ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Εγκατάσταση, παραμετροποίηση και πιλοτική λειτουργία τείχους προστασίας ΕΛ/ΑΑΚ.

ΔΗΜΗΤΡΗΣ ΖΗΤΟΥΝΗΣ

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Θ. Καμαλάκης
Επικ. Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Π. Ριζομυλιώτης
Επικ. Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Β. Δαλάκας
Επιστημονικός Συνεργάτης, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Ο Δημήτρης Ζητούνης δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Πίνακας Περιεχομένων

Περίληψη

Η παρούσα πτυχιακή μελέτη επιδιώκει την παρουσίαση του Naxsi ως προωθημένου web application firewall system. Το πρώτο βήμα προς επίτευξη του ανωτέρω σκοπού είναι μια εισαγωγική περιγραφή τόσο του τείχους προστασίας εφαρμογών ιστού ως έννοια όσο και του Naxsi επι τούτου και των πλεονεκτημάτων που το συνοδεύουν. Στη συνέχεια, ακολουθεί μια εις βάθος επεξήγηση των κρίσιμων εργαλείων που επιστρατεύτηκαν και οι εν γένει λειτουργίες τους οι οποίες τοποθετημένες σε ένα αρμονικό πλέγμα, απέφεραν το επιθυμητό αποτέλεσμα. Έπειτα, αναλύεται η σύνταξη των κανόνων του Naxsi και η ικάνοτητα να εγκατασταθεί σε Load Balancer, ώστε να αναδειχθεί η βέλτιστη και αποδοτικότερη λειτουργία του. Εν κατακλείδι, έχοντας αναδείξει τις επωφελείς πτυχές του αντικειμένου αυτής της μελέτης, δηλαδή του Naxsi, γίνεται ιδιαιτέρως εύκολο να διαπιστωθεί η ιδανική εφαρμογή του σε κάθε είδους βιομηχανικής, επιχειρησιακής ή ακόμα και οικιακής χρήσης.

Λέξεις κλειδιά: Λογισμικό, Προστασία, Καινοτομία, Επίθεση, Ευπάθεια

Abstract

The present thesis aims at presenting Naxsi as an advanced web application firewall system. The first step towards achieving the abovementioned goal is an introductory description of the notion of web application system as well as of Naxsi per se and the benefits that it offers. Whereupon, it follows an in-depth explanation of the crucial tools that have been employed and their specific functions, which, placed in a harmonious nexus, led to the desired outcome. Afterwards, it is elaborated the composition of Naxsi rules, along with the ability of Naxsi to be installed to a Load Balancer in order to achieve its most ideal and efficient aspect. In conclusion, having designated the beneficial aspects of the object if this thesis, namely of Naxsi, it is pretty evident the conclusion that Naxsi is the optimum option for all kinds of industrial, operational or even domestic appropriations.

Keywords: Software, Protection, Innovation, Firewall, Vulnerability

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Σχημάτων/Σχεδιαγραμμάτων

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

ΓΣΠ	Γεωγραφικά Συστήματα Πληροφοριών
ΕΛ/ΛΑΚ	Ελεύθερο Λογισμικό/Λογισμικό Ανοικτού Κώδικα
API	Application Programming Interface
CSV	Comma Separated Values
DDoS	Distributed Denial-of-Service
HAProxy	High Availability Proxy
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
OWASP	Open Web Application Security Project
rx	regular expression
str	string matcher
TCP	Transmission Control Protocol
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAF	Web Application Firewall
XML	Extensible Markup Language
XSS	Cross-site Scripting

Κεφάλαιο 1ο : Εισαγωγή

Ένα τείχος προστασίας εφαρμογών ιστού είναι ένα ειδικό τείχος προστασίας εφαρμογών που εφάπτεται ειδικά των εφαρμογών ιστού. Εφαρμόζεται μπροστά από τις εφαρμογές ιστού και αναλύει την αμφίδρομη διαδικτυακή (HTTP – Hypertext Transfer Protocol) κυκλοφορία, δηλαδή ανιχνεύει και εμποδίζει οτιδήποτε κακόβουλο. Με την ανάλυση της κίνησης HTTP μπορεί να αποτρέψει τις επιθέσεις που προέρχονται από ελαττώματα ασφαλείας εφαρμογών ιστού, όπως SQL injection, cross-site scripting (XSS), συμπερίληψη αρχείων και εσφαλμένες ρυθμίσεις ασφαλείας. Πιο συγκεκριμένα, οι παρακάτω 10 τεχνικές hacking ήταν αυτές που έθεσαν τα θεμέλια και την ανάγκη για την ανάπτυξης των web application firewall.

- Hidden field manipulation
- Cookie poisoning
- Parameter tampering
- Buffer overflow
- Cross site scripting (XSS)
- Backdoor or debug options
- Stealth commanding

- Forced browsing
- Third party misconfigurations
- Known vulnerabilities

Το 2002 το πρώτο open source WAF με το όνομα ModSecurity έκανε την εμφανισή του, προκειμένου να αντιμετωπίσει προβλημματα που είχαν αναπτυχθεί σε βιομηχανικό επίπεδο, επιχειρησιακό επίπεδο, ακόμα κα τα προβλήματα κόστους. Ειδικότερα, η πρώτη έκδοση κυκλοφόρησε τον Νοέμβριο του 2002 και υποστήριζε τον Apache HTTP Server 1.3.x. Ένα βασικό μειονέκτημα του modsec ήταν αρχικά ότι αποτελούσε μια υπομονάδα Apache και η μεταφορά του ModSecurity σε άλλες πλατφόρμες ήταν χρονοβόρα και είχε υψηλό κόστος συντήρησης. Ως αποτέλεσμα τούτου, ξεκίνησε μια πλήρης επανεγγραφή τον Δεκέμβριο του 2015. Αυτή η νέα επανάληψη, libmodsecurity, αλλάζει την υποκείμενη αρχιτεκτονική, χωρίζοντας το ModSecurity σε αυτόνομο μηχανισμό που επικοινωνεί με τον διακομιστή μέσω API. Αυτό το αρθρωτό αρχιτεκτονικό WAF, το οποίο ανακοινώθηκε για δημόσια χρήση τον Ιανουάριο του 2018, έγινε libmodsecurity (ModSecurity έκδοση 3.0) και έχει υποστηρίξει συνδέσμους για το NGINX και το Apache.

Παρακάτω αναφέρονται συνοπτικά μερικές WAF λύσεις:

	Enterprise	Cloud	Open-source
1.	Barracuda Networks WAF	Akamai Technologies Kona	NAXSI
2.	Citrix Netscaler Application Firewall	Alibaba Cloud	ModSecurity
3.	F5 Big-IP ASM	Amazon Web Services AWS WAF	IronBee
4.	Fortinet FortiWeb	Cloudbric	WebKnight
5.	Imperva SecureSphere	Cloudflare	Shadow Deamon
6.	Penta Security WAPPLES	Fastly	
7.	Radware AppWall	Sucuri Firewall	
8.	Sophos XG Firewall	Radware	

(Πιν.1)

Εν κατακλείδι, υπάρχουν ποικίλες επιλογές προϊόντων όσων αφορά τα application firewall. Μεγάλες βιομηχανίες και επιχειρήσεις τείνουν να προμηθεύονται τις enterprise και cloud λύσεις περισσότερο λόγω του άμμεσου Support που παρέχεται, χωρίς αυτό να σημαίνει ότι οι open-source λύσεις δεν μπορούν να σταθούν αντάξια ή και να λειτουργήσουν πολλές φορες καλύτερα απο τις προηγούμενες.

1.1. Γενικά χαρακτηριστικά του NAXSI

Σκοπός του παρόντος προγράμματος είναι να βοηθήσει τόσο ατομικά τα υποκείμενα όσο και τις εταιρίες να εξασφαλίσουν τις εφαρμογές τους στο διαδίκτυο από επιθέσεις όπως SQL Injections, Cross Site Scripting, Cross Site Requesting False, τοπικές και απομακρυσμένες καταχωρίσεις αρχείων.

Το Naxsi μπορεί να ελέγξει διαφορετικές τιμές, όπως διευθύνσεις URL, παραμέτρους αιτήματος, cookies, κεφαλίδες ή το σώμα POST και μπορεί να ενεργοποιηθεί ή να απενεργοποιηθεί στη διαμόρφωση του Nginx. Άλλα εργαλεία, όπως τα NX-Utils και Doxi, διευκολύνουν τη διαχείριση, την παραγωγή αναφορών και τις ενημερώσεις των κανόνων.Το Naxsi συνεργάζεται με το NX-utils, το οποίο είναι πολύ χρήσιμο εργαλείο για τη δημιουργία whitelist και αναφορών. Συγκεκριμένα, το NX-utils περιλαμβάνει (i) λειτουργία συλλογής, η οποία επιτρέπει στη Naxsi να αποθηκεύει αιτήματα που έχουν αποκλειστεί από το WAF για μελλοντικές αναφορές, και (ii) whitelist, η οποία απεικονίζει τα αποθηκευμένα συμβάντα.

Για να χρησιμοποιήσει κάποιος ένα WAF θα πρέπει να αντιμετωπίσει το συνδυασμό των

ακόλουθων ζητημάτων ασφαλείας σχετικά με μια εφαρμογή στο διαδίκτυο:

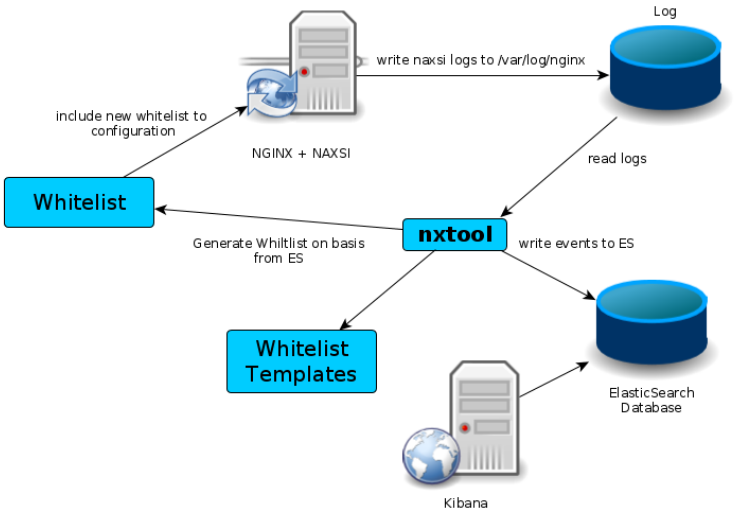
- Κλειστή πηγή και καμία δυνατότητα επιθεώρησης του πηγαίου κώδικα
- Ευπάθειες που δεν μπορούν να διορθωθούν ή να μετριαστούν στο σύστημα
- Τρωτά σημεία που απαιτούν πάρα πολύ προσπάθεια (κόστος) να διορθωθούν
- Πολύ παλιά για να διορθωθούν
- Πολύ περίπλοκο για να εξασφαλιστεί αποτελεσματικά
- Πιστοποίηση / εξουσιοδότηση που δεν είναι διαθέσιμη στο σύστημα

Υπάρχουν ποικίλες λύσεις τείχους προστασίας που είναι αφιερωμένες σε εφαρμογές, οι περισσότερες από τις οποίες βασίζονται σε μαύρες λίστες. Στόχος του τείχους προστασίας εφαρμογών είναι να αποκλείσει τις επιθέσεις που αναγνωρίζει και να αφήσει όλα τα άλλα ερωτήματα στον διακομιστή.

Τα αρχικά NAXSI σημαίνουν «Nginx Anti Xss & Sql Injection». Το NAXSI WAF ανιχνεύει μη αναμενόμενους χαρακτήρες σε αιτήματα / επιχειρήματα HTTP και τα αποκλείει. Αποτρέπει τον

εισβολέα από την αξιοποίηση ευπαθειών ιστού ενός ιστότοπου, ανεξάρτητα από τη γλώσσα στην οποία αναπτύσσεται ο ιστότοπος. Στην πραγματικότητα, προστατεύει τον ιστότοπο από τις TOP 10 απειλές OWASP. Το NAXSI ωστόσο, είναι ένα διαφορετικό σύστημα προστασίας, καθώς λειτουργεί βάσει ενός λευκού καταλόγου, από προεπιλογή, αποκλείει όλα τα ερωτήματα και επιτρέπει μόνο εκείνα που μέσω της κίνησης θεωρεί νόμιμα. Η διαφορά αυτή προστατεύει ακόμη και από τις αναδυόμενες επιδρομές στον κυβερνοχώρο, των οποίων η υπογραφή είναι ακόμη άγνωστη και συνεπώς δεν μπορεί να αντιμετωπιστεί με συγκεκριμένους κανόνες. Επιπρόσθετα, το NAXSI προσαρμόζεται σε κάθε περιβάλλον και σε κάθε τοποθεσία για να αποφύγει τις ψευδώς θετικές κινήσεις και να επιτρέψει μόνο τις νόμιμες, προσφέροντας προστασία που προσαρμόζεται στους κινδύνους και τα προβλήματά. Επιπλέον, το NAXSI δεν προκαλεί απώλειες απόδοσης και δεν απαιτεί καμία ενημέρωση (εκτός από το whitelist, φυσικά), απαλλάσοντας ουσιαστικά από τακτικές καθυστερήσεις ή διακοπές στη παραγωγή.

1.2. Πλεονεκτήματα του NAXSI



(Σχ.1)

Η απλότητα με την οποία ένας χρήστης μπορεί να χειριστεί την πρόσβαση στο NAXSI είναι ένα βασικό χαρακτηριστικό που το διαφοροποιεί από τα υπόλοιπα firewalls εφαρμογών ιστού με παρόμοιες λειτουργίες όπως το ModSecurity. Αν και το ModSecurity έρχεται με ένα πλούσιο σετ χαρακτηριστικών, είναι αρκετά πιο δύσκολο να διατηρηθεί από το NAXSI. Αυτό κάνει το NAXSI μια απλή και προσαρμόσιμη επιλογή που παρέχει εύκολα διαθέσιμους κανόνες που λειτουργούν αποτελεσματικά με δημοφιλείς εφαρμογές ιστού όπως το WordPress.

1.2.1. Απλό σύνολο κανόνων

Το NAXSI προστατεύει τους ιστότοπους με ένα απλό σύνολο κανόνων που χρησιμοποιεί ένα σύστημα με βάση τα αποτελέσματα. Καταγράφει κάθε αίτηση url με ένα score (βαθμολογία). Όταν η βαθμολογία αυτή είναι μεγαλύτερη από την τιμή που έχει οριστεί στη διαμόρφωση, το NAXSI αποκλείει αυτόματα την αίτηση ιστότοπου. Τέλος, εάν η διεύθυνση URL αιτήματος περιέχει πιθανές κακόβουλες ενδείξεις όπως « **<** «, **/ [slash]** ή **drop,** αυτόματα αυξάνει το score. Αυτές οι διευθύνσεις URL είναι αποκλεισμένες από την εκτέλεση στο διακομιστή.

1.2.2. Υποστηρίζει Whitelist

Αναμφίβολα, συνιστά μεγάλο προτέρημα το γεγονός ότι το NAXSI επιτρέπει τη δημιουργία ενός συνόλου κανόνων whitelist. Για να διευκολυνθούν οι διαδικασίες προστασίας, το NAXSI συνοδεύεται από ένα εργαλείο που ονομάζεται Nxtool . Αυτό το εργαλείο μαθαίνει αυτόματα από την επισκεψιμότητα του ιστότοπου και δημιουργεί τη whitelist. Επιπλέον, εάν πάνω από το 20% των

χρηστών κάνουν trigger την ίδια αίτηση από τον ιστότοπο, θα καταγραφούν ως νόμιμες.

1.2.3. Αντοχή σε τεχνικές bypass WAF

Το NAXSI φροντίζει και παρέχει μία μεγάλη ποικιλία από τις πιθανές τεχνικές παράκαμψης όπως την κωδικοποίηση της διεύθυνσης url, τη σύζευξη των συμβολοσειρών στο αίτημα κλπ. Παρά τους εναλλακτικούς τρόπους που χρησιμοποιούν οι χάκερ, προκειμένου να δημιουργήσουν κενό ασφαλείας.

1.2.4. Γρήγορη & εύκολη στη συντήρηση

Το NAXSI δεν καταναλώνει μεγάλο μερίδιο πόρων ενός server. Επίσης, δεν χρειάζεται περιοδικές ενημερώσεις, όπως στο ModSecurity. Μόλις εγκατασταθεί, λειτουργεί συνεχώς χωρίς κανένα χρόνο διακοπής λειτουργίας.

Κεφάλαιο 2ο: Step-by-step περιγραφή επεξεργασίας

Προκειμένου να λειτουργήσει το Naxsi ως web application firewall system χρησιμοποιήθηκαν ποικίλα προγράμματα και βάσεις δεδομένων. Ειδικότερα, επιστρατεύθηκαν τα προγράμματα nginx, nxtool, elastic search, kibana και spike, στα οποία παρακάτω γίνεται μια σύντομη αναφορά. Όλα τα εν λόγω εργαλεία χρησιμοποιήθηκαν προς επιδίωξη του σκοπού της εργασίας, ωστόσο η εξελιγμένη χρήση του Naxsi ως web application firewall system, είναι αποτέλεσμα πολλαπλών και πολύπλοκων χρήσεων των κατωτέρω αναλυτικά περιγραφόμενων εργαλείων.

2.1. Nginx

Πρόκειται για έναν εξειδικευμένο HTTP, reverse proxy και mail proxy server. Ωστόσο, στην παρούσα εργασία χρησιμοποιείται και επεξεργάζεται περαιτέρω η πρώτη λειτουργία του. Αξιοσημείωτο είναι ότι οι περισσότερες nginx εκδόσεις δεν είναι συμβατές με το module Naxsi, έχοντας ως

αποτέλεσμα να επιβάλλεται η χειροκίνητη εγκατάσταση του Nginx. Συγκεκριμένα, μέσω του ανωτέρω προγράμματος υποβλήθηκαν οι εξής διαδικασίες:

2.1.1. Λήψη nginx/naxsi και μεταγλώττιση (compile) nginx

Λήψη Nginx

```
wget http://nginx.org/download/nginx-1.13.12.tar.gz
tar -vzxf nginx-1.13.12.tar.gz
```

Λήψη Naxsi

```
git clone https://github.com/nbs-system/naxsi.git
```

Κατεύθυνση & Μεταγλώτηση

```
cd nginx-1.13.12/
```

```
 ./configure –conf-path=/etc/nginx/nginx.conf \
  –add-module=../naxsi/naxsi_src/ \
  –error-log-path=/var/log/nginx/error.log \
  –http-client-body-temp-path=/var/lib/nginx/body \
```

```
  --http-fastcgi-temp-path=/var/lib/nginx/fastcgi \
  --http-log-path=/var/log/nginx/access.log \
  --http-proxy-temp-path=/var/lib/nginx/proxy \
  --lock-path=/var/lock/nginx.lock \
  --pid-path=/var/run/nginx.pid \
  --with-http_ssl_module \
  --with-http_v2_module \
  --with-http_gzip_static_module \
  --with-http_realip_module \
  --with-http_flv_module \
  --with-http_mp4_module \
  --without-mail_pop3_module \
  --without-mail_smtp_module \
  --without-mail_imap_module \
  --without-http_uwsgi_module \
  --without-http_scgi_module \
make && make install
```

2.1.2. Script εκκίνησης του Nginx

Path:

```
/etc/init.d/nginx
```

Προορισμένο περιεχόμενο για φάκελο nginx

```
#!/bin/bash
# nginx Startup script for the Nginx HTTP Server
nginxd=/usr/sbin/nginx

# config
```

```
6.  nginx_config=/etc/nginx/nginx.conf
7.
8.  # CentOS PID /run/
9.  nginx_pid==/run/nginx.pid
10.
11. RETVAL=0
12. prog="nginx"
13. # Source function library.
14. . /etc/rc.d/init.d/functions
15. # Source networking configuration.
16. . /etc/sysconfig/network
17. # Check that networking is up.
18. [ "${NETWORKING}" = "no" ] && exit 0
19. [ -x $nginxd ] || exit 0
20. # Start nginx daemons functions.
21. start() {
22. if [ -e $nginx_pid ];then
23.   echo "nginx already running...."
24.   exit 1
25. fi
26.   echo -n $"Starting $prog: "
27.   daemon $nginxd -c ${nginx_config}
28.   RETVAL=$?
29.   echo
30.   [ $RETVAL = 0 ] && touch /var/lock/subsys/ngi
    nx
31.   return $RETVAL
32. }
33. # Stop nginx daemons functions.
34. stop() {
35.     echo -n $"Stopping $prog: "
36.     killproc $nginxd
37.     RETVAL=$?
38.     echo
39.
```

```
40.     #PID
41.     [ $RETVAL = 0 ] && rm -
    f /var/lock/subsys/nginx /run/nginx.pid
42. }
43. # reload nginx service functions.
44. reload() {
45.   echo -n $"Reloading $prog: "
46.   #kill -HUP `cat ${nginx_pid}`
47.   killproc $nginxd -HUP
48.   RETVAL=$?
49.   echo
50. }
51. # See how we were called.
52. case "$1" in
53. start)
54.     start
55.     ;;
56. stop)
57.     stop
58.     ;;
59. reload)
60.     reload
61.     ;;
62. restart)
63.     stop
64.     start
65.     ;;
66. status)
67.     status $prog
68.     RETVAL=$?
69.     ;;
70. *)
71.     echo $"Usage: $prog {start|stop|restart|rel
    oad|status|help}"
72.     exit 1
```

```
73. esac
74. exit $RETVAL
```

Ορισμός δικαιώματος εκτέλεσης & εγγραφή ως υπηρεσία

```
1.  chmod a+x /etc/init.d/nginx
2.
3.  chkconfig –add nginx
```

Εκκίνηση υπηρεσίας nginx – Input

```
1.  /etc/init.d/nginx start
```

Έλεγχος λειτουργίας

Input / Output:

```
1.  [root@localhost ~]# /etc/init.d/nginx start
2.  Starting nginx (via systemctl):
    [  OK  ]
3.  [root@localhost ~]# ps aux | grep nginx
4.  root     10063  0.0  0.0  46324  1204 ?
    Ss   19:04   0:00 nginx: master process
    /usr/sbin/nginx -c /etc/nginx/nginx.conf
5.  nobody   10065  0.0  0.0  46732  1976 ?
    S    19:04   0:00 nginx: worker process
6.  root     13457  0.0  0.0 112708   980 pts/0
    R+   22:29   0:00 grep –color=auto nginx
```

```
1.  [root@localhost ~]# /etc/init.d/nginx stop
```

```
2.  Stopping nginx (via systemctl):
    [  OK  ]
3.  [root@localhost ~]# ps aux | grep nginx
4.  root     13554  0.0  0.0 112708   984 pts/0
    R+   22:35   0:00 grep –color=auto nginx
```

```
1.  [root@localhost ~]# service nginx start
2.  Starting nginx (via systemctl):
    [  OK  ]
3.  [root@localhost ~]#  ps aux | grep nginx
4.  root     13618  0.0  0.0  46324  1200 ?
    Ss   22:36   0:00 nginx: master process
    /usr/sbin/nginx -c /etc/nginx/nginx.conf
5.  nobody   13620  0.0  0.0  46732  1976 ?
    S    22:36   0:00 nginx: worker process
6.  root     13622  0.0  0.0 112708   984 pts/0
    S+   22:36   0:00 grep –color=auto nginx
```

```
7.  [root@localhost ~]# service nginx stop
```

```
8.  Stopping nginx (via systemctl):
    [  OK  ]
```

```
9.  [root@localhost ~]# ps aux | grep nginx
10. root     13655  0.0  0.0 112708   984 pts/0
    R+   22:37   0:00 grep –color=auto nginx
```

2.1.3. Βασική διαμόρφωση Nginx και Naxsi

Αντιγραφή των naxsi_core_rules στον κατάλογο nginx

```
cp /usr/local/naxsi/naxsi_config/naxsi_core.rules /etc/nginx/naxsi_core.rules
```

Path

```
/etc/nginx/nginx.conf
```

Τροποποιήσεις του nginx.conf

```
http {
      nclude       naxsi_core.rules;
     ...
}
```

```
location / {
   #naxsi
   SecRulesEnabled;
   #location, block
   #LearningMode;
   DeniedUrl "/50x.html";
   #CheckRules,
   CheckRule "$SQL >= 8" BLOCK;
   CheckRule "$RFI >= 8" BLOCK;
   CheckRule "$TRAVERSAL >= 4" BLOCK;
   CheckRule "$EVADE >= 4" BLOCK;
   CheckRule "$XSS >= 8" BLOCK;
   #naxsi
```

```
24.    error_log /var/log/nginx/security.log;
25.  }
```

2.1.4. Εγκατάσταση module GeoIP

Το python-geoip είναι μια βιβλιοθήκη που παρέχει πρόσβαση σε βάσεις δεδομένων GeoIP. Οι βάσεις δεδομένων GeoIP παρέχουν πληροφορίες μέχρι το επίπεδο χώρας / πόλης. Το GeoIP της Maxmind είναι μια διάσημη υπηρεσία για γεωγραφικές πληροφορίες IP διευθύνσεων μεταξύ προγραμματιστών. Στην προκειμένη εκτέλεση χρησιμοποιούμε το module της python, προκειμένου να εντοπιστεί και στη συνέχεια να εμφανιστεί με τη βοήθεια της Kibana στον χάρτη η προέλευση των επιθέσεων. Παρακάτω επισυνάπτονται οι αλλαγές που χρήζουν να γίνουν προκειμένου να ενσωματωθεί σωστά με τον Naxsi.

Λήψη & εγκατάσταση πακέτου

```
1.  yum install python-devel GeoIP-devel
2.  pip install geoip
```

Τροποποίηση αρχείου nxapi.json

```
1.  cd /usr/local/etc/nxapi.json
```

```
1. “naxsi” : {
2.      “rules_path” :
   “/etc/nginx/naxsi_core.rules”,
3.      “template_path” : [
   “/usr/local/nxapi/tpl/”],
4.      “geoipdb_path” :
   “/usr/local/nxapi/country2coords.txt”
5.      },
```

2.2. Nxtool/Nxapi

Επρόκειτο για ένα εργαλείο που βοηθάει στην εκχώρηση αλλά και εμφάνιση δεδομένων από και προς την Elastic. Ειδικότερα, με αυτό επιτυγχάνεται τόσο η εισαγωγή συμβάντων, δηλαδή η εκχώρηση γεγονότων Naxsi σε μια βάση δεδομένων της Elastic, όσο και η διαχείριση συμβάντων, δηλαδή η επισήμανση τους σε βάση δεδομένων από τη διαδικασία. Επιπλέον, έχει τη δυνατότητα να διαβάσει τα δεδομένα της elastic και να τα παρουσιάσει είτε σε μορφή στατιστικών είτε σε απλούστερη μορφή.

Επειδή στην προκειμένη εκτέλεση χρησιμοποιείται η έκδοση 5.4.0 της elastic search, τα αρχεία του nxtool χρίζουν παραμετροποίησης προκειμένου να

λειτουργήσει εύρυθμα η εφαρμογή. Παρακάτω επισυνάπτονται οι αλλαγές που πρέπει να γίνουν στα αρχεία nxtool.py, nxparse.py και nxtypificator.py.

Path:

```
1.  cd /usr/local/naxsi/nxapi
```

```
1.  vi nxtool.py
2.  import elasticsearch5 as elasticsearch
```

```
1.  vi nxapi/nxparse.py
2.  from elasticsearch5.helpers import bulk
```

```
1.  vi nxapi/nxtypificator.py
2.  from elasticsearch5 import Elasticsearch
```

Επιπρόσθετα η λειτουργία του nxtool γίνεται καλύτερα αντιληπτή μέσω των παρακάτω πινάκων, οι οποίοι συνιστούν παράλληλα βήματα που ακολουθήθηκαν στην παρούσα εργασία. Συγκεκριμένα, η ακόλουθη εντολή εμφανίζει τους

κορυφαίους διακομιστές, οι οποίοι έκαναν τις περισσότερες εξαιρέσεις. Επίσης, εμφανίζονται τα κορυφαία URI και οι ζώνες που έφεραν εξαιρέσεις:

Input:

```
nxtool.py -c /usr/local/etc/nxapi.json -x
```

Output:

```
1.Starting new HTTP connection (1): 127.0.0.1
GET http://127.0.0.1:9200/ [status:200 request: 0.025s]
# size :1000
# Whitelist(ing) ratio :
GET http://127.0.0.1:9200/nxapi/events/_search [status:200 request:0.074s]
# false 50.0% (total:3521/7042)
# Top servers :
GET http://127.0.0.1:9200/nxapi/events/_search [status:200 request:0.084s]
# 192.168.69.102 95.46% (total:3361/3521)
# 192.168.69.103 4.54% (total:160/3521)
# Top URI(s) :
GET http://127.0.0.1:9200/nxapi/events/_search [status:200 request:0.086s]
# / 85.35% (total:3005/3521)
# /index.asp 14.65% (total:516/3521)
# Top Zone(s) :
```

```
18. GET http://127.0.0.1:9200/nxapi/events/_search
    [status:200 request:0.046s]
19. # ARGS 95.57% (total:3365/3521)
20. # BODY 4.43% (total:156/3521)
21. # Top Peer(s) :
22. GET http://127.0.0.1:9200/nxapi/events/_search
    [status:200 request:0.037s]
23. # 192.168.69.100 71.85% (total:2530/3521)
24. # 192.168.69.103 23.6% (total:831/3521)
25. # 192.168.69.101 4.54% (total:160/3521)
26. # Top Country(ies) :
27. GET http://127.0.0.1:9200/nxapi/events/_search
    [status:200 request:0.046s]
28. # ZZ 100.0% (total:3521/3521)
```

Ακριβής στατιστικά στοιχεία για τον 192.168.69.101, προκειμένου να δημιουργηθεί κανόνας whitelist

Input:

```
nxtool.py -c /usr/local/etc/nxapi.json -
s 192.168.69.103 -f –filter ‘uri /’ –slack
```

Output:

```
1.  Starting new HTTP connection (1): 127.0.0.1
2.  GET http://127.0.0.1:9200/ [status:200 request:
    0.017s]
3.  # size :1000
4.  GET http://127.0.0.1:9200/nxapi/events/_search
    [status:200 request:0.043s]
5.  GET http://127.0.0.1:9200/nxapi/events/_search
    [status:200 request:0.009s]
```

```
6.  GET http://127.0.0.1:9200/nxapi/events/_search
    [status:200 request:0.003s]
7.  #  template :/usr/local/nxapi/tpl/BODY/precise-
    id.tpl
8.  Nb of hits : 0
9.  #  template :/usr/local/nxapi/tpl/BODY/site-
    wide-id.tpl
10. Nb of hits : 0
11. #  template :/usr/local/nxapi/tpl/BODY/url-
    wide-id-BODY-NAME.tpl
12. Nb of hits : 0
13. #  template :/usr/local/nxapi/tpl/BODY/url-
    wide-id.tpl
14. Nb of hits : 0
15. #  template :/usr/local/nxapi/tpl/BODY/var_name
    -wide-id.tpl
16. Nb of hits : 0
17. #  template :/usr/local/nxapi/tpl/ARGS/precise-
    id.tpl
18. Nb of hits : 255
19. #  template matched, generating all rules.
20. 1 whitelists ...
21. #msg: A generic, precise wl tpl (url+var+id)
22. #Rule (1302) html open tag
23. #total hits 255
24. #peers : 192.168.69.101
25. #country : ZZ
26. #uri : /
27. #var_name : test
28.
29. BasicRule  wl:1302 "mz:$URL:/|$ARGS_VAR:test";

30. #  template :/usr/local/nxapi/tpl/ARGS/site-
    wide-id.tpl
31. Nb of hits : 255
```

```
32. #  template matched, generating all rules.
33. 1 whitelists ...
34. #msg: A generic, wide (id+zone) wl
35. #Rule (1302) html open tag
36. #total hits 255
37. #peers : 192.168.69.101
38. #country : ZZ
39. #uri : /
40. #var_name : test
41.
42. BasicRule  wl:1302 “mz:ARGS”;
43. #  template :/usr/local/nxapi/tpl/ARGS/url-
    wide-id-NAME.tpl
44. Nb of hits : 0
45. #  template :/usr/local/nxapi/tpl/ARGS/url-
    wide-id.tpl
46. Nb of hits : 255
47. #  template matched, generating all rules.
48. 1 whitelists ...
49. #msg: A generic whitelist, true for the whole u
    ri
50. #Rule (1302) html open tag
51. #total hits 255
52. #peers : 192.168.69.101
53. #country : ZZ
54. #uri : /
55. #var_name : test
56.
57. BasicRule  wl:1302 “mz:$URL:/|ARGS”;
58. #  template :/usr/local/nxapi/tpl/URI/global-
    url-0x_in_pircutres.tpl
59. Nb of hits : 0
60. #  template :/usr/local/nxapi/tpl/URI/site-
    wide-id.tpl
61. Nb of hits : 0
```

```
62. #  template :/usr/local/nxapi/tpl/URI/url-wide-id.tpl
63. Nb of hits : 0
64. #  template :/usr/local/nxapi/tpl/APPS/google_analytics-ARGS.tpl
65. Nb of hits : 0
66. #  template :/usr/local/nxapi/tpl/HEADERS/cookies.tpl
67. Nb of hits : 0
```

Στο nginx.conf έχει οριστεί ο naxsi να αποθηκεύει όλα τα logs του στο path:

```
1.  /var/log/nginx/security.conf
```

Εν συνεχεία επισυνάπτονται τα logs του naxsi απο το Security log:

```
2.  2019/09/01 17:42:29 [error] 8660#0: *23 NAXSI_FMT: ip=192.168.69.101&server=192.168.69.102&uri=/&vers=0.56&total_processed=6&total_blocked=6&config=block&cscore0=$XSS&score0=8&zone0=ARGS&id0=1302&var_name0=test, client: 192.168.69.101, server: localhost, request: “GET /?test=%3C%3E HTTP/1.1”, host: “192.168.69.102”
3.  2019/09/01 17:42:29 [error] 8660#0: *23 NAXSI_FMT: ip=192.168.69.101&server=192.168.69.102&uri=/&vers=0.56&total_processed=7&total_blocked=7&config=block&cscore0=$XSS&score0=8&zone0=ARGS&id0=1302&var_name0=test, client: 192.168.69.101,
```

```
   server: localhost, request: “GET /?test=%3C%3E
   HTTP/1.1”, host: “192.168.69.102”
4. 2019/09/01 17:42:29 [error] 8660#0: *23
   NAXSI_FMT:
   ip=192.168.69.101&server=192.168.69.102&uri=/&v
   ers=0.56&total_processed=8&total_blocked=8&conf
   ig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0=1
   302&var_name0=test, client: 192.168.69.101,
   server: localhost, request: “GET /?test=%3C%3E
   HTTP/1.1”, host: “192.168.69.102”
5. 2019/09/01 17:42:30 [error] 8660#0: *23
   NAXSI_FMT:
   ip=192.168.69.101&server=192.168.69.102&uri=/&v
   ers=0.56&total_processed=9&total_blocked=9&conf
   ig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0=1
   302&var_name0=test, client: 192.168.69.101,
   server: localhost, request: “GET /?test=%3C%3E
   HTTP/1.1”, host: “192.168.69.102”
6. 2019/09/01 17:42:30 [error] 8660#0: *23
   NAXSI_FMT:
   ip=192.168.69.101&server=192.168.69.102&uri=/&v
   ers=0.56&total_processed=10&total_blocked=10&co
   nfig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0
   =1302&var_name0=test, client: 192.168.69.101,
   server: localhost, request: “GET /?test=%3C%3E
   HTTP/1.1”, host: “192.168.69.102”
7. 2019/09/01 17:42:30 [error] 8660#0: *23
   NAXSI_FMT:
   ip=192.168.69.101&server=192.168.69.102&uri=/&v
   ers=0.56&total_processed=11&total_blocked=11&co
   nfig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0
   =1302&var_name0=test, client: 192.168.69.101,
   server: localhost, request: “GET /?test=%3C%3E
   HTTP/1.1”, host: “192.168.69.102”
```

```
8.  2019/09/01 17:42:31 [error] 8660#0: *23
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=12&total_blocked=12&co
    nfig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0
    =1302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
9.  2019/09/01 17:42:31 [error] 8660#0: *23
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=13&total_blocked=13&co
    nfig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0
    =1302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
10. 2019/09/01 17:42:32 [error] 8660#0: *23
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=14&total_blocked=14&co
    nfig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0
    =1302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
11. 2019/09/01 17:42:32 [error] 8660#0: *23
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=15&total_blocked=15&co
    nfig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0
    =1302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
12. 2019/09/01 17:42:32 [error] 8660#0: *23
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
```

```
    ers=0.56&total_processed=16&total_blocked=16&co
    nfig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0
    =1302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
13. 2019/09/01 17:42:33 [error] 8660#0: *23
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=17&total_blocked=17&co
    nfig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0
    =1302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
14. 2019/09/01 17:42:51 [error] 8701#0: *1
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=1&total_blocked=1&conf
    ig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0=1
    302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
15. 2019/09/01 17:42:52 [error] 8701#0: *1
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=2&total_blocked=2&conf
    ig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0=1
    302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
16. 2019/09/01 17:42:53 [error] 8701#0: *1
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=3&total_blocked=3&conf
    ig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0=1
    302&var_name0=test, client: 192.168.69.101,
```

```
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
17. 2019/09/01 17:43:45 [error] 8701#0: *5
    NAXSI_FMT:
    ip=192.168.69.101&server=192.168.69.102&uri=/&v
    ers=0.56&total_processed=6&total_blocked=4&conf
    ig=block&cscore0=$XSS&score0=8&zone0=ARGS&id0=1
    302&var_name0=test, client: 192.168.69.101,
    server: localhost, request: “GET /?test=%3C%3E
    HTTP/1.1”, host: “192.168.69.102”
```

Παρατηρούμε ότι μας παρέχει όλες τις απαραίτητες πληροφορίες σχετικά με το request που έγιναν στο web server, όπως το ID, το server, client’s ip, σώμα του url.

- ip : Η ip του χρήστη.
- server : Η ip του Server.
- uri: Το ερωτηθέν URI (χωρίς μεταβλητές, σταματάει στο σύμβολο ?).
- learning: Μας ορίζει αν ο Naxsi ήταν σε learning mode (0/1).
- vers : Η έκδοση του Naxsi.
- total_processed: Ο συνολικός αριθμός των αιτήσεων, που αναλύθηκαν απο τον nginx.

- total_blocked: Ο συνολικός αριθμός που αποκλείστηκαν απο τον naxsi.
- zoneN: Ζώνη στην οποία αποκλείστηκε το request.
- cscoreN : όνομα βαθμολογίας.
- scoreN : σχετική *ονομασία* τιμής βαθμολογίας.

Τα επισυναπτώμενα logs στη συνέχεια, φορτώνονται στην ElasticSearch με τη βοήθεια του nxtool.py προκειμένου να αποθηκευτούν και να αναζητηθούν στο μελλον.

Input:

```
nxtool.py -c /usr/local/etc/nxapi.json –
files=/var/log/nginx/security.log
```

Output

```
18. Starting new HTTP connection (1): 127.0.0.1
19. GET http://127.0.0.1:9200/ [status:200 request:
    0.012s]
20. # size :1000
21. PUT http://127.0.0.1:9200/nxapi [status:400 req
    uest:0.010s]
```

```
22. PUT http://127.0.0.1:9200/nxapi/_mapping/events
     [status:200 request:0.023s]
23. List of files :['/var/log/nginx/security.log']

24. Importing file /var/log/nginx/security.log
25. log open
26. POST http://127.0.0.1:9200/nxapi/events/_bulk [
    status:200 request:1.029s]
27. POST http://127.0.0.1:9200/nxapi/events/_bulk [
    status:200 request:0.295s]
28. Written 385 events
```

2.3. Crone job

Η ανάγκη να κάνουμε import τα logs του naxsi στην elastic κατά τακτά χρονικά διαστήματα, μας οδηγεί στην χρήση του crone job, καθώς είναι αδύνατον να επαναλαμβάνουμε αυτή τη διαδικασία χειροκίνητα.

Η χρησιμότητα του λογισμικού **cron** είναι με βάση το χρονοδιάγραμμα εργασίας σε Unix-όπως υπολογιστής λειτουργικά συστήματα . Οι χρήστες χρησιμοποιούν το cron για να προγραμματίσουν εργασίες, εντολες κ.α, για να εκτελούνται περιοδικά σε καθορισμένους χρόνους, ημερομηνίες ή διαστήματα. Συνήθως

αυτοματοποιεί τη συντήρηση ή τη διαχείριση του συστήματος – αν και η γενική χρήση του καθιστά χρήσιμη για πράγματα όπως η λήψη αρχείων από το Internet και η λήψη μηνυμάτων ηλεκτρονικού ταχυδρομείου σε τακτά χρονικά διαστήματα.

Το Cron ορίζεται από ένα **αρχείο crontab**, ένα αρχείο διαμόρφωσης που καθορίζει τις εντολές για να εκτελούνται περιοδικά σε ένα συγκεκριμένο χρονοδιάγραμμα. Τα αρχεία crontab αποθηκεύονται όπου διατηρούνται οι λίστες εργασιών και άλλες οδηγίες στο δαίμονα cron . Οι χρήστες μπορούν να έχουν τα δικά τους μεμονωμένα αρχεία crontab και συχνά υπάρχει ένα αρχείο crontab σε ολόκληρο το σύστημα (συνήθως σε `/etc` ή σε έναν υποκατάλογο του `/etc`) που μπορούν να επεξεργαστούν μόνο οι διαχειριστές του συστήματος.

Στον παρακάτω πίνακα περιγράφεται η σύνταξη του cron job, προκειμένου να λειτουργήσει εύρυθμα η εντολή που θα του αναθέσουμε.

Entry	Description	Equivalent to
@yearly (or @annually)	Run once a year at midnight of 1 January	0 0 1 1 *
@monthly	Run once a month at midnight of the first day of the month	0 0 1 * *
@weekly	Run once a week at midnight on Sunday morning	0 0 * * 0
@daily (or @midnight)	Run once a day at midnight	0 0 * * *
@hourly	Run once an hour at the beginning of the hour	0 * * * *
@reboot	Run at startup	N/A

(Πιν.2)

Comma (,) : Χρησιμοποιούνται για να διαχωρίσουν τα στοιχεία μιας λίστας. Για παράδειγμα, χρησιμοποιώντας το «MON, WED, FRI»

Hyphen (-) : Τα παραδείγματα ορίζουν εύρη. Για παράδειγμα, το 2000-2010 αναφέρει κάθε χρόνο μεταξύ του 2000 και του 2010, συμπεριλαμβανομένου.

Percent (%) : Τα ποσοστά (%) της εντολής, εκτός αν ξεφύγουν με την αντίστροφη κάθετο (\), μετατρέπονται σε χαρακτήρες νέας γραμμής και όλα τα δεδομένα μετά το πρώτο% αποστέλλονται στην εντολή ως τυπική είσοδο.

(L): Το «L» σημαίνει «τελευταίο». Όταν χρησιμοποιείται στο πεδίο της ημέρας της εβδομάδας, σας επιτρέπει να καθορίσετε κατασκευές όπως «την τελευταία Παρασκευή» («5L») ενός δεδομένου μήνα

(W): Ο χαρακτήρας 'W' επιτρέπεται για το πεδίο ημέρας του μήνα. Αυτός ο χαρακτήρας χρησιμοποιείται για να καθορίσει την ημέρα της ημέρας (Δευτέρα-Παρασκευή) πλησιέστερη της δεδομένης ημέρας. Για παράδειγμα, εάν ορίσετε «15W» ως τιμή για το πεδίο της ημέρας του μήνα, η έννοια είναι: «την πλησιέστερη εβδομάδα έως τις 15 του μήνα.»

Hash (#): Το '#' επιτρέπεται για το πεδίο της ημέρας της εβδομάδας και πρέπει να ακολουθείται από έναν αριθμό μεταξύ ενός και πέντε.

Question mark (?): Σε ορισμένες εφαρμογές, χρησιμοποιείται αντί για '*' για να φύγει είτε κενή ημέρα ή μήνα της εβδομάδας. Άλλες εφαρμογές cron αντικαθιστούν το «?» με το χρόνο έναρξης του δαίμονα cron

Slash (/) : οι λοξές γραμμές μπορούν να συνδυαστούν με τα διαστήματα για να καθορίσουν τις τιμές βημάτων.

(H): Το 'H' χρησιμοποιείται στο σύστημα συνεχές για να υποδείξει ότι αντικαθίσταται μια τιμή «hashed». Έτσι, αντί για '20 * * * * ' που σημαίνει σε 20 λεπτά μετά την ώρα κάθε ώρα, το' H * * * * 'υποδεικνύει ότι η εργασία εκτελείται κάθε ώρα.

Ένα παράδειγμα μίας εργασίας crone job είναι η κάτωθεν, όπου τρέχει το test_dump.sh 23:45 (11:45 PM) κάθε Κυριακή.

```
1.  45 23 * * 6 /etc/test1/scripts/test_dump.sh
```

Πιο συγκεκριμένα στην υλοποίηση του naxsi, το crone job προκειμένου να επαναλαμβάνεται η

διαδικασία log import στην elastic είναι η ακόλουθη. Δημιουργούμε ένα αρχείο timer.sh και εν συνεχεία το εκτελούμε κάθε 5 λεπτά επ αορίστου. Η χρήση αρχείου shell script γίνεται προκειμένου να συγκεντρωθούν μέσα σε αυτό όλες οι ενέργειες του application που τυχόν χρίζουν crone job.

```
2.  #!/usr/bin/bash
3.
4.  nxtool.py -c /usr/local/etc/nxapi.json –
    files=/var/log/nginx/security.log
5.
6.  exit 1
```

Εν συνεχεία στο path /tmp/crontab.NyR2UQ/crontab προσθέτουμε την παρακάτω εντολή εκτέλεσης του timer.sh script κάθε 5 λεπτα.

```
1.  */5 * * * * /root/timer.sh
```

2.4. Elastic Search

Τα προϊόντα που υπάρχουν στο ηλεκτρονικό εμπόριο και τις μηχανές αναζήτησης με τεράστιες βάσεις δεδομένων αντιμετωπίζουν προβλήματα, συμπεριλαμβανομένης της ανάκτησης πληροφοριών. Αυτό οδηγεί σε κακή εμπειρία χρήστη. Το Lag in search αποδίδεται στη σχεσιακή βάση δεδομένων που χρησιμοποιείται για το σχεδιασμό του προϊόντος, όπου τα δεδομένα είναι διασκορπισμένα μεταξύ πολλών πινάκων και η επιτυχής ανάκτηση σημαντικών πληροφοριών χρήστη απαιτεί την ανάκτηση των δεδομένων από αυτούς τους πίνακες.

Η σχεσιακή βάση δεδομένων λειτουργεί συγκριτικά αργά όταν πρόκειται για τεράστια δεδομένα και για την ανάκτηση αποτελεσμάτων αναζήτησης μέσω ερωτημάτων βάσης δεδομένων. Οι επιχειρήσεις, οι βιομηχανίες αλλά και οι χρήστες, αναζητούν σήμερα εναλλακτικές λύσεις αποθήκευσης δεδομένων με σκοπό την γρήγορη ανάκτηση και αναζήτηση δεδομένων.

Αυτές τις αδυναμίες και τα προβλήματα τα φέρει εις πέρας σε μεγάλο βαθμό η Elastic. Η elastic

μπορεί να χρησιμοποιηθεί για την αναζήτηση όλων των ειδών εγγράφων.

Η ElasticSearch είναι μια βάση δεδομένων που έχει σχεδιαστεί για την αποθήκευση, ανάκτηση και διαχείριση εγγράφων ή ημι-δομημένων δεδομένων. Όταν χρησιμοποιείτε το ElasticSearch, τα δεδομένα αποθηκεύονται σε μορφή εγγράφου JSON και στη συνέχεια, δημιουργείται ένα query για την ανάκτηση τους.

Κάθε χαρακτηριστικό της Elasticsearch εκτίθεται ως REST API. Πιο συγκεκριμένα, με τον όρο REST API, ορίζεται ο τρόπος με τον οποίο οι εφαρμογές επικοινωνούν μέσω του πρωτοκόλλου HTTP. Οι εφαρμογές που χρησιμοποιούν το REST μεταφέρουν πληροφορίες γρήγορα και αποτελεσματικά. Παρόλο που το REST δεν καθορίζει μορφές δεδομένων, συσχετίζεται συνήθως με την ανταλλαγή εγγράφων JSON ή XML μεταξύ ενός πελάτη και ενός διακομιστή.

Όσον αφορά την elastic χρησιμοποιεί τα παρακάτω REST API:

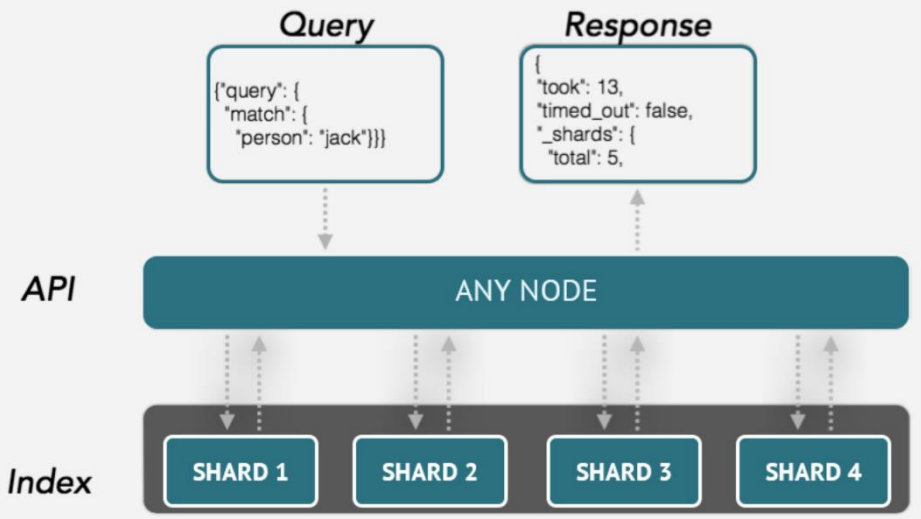
1. **Index API**: Χρησιμοποιείται για την τεκμηρίωση του ευρετηρίου.

2. **Get API**: Χρησιμοποιείται για την ανάκτηση του εγγράφου.

3. **Search API**: Χρησιμοποιείται για την υποβολή του ερωτήματος και τη λήψη ενός αποτελέσματος.

4. **Put Mapping API**: Χρησιμοποιείται για να αντικαταστήσει τις προεπιλεγμένες επιλογές και να καθορίσει το mapping.

Στο παρακάτω σχεδιάγραμμα, απεικονίζεται η μορφή ενός query προς την ElasticSearch, καθώς και η απάντησης της.



(Σχ.2)

Στην περίπτωση του Naxsi, η ElasticSearch, αποτελεί ένα απο τα πλέον χρήσιμα εργαλεία της υλοποίησης αυτής, καθώς όλα τα logs των

επιθέσεων και των request που καταγράφονται μέσα σε αυτή, πολύ εύκολα αναζητούνται και καταγράφονται με αποτέλεσμα να γίνεται πολύ ευκολότερη η χρήση του application firewall αυτού. Εν προκειμένω, χρησιμοποιείται η έκδοση 5.4.0 ως η πλέον συμβατή με το Naxsi.

Input:

```
curl “localhost:9200”
```

Output:

```
{
  “name” : “LwFqZuN”,
  “cluster_name” : “elasticsearch”,
  “cluster_uuid” : “B54C4fyxTYucj4WhVf8N4Q”,
  “version” : {
    “number” : “5.4.0”,
    “build_hash” : “780f8c4”,
    “build_date” : “2017-04-28T17:43:27.229Z”,
    “build_snapshot” : false,
    “lucene_version” : “6.5.0”
  },
  “tagline” : “You Know, for Search”
}
```

2.5. Kibana

Το Kibana είναι μια πλατφόρμα ανάλυσης και απεικόνισης, η οποία επιτρέπει να απεικονιστούν εύκολα τα δεδομένα από την ElasticSearch και να τα αναλυθούν με σκοπό την καλύτερη κατανόηση τους όπως ιστογράμματα, γραφήματα γραμμών, χάρτες κ.α. Μπορούν πολύ εύκολα να συγκεντρωθούν στοιχεία που αφορούν τους top talkers ή να κατηγοριοποιηθούν διαφόρων τύπων δεδομένα προκειμένου να διευκολύνουν τον χρήστη στη διεξαγωγή συμπερασμάτων. Επιπλέον, κάνει ευκολότερη την κατανόηση μεγάλων όγκων δεδομένων, ενώ αξίζει να σημειωθεί η δυνατότητα για γρήγορη δημιουργία δυναμικών dashboards που παρουσιάζουν αλλαγές στη βάση της ElasticSearch σε πραγματικό χρόνο. Το Kibana παρέχει επίσης ένα UI για τη διαχείριση όσον αφορά την Elasticsearch. Ουσιαστικά, το Kibana χαρακτηρίζεται ως ένα Web interface με τα δεδομένα που είναι αποθηκευμένα στην Elasticsearch. Σημαντικό είναι να επισημανθεί ότι χρησιμοποιεί τα δεδομένα από το ElasticSearch και απλά στέλνει ερωτήματα στην ElasticSearch χρησιμοποιώντας το ίδιο API REST. Η έκδοση που χρησιμοποιείται στην παρούσα υλοποίηση είναι η 5.4.0, καθώς απαιτείται συμβατότητα με την έκδοση της ElasticSearch.

Λήψη του αρχείου kibana rpm εκδοσης 5.4.0

```
1. wget
   https://artifacts.elastic.co/downloads/kib
   ana/kibana-5.4.0-linux-x86_64.tar.gz
```

Unzip του αρχείου kibana-5.4.0-linux-x86_64.tar.gz

```
1. tar -vxf kibana-5.4.0-linux-x86_64.tar.gz
```

Επεξεργασία του αρχείου kibana.yml προκειμένου να ορίσουμε την πόρτα που θα ακούει το Service αλλα και τις Ip που θα έχουν πρόσβαση. Στη συγκεκριμένη υλοποίηση δίνεται δικαίωμα σε όλους να μπορούν να έχουν πρόσβαση και γι αυτό το λόγο θέτουμε τα 0.0.0.0 στο host:

```
1. cd kibana-5.4.0-linux-
   x86_64/config/kibana.yml
```

```
1. vi kibana.yml
2. # Kibana is served by a back end server.
   This setting specifies the port to use.
```

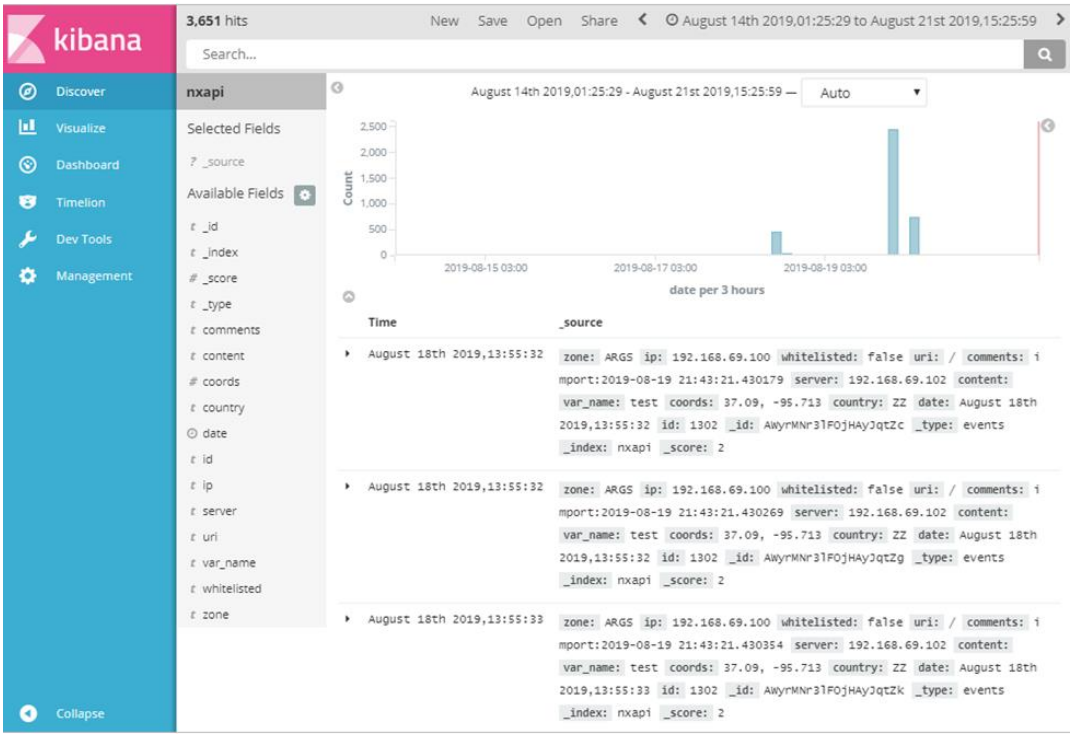
```
3. server.port: 5601
4. # To allow connections from remote users,
   set this parameter to a non-loopback
   address.
5. server.host: “0.0.0.0”
6. # The URL of the Elasticsearch instance to
   use for all your queries.
```

Στην παρακάτω απεικόνιση ενεργοποιούμε το kibana.service και ελέγχουμε για τυχών λάθη που μπορούν να προκύψουν.

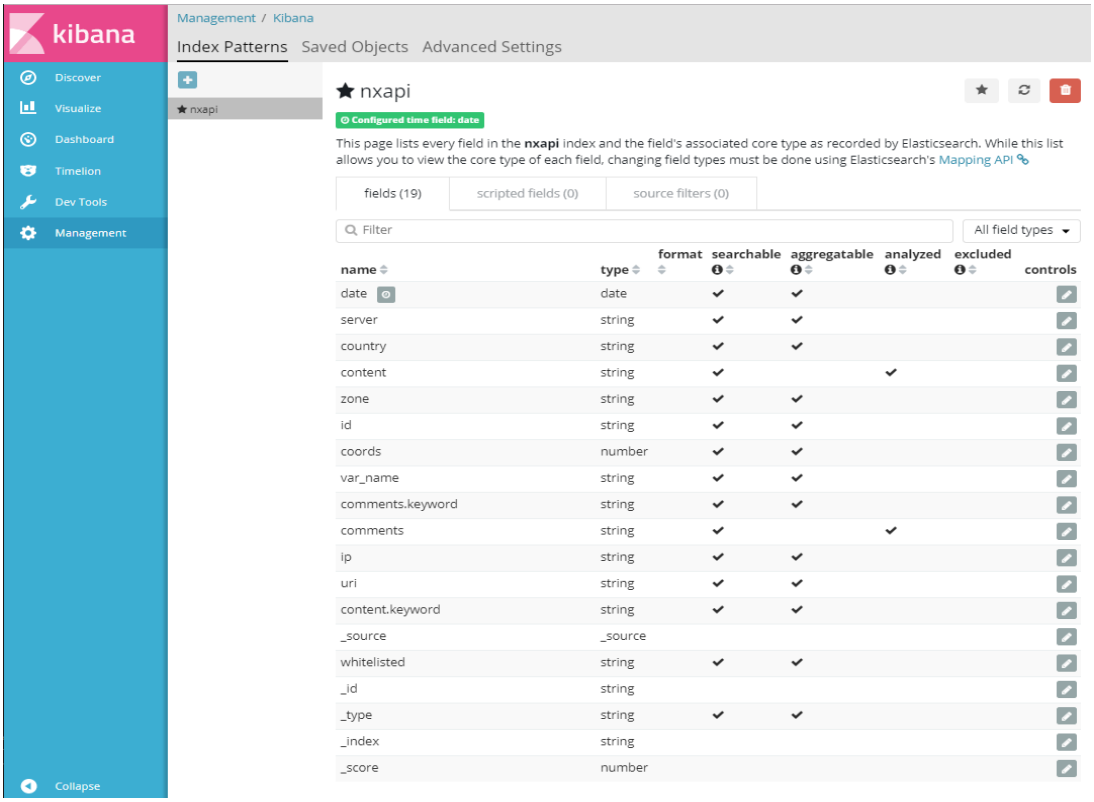
```
7. elasticsearch.url: http://localhost:9200
8.  cd kibana-5.4.0-linux-x86_64/
9. [root@localhost kibana-5.4.0-linux-
   x86_64]# ./bin/kibana
10.  log   [20:26:39.746]
   [info][status][plugin:kibana@5.4.0] Status
   changed from uninitialized to green –
   Ready
11.  log   [20:26:39.873]
   [info][status][plugin:elasticsearch@5.4.0]
   Status changed from uninitialized to
   yellow – Waiting for Elasticsearch
12.  log   [20:26:39.934]
   [info][status][plugin:console@5.4.0]
   Status changed from uninitialized to green
   – Ready
13.  log   [20:26:39.950]
   [info][status][plugin:metrics@5.4.0]
```

```
   Status changed from uninitialized to green
   – Ready
14.  log   [20:26:40.321]
   [info][status][plugin:timelion@5.4.0]
   Status changed from uninitialized to green
   – Ready
15.  log   [20:26:40.326] [info][listening]
   Server running at http://0.0.0.0:5601
16.  log   [20:26:40.327] [info][status][ui
   settings] Status changed from
   uninitialized to yellow – Elasticsearch
   plugin is yellow
17.  log   [20:26:40.618]
   [info][status][plugin:elasticsearch@5.4.0]
   Status changed from yellow to green –
   Kibana index ready
18.  log   [20:26:40.618] [info][status][ui
   settings] Status changed from yellow to
   green – Ready
```

Τέλος αφότου η πρόσβαση στο “localhost:5601” είναι επιτυχής και προκειμένου να απεικονιστούν τα δεδομένα της ElasticSearch, χρίζει αναγκαίο να τεθεί ως «index name or pattern» η λέξη «nxapi» και στη συνέχεια, επιλέγεται η ένδειξη «create».


kibana
3,651 hits
New Save Open Share
August 14th 2019,01:25:29 to August 21st 2019,15:25:59
Search...
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
Collapse
nxapi
Selected Fields
_source
Available Fields
_id
_index
_score
_type
comments
content
coords
country
date
id
ip
server
uri
var_name
whitelisted
zone
August 14th 2019,01:25:29 - August 21st 2019,15:25:59 — Auto
Count
2019-08-15 03:00
2019-08-17 03:00
2019-08-19 03:00
date per 3 hours
Time
_source
August 18th 2019,13:55:32
August 18th 2019,13:55:32
August 18th 2019,13:55:33

(Εικ.1)


kibana
Management / Kibana
Index Patterns
Saved Objects
Advanced Settings
nxapi
Configured time field: date
This page lists every field in the nxapi index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API
fields (19)
scripted fields (0)
source filters (0)
Filter
All field types
name
type
format
searchable
aggregatable
analyzed
excluded
controls
date
server
country
content
zone
id
coords
var_name
comments.keyword
comments
ip
uri
content.keyword
_source
whitelisted
_id
_type
_index
_score

(Εικ.2.)

2.6. Spike

Το Spike αποτελεί ένα εργαλείο για τη διαχείριση κανόνων του Naxsi. Οι κανόνες αυτοί αποθηκεύονται σε μια βάση sqlite και υπάρχει η δυνατότητα να προστεθούν, διαγραφούν, τροποποιηθούν, αναζητηθούν ή ακόμα να εισαχθούν και εξαχθούν σε απλό κείμενο.

Setup

```
1. git clone https://github.com/nbs-system/spike
2. pip install -r requirements.txt
3. python ./spike-server.py init
4. python ./spike-server.py run
```

Output:

```
1. /root/spike/spike/__init__.py:6: ExtDeprecation
   Warning: Importing flask.ext.bootstrap is depre
   cated, use flask_bootstrap instead.
2.   from flask.ext.bootstrap import Bootstrap
3. /root/spike/spike/model/__init__.py:1: ExtDepre
   cationWarning: Importing flask.ext.sqlalchemy i
   s deprecated, use flask_sqlalchemy instead.
4.   from flask.ext.sqlalchemy import SQLAlchemy
5. Spike app.init()
6. /usr/lib64/python3.6/site-
   packages/flask_sqlalchemy/__init__.py:835: FSAD
   eprecationWarning: SQLALCHEMY_TRACK_MODIFICATIO
   NS adds significant overhead and will be disabl
```

```
    ed by default in the future.  Set it to True or
     False to suppress this warning.
7.    'SQLALCHEMY_TRACK_MODIFICATIONS adds signific
    ant overhead and '
8.  /usr/lib64/python3.6/site-
    packages/flask_sqlalchemy/__init__.py:835: FSAD
    eprecationWarning: SQLALCHEMY_TRACK_MODIFICATIO
    NS adds significant overhead and will be disabl
    ed by default in the future.  Set it to True or
     False to suppress this warning.
9.    'SQLALCHEMY_TRACK_MODIFICATIONS adds signific
    ant overhead and '
10.  * Running on http://127.0.0.1:5555/ (Press CTR
    L+C to quit)
```

Putting Spike behind Nginx

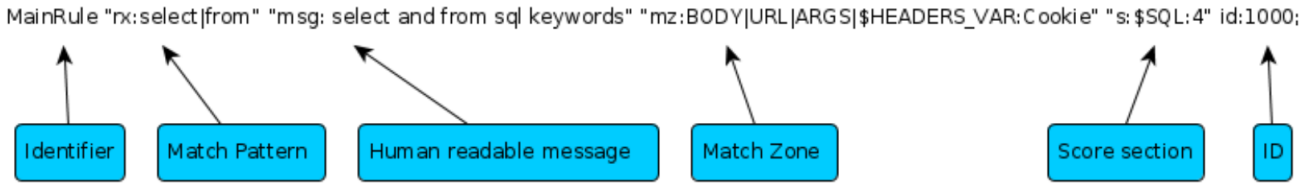
```
1.  server {
2.      server_tokens off;
3.      listen 443 ssl;
4.      server_name spike.nginx-goodies.com  ;
5.
6.      proxy_set_header    X-Forwarded-
    For   $proxy_add_x_forwarded_for;
7.      proxy_set_header    X-Real-
    IP         $remote_addr;
8.      proxy_set_header    Host              $host
    ;
9.      proxy_set_header    X-Forwarded-
    Proto $scheme;
10.
11.     access_log  /var/log/nginx/spike.access.log
    ;
12.     error_log  /var/log/nginx/error.log;
13.
```

```
14.     root /var//www/spike;
15.
16.     location  /static {
17.         autoindex off;
18.         expires 1d;
19.     }
20.
21.     location / {
22.         proxy_cache off;
23.         proxy_redirect off;
24.         proxy_pass http://127.0.0.1:5555;
25.         expires off;
26.         include /etc/nginx/doxi-rules/active-
    mode.rules;
27.         include /etc/nginx/doxi-
    rules/local.rules;
28.         include /etc/nginx/doxi-rules/spike-
    wl.rules;
29.     }
30. }
```

Επιβάλλεται να επισημανθεί ότι το συγκεκριμένο πρόγραμμα τυγχάνει εφαρμογής μόνο μέσω του Python3 και δεν θα πρέπει να εγκαθίσταται σε κοινόχρηστο server, καθώς δεν παρέχει κάποιο είδος προστασίας και χρησιμοποιείται αποκλειστικά ως βοήθημα διαχείρισης κανόνων.

Κεφάλαιο 3ο: Κανόνες Naxsi

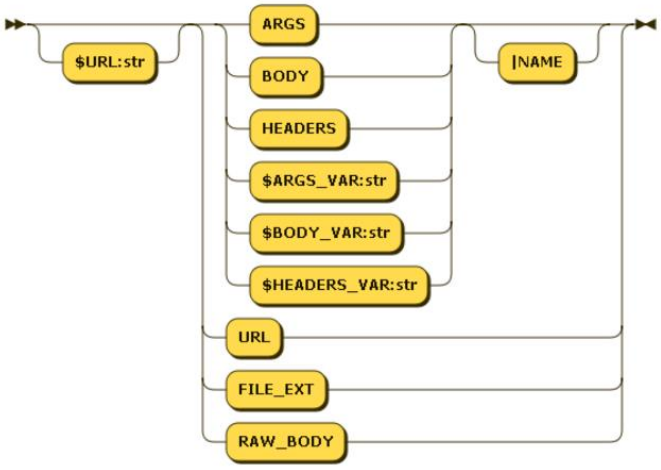
Οι κανόνες του NAXSI έχουν απλό σχεδιασμό και απαρτίζονται από τρεις βασικούς τύπους. Το ***MainRule*** ορίζει ένα πρότυπο ανίχνευσης και βαθμολογίες. Το ***BasicRule*** ορίζει λευκές λίστες για ένα MainRule. Το ***CheckRule*** ορίζει τις ενέργειες όταν επιτυγχάνεται μια βαθμολογία.



(Σχ.3)

1. MainRule: είναι ένα αναγνωριστικό που σηματοδοτεί κανόνες ανίχνευσης, σε αντίθεση με τους BasicRules, οι οποίοι συνήθως χρησιμοποιούνται για Whitelisted κάποιων MainRule.
2. Match Pattern: υποστηρίζει δύο τύπους: (i) **rx** (regular expression) και (ii) **str** (string matcher). Να σημειωθεί ότι το string matcher είναι εμφανώς ταχύτερο από το regular expression, που το καθιστά αυτομάτως προτειμώμενο.

3. MSG: Εμπεριέχει ένα μήνυμα, αναγνώσιμο από τον άνθρωπο και περιγράφει συνήθως το πρότυπο.
4. Match Zone: υπάρχουν σε κανόνες και whitelists. Χρησιμοποιείται για να διευκρινιστεί πού πρέπει να αναζητηθούν (κανόνες) ή πού πρέπει να επιτραπεί (whitelist).Υπάρχουν 4 βασικές κατηγορίες match zones:



(Σχ.4)

- ARGS: GET args
- HEADERS: HTTP Headers
- BODY: POST args (and RAW_BODY)
- URL: The URL itself (before '?')

Όσον αφορά την κατηγορία URL, μας δίνεται η δυνατότητα να διαχωρίσουμε ακόμα 2 κατηγορίες:

- $URL:string: restricted to this url
- $URL_X:regex: restricted to url matching regex (>= 0.52)

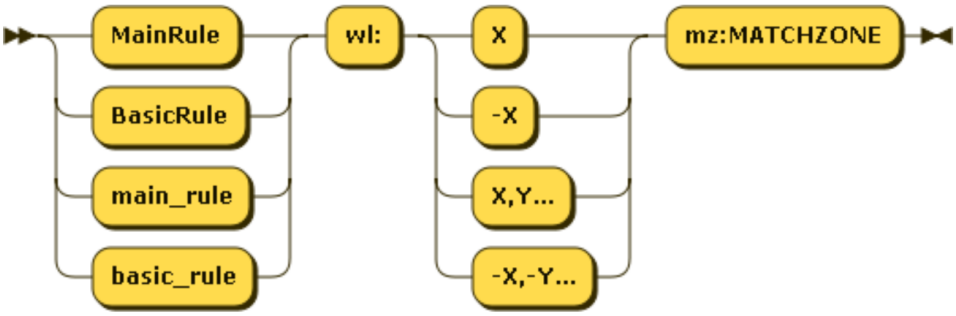
Τα Matchzones που αποτελούνται από στατικά $*_VAR: $URL:ζεύγη ( ), αποθηκεύονται σε hashtables, και έτσι είναι βέλτιστα. Αντίθετα, οι ζώνες αντιστοίχισης Regex ($*_VAR_X: $URL_X😊 απαιτούν περισσότερη επεξεργασία χρόνου εκτέλεσης.

5. Score Section: Επρόκειτο για έναν ονομαστικό μετρητή, ο οποίος θα αυξήσει τον sql μετρητή κατά 4.
6. ID: είναι το αναγνωριστικό του κανόνα. Παραπάνω έχουμε την τιμή 1000. Αυτά τα αναγνωριστικά θα χρησιμοποιηθούν στο NAXSI_FMT ή / και στην λίστα με τα λευκά αυτά ID για πιο εξιδεικευμένο αίτημα.

Κεφάλαιο 4ο: Learning mode, whitelist, blacklist

Ο κατάλογος blacklist εμπεριέχει όλους τους κανόνες που έχουν σκοπό να εμποδίσουν την πρόσβαση στους χρήστες, ενώ οι κανόνες whitelist αποσκοπούν στην αδειοδότηση αυτών. Ο διαχειριστής μπορεί είτε να προσθέσει whitelist χειροκίνητα, αναλύοντας το αρχείο καταγραφής σφαλμάτων του nginx, είτε (συνιστάται) να ξεκινήσει την εκμάθηση με την επιλογή “Learning Mode” του Naxsi, η οποία θα δημιουργήσει αυτόματα Whitelist κανόνες ανάλογα με τη συμπεριφορά και τις ανάγκες του ιστότοπου. Σε κάθε περίπτωση, ο χρήστης καλείται να γνωρίζει τη δομή και την εφαρμογή των κανόνων αυτών, όπως περιγράφεται παρακάτω, προκειμένου να αποφευχθεί οποιαδήποτε δυσκολία.

Οι λευκές λίστες μπορούν να βρίσκονται σε BasicRule επίπεδο ή στο http επίπεδο (MainRule). Οι λευκές λίστες έχουν την ακόλουθη σύνταξη:



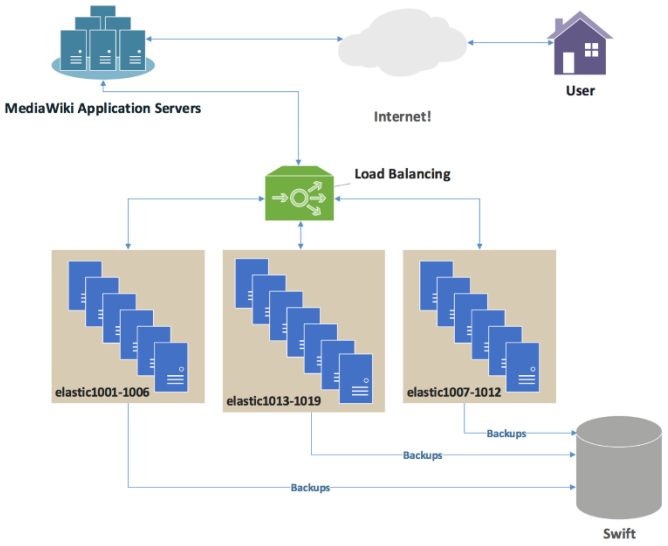
(Σχ.5)

Εν προκειμένω, σημαντικό είναι να αναφερθούμε στο κομμάτι του wl: όπως απεικονίζει και η παραπάνω φωτογραφία. Το wl πρόκειται για το ID του κανόνα και μπορεί να πάρει τις εξής μορφές:

- wl:0 : Whitelist όλους τους κανόνες
- wl:-37 : Whitelist όλους τους κανόνες του χρήστη (>= 1000), εκτός του κανόνα 37
- wl:37,38,39 : Whitelist κανόνες 37,38,39
- wl:37 : Whitelist rule #37

Κεφάλαιο 5ο : Load Balancing

Η εξισορρόπηση φορτίου (load balancing) βελτιώνει την κατανομή του φόρτου εργασίας σε πολλούς υπολογιστικούς πόρους, όπως υπολογιστές, σύμπλεγμα υπολογιστών, συνδέσεις δικτύου, κεντρικές μονάδες επεξεργασίας ή μονάδες δίσκου. Το load balancing στοχεύει στη βελτιστοποίηση της χρήσης πόρων, στη μεγιστοποίηση της απόδοσης, στην ελαχιστοποίηση του χρόνου απόκρισης και στην αποφυγή υπερφόρτωσης οποιουδήποτε μεμονωμένου πόρου. Η χρήση πολλαπλών στοιχείων με αντιστάθμιση φορτίου αντί για ένα μόνο στοιχείο μπορεί να αυξήσει την αξιοπιστία και τη διαθεσιμότητα. Η εξισορρόπηση φορτίου περιλαμβάνει συνήθως ειδικό λογισμικό ή υλικό, όπως multilayer switch ή ένα domain name system.



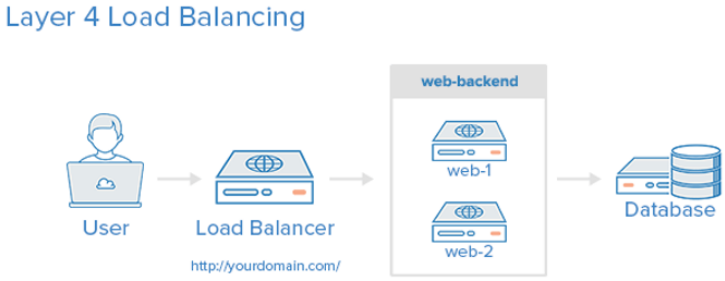
(Σχ.6)

Το HAProxy, το οποίο αντιπροσωπεύει το High Availability Proxy, είναι ένα δημοφιλές λογισμικό ανοιχτού κώδικα TCP / HTTP Load Balancer και λύση proxying που μπορεί να λειτουργήσει σε Linux, Solaris και FreeBSD. Η πιο συνηθισμένη χρήση της είναι η βελτίωση της απόδοσης και της αξιοπιστίας ενός περιβάλλοντος διακομιστών, διανέμοντας το φόρτο εργασίας σε πολλούς διακομιστές (π.χ. web, εφαρμογή, βάση δεδομένων).

(Σχ.7)

Ο Haproxy διαχωρίζεται κυρίως σε 2 κατηγορίες: (i) Layer 4 Load Balancing και (ii) Layer 7 Load Balancing.

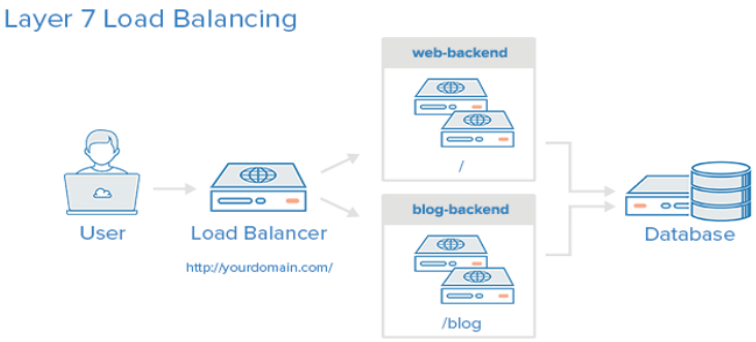
- **Layer 4 Load Balancing**: Ο απλούστερος τρόπος για επιτευχθεί η ισορροπία της κυκλοφορίας δικτύου σε πολλούς διακομιστές είναι να χρησιμοποιηθεί το επίπεδο φόρτωσης φορτίου του layer 4 (transport layer). Η αντιστοίχιση φορτίου με αυτόν τον τρόπο θα μεταφέρει την επισκεψιμότητα των χρηστών με βάση την εμβέλεια IP και τη θύρα.



(Σχ.8)

- **Layer 7 Load Balancing**: Ένας άλλος, πιο πολύπλοκος τρόπος προς επίτευξη της ισορροπίας της κυκλοφορίας δικτύου είναι να χρησιμοποιηθεί η στρώση 7 (στρώση εφαρμογής) εξισορρόπησης φορτίου. Η χρήση του στρώματος 7 επιτρέπει στο balancer φόρτωσης να διαβιβάζει τα

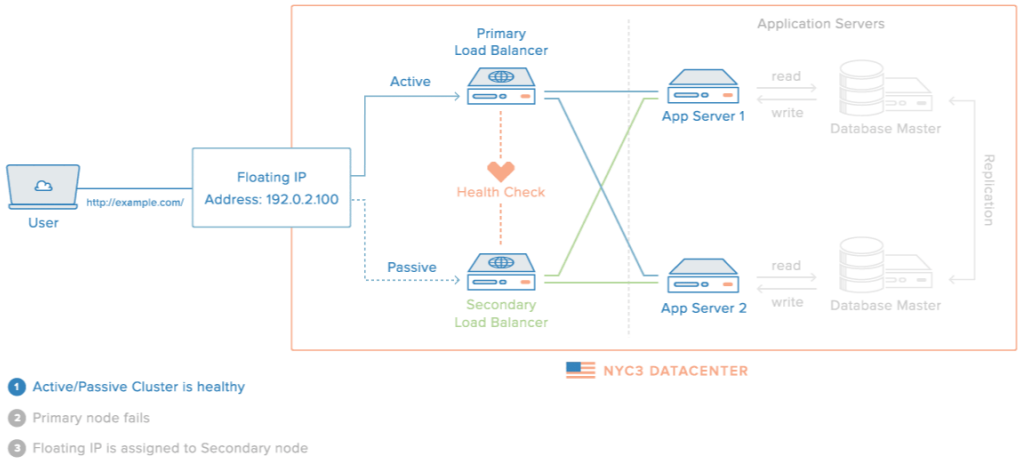
αιτήματα σε διαφορετικούς διακομιστές backend με βάση το περιεχόμενο του αιτήματος του χρήστη. Αυτή η λειτουργία εξισορρόπησης φορτίου επιτρέπει να λειτουργούν πολλοί διακομιστές εφαρμογών ιστού κάτω από τον ίδιο τομέα και θύρα.



(Σχ.9)

Μια εγκατάσταση υψηλής διαθεσιμότητας (HA) είναι μια υποδομή, η οποια αποτρέπει την αποτυχία ενός μόνο διακομιστή να γίνει η αιτία διακοπής προσθέτοντας πλεονασμό σε κάθε επίπεδο της αρχιτεκτονικής. Σε αυτό το παράδειγμα, υπάρχουν πολλαπλές αντισταθμίσεις φορτίου (ένα ενεργό και ένα ή περισσότερα παθητικά) πίσω από μια στατική διεύθυνση IP που μπορεί να μετατραπεί από ένα διακομιστή σε άλλο. Όταν ένας χρήστης έχει πρόσβαση στον ιστότοπό, το αίτημα περνά από την εξωτερική διεύθυνση IP στον ενεργό εξισορροπητή φορτίου. Αν αποτύχει ο αντισταθμιστής φορτίου, ο

μηχανισμός ανακατεύθυνσης θα το εντοπίσει και θα επαναπροσδιορίσει αυτόματα τη διεύθυνση IP σε έναν από τους παθητικούς διακομιστές.



(Σχ.10)

Προκειμένου να συνεργαστεί όμως το Naxsi με τον HAProxy απαραίτητο είναι να διαφορφωθεί το nginx ως reverse_proxy.

```
1.  server {
2.   proxy_set_header Proxy-Connection “”;
3.   listen       192.168.10.15:81;
4.   access_log  /var/log/nginx/naxsi_access.log;
5.   error_log  /var/log/nginx/naxsi_error.log
    debug;
6.   location / {
7.    include    /etc/nginx/test.rules;
```

```
8.    proxy_pass http://192.168.10.2:81/;
9.   }
10.  error_page 403 /403.html;
11.  location = /403.html {
12.   root /opt/nginx/html;
13.   internal;
14.  }
15.  location /RequestDenied {
16.   return 403;
17.  }
18. }
```

Αξίζει να σημειωθεί ότι ο HAProxy μπορεί να χρησιμοποιηθεί ως ένα είδος απλού firewall, στην περίπτωση που δημιουργηθεί το αναγαίο haproxy.cfg. Είναι ικανό να εντοπίσει και να αντιμετωπίσει DDoS επιθέσεις στο frontend, να αποκλείσει αυτόματα μια διεύθυνση IP η οποία έχει δημιουργήσει περισσότερα από 100 αιτήματα σε διάστημα 10s ή 10 σφαλμάτων. Οι χρόνοι φυσικά επιδέχονται τροποποίηση σύμφωνα με τις ανάγκες του κάθε χρήστη. Παρακάτω επισυνάπτεται ένα παράδειγμα υλοποίησης τοηυ haproxy προκειμένου να ανταπεξέλθει στις ανάγκες του naxsi εμπεριέχοντας σχετικά σχόλια.

```
1.  ######## Default values for all entries till
    next defaults section
```

```
2.  defaults
3.    option  http-server-close
4.    option  dontlognull
5.    option  redispatch
6.    option  contstats
7.    retries 3
8.    timeout connect 5s
9.    timeout http-keep-alive 1s
10.   # Slowloris protection
11.   timeout http-request 15s
12.   timeout queue 30s
13.   timeout tarpit 1m          # tarpit hold tim
14.   backlog 10000
15. # public frontend where users get connected to
16. frontend ft_waf
17.   bind 192.168.10.2:80 name http
18.   mode http
19.   log global
20.   option httplog
21.   timeout client 25s
22.   maxconn 10000
23.   # DDOS protection
24.   # Use General Purpose Couter (gpc) 0 in SC1
    as a global abuse counter
25.   # Monitors the number of request sent by an
    IP over a period of 10 seconds
26.   stick-table type ip size 1m expire 1m store
    gpc0,http_req_rate(10s),http_err_rate(10s)
27.   tcp-request connection track-sc1 src
28.   tcp-request connection reject if {
    sc1_get_gpc0 gt 0 }
29.   # Abuser means more than 100reqs/10s
30.   acl abuse sc1_http_req_rate(ft_web) ge 100
31.   acl flag_abuser sc1_inc_gpc0(ft_web)
```

```
32.   tcp-request content reject if abuse
    flag_abuser
33.   acl static path_beg /static/
    /dokuwiki/images/
34.   acl no_waf nbsrv(bk_waf) eq 0
35.   acl waf_max_capacity queue(bk_waf) ge 1
36.   # bypass WAF farm if no WAF available
37.   use_backend bk_web if no_waf
38.   # bypass WAF farm if it reaches its capacity
39.   use_backend bk_web if static waf_max_capacity
40.   default_backend bk_waf
41. # WAF farm where users' traffic is routed first
42. backend bk_waf
43.   balance roundrobin
44.   mode http
45.   log global
46.   option httplog
47.   option forwardfor header X-Client-IP
48.   option httpchk HEAD /waf_health_check
    HTTP/1.0
49.   # If the source IP generated 10 or more http
    request over the defined period,
50.   # flag the IP as abuser on the frontend
51.   acl abuse sc1_http_err_rate(ft_waf) ge 10
52.   acl flag_abuser sc1_inc_gpc0(ft_waf)
53.   tcp-request content reject if abuse
    flag_abuser
54.   # Specific WAF checking: a DENY means
    everything is OK
55.   http-check expect status 403
56.   timeout server 25s
57.   default-server inter 3s rise 2 fall 3
58.   server waf1 192.168.10.15:81 maxconn 100
    weight 10 check
```

```
59.   server waf2 192.168.10.16:81 maxconn 100
    weight 10 check
60. # Traffic secured by the WAF arrives here
61. frontend ft_web
62.   bind 192.168.10.2:81 name http
63.   mode http
64.   log global
65.   option httplog
66.   timeout client 25s
67.   maxconn 1000
68.   # route health check requests to a specific
    backend to avoid graph pollution in ALOHA GUI
69.   use_backend bk_waf_health_check if { path
    /waf_health_check }
70.   default_backend bk_web
71. # application server farm
72. backend bk_web
73.   balance roundrobin
74.   mode http
75.   log global
76.   option httplog
77.   option forwardfor
78.   cookie SERVERID insert indirect nocache
79.   default-server inter 3s rise 2 fall 3
80.   option httpchk HEAD /
81.   # get connected on the application server
    using the user ip
82.   # provided in the X-Client-IP header setup by
    ft_waf frontend
83.   source 0.0.0.0 usesrc hdr_ip(X-Client-IP)
84.   timeout server 25s
85.   server server1 192.168.10.11:80 maxconn 100
    weight 10 cookie server1 check
86.   server server2 192.168.10.12:80 maxconn 100
    weight 10 cookie server2 check
```

```
87. # backend dedicated to WAF checking (to avoid
    graph pollution)
88. backend bk_waf_health_check
89.   balance roundrobin
90.   mode http
91.   log global
92.   option httplog
93.   option forwardfor
94.   default-server inter 3s rise 2 fall 3
95.   timeout server 25s
96.   server server1 192.168.10.11:80 maxconn 100
    weight 10 check
97.   server server2 192.168.10.12:80 maxconn 100
    weight 10 check
```

Κεφάλαιο 6ο: Ιδέες προς υλοποίηση

6.1. Logstash

Το Logstash είναι ένας αγωγός επεξεργασίας δεδομένων ανοιχτού κώδικα, το οποίο επιτρέπει τη συλλογή δεδομένων από διάφορες πηγές, τη μετατροπή και την αποστολή στον επιθυμητό προορισμό. Συχνά χρησιμοποιείται ως αγωγός δεδομένων για την Elasticsearch. Λόγω της στενής ενσωμάτωσης της με την Elasticsearch, τις ισχυρές δυνατότητες επεξεργασίας αρχείων καταγραφής και πάνω από 200 προσχεδιασμένα plug-ins που μπορούν να βοηθήσουν να αναζητηθούν εύκολα τα δεδομένα, το Logstash είναι μια δημοφιλής επιλογή για τη φόρτωση των δεδομένων στο Elasticsearch.

- Input Stage: Το στάδιο εισαγωγής είναι ο τρόπος με τον οποίο το Logstash λαμβάνει τα δεδομένα. Ένα plugin θα μπορούσε να είναι ένα αρχείο έτσι ώστε το Logstash να διαβάζει γεγονότα από ένα αρχείο. Θα μπορούσε να είναι ένα τελικό σημείο HTTP ή θα μπορούσε να είναι μια σχεσιακή βάση δεδομένων.

- Filter Stage: Το στάδιο φιλτραρίσματος αφορά το πώς η Logstash θα επεξεργάζεται τα συμβάντα που ελήφθησαν από τα plugins της φάσης εισόδου. Εδώ μπορούμε να αναλύσουμε το CSV, XML, ή JSON. Μπορούμε επίσης να εμπλουτίσουμε δεδομένα, όπως αναζήτηση μιας διεύθυνσης IP και επίλυση της γεωγραφικής της θέσης ή αναζήτηση δεδομένων σε μια σχεσιακή βάση δεδομένων.

- Output Stage: Ένα plugin εξόδου είναι το σημείο όπου στέλνουμε τα επεξεργασμένα συμβάντα. Από τυπικής απόψεως, οι τόποι αυτοί ονομάζονται κηλίδες. Αυτά τα σημεία μπορεί να είναι μια βάση δεδομένων, ένα αρχείο, ένα παράδειγμα Elasticsearch.

Παρακάτω επισυνάπτεται ένα παράδειγμα του αρχείου επεξεργασίας του logtash, προκειμένου να

καταλάβουμε πως μπορεί να μετατρέψει τα δεδομένα και να μας εκδόσει το επιθυμητό output.

```
1.  input {
2.        file {
3.              path => “/path/to/your/logfile.log”
4.        }
5.   }
6.   filter {
7.        if [request] in [“/robots.txt”] {
8.              drop {}
9.        }
10.  }
11.  output {
12.       file {
13.             path => "%{type}_%{+yyyy_MM_dd}.log"
14.       }
15.  }
16.
```

Ένα από τα βασικά πλεονεκτήματα του logstash είναι ότι λειτουργεί όμορφα με δεδομένα συνεχούς ροής. Όπως και στην περίπτωση που εκινείται το logstash ως υπηρεσία, η παρακολούθηση συνεχώς την πηγή εισόδου είναι εφικτή και στη συνέχεια εύκολη η εφαρμογή καθορισμένων κανόνων μορφοποίησης.

Αξίζει να αναφερθεί ότι το logtash, περιέχει UI που επιτρέπει την παρακολούθηση της επεξεργασίας

των δεδομένων όπως και πολλά άλλα χρηστικά εργαλεία όπως επισυνάπτεται στην παρακάτω εικόνα.

(Εικ.3)

Σχετικά με το Naxsi, το logstash θα αποτελούσε ένα πλέον χρηστικό εργαλείο, προκειμένου να μετατραπούν τα δεδομένα σε μορφή χρηστική για πολλές άλλες εφαρμογές. Το logstash έχει τη δυνατότητα να λάβει ως input κατευθείαν τα logs απο τον nginx (/var/log/nginx/security.log) είτε να συνδεθεί με την elastic search. Σε κάθε περίπτωση χρηστικό θα ήταν τα δεδομένα να τα εξάγει είτε σε csv μορφή προκειμένου να κρατηθεί ένα backup,

είτε ακόμα και να τα ενγράψει σε έναν Syslog server.

6.2. Fail2Ban

Το Fail2ban σαρώνει τα αρχεία καταγραφής (/var/log/nginx/access.log) και απαγορεύει τις IP που δείχνουν τα κακόβουλες προθέσεις, δηλαδή πάρα πολλές αποτυχημένες προσπάθειες κωδικού πρόσβασης, αναζητώντας εκμεταλλεύσεις ή πάρα πολλά αιτήματα σε μικρό χρόνο. Γενικά το Fail2Ban χρησιμοποιείται για την ενημέρωση των κανόνων τείχους προστασίας για τον αποκλεισμό των διευθύνσεων IP για ένα συγκεκριμένο χρονικό διάστημα. Το Fail2Ban είναι σε θέση να μειώσει το ποσοστό λανθασμένων προσπαθειών ελέγχου ταυτότητας, ωστόσο δεν μπορεί να εξαλείψει τον κίνδυνο γι αυτό το λόγο τον επόμενο ρόλο προστασίας τον αναλαμβάνει ο Naxsi. Τα βήματα για να εγκατασταθεί και να συνεργαστεί με τον naxsi είναι τα παρακάτω:

Εγκατάσταση fail2ban:

```
1.  yum install fail2ban
```

Προσθήκη Naxsi-Filter:

```
1. cd /etc/fail2ban/filter.d/nginx-naxsi.conf
```

```
1. [INCLUDES]
2. before = common.conf
3. [Definition]
4. failregex = NAXSI_FMT: ip=<HOST>
5. ignoreregex = NAXSI_FMT:
   ip=<HOST>.*&config=learning
```

Παραμετροποίηση αρχείου jail.conf:

```
1. cd /etc/fail2ban/jail.conf
```

```
1. [nginx-naxsi]
2. enabled = true
3. port = http,https
4. filter = nginx-naxsi
5. logpath = /var/log/nginx/*error.log
6. maxretry = 3
```

Τα αποτελέσματα της ορθής λειτουργίας του fail2ban μπορεί εύκολα να ελεγχθεί απο το fail2ban.log, το οποίο βρίσκεται στο path: /var/log/fail2ban.log. Στο αρχείο αυτό καταγράφεται

η ώρα η ημερομηνία και η διεύθυνση η οποία αποκλείστηκε και έχει την παρακάτω μορφή:

```
1.  2019-09-13 15:34:44,016 fail2ban.actions:
    WARNING [nginx-naxsi] Ban 88.z.x.y`
```

Κεφάλαιο 7ο: Συμπεράσματα

Από την παραπάνω έρευνα και υλοποίηση του WAF Naxsi, συμπερένουμε ότι επρόκειτο για ένα αρκετά τροποποιήσιμο και αποδοτικό firewall που είναι ικανό να σταθεί επάξια σε κάθε είδους βιομηχανικής, επιχειρησιακής ή ακόμα και οικιακής πρόκλησης χωρίς κανένα πρόβλημα. Η ευελιξία που προσφέρει σε επίπεδο κανόνως και αυτοματισμών το καθιστούν εύκολα τροποποιήσιμο τόσο απο εξειδικευμένους χρήστες όσο και απο μη εξειδικευμένους. Αξιοσημείωτο είναι να τονιστεί ότι το Naxsi δεν απαιτεί σύστημα με μεγάλο αριθμό πόρων προκειμένου να διασφαλιστεί η εύρυθμη λειτουργία του. Αντίθετα μπορεί να φιλοξενηθεί σε έναν μέτριο application server που παράλληλα με το web hosting του server να επιβιώνει χωρίς κανένα πρόβλημα. Από μόνο του αυτο το καθιστά ένα απο τα πιο ευέλικτα και ανταγωνιστικά WAF της αγοράς χωρίς αυτό να σημαίνει ότι δεν εμπεριέχει και κάποια αρνητικά στοιχεία. Ένα απο αυτα είναι το support, που αποτελεί το μείζων πρόβλημα για αρκετά applications και αυτός είναι ένας από τους κυριότερους λόγους που δεν βρίσκεται στις κορυφαίες επιλογές μεγάλων εταιριών. Ωστόσο το Naxsi με τον κατάλληλο χειρισμό από εξιδεικευμένο

προσωπικό με γνώσεις στην προστασία ιστοτόπων θα μπορούσε να φιλοξενηθεί σε οποιονδήποτε application server μεγάλων εταιριών ή ακομα και τραπεζών. Δυστυχώς όμως οι χρήστες δεν εμπιστεύονται εύκολα μια open-source λύση για την επιχειρησή τους και προστρέχουν σε enterprise λύσεις που το marketing τους επιρεάζει να την προμηθευτούν χωρίς να τους προσφέρει την απαραίτητη διαδικτυακή προστασία που χρειάζονται. Αυτό τους καθιστα εν αγνεία τους ευάλωτους και πέφτουν θύματα επιθέσεων, με αποτέλεσμα να δυσχεραίνεται η περιήγηση στους ιστοτοπούς τους ή ακόμα και να χάνουν την αξιοπιστία της εργασίας τους.

Βιβλιογραφία

Amazon Web Services, Inc. (n.d.). *Explain of Kibana – Amazon Web Services*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://aws.amazon.com/elasticsearch-service/kibana/ [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Barr, J. (2004). *New – Amazon Elasticsearch Service | Amazon Web Services*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://aws.amazon.com/blogs/aws/new-amazon-elasticsearch-service/ [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Code.google.com. (2017). *Naxsi Introduction*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://code.google.com/archive/p/naxsi/wikis/OVIntro.wiki [**Ανακτήθηκε: 13 Sep. 2019**].

Cron-job (2015). *Scheduling Repeated Job using cron* **[Διαδίκτυο]. Διαθέσιμο στο: http://researchhubs.com/post/computing/linux-cmd/cron-repeat-jobs.html** [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Dev.maxmind.com. (2012). *GeoIP Legacy Apache Module*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://dev.maxmind.com/geoip/legacy/mod_geoip2/ [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Relan, K. (n.d.). *How To Secure Nginx with NAXSI | DigitalOcean*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-naxsi-on-ubuntu-16-04 [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

NGINX Documentation. (2019). *NGINX Docs | Installing NGINX Open Source*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-open-source/ [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

NGINX Documentation. (n.d.). *NGINX Docs | Installing NGINX Open Source*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-open-source/ [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Elastic.co. (n.d.). *Installing Elasticsearch | Elasticsearch Reference [7.3]* **[Διαδίκτυο]. Διαθέσιμο στο:** https://www.elastic.co/guide/en/elasticsearch/refer

ence/current/install-elasticsearch.html [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Elastic.co. (n.d.). *Set Up Kibana | Kibana Guide [7.3] | Elastic*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://www.elastic.co/guide/en/kibana/current/setup.html [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Fail2ban.org (2016). *Fail2ban introduction* **[Διαδίκτυο]. Διαθέσιμο στο:** https://www.fail2ban.org/wiki/index.php/Main_Page [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

elastic.co (n.d). *logstash Inputs, filters & outputs* **[Διαδίκτυο]. Διαθέσιμο στο:** https://www.elastic.co/products/logstash [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

elastic.co (n.d). *logstash Introduction* **[Διαδίκτυο]. Διαθέσιμο στο:** https://www.elastic.co/guide/en/logstash/current/introduction.html [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Deous, M. (2016). *Spike introduction and explain*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://github.com/nbs-system/spike [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Deous, M. (2017). *Naxsi rules examples*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://github.com/nbs-system/naxsi/wiki/rules-examples [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Deous, M. (2017). *nbs-system/nxtool-ng*. **[Διαδίκτυο]. Διαθέσιμο στο:** https://github.com/nbs-system/nxtool-ng [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Haproxy.org. (n.d.). *HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer*. **[Διαδίκτυο]. Διαθέσιμο στο:** http://www.haproxy.org/#secu [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].

Saunois, L. (2015). *Naxsi firewall* **[Διαδίκτυο]. Διαθέσιμο στο:** https://www.nbs-system.com/en/blog/naxsi-what-do-its-users-think-about-it/ [**Ανακτήθηκε: 13 Σεπτεμβρίου 2019**].