.

Duration: 1800 – 2700
Difficulty: Foundational
Outline:

- Overview
- Allow Listing
    - Overview
    - Technical Explanation
    - When to Use Allow Listing
- Deny Listing
    - Overview of Deny Listing
    - When to Use Deny Listing
    - Dangers of Deny Listing
- Libraries
    - Default Libraries
    - Third-Party Libraries
- Network Segregation

Bypassing SSRF Defences

Description:
This module will discuss the different ways of bypassing SSRF defences and will end with a simple exercise utilizing one of them. To start off, in the overview, outline the common defences used (deny listing, allow listing) to refresh the student's memory and explain, that there are a few different common techniques to bypass them. Name all the different techniques (outlined below). Explain that quite a few of these techniques threaten both defences and exploit faulty configuration on the developer's part.
URL Formatting is one of the simplest to explain, but also a very effective bypass technique. URL Formatting is an umbrella term for multiple different types of formatting bypasses that can threaten both deny listing and allow listing solutions. These include IP formatting, different ways to access private IPs. Domain parsing, different ways to write an URL that may bypass a deny list. Domain confusion, including a portion of a different domain in the request in order to bypass an allow list. Examples for these can be found here[1].
Open Redirection is a technique where the attacker sets up a public server which redirects to a private one, such as a private IP or file://. The idea behind this bypass is that the vulnerable application may filter by the URL of the original request, but does not check for a redirect. Example can be found here[2]
DNS Bypass is a technique where the attacker sets up a domain that makes requests to localhost or other private IPs, by having a DNS record point to that IP and supplying the SSRF vulnerable application with the domain. The idea here is to exploit input validation that checks the user inputted domain, but not what IP it resolves to.
DNS Rebinding is a technique which exploits applications that make multiple requests to an URL in order to first validate it and then request the contents. For example, the application makes a request to the URL to get it's IP in order to validate it. The URL points to a public IP and is considered valid. The application then makes another request to the same URL, now considered valid, to get it's contents. But after the first request, a script set up by the domain's owner may change the IP the URL points to, which would cause the second request to be made to an unvalidated IP, such as localhost or another private IP. The vulnerability is created by making multiple requests and trusting that the domain owner would not act maliciously. Example can be found here[3].
URL Parser Exploitation is a technique that exploits inconsistencies in URL parsers used in different libraries. The layman's explanation is that the attacker supplies a URL containing multiple URLs. The parser used to validate the URL selects one of them, which ends up being valid URL. Then the parser which is used to make a request to the previously validated URL (of which, in reality, only one part was validated) selects the other (unvalidated) URL, which can point to a private IP. Different URL confusion techniques can be used to target specific libraries or SDKs. This vulnerability is usually not created by the developer of the application but is found in the parsers they use. Thus, they must keep their libraries and SDKs up to date, as these vulnerabilities are patched when discovered. Example can be found here[4].
After the student is given an overview and technical explanation on these bypass techniques, provide them with a cheat sheet, either containing an abbreviated version of the previously linked materials or give the links to the student directly. Explain to the student that these are

[1] https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery/url-format-bypass#localhost

[2] https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery/url-format-bypass#bypass-via-redirect

[3] https://unit42.paloaltonetworks.com/dns-rebinding/

[4] https://claroty.com/2022/01/10/blog-research-exploiting-url-parsing-confusion/

useful resources for testing the security of any application they may find themselves developing and can be found again with a quick google search.
The last part of this module is a practical exercise in which the student must use what they have previously learned to retrieve a flag. The student is told that the target is using an incomplete deny list to validate the URL input. This means that the target is vulnerable to both the URL Formatting bypass and Open Redirection. The intended way to retrieve the flag is that the student tries different URL formats to access localhost, as some of them are blocked by the deny list, but most are not. However, if the student manages to retrieve the flag using a different bypass, it is considered a valid solution of the exercise. The URL where the flag is located should be given to the student.
For the purposes of this exercise, the deny list should block most regular IPv4 addresses, such as 127.0.0.1, 127.1, localhost, [::], etc, but be vulnerable to, for example, decimal or octal bypass. The goal is that the first URL formats found in the cheat sheet don't work, encouraging the student to try multiple ones or perhaps attempt a different bypass, such as Open Redirection. The goal of the practical exercise should not be to train a student to bypass SSRF defences, but to check that they paid attention to the theory part of this module. As such, it should be relatively simple to complete, but require a bypass outlined in this module.
Activity:
The student reads the overview. The student reads the explanations about the different bypass techniques. The student looks over the cheat sheet. For the practical exercise, the student is directed to the target. They are given an explanation on the defences used by the target and instructed to retrieve the flag. After the student has retrieved the flag, the module is complete.
Learning outcomes:

- Knowledge of the different bypass techniques used to commit SSRF exploits.
- Knowledge of what can cause an application to be vulnerable to bypasses.
- Ability to find resources and information about SSRF defence bypasses.

Duration: 2700
Difficulty: Intermediate
Outline:

- Overview
- Techniques
    - URL Formatting
    - Open Redirection
    - DNS Bypass
    - DNS Rebinding
    - URL Parser Exploitation
- Cheat sheet
- Exercise
    - Overview
    - Retrieve the Flag

SSRF Defence: Allow Listing

Description:
This module covers the Allow Listing defence in SSRF in more detail. The purpose of this module is to teach the student how to set up an allow listing solution to protect from SSRF.
The module starts off with an overview on allow listing, a short explanation on what it is to refresh the student's memory. Explain that when protecting against SSRF, allow listing is the preferred approach. If an application can expect what user input should be like, the allow listing solution can be used. For example, a website that tracks the prices of products in online shops can expect user input to match one of the popular online shopping websites. Anything that does not match those is denied. However, for allow listing to be effective, it must be implemented correctly.
After the overview there will be an explanation of how to set up an allow list on the application layer.
On the application layer we should accept IP addresses and domain names. There are a few steps to take to validate them. The details on these steps can be found here[1]. These steps should be abbreviated and simplified for the purposes of this module, however, provide a link to either the reference provided in this document or a similar one that outlines the steps in more detail. The abbreviated explanation should go more or less as follows:

1. If the input is an IP address, first validate that the IP is a valid IPv4 or IPv6 address and then confirm that it is one of the IPs of trusted applications in the allow list.
2. If the input is a domain name, first validate that it is a valid domain name using libraries, then confirm that the domain name is of one of the trusted applications in the allow list.
    a. After the domain name has been validated, to prevent a DNS rebinding bypass, request all IPs that the domain name resolves to and apply the allow listing check to them.

It is important to note that the application should not accept full URLs, but instead accept domain names.
After that the student moves on to the practical exercise. The goal of this exercise is to implement a simple allow list in a vulnerable application that is under attack. The student must implement an allow list in input validation that only allows through a list of IPs and domain names given to the student in the exercise description.
After the student has implemented an allow list that blocks all of these attacks, but does not block regular traffic, the exercise is considered complete.
Activity:
The student reads the overview and how to implement an allow list on the application layer. They then proceed to the exercise. The student is directed to the vulnerable application and given access to the application's code. The student is instructed to apply an allow list defence that only allows requests to trusted IPs and domain names. If all attacks are blocked and legitimate requests are not allowed through, the exercise is considered complete.
Learning outcomes:

---

[1]https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html#application-layer

- Knowledge of how to implement an allow list on the application layer.
- Knowledge of when an allow list can be used to protect from SSRF.
- Ability to implement a simple allow list.

Duration: 2700
Difficulty: Intermediate
Outline:

- Overview
- Protections
- Exercise

SSRF Defence: Deny Listing

Description:
This module covers the Deny Listing defence in SSRF in more detail. The purpose of this module is to teach the student how to set up a deny list solution in the case that allow listing can not be used for some reason.
The module starts off with an overview on deny listing, a short explanation on what it is to refresh the student's memory. First, we must explain why one would use the deny listing solution in the first place. Mention that allow listing is preferable in any scenario where it can be applied, however, it cannot be applied in every scenario. For example, a website that allows the use of webhooks cannot use the allow listing solution because the IPs and domains are often unknown and can dynamically change. As such, the deny list solution should be used, but it comes with challenges.
The deny listing solution is often not recommended, because for it to operate safely, the application must be able to detect whether an IP, both IPv4 and IPv6, is not a private IP, not localhost or a link-local address. The functionality to detect that is not a built-in feature in all SDKs, and thus the developer must implement a deny list blocking all possible variations of such IPs. The same goes for domains and the IPs the domain names resolve to. An organization must maintain an up-to-date list of all internal domain names.
Next, we will discuss how to protect from SSRF while using the deny listing defence. We will discuss what can be done on the application layer.
On the application layer we can implement a validation flow that conducts a series of validation steps and if all steps pass, the input is considered valid. The details on these steps can be found here[1]. These steps should be abbreviated and simplified for the purposes of this module, however, provide a link to either the reference provided in this document or a similar one that outlines the steps in more detail. The abbreviated explanation should go more or less as follows:

1. If the input is an IP address, we can use built-in libraries in most SDKs to confirm that the IP is a valid one and does not direct to the private network.
2. If the input is a domain name, to prevent DNS rebinding bypass, we will retrieve all IP addresses behind the domain name and verify them as we did in the previous step.
3. Check the that the protocol is either HTTP or HTTPS.
4. Using only the previously validated information, make a request to the IP or domain name with redirects disabled.

It is important to note that the application should not accept full URLs, but instead accept domain names.
After covering application layer protection against SSRF we move on to the practical exercise. The goal of this exercise is to implement a simple deny list to a vulnerable application that is being attacked. To make protecting from the attack easier, the student should have access to the log of requests made to the application, where they can see what inputs are being made. The log should only contain inputs that go through (are not blocked by deny list). The SSRF attacks committed should be requests to localhost, 127.0.01, obvious local IP addresses.

[1]https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html#application-layer_1

After the student has implemented a deny list that blocks all of these attacks, but does not block regular traffic, the exercise is considered complete.
To provide better feedback on progress, the task's completion percentage should reflect what amount of malicious requests are blocked. If any legitimate requests are blocked, the completion percentage should go back to zero percent. To communicate this to the student, it should be outlined in the exercise description.
After the exercise is finished, it should be outlined to the student that while the deny list implemented blocks the specific attacks used in this module, real world applications should be supplemented with more protection as was described in the materials.

Activity:

The student reads the overview and protections part, then proceed to the exercise. The student is directed to the vulnerable application, given access to the code and application logs. The student is instructed to use the logs to apply a deny list defence that blocks the attacks. If all attacks are blocked and legitimate requests are not blocked, the exercise is considered complete.

Learning outcomes:

- Knowledge of how to implement a secure deny list on the application layer.
- Knowledge of when to use a deny list over allow list.
- Knowledge of the specific weaknesses of deny listing.
- Knowledge of how those weaknesses can be addressed.
- Ability to implement a simple deny list.

Duration: 3600
Difficulty: Intermediate
Outline:

- Overview
    - When To Use
    - Weaknesses
- Protections
- Exercise

Defence Challenge

Description:
In this challenge module, the student's task is to protect the target, which is being exploited with SSRF. First, introduce the challenge to the student. Because the target can accept almost any URL, the allow list approach cannot be used. It is their task to create a deny listing defence that protects the application. To help them understand how the target is being exploited, they have access to the logs of requests made. Logs should include request and response, to make it easier to find exploits.
The attacks against the target are of several different types:

- Requests to 127.0.0.1, localhost, and other ways to access localhost
- Formatted IPs that access localhost, such as [::] and 2130706433 (decimal bypass)
    - Combinations of different formatting bypasses.
- Open redirection bypasses to localhost. (Must disable redirects)

Make use of as many of these[1] as possible.
The student is given access to the target's code and logs. The student's goal is to block all SSRF exploits without blocking any of the regular application traffic. The regular traffic should have randomized IPs and domain names so that the student could not easily implement an allow list, as that is not the intended solution of this challenge.
To make progress easier to ascertain, the completion percentage of the "Defend the Application" task should reflect what number of malicious requests are blocked. If any regular site traffic is blocked, the percentage should be set to zero. To better communicate this to the student, this should be outlined in the challenge introduction.
Activity:
Student reads the challenge introduction. The student is given access to target's code and logs. The challenge is complete when all malicious requests are blocked without blocking legitimate requests.
Learning outcomes:

- Knowledge of how to use logs to find malicious site traffic.
- Knowledge on how to implement a basic deny list solution against SSRF.

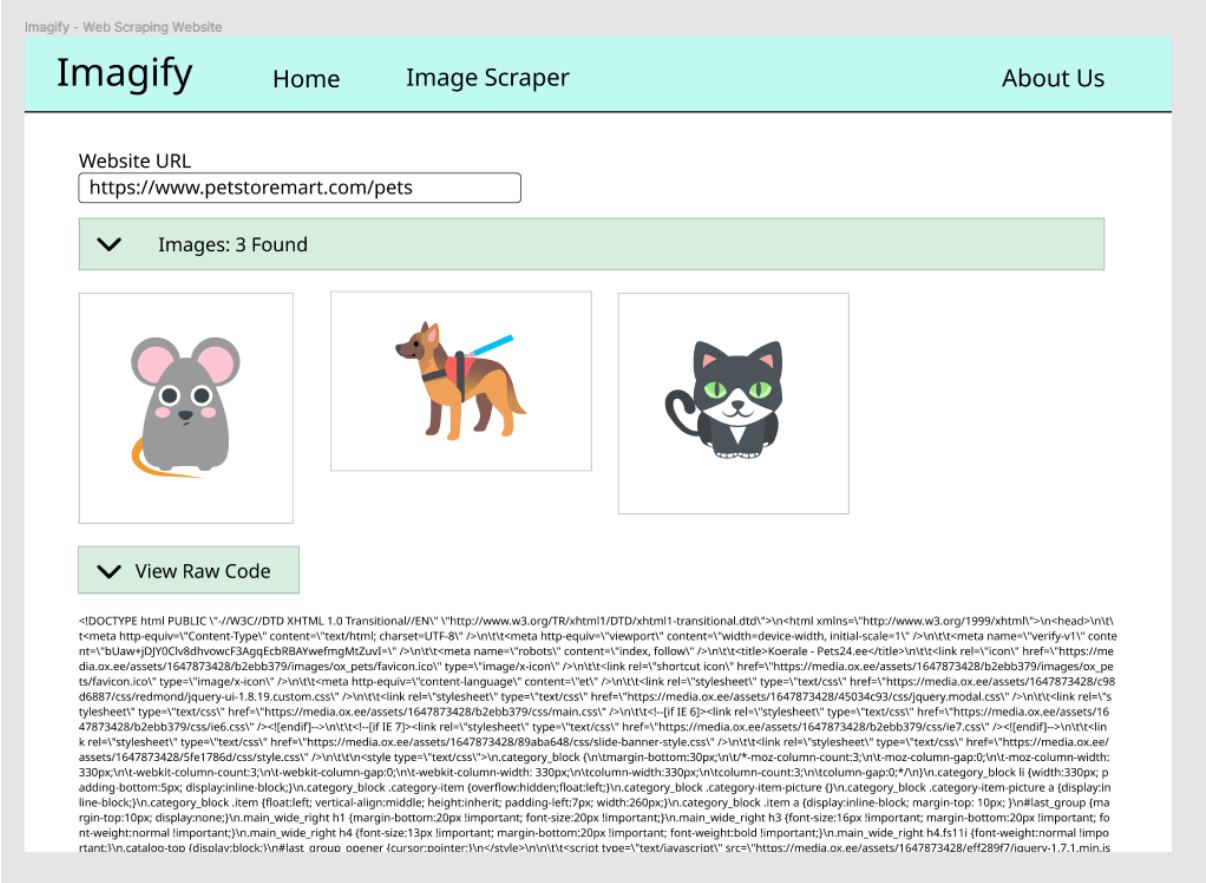
Duration: 3600
Difficulty: Advanced
Outline:

- Challenge Introduction
- Defend the Application

[1] https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery/url-format-bypass

Target Description

The target in these modules can be a web scraping website, that accepts an URL and returns images it can find, such as sources from <img> tags.
The entry point for an SSRF attack is the “Website URL” that an user would provide to the website. To make these exercises not be blind SSRF, the website provides images and the source code in expandable sections. Response is given even if no images are found.



Prototype of what the target could look like