Lauri Partinen (ktkt23sp)

ASSIGNMENT 7 ja 8

2025

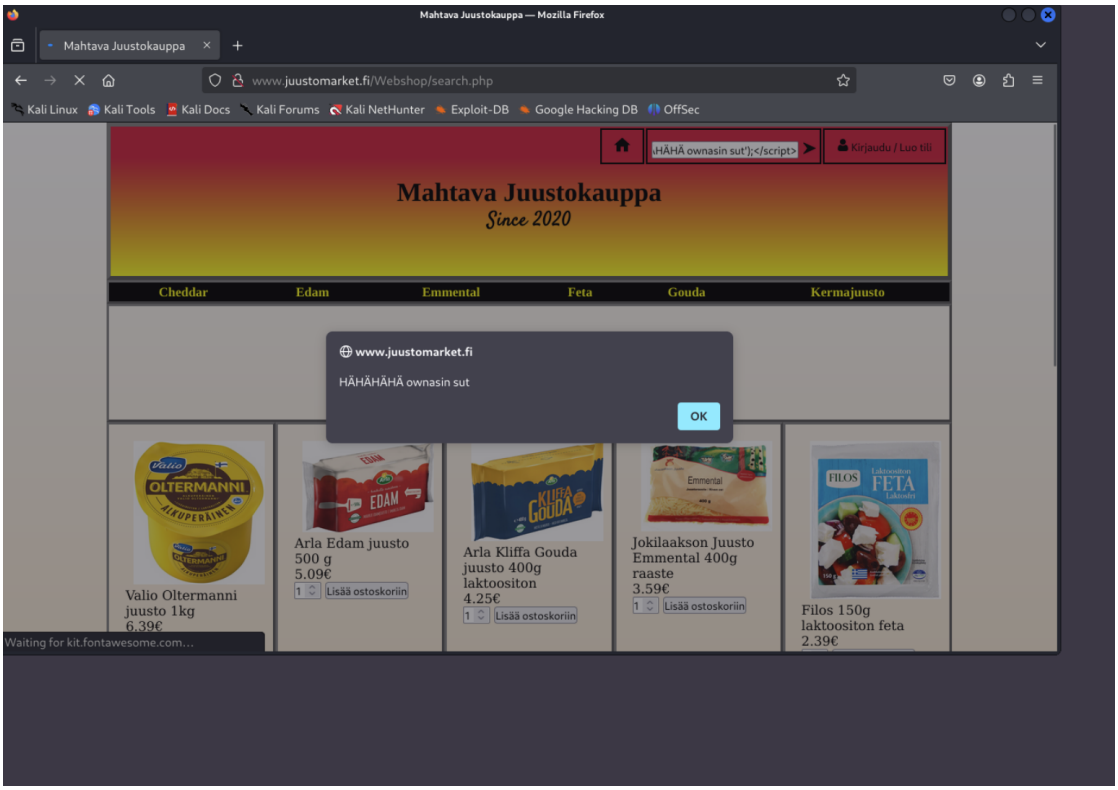
Kaakkois-Suomen
ammattikorkeakoulu

SISÄLLYS

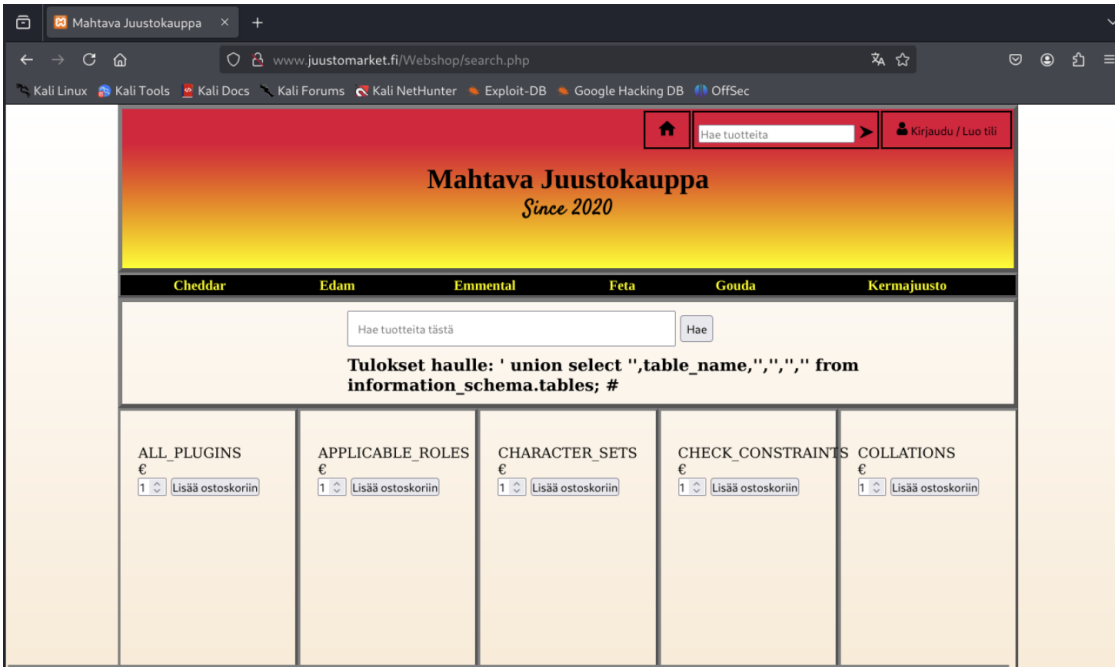
1 WAF

Aluksi testasin xss ja sql injektioita juustomarketissa

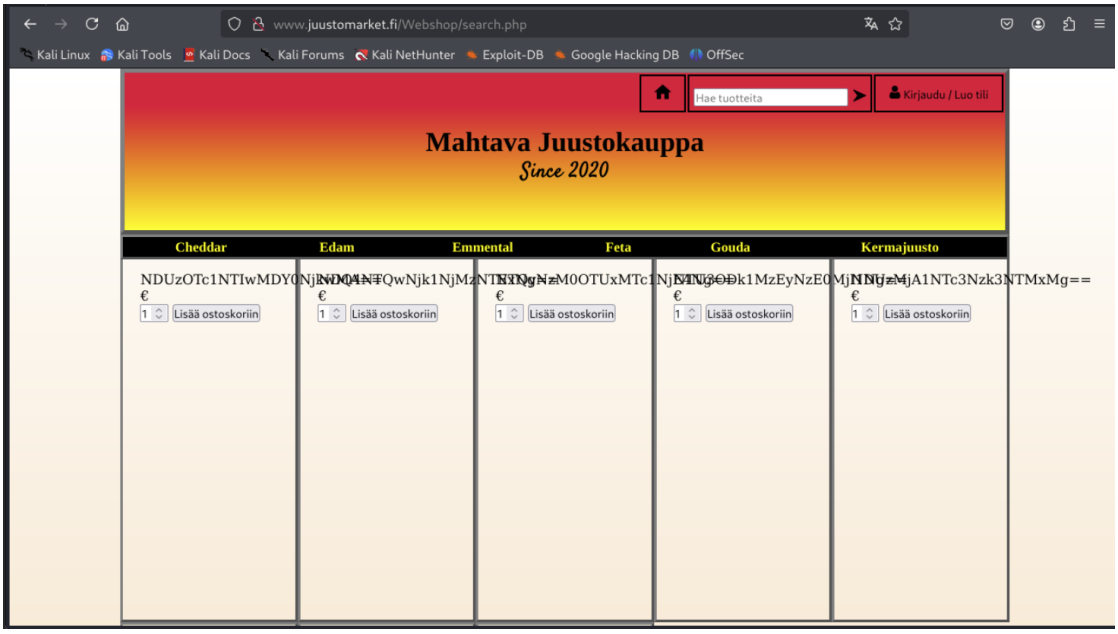
<script>alert('HÄHÄHÄHÄ ownasin sut');</script>



Dump the names of all databases: ' union select '',table_name,'','','' from information_schema.tables; #



Dump all credit card numbers: ' union select '',number,'','','' from creditcards; #



Openresty

Aloitin tehtävän tekemällä openrestyyn tarvittavat muutokset.

Muutin netplanista osoitteet vastaamaan segmenttejä.

```
  GNU nano 6.2
# This is the network config written by 'subiquity'
network:
  ethernets:
    ens4:
     dhcp4: no
     addresses: [10.10.20.2/24]
     nameservers:
       addresses: [192.168.2.100]
    ens3:
      dhcp4: no
      addresses: [10.10.10.5/24
      gateway4: 10.10.10.1]
  version: 2
```

Asetin nameserverin osoitteen 192.168.2.100 osoitteen osoitteeeksi 10.10.20.2/24 sama kuin nameserverillä

```
  GNU nano 4.8
# This is the network config written by 'subiquity'
network:
  ethernets:
    ens3:
      dhcp4: no
      addresses: [10.10.20.1/24]
      gateway4: 10.10.10.1
      nameservers:
        addresses: [10.0.0.5]

  version: 2
```

Muokattiin nginx.conf tiedostosta server nimi ja proxy_pass.

```
#user  nobody;
worker_processes  1;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    server {
        listen       80;
        server_name  www.juustomarket.fi;
        location / {
                modsecurity on;
                modsecurity_rules_file /usr/local/openresty/nginx/conf/modsecurity.conf;

                proxy_pass http://10.10.20.1;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;

        }
```

Testataan kommenttikenttään

' union select '',table_name,'','','' from information_schema.tables; #

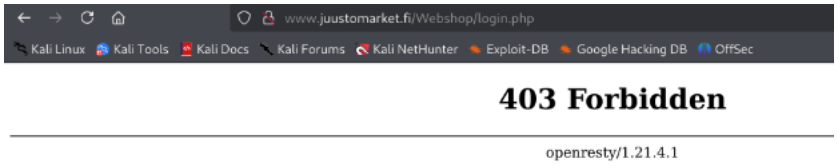
Toimii ei tule mitään

Policy 1: Testaa ensimmäinen SecRule estämään Juustomarketin Webstoren kirjautumissivu.

- **Tilanne ja mitä ollaan tekemässä?**
    - **Tilanne:** Halutaan testata ModSecurityn perustoiminnallisuutta estämällä pääsy tiettyyn URL-osoitteeseen, tässä tapauksessa Juustomarketin Webstoren oletettuun kirjautumissivuun
    - **Tekemässä:** Poistetaan nginx.conf tiedostossa olevan säännön kommentit.
- **Miten ongelma saatiin ratkaistua tai miten jokin asia toteutettiin?**
- Sääntö oli valmiina nginx.conf tiedostossa.

```
#Rule to disable login.php. Uncoomment SecRule line
SecRule ARGS|REQUEST_URI \
        "login.php" \
        "id:900002,\
        phase:1,\
        block,\
        deny,\
        status:403,\
        log,\
       msg:'Access denied - Login to system disabled'"
```

Testataan toiminta



- **Perustele ratkaisu ja tarvittaessa havainnollista.**
    - **Perustelu:** Yksinkertainen tapa varmistaa, että Modsecurity-sääntöjen muokkaus ja aktivointi toimii
- **Analyysi:**
    - **Miten järjestelmän turvallisuus muuttui vs aiempi tilanne? Miksi?**
        - Turvallisuus parani aiemmasta. Suora pääsy kirjautumissivulle tämän säännön avulla ei onnistu.
    - **Vaikuttaako tehty ratkaisu kokonaisuuden muuhun toimintaan? Miten, miksi?**
        - Estää pääsyn vain määriteltyyn URL-osoitteeseen. Jos tämä on ainoa tapa kirjautua sisään, se estää kirjautumisen kokonaan.
- **Arviointi:**
    - **Onko tehdyt ratkaisut tarpeellisia? Miksi?**
        - Tämä yksittäinen sääntö itsessään ei välttämättä ole tarpeellinen, koska kirjautuminen sivustolle evätään.
    - **Voiko niitä kiertää? Miten?**

        - Kirjautumissivun url muuttuu, sääntö ei enää toimi.
    - **Onko olemassa muita parempia vaihtoehtoja? Mitä?**
        - **IP-rajoitukset:** Sallitaan pääsy hallintasivuille vain tietyistä ip-osoitteista.

Policy 2: Estä Webstoren käyttäjiä käyttämästä HTML-tageja kommenteissa.

- **Tilanne ja mitä ollaan tekemässä?**
    - **Tilanne:** Webstoren tuotteisiin voi jättää kommentteja. Jos käyttäjät voivat syöttää kommentteihin raakaa HTML-koodia, se avaa mahdollisuude xss-hyökkäyksille.
    - **Tekemässä:** Konfiguroidaan waf tunnistamaan ja estämään kommenttikenttiin lähetetyt pyynnöt, jotka sisältävät html-tageja.
- **Miten ongelma saatiin ratkaistua tai miten jokin asia toteutettiin?**
  Luotiin modsecurity.conf sääntö, joka estää html tagien käyttö.

- **Perustele ratkaisu ja tarvittaessa havainnollista.**
    - **Perustelu:** Estämällä HTML-tagien käytön käyttäjien syötteissä estetään tehokkaasti yleisimmät xss-hyökkäykset.
- **Analyysi:**
    - **Miten järjestelmän turvallisuus muuttui vs aiempi tilanne? Miksi?**
        - Turvallisuus parani. Xss hyökkäyksen hyödyntäminen kommenttikentässä evätään, mikä parantaa juustomarketin turvallisuutta. Aiemmin haavoittuva kommenttikenttä olisi voinut mahdollistaa hyökkääjälle muiden käyttäjien tietojen saamisen.
    - **Vaikuttaako tehty ratkaisu kokonaisuuden muuhun toimintaan? Miten, miksi?**
        - Estää käyttäjiä käyttämästä html merkistöä kommenteissaan.
- **Arviointi:**
    - **Onko tehdyt ratkaisut tarpeellisia? Miksi?**

        - Tärkeää mille tahansa web-sovellukselle, joka hyväksyy ja näyttää käyttäjien syötteitä.
    - **Voiko niitä kiertää? Miten?**
        - **Erilaiset enkoodaukset**
        - **Merkkijonojen pilkkominen:** Kommenttien tai javscript koodin osien syöttäminen eri tavoin.
    - **Onko olemassa muita parempia vaihtoehtoja? Mitä?**
        - **Sovelluksen koodin korjaukset:** Paras tapa estää xss hyökkäykset on varmistaa, että itse web-sovellus hyväksyy kaikki käyttäjien syötteet ja enkoodaa kaiken ulostulon oikein.
        - **Content Security Policy:** Voidaan kertoa selaimelle, mistä lähteistä se saa ladata ja suorittaa skriptejä.

Policy 3: Estä Webstoren käyttäjiä käyttämästä permutoituja HTML-tageja kommenteissa.

- **Tilanne ja mitä ollaan tekemässä?**
    - **Tilanne:** Hyökkääjät eivät välttämättä yritä syöttää suoraan html, vaan käyttävät erilaisia muunneltuja muotoja yrittäessään kiertää suodatusta.
    - **Tekemässä:** Varmistetaan, että käytössä oleva waf tunnistaa ja estää muunnellut xss tai html scriptit.
- **Miten ongelma saatiin ratkaistua tai miten jokin asia toteutettiin?**
- Lisäsin modsecurity.conf tiedostoon säännön, jota käytin policy 2. Se estää sekä permutoidut- ja html tagit.

```
SecRule ARGS "<[^>]+>" \
    "id:900004,\
    phase:2,\
    deny,\
    status:403,\
    log,\
    msg:'HTML tags detected in user input - request blocked.'"
```

Policy 4: Estä SQL-injektio tuotehaun kautta Juustomarketin Webstoressa.

- **Tilanne ja mitä ollaan tekemässä?**
    - **Tilanne:** Webstoren tuotehakutoiminto ottaa vastaan käyttäjän syötteen ja käyttää sitä tietokantakyselyn rakentamiseen

tuotteiden etsimiseksi. Tavoite on estää sql injektion käyttö juustomarketin nettikaupan tuotehaussa.

- **Tekemässä:** Konfiguroidaa waf tunnistamaan ja estämään pyynnöt, jotka sisältävät tunnettuja sql-injektioyritysten merkkejä ja komentoja.

- **Miten ongelma saatiin ratkaistua tai miten jokin asia toteutettiin?**
- Tehdään modsecurity.conf sääntö, joka estää yleisten sql injektioiden merkkien ja termistön käytön.

```
ecRule ARGS "(?i:(\%27)|(\')|(\-\-)|(\%23)|(#)|(\b(select|update|delete|insert|drop|union|or|and)\b))" \
   "id:900006,\
   phase:2,\
   deny,\
   status:403,\
   log,\
   msg:'Generic SQL Injection attempt detected in user input.'"
```

- Testataan kommenttikenttään scriptiä

' union select '',table_name,'','','' from information_schema.tables; #

- **Perustele ratkaisu ja tarvittaessa havainnollista.**
  - **Perustelu:** Estämällä tunnetut sql injektiot waf säännön avulla, voi estää hyökkäyksen onnistumisen.
- **Analyysi:**
  - **Miten järjestelmän turvallisuus muuttui vs aiempi tilanne? Miksi?**

        - Sääntö luo jonkin verran turvaa sql-injektioita vastaan. Aiemmin haavoittuva hakutoiminto olisi voinut johtaa mahdollisesti vakaviin seuraamuksiin juustomarketissa.
    - **Vaikuttaako tehty ratkaisu kokonaisuuden muuhun toimintaan? Miten, miksi?**
        - Yleensä ei vaikuta negatiivisesti normaaliin toimintaan. SQLi-säännöt ovat yleensä melko tarkkoja kohdistumaan haitallisiin kuvioihin. On kuitenkin mahdollista, että jokin erittäin epätavallinen, mutta legitiimi hakusana tai syöte voi laukaista "false positive" -eston.
- **Arviointi:**
    - **Onko tehdyt ratkaisut tarpeellisia? Miksi?**
        - Tärkeää, koska juustomarketti käyttää tietokantaa osana toimintaa.
    - **Voiko niitä kiertää? Miten?**
        - **Second-order SQL injection:** Haitallinen syöte tallennetaan ensin tietokantaan ja se aiheuttaa injektion vasta myöhemmin, kun dataa luetaan ja käytetään toisessa kyselyssä. Tästä tekee vaarallisen se, että waf ei välttämättä havaitse tätä.
        - **Blind SQL injection:** Hyökkääjä päättelee tietoa tietokannasta epäsuorasti käyttämättä suoria esim. UNION SELECT tyyppisiä komentoja, jotka waf helpommin tunnistaa.
        - **WAF-tunnistuksen kiertäminen:** Erilaiset tekniikat, joilla yritetään saada waf tulkitsemaan pyyntö eri tavalla kuin taustajärjestelmä.
    - **Onko olemassa muita parempia vaihtoehtoja? Mitä?**
        - **Parametrisoidut kyselyt:** Tapa estää sql injektioita sovelluskoodissa. Käyttäjän syötettä ei koskaan yhdistetä suoraan SQL-komentoon, vaan se käsitellään erillisenä parametrina.

Policy 5: Estä arkaluontoinen sisältö (luottokorttinumerot).

- **Tilanne ja mitä ollaan tekemässä?**

    - **Tilanne:** Webstoren vastauksissa ei saisi koskaan paljastua arkaluontoista tietoa, kuten luottokorttinumeroita. Tavoite on blokata kaikki pankkikortin numeroita näyttävät syötteet.
    - **Tekemässä:** Konfiguroidaan waf tarkastamaan web-palvelimen lähettämät vastaukset ja estämään ne vastaukset, jotka sisältävät tunnistettavia luottokorttinumeroiden kaltaisia numerosarjoja.

- **Miten ongelma saatiin ratkaistua tai miten jokin asia toteutettiin?**
- Tein ensin webstoreen tekstitiedoston creditcard.txt, johon lisäsin tehtävän annossa tulleet korttitiedot, mutta en onnistunut hacker koneelta saamaan url kautta yhteyttä.
- **Perustele ratkaisu ja tarvittaessa havainnollista.**
    - **Perustelu:** Ratkaisu tarjoaa tärkeän turvakerroksen estämällä arkaluontoisten luottokorttitietojen tahatonta vuotamista web sovelluksen vastauksissa.
- **Analyysi:**
    - **Miten järjestelmän turvallisuus muuttui vs aiempi tilanne? Miksi?**
        - Turvallisuus parani. Pankkikorttitietojen turvaaminen on juustomarketille liiketoiminnan kannalta tärkeää.
    - **Vaikuttaako tehty ratkaisu kokonaisuuden muuhun toimintaan? Miten, miksi?**
        - Toiminnan kannalta riskinä on false positives, joissa waf virheellisesti tunnistaa tavallista dataa luottokorttinumeroksi ja estää sivun näkymisen.
- **Arviointi:**
    - **Onko tehdyt ratkaisut tarpeellisia? Miksi?**
        - Kyllä, koska juustomarket käsittelee maksu- tai muita arkaluontoisia tietoja palvelussaan.
    - **Voiko niitä kiertää? Miten?**
        - **Enkoodaus:** Jos luottokorttinumero on vastauksessa koodattu tai muunneltu tavalla, jota sääntö ei tunnista.
    - **Onko olemassa muita parempia vaihtoehtoja? Mitä?**

- **Tokenien käyttö:** Korvataan oikeat korttinumerot tokeneilla sovelluksen sisällä, jolloin oikeita numeroita käsitellään vain erittäin rajatussa ja suojatussa ympäristössä.

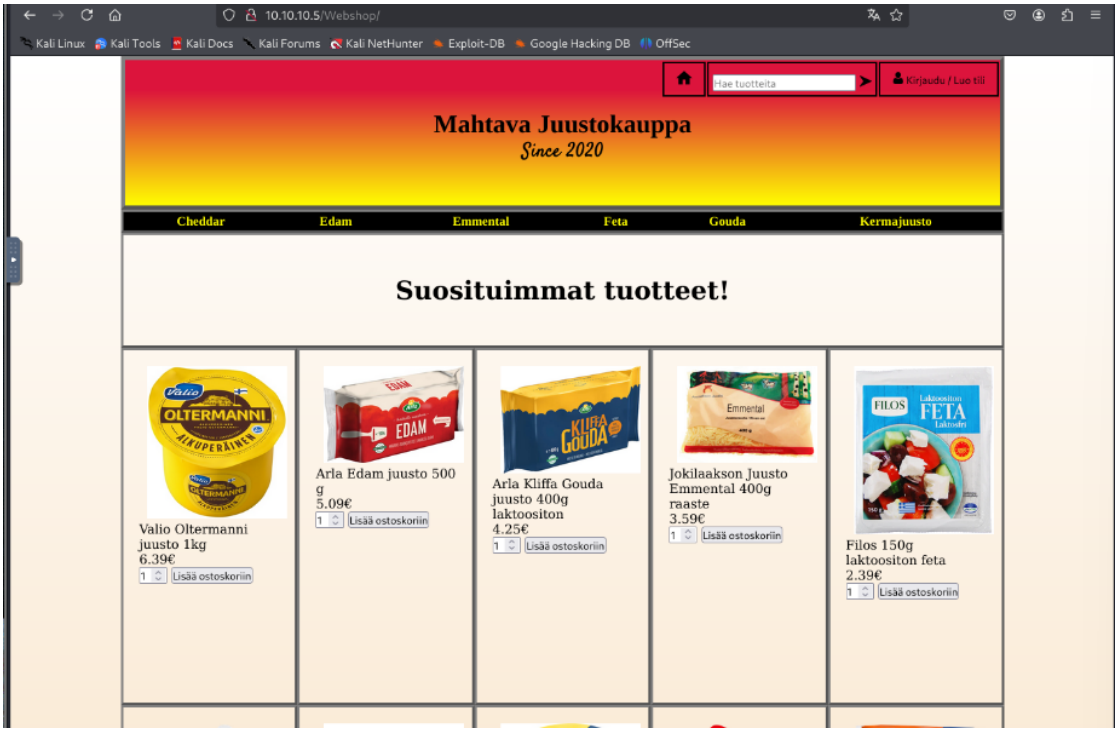
2 8A) TCP SYN FLOOD

Aloitin hyökkäyksen hacker koneelta kommennolla.

sudo hping3 10.10.10.5 --flood -p 80 -S

Dashboardista huomaan sisään tulevan liikenteen nousun huomattavasti aiemmasta ja samoin prosessorin käytössä on piikkejä.

Netin selaaminen toimii hyvin Worker_pc. Hacker koneella huomaa viivettä sivun lataamisessa.

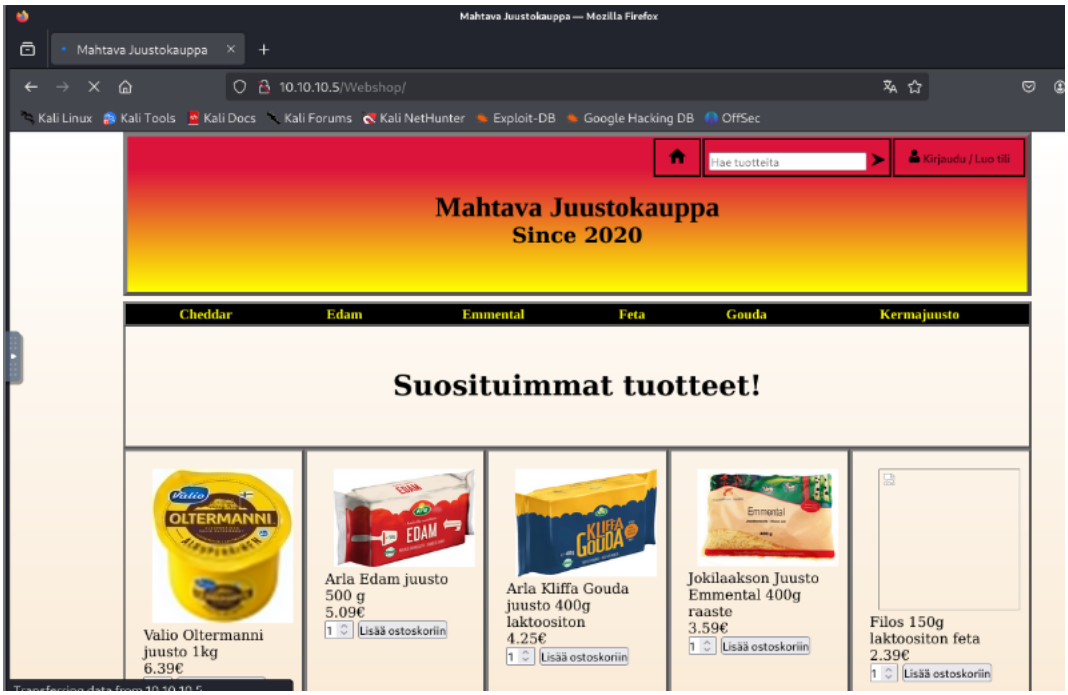


Lisätään hyökkäys scriptiin random source.

sudo hping3 10.10.10.5 --flood -p 80 -S --rand-source

Muokkauksen jälkeen netin selaus hidastui huomattavasti Worker_pc. Hacker koneella juustomarketin sivuston lataamisessa meni vielä pidempään.

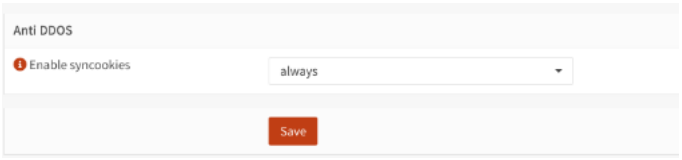
Lopulta juustomarket.fi latautui.



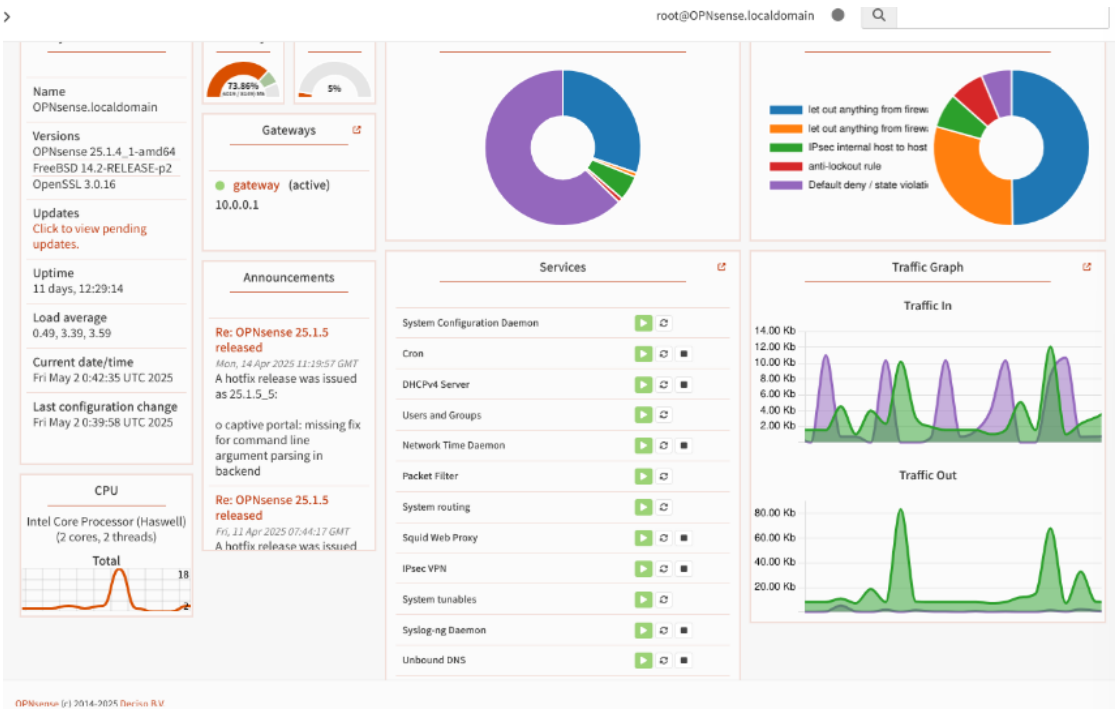
Dashboardissa prosessorin käyttö pomppasi 100%.

Part 8b: TCP SYN Flood Mitigation (Enable SYN Cookies)

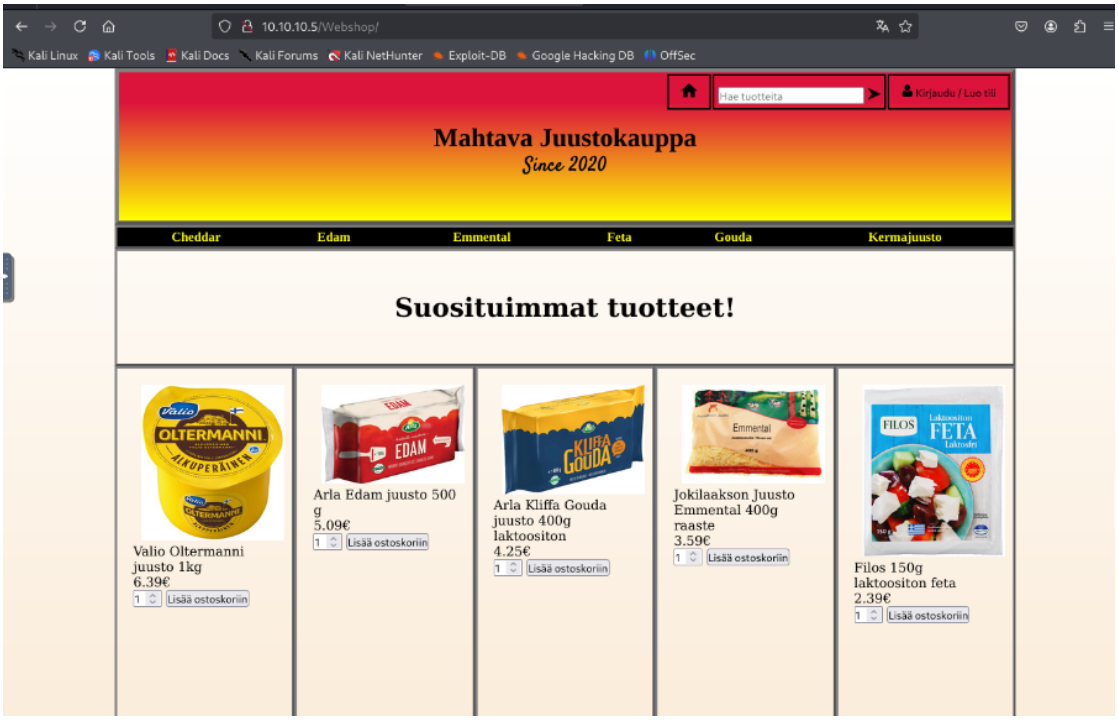
Syn cookies always



Dashboardissa huomaa merkittäviä liikenne piikkejä sisään tulevassa liikenteessä. Prosessorin käytössä huomattavissa samoja piikkejä.

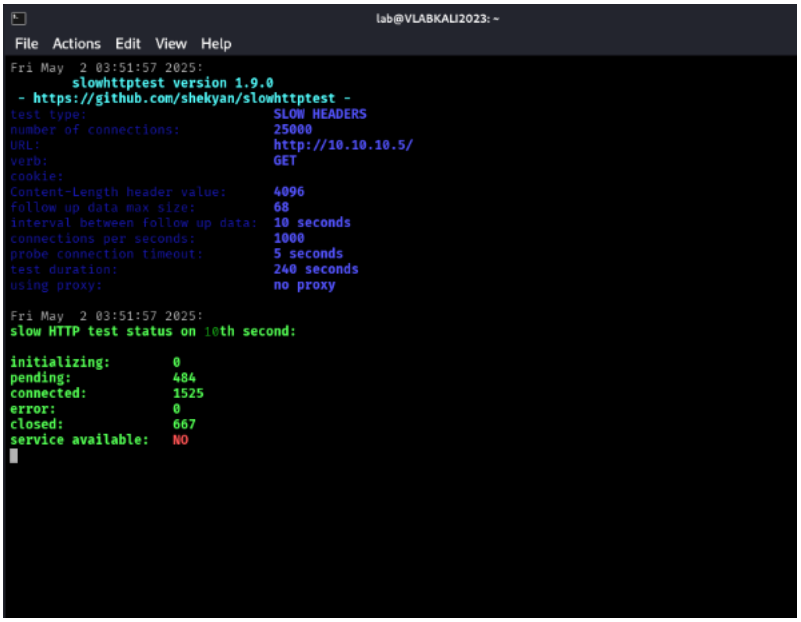


Internetin selaaminen onnistuu normaalisti.

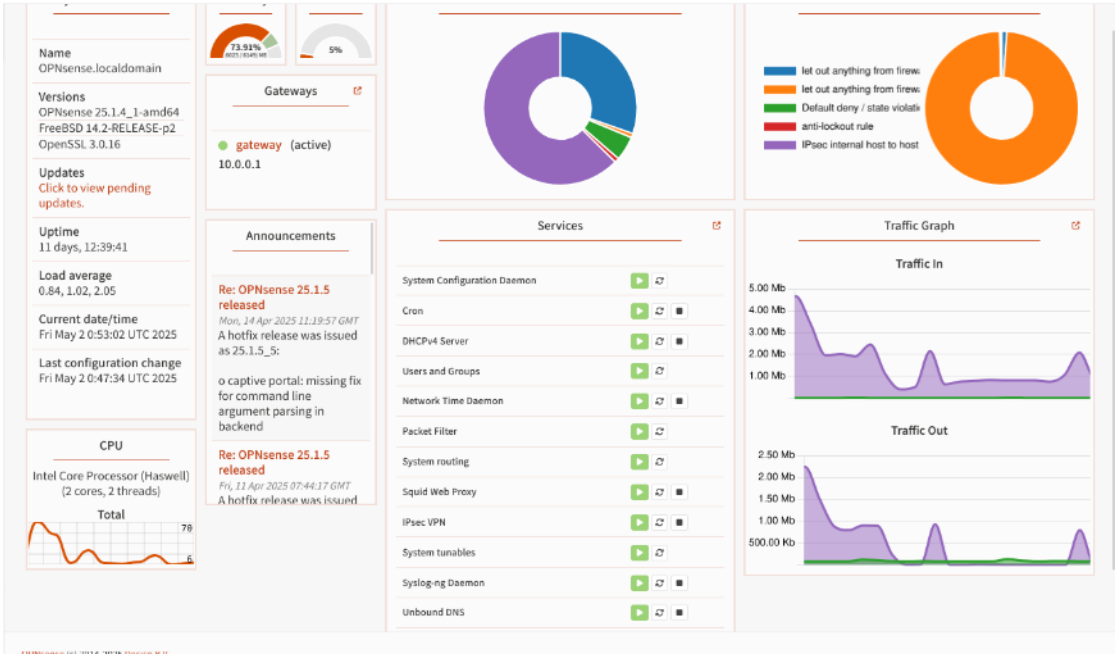


8c) Slowloris

slowhttptest -H -u http://10.10.10.5 -c 25000 -r 1000

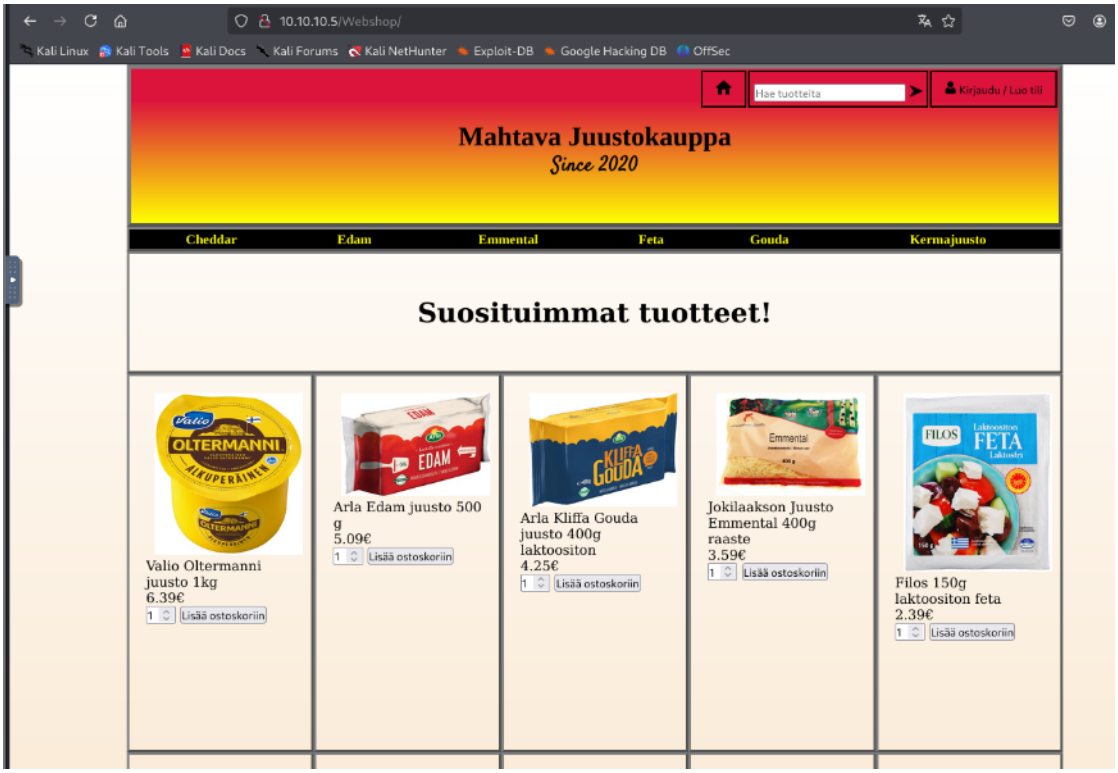


Dashboardissa firewall kohdassa. Let out anything from firewall host itself hallitsee säännön jälkeen. Prosessorin käyttö vaikuttaa olevan normaali ei suurta eroa normaali tilanteeseen.



ss -a Waf

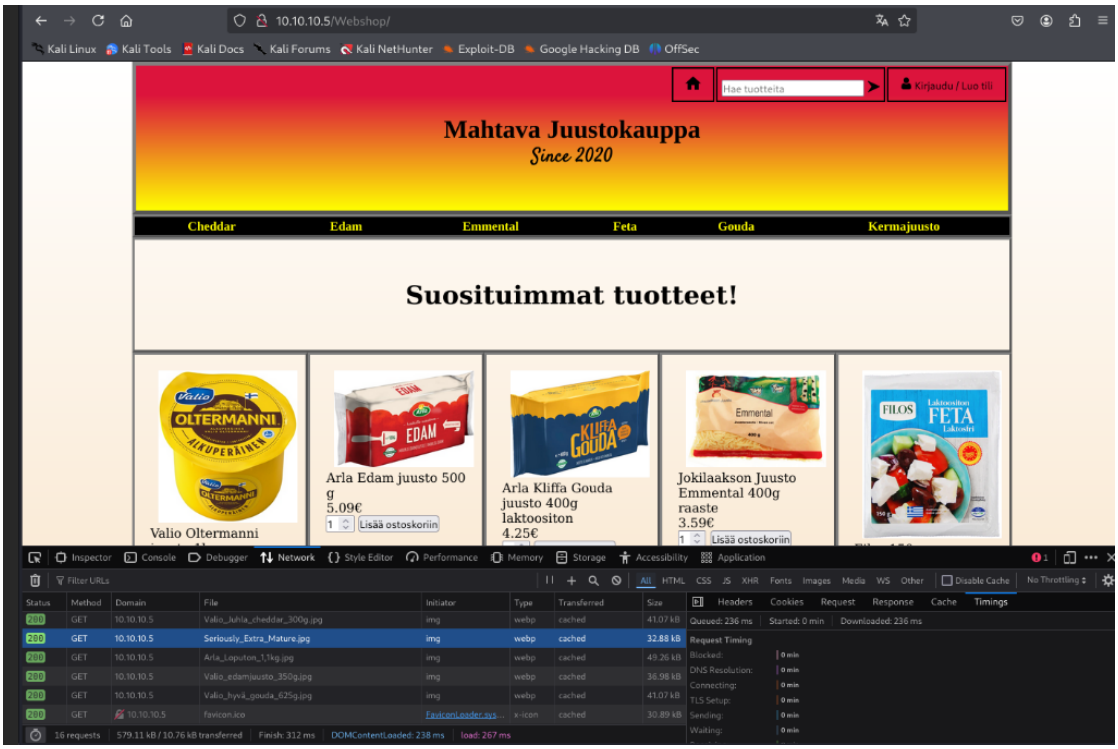
Inside verkon selaaminen toimii normaalisti



Hacker koneessa juustomarket toimii normaalisti.

8d) Slowloris mitigation

Juustomarket.fi sivun lataamiseen meni normaalisti 236ms.



/usr/local/openresty/nginx/conf/nginx.conf

En löytänyt nginx.conf tiedostoa, joten tehtävä jäi kesken.

3 REFLEKTIO

Mitä saavutin?
Waf peruskonfiguraatio: Onnistuin konfiguroimaan OpenRestyn toimimaan käänteisenä välityspalvelimena

- ModSecurity Sääntöjen Toteutus ja Testaus
- DoS-Hyökkäysten Simulointi
- SYN Flood Torjunta: Aktivoin palomuurin syn cookie suojauksen ja todensin sen tehokkuuden.

- Mitä jäi saavuttamatta?
- Policy 5 (Luottokorttinumeroiden Estäminen): Tämän säännön toteutus ja testaus jäi kesken. En onnistunut luomaan toimivaa sääntöä.
- Slowloris-Torjunta: En onnistunut toteuttamaan Slowloris-hyökkäyksen torjuntaa, koska en löytänyt nginx.conf-tiedostoa annetusta polusta.

Miten opin asian?

Opin asiat pääasiassa käytännön tekemisen kautta:

- Konfigurointi ja Testaus: Muokkaamalla suoraan konfiguraatiotiedostoja ja testaamalla niiden toimivuus.
- Työkalujen Käyttö: Hyökkäystyökaluja esim. Tcp synflood ja seurantatyökaluja

Kuinka taitoni kehittyivät?

- WAF-Konfigurointi: Taitoni OpenRestyn ja ModSecurityn peruskonfiguroinnissa ja sääntöjen kirjoittamisessa kehittyivät konkreettisesti.

- Haavoittuvuuksien Ymmärrys: Ymmärrykseni yleisistä web-haavoittuvuuksista xss ja niiden torjunnasta waf avulla syveni.
- DoS: Opin tunnistamaan ja simuloimaan yleisiä DoS-hyökkäystyyppejä esim synflood.

Miten lähestyisin asiaa seuraavalla kerralla?

- Dokumentointi: Kirjaisin tarkemmin ylös havaintoja, erityisesti virhetilanteita ja mahdollisia ratkaisuja

Miten pystyn käyttämään tätä taitoa/osaamista tulevaisuudessa?

- Web-Sovellusten Suojaaminen: Voin hyödyntää WAF-konfigurointitaitojani web-sovellusten suojaamisessa yleisiltä hyökkäyksiltä.
- Turvallisuustestaus: Opin käyttämään työkaluja, joita voi hyödyntää perustason tietoturvatestauksessa.