ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ ΤΩΝ ΓΛΩΣΣΩΝ ΠΕΡΙΓΡΑΦΗΣ-ΧΑΡΑΚΤΗΡΙΣΜΟΥ ΚΑΚΟΒΟΥΛΟΥ ΛΟΓΙΣΜΙΚΟΥ

Διπλωματική Εργασία

του

Διαμαντούδη Απόστολου

Θεσσαλονίκη, Φεβρουάριος 2019

ΣΥΓΚΡΙΤΙΚΗ ΜΕΛΕΤΗ ΤΩΝ ΓΛΩΣΣΩΝ ΠΕΡΙΓΡΑΦΗΣ-ΧΑΡΑΚΤΗΡΙΣΜΟΥ ΚΑΚΟΒΟΥΛΟΥ ΛΟΓΙΣΜΙΚΟΥ

Διαμαντούδης Απόστολος

Πτυχίο Πληροφορικής, ΕΑΠ, 2010

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Μαυρίδης Ιωάννης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την / / 2019

Μαυρίδης Ιωάννης	Χατζηγεωργίου Αλέξανδρος	Γεωργιάδης Χρήστος
.....	.....	.....

Απόστολος Διαμαντούδης

.....

Περίληψη:

Η δημιουργία νέων κακόβουλων λογισμικών υψηλής πολυπλοκότητας που λειτουργούν ως μέρος τεχνολογικά προηγμένης κακόβουλης δραστηριότητας, έχει στρέψει την έρευνα από την ανίχνευση στηριζόμενη στις υπογραφές (*signature-based detection*) στην ανίχνευση που στηρίζεται στην συμπεριφορά και στην ανώμαλη δραστηριότητα (*behavioral or anomaly-based detection*). Στα πλαίσια αυτής της τάσης, έχουν προταθεί διάφορα πρότυπα (MAEC, OpenIOC, BSL κ.τ.λ.) καταγραφής της πληροφορίας που αντιπροσωπεύει ένα κακόβουλο λογισμικό, με σκοπό την κατασκευή τεχνουργημάτων που απαντώνται με τον γενικό όρο "Δείκτης Επικινδυνότητας". Σκοπός της εργασίας είναι η μελέτη των διάφορων γλωσσών-προτύπων που έχουν προταθεί στην βιβλιογραφία, η παρουσίαση κριτηρίων αξιολόγησης και στη βάση αυτών η συγκριτική αξιολόγηση των γλωσσών-προτύπων.

Abstract:

The creation of new high-complexity malware, which function as part of a technologically advanced malicious activity, has turned research efforts from signature-based detection to behavioural or anomaly-based detection. As part of this trend, various standards (MAEC, OpenIOC, BSL etc.) for the recording of the information which represents malicious software have been proposed, aiming at creating artefacts that are encountered under the general term “Indicator(s) of Compromise”. The objective of this thesis is the study of the various languages-standards that have been proposed in the bibliography, the presentation of evaluation criteria, and based on them a comparative evaluation of the languages-standards.

Κατάλογος εικόνων

Κατάλογος Πινάκων

Πίνακας Περιεχομένων

1 Εισαγωγή

Στις μέρες μας, το κακόβουλο λογισμικό ("*malicious*[1] *software*", ή αλλιώς, "*malware*") απαντάται όλο και πιο συχνά, και σε πολλές και διάφορες μορφές. Έτσι, έχει υπάρξει η ανάγκη για την περιγραφή του, ούτως ώστε να είμαστε σε θέση να αποκρούουμε και να αντιμετωπίζουμε ("*mitigate*") τις επιθέσεις που δέχονται τα υπολογιστικά συστήματα, επί καθημερινής βάσης σχεδόν, βασισμένοι σε συγκεκριμένες ιδιότητες του λογισμικού αυτού, ώστε να βελτιώνουμε τους μηχανισμούς προστασίας από αυτό.

Είναι ένα πρόβλημα το οποίο έχει λάβει παγκόσμιες διαστάσεις, καθώς, το κακόβουλο λογισμικό, μέσω του διαδικτύου, μπορεί και διαδίδεται σε υπολογιστές σε κάθε γωνιά της υδρογείου, είτε αυτοί ανήκουν σε κυβερνήσεις, ή σε δημόσιους, ή σε ιδιωτικούς οργανισμούς, ή σε τελικούς χρήστες ("*end-users*"). Εκτός αυτού, έχει τεράστιο οικονομικό κόστος (Accenture, 2017), για τα μέρη τα οποία έχουν υποστεί βλάβη εκ της ακούσιας χρήσεως τέτοιων λογισμικών. Το δε πραγματικό κόστος από την χρήση κακόβουλου λογισμικού δεν είναι εύκολο να υπολογιστεί (Aycock, Computer Viruses and Malware, 2006).

Από την άλλη, η γνώση περί του κακόβουλου λογισμικού, και κυρίως των χαρακτηριστικών αυτού, είναι η βάση πάνω στην οποία μπορούν να χτιστούν πιο ασφαλή υπολογιστικά συστήματα, τέτοια ώστε να μην είναι επιρρεπή ("*vulnerable*") στην εκτέλεση τέτοιου κακόβουλου λογισμικού.

Γενικά, στις μέρες μας, το κακόβουλο λογισμικό έχει γίνει ιδιαίτερα πολύπλοκο (*complex*) (Hernandez-Ardieta & Tapiador, 2013). Καθώς αυτό τείνει να είναι πολυμορφικό και μεταμορφικό, δηλαδή έχει την ιδιότητα να αλλάζει τον κώδικά του κατά την εκτέλεσή του (Gandotra, Bansal, & Sofat, 2014), απαιτούνται πιο προηγμένοι αμυντικοί μηχανισμοί.

Σε αυτή την εργασία θα μελετήσουμε τις διάφορες γλώσσες – πρότυπα ("*standards*") που έχουν προταθεί στην βιβλιογραφία παγκοσμίως για την περιγραφή και τον χαρακτηρισμό του κακόβουλου λογισμικού, κι εν συνεχεία θα τις συγκρίνουμε.

[1] Ένας καλύτερος όρος για την απόδοση της λέξης «*κακόβουλος*» στην αγγλική γλώσσα, θα μπορούσε να είναι η λέξη "*malevolent*", που σημαίνει κυριολεκτικά κακόβουλος, αντί της λέξης "*malicious*" που κυριολεκτικά σημαίνει κακεντρεχής.

1.1 Κακόβουλο Λογισμικό

Σύμφωνα με τον Aycock, ως κακόβουλο λογισμικό ορίζεται, «*το λογισμικό εκείνο, του οποίου είτε ο σκοπός του (intent) είναι κακόβουλος, ή το αποτέλεσμά του (effect) είναι κακόβουλο»* (Aycock, Computer Viruses and Malware, 2006). Κατά έναν άλλον ορισμό, το κακόβουλο λογισμικό, «*είναι εκείνο το πρόγραμμα, ή μέρος προγράμματος, το οποίο έχει μη αναμενόμενα ή ανεπιθύμητα αποτελέσματα, προκαλούμενα από έναν πράκτορα (agent) με απώτερο σκοπό την καταστροφή*» (Swimmer, 2008). Επιπροσθέτως, «*ο κακόβουλος κώδικας μπορεί να απευθύνεται είτε προς έναν μόνο συγκεκριμένο χρήστη, ή προς μια κλάση χρηστών, ή προς οποιονδήποτε*» (Pfleeger, Pfleeger, & Margulies, 2015).

Πιο απλουστευμένα, μπορούμε να πούμε ότι το κακόβουλο λογισμικό, είναι λογισμικό το οποίο έχει δημιουργηθεί από κακόβουλους χρήστες ("*hackers*", ή αλλιώς, "*malware authors*" ή "*malicious actors*") και ο σκοπός του είναι να βλάψει γενικώς το σύστημα στο οποίο θα εκτελεσθεί, είτε αυτό είναι το περιβάλλον εργασίας του τελικού χρήστη ("*end-user*"), ή το κεντρικό σύστημα (π.χ. *server*) ενός οργανισμού ή μιας εταιρείας.

1.2 Προηγμένη Επίμονη Απειλή

Ως «*Προηγμένη Επίμονη Απειλή»* (*Advanced Persistent Threat* – ακρωνύμιο "*APT"*), ορίζουμε εκείνη την δικτυακή επίθεση (*network attack*) προς το σύστημα, κατά την οποία κάποιο άτομο, ή ομάδα ατόμων, αποκτά μη-εξουσιοδοτημένη πρόσβαση (*non-authorized access*) στο σύστημα, και παραμένει ανεξιχνίαστο (*undetected*) επί μεγάλου χρονικού διαστήματος (Advanced persistent threat, 2018). Πρόκειται για μία επίθεση η οποία είναι καλά προσχεδιασμένη, κι επιπλέον μπορεί να προκαλέσει μεγάλη ζημία στο σύστημα του χρήστη.

Ο δε στόχος μίας τέτοιας επίθεσης, είναι συνήθως, η κλοπή δεδομένων.

Σύμφωνα με την εταιρεία FireEye (FireEye, 2018), τα κυρίως στάδια (*steps*) της *APT*, είναι τα κάτωθι (6) έξι:

1. *«Ο διενεργητής της απειλής (threat actor) αποκτά πρόσβαση στο σύστημα, είτε μέσω ηλεκτρονικού ταχυδρομείου (e-mail), ή δικτύου (network), ή αρχείου (file), ή κάποιας ευπάθειας εφαρμογής (application vulnerability), κι εν συνεχεία*

εισάγει κακόβουλο λογισμικό (malware) στο σύστημα. Το σύστημα, σε αυτήν τη φάση, θεωρείται συμβιβασμένο (compromised), αλλά όχι ακόμη παραβιασμένο (breached).

2. *Το εισαχθέν προηγμένο κακόβουλο λογισμικό, αναζητάει επιπλέον ευπάθειες (vulnerabilities) στο σύστημα, κι επικοινωνεί με τους Command-and-Control (CnC) εξυπηρετητές (servers), ούτως ώστε να λάβει επιπρόσθετες οδηγίες.*
3. *Το κακόβουλο λογισμικό αναζητάει και εγκαθιστά επιπλέον σημεία συμβιβασμού (points of compromise) στο σύστημα, για να συνεχιστεί απρόσκοπτη η κυβερνο-επίθεση σε περίπτωση που κάποια άλλα σημεία πρόσβασης έχουν κλείσει.*
4. *Ο διενεργητής της επίθεσης (threat actor), με το που αντιληφθεί ότι έχει αποκτήσει πρόσβαση στο σύστημα, συγκεντρώνει δεδομένα από το σύστημα, όπως διαπιστευτήρια (credentials) χρηστών.*
5. *Το κακόβουλο λογισμικό συγκεντρώνει δεδομένα σε κάποιον ενδιάμεσο εξυπηρετητή (staging server), υπό τον έλεγχο του διενεργητή της επίθεσης. Το σύστημα πλέον θεωρείται παραβιασμένο.*
6. *Τυχόν αποδείξεις αυτής της επίθεσης (APT), αφαιρούνται από το σύστημα (π.χ., system-access logs), αν και αυτό ακόμη θεωρείται συμβιβασμένο. Σε αυτό το σημείο, ο διενεργητής της επίθεσης, δύναται να επιστρέψει μεταγενέστερα, για να συνεχίσει την εγκληματική δραστηριότητά του.»*

1.3 Δείκτες Επικινδυνότητας – Indicators of Compromise

Ως «Δείκτης Επικινδυνότητας» (*Indicator of Compromise*, ακρωνύμιο: "*IoC*") ορίζεται, εκείνο το παρατηρηθέν τεχνούργημα (*artifact*), δηλαδή ό,τι απομένει στο σύστημα μετά από μία κυβερνο-επίθεση, το οποίο καταδεικνύει την ύπαρξη κάποιας επίθεσης προς το σύστημα μας. (Indicator of Compromise, 2018), και αυτός είναι ο λόγος για τον οποίον χρησιμοποιείται (Koen, 2017).

Πρέπει να σημειώσουμε εδώ, ότι υπάρχουν πολλοί και διαφορετικοί δείκτες επικινδυνότητας (Koen, 2017). Τυπικοί *Δείκτες Επικινδυνότητας* μπορεί να είναι, από ένα απλό όνομα αρχείου (*filename*), ή ένα MD5 file hash[2], έως την ίδια την

[2] Συνήθως, δεκαεξαδική συμβολοσειρά (hexadecimal string), συγκεκριμένου μήκους, η οποία προκύπτει εάν εφαρμόσουμε κάποιον αλγόριθμο κατακερματισμού (hash algorithm), π.χ. MD5, στο ίδιο το αρχείο.

συμπεριφορά (*behavior*) του κακόβουλου λογισμικού στο σύστημα. Έτσι, για παράδειγμα, μεταξύ άλλων συμπεριφορικών δεικτών έχουμε και τους εξής παρακάτω (Andress, 2015):

- Εάν έχουν εκδηλωθεί εισερχόμενες ή εξερχόμενες (*ingoing / outgoing*) αιχμές (*spikes*) στην δραστηριότητα του δικτύου μας (*network activity*), δηλαδή μη συνήθης κίνηση δικτύου.
- Εάν έχει υπάρξει πρόσβαση στο σύστημα από μη συνήθεις γεωγραφικές τοποθεσίες.
- Εάν έχει υπάρξει μη συνήθης δραστηριότητα στην Βάση Δεδομένων, όπως για παράδειγμα διόγκωση στην ανάγνωση δεδομένων.
- Εάν έχει υπάρξει υψηλός αριθμός αποτυχημένων προσπαθειών εισόδου στο σύστημα (*log-in attempts*), ή και ύπαρξη ανωμαλιών στην δραστηριότητα των προνομιούχων (*privileged*) λογαριασμών χρηστών.
- Εάν έχουν υπάρξει συνδέσεις σε μη συνήθεις θύρες (*ports*) ή πρωτόκολλα δικτύου.
- Εάν έχουν υπάρξει ύποπτες αλλαγές στο μητρώο (*registry*) ή στα αρχεία συστήματος (*system files*).
- Εάν έχει υπάρξει μεγάλος αριθμός αιτήσεων για το ίδιο αρχείο.

2 Ταξινομία, Ανίχνευση και Ανάλυση

Ως ταξινομία κακόβουλου λογισμικού (Malware Taxonomy), ορίζουμε «*εκείνη την διαδικασία, κατά την οποία ταξινομούμε το κακόβουλο λογισμικό σε διαφορετικές ομάδες, χρησιμοποιώντας μια συστηματική προσέγγιση, βασισμένη στα χαρακτηριστικά του»* (Elisan, 2015).

2.1 Τύποι Κακόβουλου Λογισμικού

Το κακόβουλο λογισμικό απαντάται σε πολλές και διάφορες μορφές. Μια κατηγοριοποίησή του σύμφωνα με την λειτουργικότητά του (*functionality*) αναφέρουμε παρακάτω, αν και συνήθως το κακόβουλο λογισμικό εμπίπτει σε περισσότερες από μία κατηγορίες (Sikorski & Honig, 2012):

- Backdoor: Είναι εκείνο το κακόβουλο λογισμικό, το οποίο εγκαθίσταται στον υπολογιστή, και επιτρέπει στον επιτιθέμενο (*attacker*) να αποκτήσει πρόσβαση στο σύστημα.
- Botnet: Παρόμοιο με το backdoor, αλλά υπάρχει κεντρικός συντονισμός από απομακρυσμένο υπολογιστή/εξυπηρετητή (*server*).
- Downloader: Μεταφορτώνει (*download*), άλλο κακόβουλο λογισμικό στον υπολογιστή.
- Information-stealing malware: Υποκλέπτει διαφόρων τύπων πληροφορίες από τον υπολογιστή τον οποίο έχει μολύνει, και τις αποστέλλει στον επιτιθέμενο.
- Launcher: Χρησιμοποιώντας ειδικές τεχνικές, εκτελεί άλλον κακόβουλο κώδικα.
- Rootkit: Δύσκολος στην ανίχνευση κακόβουλος κώδικας, καθώς έχει την ιδιότητα να αποκρύπτει την ύπαρξη άλλου κακόβουλου κώδικα.
- Scareware: Κακόβουλο λογισμικό το οποίο "εκφοβίζει" τον χρήστη στο να αγοράσει κάποιο άλλο λογισμικό.
- Spam-sending malware: Αφ'ότου εγκατασταθεί στο σύστημα, αποστέλλει ανεπιθύμητα μηνύματα ηλεκτρονικού ταχυδρομείου (*e-mail*).
- Worm ή Virus: Έχει την ιδιότητα να αυτο-διαδίδεται (*propagate*) από σύστημα σε σύστημα.

Διάφοροι συγγραφείς αναφέρουν διαφορετικές κατηγορίες / τύπους κακόβουλου λογισμικού. Για παράδειγμα, ο Aycock (Aycock, Computer Viruses and Malware, 2006), προτείνει 3 γενικά χαρακτηριστικά (*attributes*) τα οποία χαρακτηρίζουν *κάθε* τύπο malware:

- Self-replication (Αυτο-αντιγραφή): Την δυνατότητα που έχει το κακόβουλο λογισμικό να αυτο-αντιγράφεται σε άλλους υπολογιστές, ή όχι.
- Population Growth (Ανάπτυξη πληθυσμού): Έτσι περιγράφεται η γενική αλλαγή στον αριθμό των περιστατικών (*instances*) κακόβουλου λογισμικού, εξαιτίας της ιδιότητας της *αυτο-αντιγραφής* (*self-replication*).
- Parasitic (Παρασιτικό): Εάν το κακόβουλο λογισμικό είναι *παρασιτικό* ή όχι, δηλαδή, την ιδιότητα εκείνη του κακόβουλου λογισμικού, να χρησιμοποιεί κάποιο άλλο εκτελέσιμο αρχείο για την ύπαρξή του.

Αναλυτικότερα, ο Aycock (Aycock, Computer Viruses and Malware, 2006), προτείνει τους εξής παρακάτω τύπους κακόβουλου λογισμικού:

Όνομα/Κατηγορία	Self-Replication	Population Growth	Parasitic
Logic bomb	Όχι	0	Πιθανόν
Trojan Horse	Όχι	0	Ναι
Back Door	Όχι	0	Πιθανόν
Virus	Ναι	Θετικό	Ναι
Worm	Ναι	Θετικό	Όχι
Rabbit	Ναι	0	Όχι
Spyware	Όχι	0	Όχι
Adware	Όχι	0	Όχι
Hybrid, Droppers & Blended Threats	-	-	-
Zombies	-	-	-

Πίνακας 1. Τύποι κακόβουλου λογισμικού (Aycock, 2006)

Ο Elisan (Elisan, 2015), προτείνει τύπους βάσει της συμπεριφοράς του malware:

- Infectors,
- Network worms,
- Trojan horses,
- Backdoors,
- Remote-access Trojans (RAT),
- Information stealers,
- Ransomware,
- Scareware,
- Fakeware,
- Greyware.

2.2 Μέθοδοι Ανίχνευσης Κακόβουλου Λογισμικού

Οι τρεις βασικές μέθοδοι ανίχνευσης που απαντώνται στην βιβλιογραφία είναι (Deeg, Nerz, & Sauder, 2014), (Idika & Mathur, 2007):

1. Ανίχνευση βασισμένη σε υπογραφές (*signature-based detection*).
2. Ανίχνευση βάσει συμπεριφοράς (*behavior-based or anomaly-based detection*).
3. Ανίχνευση βάσει προσδιορισμών (*specification-based detection*)

2.2.1 Ανίχνευση βασισμένη σε υπογραφές

Με την ανίχνευση βασισμένη σε υπογραφές εκτελέσιμων αρχείων λογισμικού (*signature-based detection*), το αντι-ϊικό λογισμικό (*anti-virus software*) αναζητάει «γνωστά» μοτίβα (*patterns*) με την μορφή σειρών από bytes (*byte sequences*) στο εκτελέσιμο αρχείο λογισμικού, και με αυτόν τον τρόπο μπορεί να ανιχνεύει το κακόβουλο λογισμικό. Εν συνεχεία, το αντι-ϊικό λογισμικό, όταν αναγνωρίζει το μοτίβο των bytes, εμποδίζει / απαγορεύει την εκτέλεση του εν λόγω προγράμματος στο σύστημα, θέτοντας το σε μαύρη λίστα (*black-list*). Σε αντίθετη περίπτωση το πρόγραμμα εισέρχεται στην λευκή λίστα (*white-list*) του αντι-ϊικού λογισμικού, κι έτσι επιτρέπεται ελεύθερα η εκτέλεσή του στο σύστημα (Deeg, Nerz, & Sauder, 2014).

Οι υπογραφές των εκτελέσιμων αρχείων, δεν είναι τίποτα άλλο παρά σειρές από bytes (*byte sequences*) οι οποίες βρίσκονταν στον κώδικα του εκτελέσιμου αρχείου. Η δημιουργία τους όμως «επακολουθεί» χρονικά της δημιουργίας του κακόβουλου λογισμικού από τους κακόβουλους προγραμματιστές (*malicious actors*), καθώς ανακαλύπτονται κατά την ενδελεχή μελέτη του κακόβουλου λογισμικού αφού αυτό έχει συγγραφεί, κι εφαρμόζονται στο σύστημα για την καταπράϋνσή του (*mitigation*), σε μεταγενέστερο χρονικό σημείο. Έτσι, ένα πρώτο μειονέκτημα αυτής της μεθοδολογίας, είναι ότι το αντι-ϊικό λογισμικό μπορεί να ανιχνεύσει μόνο ήδη γνωστές υπογραφές, κάτι το οποίο δεν είναι ιδιαίτερα αποτελεσματικό για την καταπολέμηση του κακόβουλου λογισμικού.

2.2.2 Ανίχνευση βάσει συμπεριφοράς

Η ανίχνευση βάσει συμπεριφοράς του εκτελέσιμου αρχείου (*behavior-based detection*), είναι διαφορετική από την παραπάνω προσέγγιση βασισμένη σε υπογραφές εκτελέσιμων αρχείων. Σε αυτήν την περίπτωση το εν λόγω κακόβουλο λογισμικό, ταξινομείται (*classified*) σύμφωνα με την συμπεριφορά του, δηλαδή, με το ποιές ακριβώς ενέργειες αυτό διαπράττει στο σύστημα (Deeg, Nerz, & Sauder, 2014).

Συνήθως, υπάρχει ένα σύστημα βαθμολόγησης (*scoring system*), για να αποφανθούμε κατά πόσο κάποιο λογισμικό είναι καλοήθες (*benign*), ή αυτό να χαρακτηριστεί ως κακόβουλο λογισμικό (Deeg, Nerz, & Sauder, 2014). Σύμφωνα με αυτό το σύστημα βαθμολόγησης, υπάρχουν ορισμένα κατώφλια βαθμολογίας (*score thresholds*), τα οποία χρησιμοποιούνται καταλλήλως, ούτως ώστε να καταταχθεί κάποιο λογισμικό είτε στην μία κατηγορία, ή στην άλλη, δηλαδή σε αυτήν του κακόβουλου λογισμικού.

2.2.3 Ανίχνευση βάσει προσδιορισμών

Κατά τους Idika & Mathur (Idika & Mathur, 2007), αυτή η μέθοδος ανίχνευσης, είναι ειδική περίπτωση της μεθόδου βασισμένης στην συμπεριφορά του κακόβουλου λογισμικού (*behavioral detection*), και η οποία μέθοδος ανίχνευσης προσπαθεί να επιλύσει το πρόβλημα του αρκετά μεγάλου ποσοστού της λανθασμένης ανίχνευσης με την μέθοδο βάσει συμπεριφοράς. Η μέθοδος βάσει προσδιορισμών, δίνει το βάρος της στις απαιτήσεις (*requirements*) της εφαρμογής, αντί στην υλοποίηση (*implementation*) της εφαρμογής.

2.2.4 Σύγκριση μεθόδων ανίχνευσης κακόβουλου λογισμικού

Ως μία πρώτη σύγκριση που θα μπορούσαμε να κάνουμε μεταξύ των δύο πρώτων μεθόδων ανίχνευσης, είναι ότι η μεν πρώτη μέθοδος, αυτή της ανίχνευσης βάσει υπογραφών, έχει το βασικό μειονέκτημα ότι εφαρμόζει την καταπράϋνση (*mitigation*) εκ των υστέρων και αφού έχουν δυνητικά προκληθεί ζημίες στο σύστημα του χρήστη, ενώ η δεύτερη μέθοδος, η βασισμένη στην συμπεριφορά του πιθανόν κακόβουλου λογισμικού, φαίνεται να έχει πιο υποσχόμενα αποτελέσματα, καθώς πρόκειται για μία πιο ενδελεχή και ακριβέστερη μελέτη του κακόβουλου λογισμικού, και δεν μένει μόνο στις υπογραφές (π.χ., με την πρώτη μέθοδο ανίχνευσης, κάποιο πρόγραμμα / λογισμικό θα μπορούσε λανθασμένα να χαρακτηριστεί ως κακόβουλο λογισμικό, λόγω της πιθανότητας της ύπαρξης της υπογραφής στον κώδικά του, χωρίς αυτό να αποτελεί απειλή για το σύστημα του χρήστη).

Βεβαίως, δεν θα ήταν δυνατόν να αποφανθούμε ακόμη για το ποια μέθοδος είναι καλύτερη ως προς την ανίχνευση του κακόβουλου λογισμικού. Αντιθέτως, θα πρέπει να υπάρξει διεξοδική μελέτη, ως προς το εύρος του τί μπορεί να ανιχνεύσει η μία μέθοδος, και τί η άλλη, ούτως ώστε να αποφανθούμε με μεγαλύτερη σιγουριά.

2.3 Τεχνικές Ανάλυσης

Η ανάλυση του κακόβουλου λογισμικού είναι μια επίπονη διαδικασία συνήθως, καθώς το κακόβουλο λογισμικό από την φύση του, είναι τέτοιο που δύναται να προκαλέσει ζημίες στο σύστημα, καθώς πρόκειται για Advanced Persistent Threat (*APT*), δηλαδή μια καλά προσχεδιασμένη επίθεση στο σύστημα (Advanced persistent threat, 2018).

Επίσης, πρέπει να αναφέρουμε εδώ, ότι υπάρχουν πλέον αυτοματοποιημένες διαδικασίες ανάλυσης του κακόβουλου λογισμικού.

2.3.1 Στατική Ανάλυση

Η στατική ανάλυση είναι εκείνη η διαδικασία με την οποία ο κώδικας του κακόβουλου λογισμικού αναλύεται μέσω ειδικού προγράμματος το οποίο αποκαλείται *disassembler*, κι έτσι μπορούμε -ως έναν βαθμό- να αποφανθούμε για την λειτουργικότητα του κακόβουλου προγράμματος, καθώς μας δίνει την δυνατότητα να μελετήσουμε το στατικό σύνολο των οδηγιών (*instruction-set*) του προγράμματος (Sikorski & Honig, 2012).

2.3.2 Δυναμική Ανάλυση

Ως δυναμική ανάλυση του κακόβουλου λογισμικού, ορίζεται εκείνη η διαδικασία κατά την οποία το κακόβουλο λογισμικό εκτελείται, και εν συνεχεία εξετάζεται η συμπεριφορά του μέσω ενός ειδικού προγράμματος *εκσφαλματωτή* (*debugger*), πάντα στα πλαίσια ενός ασφαλούς περιβάλλοντος, λόγω της απρόβλεπτης φύσης του κακόβουλου λογισμικού, κι έτσι αποφαινόμαστε για την συμπεριφορά του κακόβουλου λογισμικού, καθώς αυτό εκτελείται. Αυτή η διαδικασία της δυναμικής ανάλυσης, είναι αποτελεσματικότερη της στατικής ανάλυσης (Sikorski & Honig, 2012).

Επειδή η εκτέλεση ενός κακόβουλου λογισμικού από έναν αναλυτή μπορεί να επιφέρει πάσης φύσεως ζημίες, είναι απαραίτητο αυτή η εκτέλεση να επιτελεστεί στα πλαίσια κάποιου εικονικού (*virtual*) περιβάλλοντος (συνήθως αποκαλούμενου και ως *sandbox*), το οποίο δεν είναι τίποτα άλλο παρά ένα ασφαλές (*secure*) περιβάλλον, που δεν επιτρέπει την άλλως ζημιογόνα εκτέλεση του κακόβουλου λογισμικού. Ένα τέτοιο

ασφαλές περιβάλλον, είναι το Cuckoo Sandbox (Cuckoo Sandbox, 2017), (Oktavianto & Murhadianto, 2013).

Επίσης, στην βιβλιογραφία έχει προταθεί η δημιουργία ασφαλούς εργαστηρίου για την επεξεργασία ιών (*computer viruses / virii*) από τον Aycock (Aycock & Barker, Creating a Secure Computer virus Laboratory (Case Study), 2004), αλλά και από τον Zeltser, γενικότερα για κακόβουλο λογισμικό (*malware*) (Zeltser, 2015).

3 Πρότυπα και γλώσσες περιγραφής

Παρ'όλο που ο χαρακτηρισμός του κακόβουλου λογισμικού δεν είναι μία νέα έννοια (Kirillov I. , Beck, Chase, & Martin, 2014), εντούτοις, στις μέρες μας υπάρχει η ανάγκη για νέες μεθόδους / τεχνολογίες, με τις οποίες θα μπορούμε όχι μόνο να ανιχνεύσουμε το κακόβουλο λογισμικό, αλλά και να το αντιμετωπίσουμε, με έναν αποτελεσματικό τρόπο, καθώς και να ελαχιστοποιήσουμε, να αμβλύνουμε, και να "καταπραΰνουμε" (*mitigate*) τις συνέπειές του στο σύστημα.

Η συμβατική, έως τώρα, μέθοδος αναγνώρισης του κακόβουλου λογισμικού, είναι βασισμένη σε εργαλεία (*anti-virus & anti-malware tools*) τα οποία βασίζονται σε φυσικές υπογραφές (*physical signatures*) λογισμικών, και η οποία μέθοδος δεν είναι τόσο αποτελεσματική, καθώς το κακόβουλο λογισμικό έχει την ιδιότητα να είναι, -πολλές φορές-, πολυμορφικό (*polymorphic*) ή και μεταμορφικό (*metamorphic*) (Kirillov I. , Beck, Chase, & Martin, 2014), κι έτσι να καθίσταται αδύνατη η αναγνώρισή του από τα συμβατικά εργαλεία αναγνώρισης που υπάρχουν. Το κακόβουλο λογισμικό μπορεί επίσης να είναι κρυπτογραφημένο (*encrypted*), πακεταρισμένο (*packed*), ή και κατακερματισμένο (*obfuscated*) (Kirillov I. , Beck, Chase, & Martin, 2014). Συνεπεία αυτού, είναι να διαφεύγει και να παρακάμπτεται (*circumvent*) των τωρινών μεθόδων αναγνώρισης κακόβουλου λογισμικού, η αποτελεσματική αναγνώριση του, με -ίσως- καταστροφικές συνέπειες για το σύστημα.

Οι γλώσσες περιγραφής κακόβουλου λογισμικού μας παρέχουν το ιδανικό πλαίσιο / μηχανισμό, μία πλέον προηγμένη μέθοδο, με την οποία μπορούμε να κινηθούμε προς τον σκοπό αυτό. Ο αντικειμενικός σκοπός αυτών των γλωσσών χαρακτηρισμού, είναι η ακριβής περιγραφή του λογισμικού, -το οποίο και διαφοροποιούμε από το καλοήθες (*benign*) λογισμικό-, ως κακόβουλο λογισμικό ("*malicious software*" ή αλλιώς "*malware*").

Μία σωστά ορισμένη γλώσσα περιγραφής θα πρέπει να λαμβάνει υπ'όψιν κάποια χαρακτηριστικά. Για παράδειγμα, δεν θα πρέπει να βασίζεται μόνο στις υπογραφές (*signatures*) του λογισμικού, ή μόνο στην έξοδο (*output*) της στατικής και της δυναμικής ανάλυσης, αλλά να λαμβάνει υπ'όψιν όλα εκείνα τα χαρακτηριστικά συμπεριφοράς (*behavioral attributes*), τα τεχνουργήματα (*artifacts*), και τα σχήματα

επίθεσης (*attack patterns*) (Kirillov I. , Beck, Chase, & Martin, 2014), κι εν συνεχεία να τα κατηγοριοποιεί, αναλόγως με την σπουδαιότητά τους.

Ο χαρακτηρισμός του κακόβουλου λογισμικού θα πρέπει να είναι λεπτομερής, δηλαδή να λαμβάνει υπ'όψιν όσο περισσότερα χαρακτηριστικά γίνεται, και συνάμα να είναι ακριβής ως προς την περιγραφή του. Εξάλλου, οι γλώσσες περιγραφής, μας επιτρέπουν να συλλέγουμε περισσότερες πληροφορίες για το κακόβουλο λογισμικό (π.χ., η γλώσσα *OpenIOC* περιλαμβάνει περί τους 600 όρους στην έκδοση 1.1), με συνέπεια την καταλληλότερη -άμεση, αλλά και πιο γρήγορη- αντιμετώπισή του.

Σε αυτήν την ενότητα θα εξετάσουμε αναλυτικά, ορισμένες από τις πιο δημοφιλείς γλώσσες ή/και πρότυπα (*standards*) περιγραφής κακόβουλου λογισμικού, που έχουν προταθεί στην βιβλιογραφία. Οι γλώσσες περιγραφής / χαρακτηρισμού του κακόβουλου λογισμικού, θα μπορούσαν να χαρακτηριστούν και ως Domain-Specific Languages (*DSL*), δηλαδή, γλώσσες οι οποίες εφαρμόζονται σε ένα συγκεκριμένο τομέα (*application domain*), ή στην περίπτωσή μας, στον τομέα του κακόβουλου λογισμικού (*malware domain*).

Οι γλώσσες που θα εξετάσουμε (με αλφαβητική σειρά), είναι οι εξής κάτωθι (Kampanakis P. , 2014):

AMBL	Γλώσσα προσδιορισμών (*specification language*), η οποία παρέχει μία πλατφόρμα για την ανίχνευση του κακόβουλου λογισμικού (Deschamps, 2008).
BSL	Πρόκειται για μία γλώσσα η οποία αποτελείται από ένα σύνολο γράφων. Είναι αρκετά αποτελεσματική στην ανίχνευση κακόβουλου λογισμικού (Martignoni, Stinson, Fredrikson, Jha, & Mitchell, 2008).
CAPEC™	Πρόκειται για γλώσσα, η οποία χρησιμεύει στο να ορίζουμε την λίστα των πλέον κοινών σχημάτων επίθεσης (*common attack patterns*) (Rattan, Kaur, Chamotra, & Bhushan, 2017).
CybOX™	Πρόκειται για γλώσσα περιγραφής, η οποία, "*έχει σχεδιαστεί ώστε να μπορεί να περιγράφει μια μεγάλη ποικιλία περιπτώσεων χρήσης (use cases) του κυβερνο-πεδίου (cyber domain)*" (Obrst, Chase, & Markeloff, 2012).

IODEF Πρόκειται για μια μορφοποίηση δεδομένων (*data format*) που περιγράφουν συμβάντα ασφαλείας, με σκοπό τον διαμοιρασμό τους (IODEF, 2018).

MAEC™ Δημιουργήθηκε από την εταιρεία κυβερνο-ασφάλειας (*cyber security*) Mitre Corporation. Είναι γλώσσα περιγραφής ειδικά φτιαγμένη για κακόβουλο λογισμικό, με προεκτάσιμο (*extensible*) χαρακτήρα.

MAIL Είναι μία ενδιάμεση (*intermediate*) γλώσσα η οποία χρησιμοποιεί εντολές γλώσσας μηχανής (*assembly*), κι έχει ικανοποιητικά αποτελέσματα ως προς την ανίχνευση του κακόβουλου λογισμικού.

MBO Πρόκειται για οντολογία (*ontology*) η οποία περιγράφει την συμπεριφορά του κακόβουλου λογισμικού. (Grégio et al., 2016).

MIST Είναι μία μετα-γλώσσα (*meta-language*) η οποία δεν χρησιμοποιεί XML για την περιγραφή του κακόβουλου λογισμικού, αλλά είναι βελτιστοποιημένη (*optimized*) για περαιτέρω ανάλυση (Trinius et al., 2009).

OpenIOC Πρόκειται για πρότυπο/σκελετό (*framework*) το οποίο λαμβάνει δεδομένα σε μορφή XML (eXtensible Markup Language), για την περιγραφή των δεικτών επικινδυνότητας (*IoC, Indicators of Compromise*) (Mandiant, 2017).

OVAL® Η γλώσσα OVAL δημιουργήθηκε από την εταιρεία Mitre Corporation. Μεταξύ άλλων περιπτώσεων χρήσης (*use cases*), χρησιμοποιείται και για την ανίχνευση κακόβουλου λογισμικού (Open Vulnerability and Assessment Language, 2016).

RBS Είναι μία γλώσσα προσδιορισμών (*specification language*), η οποία αποθηκεύει την ολοκληρωμένη διαφαινόμενη συμπεριφορά κάποιου δείγματος κακόβουλου λογισμικού, για την μεταγενέστερη ανάλυσή της (Deschamps, 2008).

VERIS Είναι ένα πρότυπο/πλαίσιο (*framework*) το οποίο αποτελείται από ένα σύνολο στοιχείων με τα οποία μπορούμε να περιγράψουμε συμβάντα ασφαλείας (*security incidents*) σε μια δομημένη μορφή (*structured format*) (VERIS Framework, 2018).

YARA Πρόκειται για εργαλείο, γραμμένο σε γλώσσα προγραμματισμού Python, το οποίο βασίζεται σε υπογραφές γραμμένες σε γλώσσα JSON

(*JavaScript Object Notation*). Είναι μια απλή γλώσσα περιγραφής κακόβουλου λογισμικού, αλλά είναι περιορισμένη ως προς το εύρος του τί μπορεί να περιγράψει (YARA, 2018).

3.1.1 AMBL

Η γλώσσα AMBL (Abstract Malicious Behavioral Language), είναι μία γλώσσα προσδιορισμών (*specification language*), και ως τέτοια, παρέχει μία πλατφόρμα για την ανίχνευση του κακόβουλου λογισμικού (Yu, Fang, Yang, Tang, & Liu, 2017).

Έχει σχεδιαστεί για να είναι μία συμπαγής (*compact*), και κατανοητή από τον άνθρωπο (*human-readable*) αντιπροσώπευση των συλλεχθέντων δεδομένων, και έχει ως αποκλειστικό σκοπό την ανίχνευση (*detection*) του κακόβουλου λογισμικού (Deschamps, 2008).

3.1.1.1 Αρχιτεκτονική / Δομή της γλώσσας AMBL

Η γλώσσα AMBL, αποτελείται από δηλώσεις συμπεριφορικών υπογραφών (*behavioral signatures*), κάτι που το καθιστά εύκολο να χτίζει αποτελεσματικά και ανθεκτικά αυτόματα (*parsing automata*) προς την ανίχνευση του κακόβουλου λογισμικού (Yu, Fang, Yang, Tang, & Liu, 2017).

Είναι στην βάση της μία αντικειμενοστραφής γλώσσα (*object-oriented*), και συγκεντρώνει όλα εκείνα τα στοιχεία μίας σύγχρονης γλώσσας προγραμματισμού, όπως π.χ., ότι είναι μία Turing-Complete γλώσσα.

Πρόκειται για γραμματική-χωρίς-συμφραζόμενα (*Context-Free Grammar, CFG*), η οποία είναι εμπλουτισμένη από σημασιολογικά χαρακτηριστικά (*semantic attributes*), και κανόνες (*rules*). Σε μια τέτοια δομή, οι συντακτικοί κανόνες της γραμματικής ορίζουν τους συνδυασμούς των χαρακτηριστικών του κακόβουλου λογισμικού, και οι σημασιολογικοί κανόνες ελέγχουν την ροή των δεδομένων μεταξύ των στοιχείων που λαμβάνουν μέρος σε αυτή τη δομή (Deschamps, 2008).

3.1.2 BSL

Η γλώσσα BSL (Behavior Specification Language), αποτελείται από ένα σύνολο πρωτόγονων δομών (*primitives*), οι οποίες μπορούν να χρησιμοποιηθούν για τον ορισμό επιπλέον συμπεριφορών του κακόβουλου λογισμικού. Επίσης, δίνεται η δυνατότητα σύνθεσης αυτών των πρωτόγονων δομών, σε πιο πολύπλοκες δομές, έτσι ώστε η γλώσσα BSL να μπορεί να προσδιορίζει, και να ταυτοποιεί νέες σημασιολογικές (*semantic*) συμπεριφορές (Yu, Fang, Yang, Tang, & Liu, 2017). Για αυτόν τον λόγο κάνει χρήση ιεραρχικών γράφων συμπεριφορών, ούτως ώστε να εξαγάγει συμπεριφορές υψηλού επιπέδου (Martignoni, Stinson, Fredrikson, Jha, & Mitchell, 2008).

Πρόκειται για γλώσσα, η οποία είναι πολύ αποτελεσματική στην διάγνωση και ανίχνευση του κακόβουλου λογισμικού, καθώς μπορούμε να ταυτοποιήσουμε κακόβουλες συμπεριφορές μέσα από ένα πλήθος βάσεων κώδικα (*code-bases*). (Martignoni, Stinson, Fredrikson, Jha, & Mitchell, 2008).

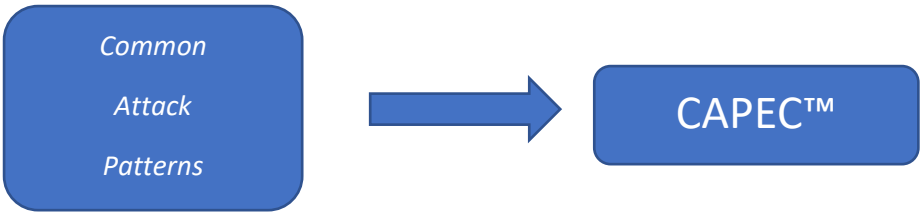
Έτσι, με την γλώσσα BSL μπορούμε να βελτιώσουμε το ποσοστό ανίχνευσης του κακόβουλου λογισμικού, κάτι που έρχεται σε αντίθεση με τα ποσοστά ανίχνευσης του κακόβουλου λογισμικού από προγράμματα ανίχνευσης κακόβουλου λογισμικού βασισμένα σε υπογραφές (*signature-based anti-malware applications*) – πρόκειται για μία προσέγγιση βασισμένη στην συμπεριφορά του κακόβουλου λογισμικού (*behavior-based approach*).

3.1.2.1 Δομή / Χαρακτηριστικά της γλώσσας BSL

Η γλώσσα BSL αποτελείται από πρωτόγονες δομές (*primitives*) οι οποίες είναι στην ουσία γράφοι συμπεριφορών. (Martignoni, Stinson, Fredrikson, Jha, & Mitchell, 2008). Στην βάση του, το σύστημα αποτελείται από γράφους σε κάθε επίπεδο (*layer*). Κάθε μία από αυτές τις συμπεριφορές είναι μία πρωτόγονη δομή (*primitive*), η οποία μπορεί να χρησιμοποιηθεί περαιτέρω, για την περιγραφή πιο σύνθετων συμπεριφορών.

Στους γράφους συμπεριφορών, κάθε ορθογώνιο αντιπροσωπεύει ένα γεγονός (*event*), και αυτά τα γεγονότα συνδέονται μεταξύ τους με ακμές, δηλαδή τις εξαρτήσεις (*dependencies*) μεταξύ τους.

3.1.3 CAPEC™



Εικόνα 1. Common Attack Pattern Enumeration and Classification

Η γλώσσα CAPEC™ (Common Attack Pattern Enumeration and Classification) αποτελεί δημιούργημα της εταιρείας Mitre Corporation (Mitre Corporation, 2017), (CAPEC™, 2018).

Χρησιμεύει στο να ορίζουμε την λίστα των πλέον κοινών σχημάτων επίθεσης (*common attack patterns*) (Rattan, Kaur, Chamotra, & Bhushan, 2017). Κατά τους Mavroeidis *et al.* (Mavroeidis & Bromander, 2017), η γλώσσα CAPEC™, είναι μία συλλογή από τις πιο κοινές τεχνικές / μεθόδους, οι οποίες χρησιμοποιούνται στις κυβερνο-επιθέσεις (*cyber attacks*).

3.1.3.1 Δομή της γλώσσας CAPEC™

Η γλώσσα ενδείκνυται για να βοηθήσει τους προγραμματιστές ως προς το να δημιουργήσουν (*build*) πιο ασφαλές λογισμικό (Mitre Corporation, 2017). Για αυτό τον λόγο στην δομή της γλώσσας συμπεριλαμβάνονται όροι, όπως, “SQL injection”, “XSS – Cross Site Scripting”, “Phishing”, κ.ά.

Τα σχήματα επίθεσης, είναι δομημένες (*structured*) περιγραφές των πλέον κοινών μεθόδων «εκμετάλλευσης» λογισμικού (*software exploiting*), κι έχουν προέλθει από παραδείγματα στον πραγματικό κόσμο, και τυγχάνουν εφαρμογής σε όλες τις φάσεις του κύκλου ζωής ανάπτυξης του λογισμικού (*SDLC – Software Design Life Cycle*), με στόχο την αποτελεσματική τους αντιμετώπιση (*mitigation*).

Η δομή της γλώσσας CAPEC™, διαφαίνεται παρακάτω:

1. Όνομα σχήματος επίθεσης
2. Περιγραφή επίθεσης
3. Προαπαιτούμενα επίθεσης
4. Τυπική πιθανότητα (*likelihood*) της εκμετάλλευσης (*exploit*)
5. Μέθοδοι της επίθεσης

6. Παραδείγματα – Αναφορές
7. Ικανότητα του επιτιθέμενου ή γνώση που απαιτείται
8. Απαιτούμενοι πόροι (*resources*)
9. Τεχνικές ανίχνευσης (*probing*)
10. Δείκτες – Προειδοποιήσεις επίθεσης
11. Λύσεις και αντιμετώπιση (*mitigation*)
12. Κίνητρο επίθεσης – Συνέπειες
13. Περιγραφή συμφραζομένων
14. Διάνυσμα εμβόλισης (*injection vector*)
15. Φορτίο (*payload*)
16. Ζώνη ενεργοποίησης
17. Αντίκτυπος ενεργοποίησης φορτίου (*payload*)
18. Αντίκτυπος CIA
19. Σχετιζόμενες αδυναμίες
20. Σχετιζόμενες απαιτήσεις ασφάλειας
21. Σχετιζόμενες αρχές ασφάλειας
22. Σχετιζόμενες οδηγίες (*guidelines*)
23. Αναφορές

3.1.4 CybOX™

Η γλώσσα CybOX™ (Cyber Observable eXpression), έχει δημιουργηθεί από την εταιρεία Mitre Corporation.

Πρόκειται για γλώσσα περιγραφής, η οποία, "*έχει σχεδιαστεί ώστε να μπορεί να περιγράφει μια μεγάλη ποικιλία περιπτώσεων χρήσης (use cases) του κυβερνο-πεδίου (cyber domain)*" (Obrst, Chase, & Markeloff, 2012).

Είναι μία ευέλικτη (*flexible*), δομημένη (*structured*) και τυποποιημένη (*standardized*) γλώσσα, η οποία χρησιμοποιείται για τον προσδιορισμό, την σύλληψη (*capture*), τον χαρακτηρισμό, και την επικοινωνία των παρατηρηθέντων συμβάντων (*cyber-observable events*). Τέτοια παρατηρηθέντα συμβάντα μπορεί να είναι η δημιουργία ενός κλειδιού μητρώου συστήματος (*registry key*), η διαγραφή κάποιου αρχείου, η ύπαρξη κάποιου αρχείου με συγκεκριμένο MD5 hash, η λήψη συγκεκριμένης αίτησης HTPP (*HTTP request*), κ.α. (Mitre Corporation, 2017)

Μία από τις εφαρμογές της γλώσσας CybOX™, είναι και για τον χαρακτηρισμό του κακόβουλου λογισμικού (*malware characterization*) (Mitre Corporation, 2017). Σύμφωνα με τους Barnum et al. (Barnum, Martin, Worrell, & Kirillov, 2012), η γλώσσα CybOX™, ως προς την περίπτωση χρήσης του κακόβουλου λογισμικού, έχει τις εξής δυνατότητες:

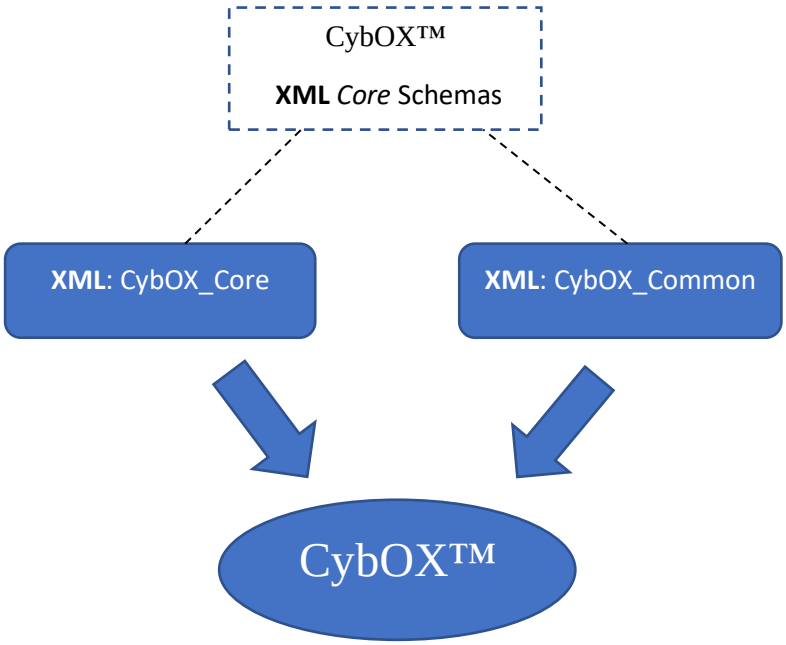
- Ανάλυση μοτίβων (*patterns*) κακόβουλου λογισμικού, καθώς και του ιδίου του κακόβουλου λογισμικού.
- Αναζήτηση (*hunting*) τεχνουργημάτων (*artifacts*) κακόβουλου λογισμικού.
- Κατηγοριοποίηση (*indexing*) μετα-δεδομένων (*metadata*) κακόβουλου λογισμικού.
- Διαμοιρασμό πληροφορίας χαρακτηρισμού του κακόβουλου λογισμικού.

Ως προς το κακόβουλο λογισμικό, επίσης, η γλώσσα CybOX™, μπορεί να χαρακτηρίσει την συμπεριφορά του, να ανιχνεύσει τις συνέπειες του στο σύστημα, και να βοηθήσει την περαιτέρω ανάλυση του (Mitre Corporation, 2017).

3.1.4.1 Δομή της γλώσσας CybOX™

Οι δομές της γλώσσας CybOX™ αντιπροσωπεύονται από την γλώσσα XML. Μπορεί να περιέχει ως στοιχεία της (*elements*), *hashes*, συμβολοσειρές (*strings*), ή κλειδιά μητρώου συστήματος (*registry keys*) (Van Impe, 2015).

Η δομή της γλώσσας CybOX™, διαφαίνεται στο παρακάτω σχήμα:



Εικόνα 2. Δομή της γλώσσας CybOX™

Η λειτουργικότητα της γλώσσας προσδίδεται μέσω των 2 κυρίως σχημάτων XML: (1) *CybOX_Core*, (2) *CybOX_Common*, στα οποία και απαριθμούνται (*enumerated*) τα αντικείμενα της γλώσσας, κατά τέτοιον τρόπο ώστε να χρησιμοποιούνται μόνο αυτά που χρειάζονται, κατά βούληση (Mitre Corporation, 2017).

Οι κυρίως κλάσεις-αντικείμενα της γλώσσας, έχουν ως εξής (CybOX™ Design: Object Hierarchy Structuring, 2018):

- API
- ARP Cache
- AS
- Account
    - User Account Extension
    - Computer Account Extension
    - Unix Account Extension
    - Unix User Account Extension
    - Windows User Account Extension
- IP Address
- IPv6 Address
- IPv4 Address
- MAC Address

- Email Address
- File
    - File Metadata Extension
    - EXT3 File Extension
    - NTFS File Extension
    - Image File Extension
    - PDF File Extension
    - Archive File Extension
    - PE Binary File Extension
    - Library File Extension
    - Linux Package Extension
    - Windows Driver Extension
- Artifact
- Code
- Custom
- DNS Cache
- DNS Query
- DNS Response
- DNS Record
- Device
    - Disk Device Extension
- Disk Partition
- Domain Name
- Email Message
- GUI Dialogbox
- GUI Window
- HTTP Request
- HTTP Response
- Memory
    - Windows Memory Page Region Extension
- Mutex
    - Windows Mutex Extension
- Network Connection
- Network Flow
- Network Packet
- Network Route Entry
    - Unix Network Route Entry Extension
    - Windows Network Route Entry Extension
- Network Route
- Network Socket
- Network Subnet
- Pipe

    - Unix Pipe Extension
    - Windows Pipe Extension
- Process
    - Unix Process Extension
    - Windows Process Extension
    - Windows Service Extension
- Product
- SMS Message
- Semaphore
    - Windows Semaphore Extension
- Socket Address
- System
    - Windows System Extension
- Thread
    - Windows Thread Extension
- URI
- URL History
- User Session
- Volume
    - Unix Volume Extension
    - Windows Volume Extension
- Windows Critical Section
- Windows Event Log
- Windows Event
- Windows Filemapping
- Windows Handle
- Windows Hook
- Windows Kernel Hook
- Windows Kernel
- Windows Mailslot
- Windows Network Share
- Windows Registry Key
- Windows System Restore
- Windows Task
- Windows Waitable Timer
- X.509 Certificate

Η γλώσσα CybOX™ έχει στην έκδοσή της 3.0, 61 αντικείμενα και 30 επεκτάσεις. Μεταξύ άλλων κλάσεων, μπορούμε να διακρίνουμε: API, IP Address, DNS κλάσεις, Network κλάσεις, Windows κλάσεις, File κλάσεις, Artifact, κ.α.

Θα πρέπει εδώ να σημειωθεί, ότι η γλώσσα CybOX™, έχει πλέον ενσωματωθεί στην έκδοση 2.0 της γλώσσας STIX™ (Mitre Corporation, 2017).

3.1.5 IODEF

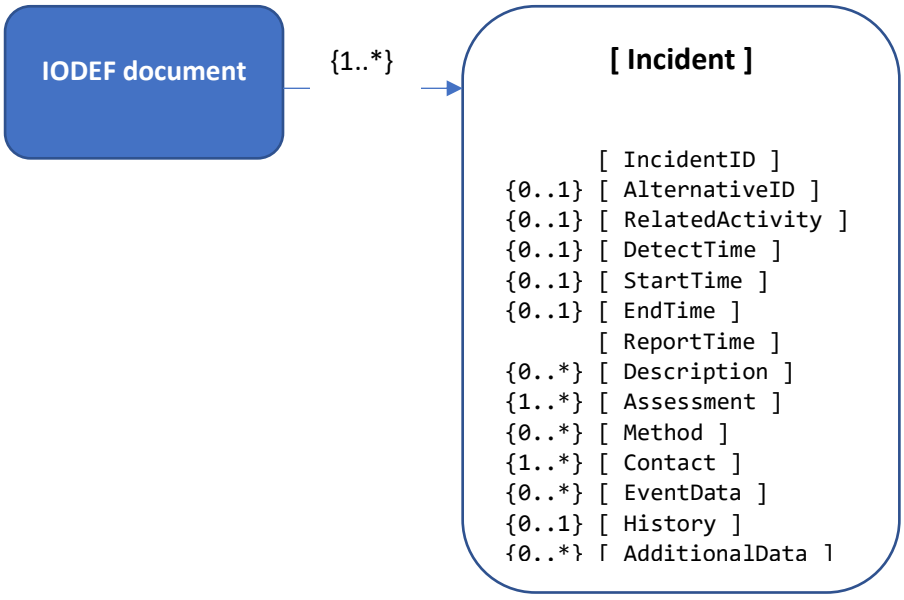
Η γλώσσα IODEF (Incident Object Description and Exchange Format), δημιουργήθηκε από μέλη της ομάδας IETF[3] - Extended Incident Handling (INCH) Working Group (IODEF, 2018).

Πρόκειται για μια μορφοποίηση δεδομένων (*data format*), γραμμένη σε γλώσσα περιγραφής δεδομένων XML, που περιγράφουν λειτουργικές (*operational*) και στατιστικές πληροφορίες συμβάντων ασφαλείας (*security incidents*), με σκοπό τον διαμοιρασμό τους (*sharing*) ή και την ανταλλαγή τους (*exchange*) μεταξύ των ομάδων απόκρισης συμβάντων ασφάλειας υπολογιστών - CSIRT[4] (Incident Object Description Exchange Format, 2018), (IODEF, 2018).

Η γλώσσα IODEF, περιγράφεται αναλυτικά στα έγγραφα RFC[5], RFC-5070 (Danilyw, Meijer, & Demchenko, 2007), RFC-6685, RFC-7970, καθώς και στο RFC-8274 (Kampanakis & Suzuki, 2017), (Incident Object Description Exchange Format, 2018).

3.1.5.1 Δομή της γλώσσας IODEF

Η γλώσσα IODEF περιγράφεται μέσω της γλώσσας περιγραφής δεδομένων XML.



Εικόνα 3. Δομή εγγράφου της γλώσσας IODEF

[3] IETF: Internet Engineering Task Force.
[4] CSIRT: Computer Security Incident Response Teams, Ομάδες Απόκρισης σε Συμβάντα Ασφαλείας Υπολογιστών.
[5] RFC: Έγγραφο (Document), Request For Comments.

Το έγγραφο IODEF, αποτελείται από την κλάση *Incident* (μία ή περισσότερες), η οποία περιγράφει το συμβάν ασφαλείας (Danilyw, Meijer, & Demchenko, 2007), και αποτελείται από 4 χαρακτηριστικά (*attributes*):

1. purpose: Απαρίθμηση / ENUM (απαιτείται / required). Ο σκοπός, η αιτία δημιουργίας του εγγράφου IODEF.
2. ext-purpose: Συμβολοσειρά / STRING (προαιρετικό πεδίο / optional). Ο τρόπος επέκτασης του παραπάνω ("*purpose*") χαρακτηριστικού.
3. lang: Απαρίθμηση / ENUM (προαιρετικό πεδίο / optional). Η γλώσσα στην οποία περιγράφεται το συμβάν, κατά το έγγραφο RFC-4646.
4. restriction: Απαρίθμηση / ENUM (προαιρετικό πεδίο / optional). Αυτό το χαρακτηριστικό περιγράφει τους περιορισμούς στον διαμοιρασμό (*sharing*) του εγγράφου. Έτσι, έχουμε τα παρακάτω υπο-χαρακτηριστικά: δημόσιο (*public*), ιδιωτικό (*private*), απαραίτητο προς γνώση (*need-to-know*), και, προκαθορισμένο (*default*), στην οποία περίπτωση, ο διαμοιρασμός θα γίνει σύμφωνα με προσυμφωνημένο έγγραφο.

Τα κυρίως συστατικά (*components*) της κυρίως κλάσης Incident, έχουν ως εξής (Danilyw, Meijer, & Demchenko, 2007) :

- `IncidentID`: (1 πεδίο). Πρόκειται για τον αριθμό ταυτοποίησης (*ID number*) με τον οποίο αριθμό αναφέρεται το έγγραφο από τις ομάδες απόκρισης σε συμβάντα ασφαλείας ("*CSIRT*"), δηλαδή από τους επαγγελματίες ασφάλειας υπολογιστικών συστημάτων.
- `AlternativeID`: (0 ή 1 πεδία). Εναλλακτικός αριθμός ταυτοποίησης του εγγράφου (ID number), που χρησιμοποιείται από τις ομάδες απόκρισης.
- `RelatedActivity`: (0 ή 1 πεδία). Συσχετιζόμενη δραστηριότητα: οι αριθμοί ταυτοποίησης των εγγράφων που σχετίζονται/συνδέονται με το συγκεκριμένο έγγραφο IODEF.
- `DetectTime`: (0 ή 1 πεδία). Ο χρόνος κατά τον οποίο ανιχνεύτηκε πρώτα το συμβάν.
- `StartTime`: (0 ή 1 πεδία). Ο χρόνος κατά τον οποίο ξεκίνησε το συμβάν.
- `EndTime`: (0 ή 1 πεδία). Ο χρόνος κατά τον οποίο έληξε το συμβάν.
- `ReportTime`: (1 πεδίο). Ο χρόνος κατά τον οποίο αναφέρθηκε το συμβάν.

- `Description`: (0 ή περισσότερα πεδία). Η περιγραφή του συμβάντος: κειμενική (*textual*) ελεύθερη περιγραφή του συμβάντος.
- `Assessment`: (1 ή περισσότερα πεδία). Χαρακτηρισμός της επίπτωσης (*impact*) του συμβάντος.
- `Method`: (0 ή περισσότερα πεδία). Οι διάφορες τεχνικές που χρησιμοποιήθηκαν από τον εισβολέα (*intruder*), κατά την διάρκεια του συμβάντος.
- `Contact`: (1 ή περισσότερα πεδία). Πληροφορίες επικοινωνίας για τις ομάδες που αναμείχθηκαν στο συμβάν.
- `EventData`: (0 ή περισσότερα πεδία). Περιγραφή των γεγονότων που έλαβαν χώρα κατά το συμβάν.
- `History`: (0 ή 1 πεδία). Το ημερολόγιο (*log*) των συμβάντων που έλαβαν χώρα κατά την διαχείριση του συμβάντος.
- `AdditionalData`: (0 ή περισσότερα πεδία). Επιπρόσθετα δεδομένα: ένας μηχανισμός προσθήκης/επέκτασης του μοντέλου δεδομένων της IODEF.

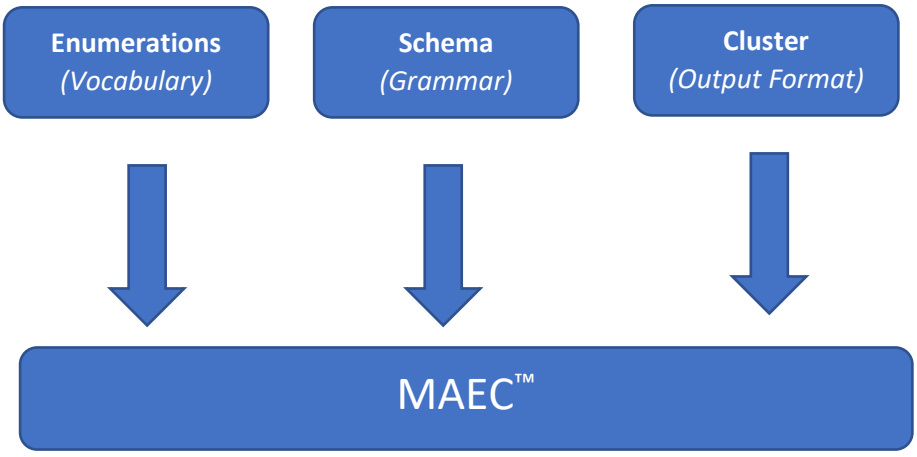
3.1.6 MAEC™

Η γλώσσα MAEC™ (Malware Attribute Enumeration and Characterization) δημιουργήθηκε από την εταιρεία Mitre Corporation, και είναι μία δομημένη (*structured*), τυποποιημένη (*standardized*) γλώσσα περιγραφής κακόβουλου λογισμικού, η οποία αναπτύσσεται από την κοινότητα, με απώτερο σκοπό την κωδικοποίηση χαρακτηριστικών του κακόβουλου λογισμικού, όπως για παράδειγμα, συμπεριφορικών χαρακτηριστικών (*behavioral attributes*), τεχνουργημάτων (*artifacts*), καθώς και των σχέσεων (*relationships*) μεταξύ δειγμάτων κακόβουλου λογισμικού (Kirillov I. A., Beck, Chase, & Martin, 2010). Επίσης, πρέπει να σημειώσουμε ότι έχει επεκτάσιμο (*extensible*) χαρακτήρα.

3.1.6.1 Δομή της γλώσσας MAEC™

Η δομή της MAEC™ αποτελείται από 3 κυρίως μέρη, όπως μπορούμε να δούμε στο παρακάτω σχήμα:

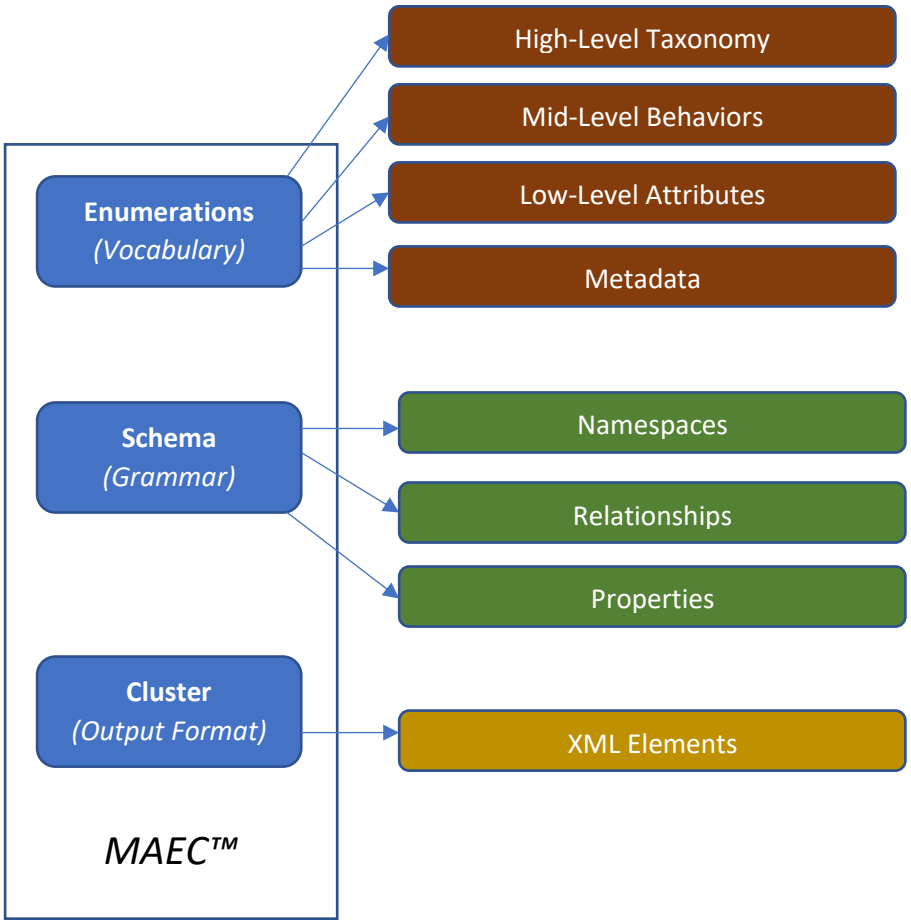
1. Enumerations (Vocabulary) – Απαριθμήσεις (Λεξιλόγιο)
2. Schema (Grammar) – Σχήμα (Γραμματική)
3. Cluster (Output Format) – Συστοιχία (Μορφή Εξόδου)



Εικόνα 4. Αρχιτεκτονική της γλώσσας MAEC

Τα επιμέρους συστατικά (*components*) της κάθε κατηγορίας είναι τα εξής:

1. **Enumerations (Vocabulary)**
    1. *High-level taxonomy – Ταξινόμηση υψηλού επιπέδου*
    2. *Mid-level behaviors – Συμπεριφορές μεσαίου επιπέδου*
    3. *Low-level attributes – Χαρακτηριστικά χαμηλού επιπέδου*
    4. *Metadata – Μετα-δεδομένα*
2. **Schema (Grammar)**
    1. *Namespaces – Χώρος ονομάτων*
    2. *Relationships – Συσχετισμοί, σχέσεις*
    3. *Properties - Ιδιότητες*
3. **Cluster (Output Format)**
    1. *XML Elements – Στοιχεία γλώσσας XML*



Εικόνα 5. Δομή της γλώσσας MAEC

Αναλυτικότερα, έχουμε τα εξής (Kirillov I. A., Beck, Chase, & Martin, 2010):

- Low-level attributes – Χαρακτηριστικά χαμηλού επιπέδου: Πρόκειται για τις αλλαγές στην κατάσταση του συστήματος που επιφέρει το κακόβουλο λογισμικό. Παραδείγματα είναι, αλλαγές στο αρχείο ή στο μητρώο (*registry*), η δημιουργία κάποιας σύνδεσης TCP/IP, αλλαγές σε διεργασίες στη μνήμη του συστήματος, ή οποιαδήποτε άλλη αλλαγή μπορεί να επιφέρει η εκτέλεση του κακόβουλου λογισμικού.
- Mid-level behaviors – Συμπεριφορές μεσαίου επιπέδου: Πρόκειται για την οργάνωση σε συστοιχίες (*clusters*), των χαρακτηριστικών χαμηλού επιπέδου, έτσι ώστε να γίνεται προφανής ο σκοπός ή ο στόχος αυτών των χαρακτηριστικών, δηλαδή το *γιατί* αυτά έγιναν.
- High-level taxonomy – Ταξινόμηση υψηλού επιπέδου: Πρόκειται για την ταξινόμηση σε υψηλότερο επίπεδο των συστοιχιών των συμπεριφορών μεσαίου επιπέδου, με απώτερο σκοπό την έκθεσή τους (*view*) σε διαφορετικά κοινά (*audiences*). Για παράδειγμα, ο αναλυτής κακόβουλου λογισμικού (*forensic analyst*), ενδιαφέρεται μόνο για την συμπεριφορά του φορτίου (*payload*) του κακόβουλου λογισμικού, και αυτή την πληροφορία θα πρέπει να λάβει, με έναν οργανωμένο τρόπο.
- Metadata – Μετα-δεδομένα: Τα μετα-δεδομένα (δηλαδή "*δεδομένα αναφορικά / σχετικά με δεδομένα*"), είναι επιπλέον πληροφορίες σχετικά με το κακόβουλο λογισμικό, τέτοιες ώστε να περιγράφονται καλύτερα τα χαρακτηριστικά του χαμηλού επιπέδου.
- Namespaces – Χώροι ονομάτων: Σε ένα σχήμα (*schema*) της MAEC, είναι η ομαδοποίηση (*grouping*) εκείνων των παρατηρητέων (*observables*) και άλλων χαρακτηριστικών, σε μία τάξη (*class*).
- Relationships – Συσχετισμοί: Πρόκειται για τις σχέσεις / συσχετισμούς μεταξύ των διαφόρων χώρων ονομάτων σε ένα σχήμα MAEC. Για παράδειγμα, ένας χώρος ονομάτων (*namespace*) μεσαίου επιπέδου συμπεριφορών, μπορεί να αποτελείται από έναν ή περισσότερους χώρους ονομάτων μεσαίου επιπέδου χαρακτηριστικών.
- Properties – Ιδιότητες: Πρόκειται για ένα γενικό σύνολο ιδιοτήτων, εφαρμοστέων σε χαρακτηριστικά και χώρους ονομάτων, με συγκεκριμένες ιδιότητες για συμπεριφορές.

- XML Elements – Στοιχεία γλώσσας XML: Πρόκειται για την μορφή σε XML, των διαφόρων χαρακτηριστικών του κακόβουλου λογισμικού, όπως αυτά περικλείονται σε διάφορες συστοιχίες (*clusters*).

Ως υποκείμενη γλώσσα, για την περιγραφή των παραπάνω, χρησιμοποιείται η γλώσσα XML[6], ενώ σε μελλοντικές εκδόσεις της θα υιοθετηθεί η γλώσσα JSON[7] και η γλώσσα RDF[8], καθώς και άλλες «ελαφριές» (*light-weight*) γλώσσες (Kirillov I. , Beck, Chase, & Martin, 2014).

[6] XML: Extensible Markup Format Language.
[7] JSON: JavaScript Object Notation.
[8] RDF: Resource Description Framework.

3.1.7 MAIL

Η γλώσσα MAIL (Malware Analysis Intermediate Language) (Alam, 2014), είναι μία «ενδιάμεση» (*intermediate*) γλώσσα, σκοπός της οποίας είναι να περιγράψει την ροή της πληροφορίας σε ένα πρόγραμμα ως έναν γράφο (Yu, Fang, Yang, Tang, & Liu, 2017), και έτσι να μπορέσει να διακρίνει μεταξύ ενός καλοήθους (*benign*) προγράμματος κι ενός κακόβουλου (*malicious*) προγράμματος, με το να συγκρίνει, μέσω της διαδικασίας προσαρμογής μοτίβου (*pattern matching*), αυτούς τους γράφους ροών (*flow graphs*) στην πληροφορία του κάθε προγράμματος.

3.1.7.1 Αρχιτεκτονική / Δομή της γλώσσας MAIL

Η δομή της γλώσσας MAIL, μοιάζει σε αυτή της ενδιάμεσης γλώσσας η οποία παράγεται από προγράμματα γραμμένα σε γλώσσες προγραμματισμού όπως η Java της οποίας ο κώδικας μεταφράζεται σε "*bytecode*", ή η γλώσσα C# της οποίας ο κώδικας πρώτα μεταγλωττίζεται σε ενδιάμεση γλώσσα CIL (*Common Intermediate Language*).

Θα πρέπει να σημειωθεί εδώ, ότι αυτές οι δύο προαναφερθείσες ενδιάμεσες γλώσσες, μοιάζουν ως έναν βαθμό, ως προς τη δομή τους και ως προς τις εντολές που αυτές χρησιμοποιούν, προς την γλώσσα προγραμματισμού χαμηλού επιπέδου (*low-level language*) Assembly, ή κώδικα μηχανής (*machine code*).

Ο ρόλος της γλώσσας MAIL, είναι να αντιπροσωπεύσει την δομική (*structural*) καθώς και συμπεριφορική (*behavioral*) πληροφορία που υποκρύπτεται σε ένα πρόγραμμα γραμμένο σε γλώσσα μηχανής (*assembly*), το οποίο προορίζεται για την ανάλυση και ανίχνευση του κακόβουλου λογισμικού. Επίσης, θα καταστήσει το πρόγραμμα αυτό, αναγνώσιμο και κατανοητό από έναν αναλυτή κακόβουλου λογισμικού (Alam, 2014).

Η γλώσσα MAIL εμπεριέχει τις κάτωθι κατηγορίες εντολών γλώσσας μηχανής (Alam, 2014):

1. Εντολές ελέγχου (Control instructions): Εντολές οι οποίες αλλάζουν την ροή του προγράμματος, όπως οι, `JMP`, `CALL`, `RET`, `CMP` κ.ά.
2. Αριθμητικές εντολές (Arithmetic instructions): Εντολές οι οποίες εκτελούν αριθμητικές πράξεις, όπως οι, `ADD`, `SUB`, `DIV`, `MUL`, `SHR`, `SHL`, κ.ά.
3. Λογικές εντολές (Logical instructions): Εντολές οι οποίες κάνουν λογικές λειτουργίες, όπως οι, `AND`, `OR`, `NOT`.

4. Εντολές μεταφοράς δεδομένων (Data transfer instructions): Εντολές οι οποίες εκτελούν μεταφορά δεδομένων, όπως οι, `MOV`, `XCHG`, `PUSH`, `POP`, κ.ά.
5. Εντολές συστήματος (System instructions): Εντολές οι οποίες εκτελούν κλήσεις συστήματος, όπως οι, `LOCK`, `LTR`, `STR`, `XSAVE`, κ.ά.
6. Διάφορες εντολές: Αυτές οι εντολές δεν εμπίπτουν στις παραπάνω κατηγορίες εντολών, όπως οι `NOP`, `HLT`, `WAIT`, κ.ά.

Η δε γραμματική της γλώσσας MAIL δίνεται παρακάτω, σε μορφή *Backus-Naur Form*, BNF (Alam, 2014):

```
statements ::= ( statement* ) ;
statement  ::= assignment_s+
             | control_s+
             | condition_s+
             | function_s+
             | jump_s+
             | lib_call_s+
             | 'halt'
             | 'lock' ;
```

Εικόνα 6. BNF μορφή της γλώσσας MAIL

, όπου κάθε δήλωση (statement) μπορεί να εμπεριέχει περισσότερες από μία υποδηλώσεις, και όπου υποδηλώσεις μπορεί να αποτελούνται από δομές ανάθεσης (*assignment*), ελέγχου (*control*), συνθήκες (*conditions*), ρουτίνες (*functions*), και κλήσεις βιβλιοθηκών (*library calls*).

Η διαδικασία που ακολουθείται για την ανίχνευση κακόβουλου κώδικα σε κάποιο πρόγραμμα κακόβουλου λογισμικού, είναι πρώτα αυτό να μεταφραστεί σε κώδικα της γλώσσας MAIL, και μετά να εφαρμοστούν οι τεχνικές ταιριάσματος μοτίβων (*pattern matching*). Η ίδια διαδικασία ακολουθείται είτε πρόκειται για γνωστό ή για άγνωστο κακόβουλο λογισμικό (Alam, 2014).

3.1.7.2 Ποσοστά ανίχνευσης κακόβουλου λογισμικού, κατά MAIL

Σύμφωνα με τον Alam (Alam, 2014), η ενδιάμεση γλώσσα MAIL έχει υψηλά ποσοστά ανίχνευσης κακόβουλου λογισμικού: 100% επιτυχία για γνωστό κακόβουλο λογισμικό, και ~(94-100)% για πρότινος άγνωστο κακόβουλο λογισμικό.

3.1.8 MBO

Η MBO (Malicious Behavior Ontology) είναι μία οντολογία *(ontology),* η οποία περιγράφει την συμπεριφορά του κακόβουλου λογισμικού. (Grégio, Bonacin, & Nabuco, 2016). Με τον όρο *οντολογία*, εννοούμε, εκείνη την αντιπροσώπευση εννοιών, δεδομένων, και οντοτήτων, η οποία περιλαμβάνει την επίσημη ονοματολογία, καθώς και τους ορισμούς των κατηγοριών, ιδιοτήτων και σχέσεων μεταξύ των εννοιών, δεδομένων, και οντοτήτων οι οποίες περιλαμβάνονται σε αυτήν (Ontology (information science), 2018).

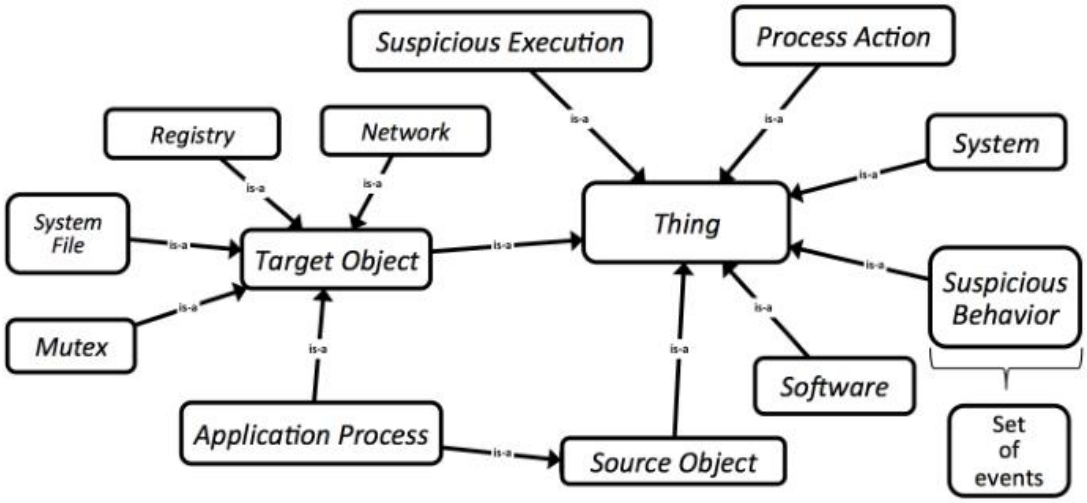
Οι Mavroeidis & Bromander (Mavroeidis & Bromander, 2017), αναφέρουν για την οντολογία MBO, ότι πρόκειται για μία οντολογία η οποία ασχολείται με το πρόβλημα της ανίχνευσης μοντέρνων οικογενειών κακόβουλου λογισμικού, των οποίων οι μολύνσεις (*infections*), αποτελούνται από σύνολα πολλαπλών μεθόδων εκμετάλλευσης (*exploit methods*). Οι δημιουργοί της οντολογίας MBO, δημιούργησαν μια ιεραρχία κυρίως συμπεριφορών, όπου η καθεμία αποτελείται από ένα σύνολο ύποπτων συμπεριφορών.

Εν συνεχεία πρότειναν την οντολογία, η οποία μοντελοποιεί την γνώση (*knowledge*) αυτών των συμπεριφορών, ως προς την παρουσία 6 συμβάντων (*events*) τα οποία με τη σειρά τους αποτελούνται από διάφορα χαρακτηριστικά. Τα 6 αυτά συμβάντα είναι τα εξής παρακάτω:

1. Attack launching (Ξεκίνημα επίθεσης)
2. Evasion (Υπεκφυγή)
3. Remote control (Απομακρυσμένος έλεγχος)
4. Self-defense (Αυτο-άμυνα)
5. Stealing (Κλοπή)
6. Subversion (Ανατροπή)

Σύμφωνα με τα παραπάνω 6 συμβάντα, η οντολογία θα βρει την χρησιμότητά της στο να συμπεράνει (*infer*) για το εάν κάποια ύποπτη εκτέλεση του προγράμματος (*suspicious execution*) συνδέεται με κάποιο δείγμα κακόβουλου λογισμικού.

3.1.8.1 Αρχιτεκτονική / Δομή της οντολογίας MBO



Εικόνα 7. Γενική άποψη των κλάσεων της οντολογίας MBO (Grégio, Bonacin, & Nabuco, 2016)

Η οντολογία MBO, αποτελείται από κλάσεις καθώς και από συσχετισμούς (σχέσεις) μεταξύ των διαφόρων κλάσεων. Η κυρίως κλάση ονομάζεται `SuspiciousExecution` (Grégio, Bonacin, & Nabuco, 2016):

- Κάθε κλάση `SuspiciousExecution` συσχετίζεται με μία κλάση `SuspiciousSoftware`.
- Όλα τα αντικείμενα (instances) της `SuspiciousExecution`, εκτελούνται σε μία κλάση `System`.
- Η κλάση `SuspiciousExecution` εμπεριέχει ένα σύνολο από `ProcessAction` κλάσεις.
- Η κλάση `ProcessAction` συσχετίζεται με τις κλάσεις `SourceObject` και `TargetObject`.
- Αντικείμενα της `ProcessAction` μπορούν να συνδεθούν με αντικείμενα της κλάσης `SuspiciousBehavior`.

Η οντολογία MBO, έχει την δυνατότητα να ανιχνεύει τις μοντέρνες οικογένειες κακόβουλου λογισμικού, των οποίων οι μολύνσεις (*infections*) αποτελούν σύνολα πολλαπλών μεθόδων εκμετάλλευσης (*exploit methods*), με το να εφαρμόζει ειδικούς κανόνες πάνω στην οντολογία για την εξαγωγή συμπερασμάτων (*inferencing*). Επίσης, η οντολογία MBO, έχει την δυνατότητα να ανιχνεύει άγνωστο κακόβουλο λογισμικό, όταν τα συνηθισμένα προγράμματα antivirus δεν μπορούν να το ανιχνεύσουν.

Όμως, η οντολογία MBO έχει κάποιες αδυναμίες. Για παράδειγμα, δεν μπορεί επί του παρόντος να ανιχνεύσει κακόβουλο λογισμικό σε πραγματικό χρόνο (*real-time detection*), ή αλλιώς έχει κάποιους περιορισμούς ως προς τα θέματα απόδοσης (*performance issues*) (Mavroeidis & Bromander, 2017).

3.1.9 MIST

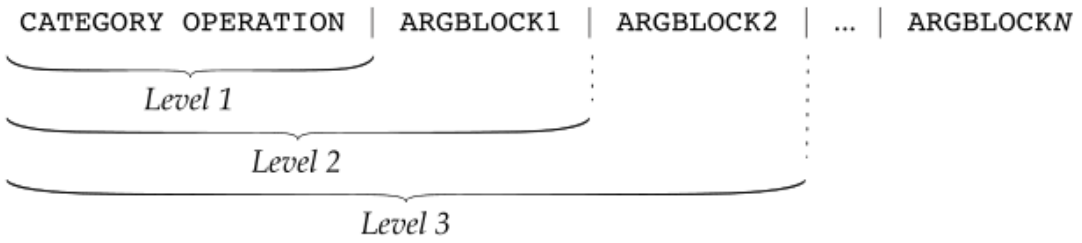
Η γλώσσα MIST (Malware Instruction-Set for Behavior-Based Analysis), είναι μία μετα-γλώσσα[9] (*meta-language*), η οποία δεν χρησιμοποιεί την γλώσσα περιγραφής δεδομένων XML για την περιγραφή του κακόβουλου λογισμικού, αλλά είναι βελτιστοποιημένη (*optimized*) για περαιτέρω ανάλυση, με την δυνατότητα σύγκρισης μεταξύ διαφορετικών συστημάτων συμπεριφορικής παρακολούθησης (*behavior monitoring systems*) (Trinius, Willems, Holz, & Rieck, 2009).

Κατά τους Yu *et al*. (Yu, Fang, Yang, Tang, & Liu, 2017), η γλώσσα MIST είναι μία αντιπροσώπευση κλήσεων του συστήματος (*system calls*) με εισερχόμενα (*input*) και εξερχόμενα (*output*) επιχειρήματα (*arguments*), και πρόκειται για μία βελτιστοποιημένη αντιπροσώπευση, για την αποτελεσματική ανάλυση της συμπεριφοράς του λογισμικού, χρησιμοποιώντας τεχνικές μηχανικής μαθήσεως (*machine learning*), καθώς και τεχνικές εξόρυξης δεδομένων (*data mining*).

Το δείγμα της γλώσσας MIST, μπορεί είτε να παραχθεί με αυτόματο τρόπο κατά την διάρκεια της δυναμικής ανάλυσης του κακόβουλου λογισμικού, μέσω της χρήσης κάποιου εργαλείου συμπεριφορικής ανάλυσης, ή να δημιουργηθεί με μη αυτόματο τρόπο, μέσω της μετατροπής συμπεριφορικών αναφορών (*behavior reports*) (Trinius, Willems, Holz, & Rieck, 2009).

3.1.9.1 Αρχιτεκτονική / Δομή της γλώσσας MIST

Η δομή της εντολής στην γλώσσα MIST διαφαίνεται παρακάτω.



Εικόνα 8. Γενικευμένη αντιπροσώπευση των εντολών της γλώσσας MIST (Trinius, Willems, Holz, & Rieck, 2009)

Οι εντολές στην γλώσσα MIST είναι πολυ-επίπεδες. Αποτελούνται από το 1ο επίπεδο, το οποίο αντιστοιχεί στο όνομα και στην κατηγορία της παρακολουθούμενης κλήσης του συστήματος (`CATEGORY & OPERATION`), και τα επόμενα επίπεδα -έως το

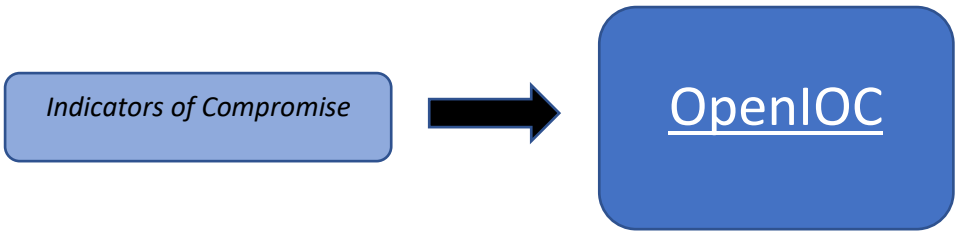
[9] Meta-language: Είναι μία γλώσσα, ή ένα σύνολο όρων, η οποία χρησιμοποιείται για την περιγραφή μίας άλλης γλώσσας.

επίπεδο N-, είναι το σύνολο των επιχειρημάτων (ARGBLOCK1 - ARGBLOCKN) , έχοντας τα κυρίως επιχειρήματα προς τα αριστερά, ενώ τα υπόλοιπα επιχειρήματα που προσδίδουν «θόρυβο» συμπληρώνονται προς τα δεξιά των κυρίων επιχειρημάτων, δηλαδή προς το τέλος της εντολής.

Η γλώσσα MIST αντιπροσωπεύει περί τις 120 κλήσεις συστήματος (*system calls*). Παράδειγμα τέτοιας κλήσης είναι: «μετακίνηση_αρχείου» (Trinius, Willems, Holz, & Rieck, 2009).

Τέλος, αυτό που επιτυγχάνεται με την χρήση της γλώσσας MIST είναι μία ελάττωση στο μέγεθος των δεδομένων (*data reduction*) προς περαιτέρω ανάλυση, και αυτό, προς αντιδιαστολή με το μέγεθος των δεδομένων XML που προσλαμβάνονται από άλλες γλώσσες ανάλυσης του κακόβουλου λογισμικού.

3.1.10 OpenIOC



Εικόνα 9. Η γλώσσα OpenIOC

Η γλώσσα OpenIOC (Open Indicators of Compromise), δημιουργήθηκε αρχικά από την εταιρεία Mandiant[10] (Mandiant, 2017), με σκοπό να μειωθεί ο χρόνος ανίχνευσης (*detection*) καθώς και ο χρόνος απόκρισης (*response time*) σε συμβάντα κυβερνο-ασφάλειας (*cyber-security incidents*), στο μέγιστο, δηλαδή σε χρόνους υπολογισμού των υπολογιστών (*machine-speed calculations*).

Επίσης, ένας ακόμη σκοπός, είναι ο διαμοιρασμός (*sharing*) πληροφορίας κυβερνο-επίθεσης (*cyber-attack*) στα ενδιαφερόμενα μέρη, ούτως ώστε να μπορούν να προσθέτουν πληροφορίες άμεσα (*on the fly*) και αποτελεσματικά.

Η γλώσσα OpenIOC είναι ένα ανοικτό (*open*), τυποποιημένο (*standardized*), επεκτάσιμο (*extensible*), κι ευέλικτο (*flexible*) σχήμα XML (*XML schema*), που εμπεριέχει ως πληροφορία, λογικά ομαδοποιημένους Δείκτες Επικινδυνότητας (*Indicators of Compromise*), με σκοπό τον διαμοιρασμό (*sharing*) και την επικοινωνία τους, με απώτερο σκοπό να βελτιωθεί η άμυνα απέναντι σε κυβερνο-επιθέσεις (Mandiant, 2016).

Κατά τους Mavroeidis *et al.* (Mavroeidis & Bromander, 2017), αυτό το επεκτάσιμο (*extensible*) σχήμα XML, μας επιτρέπει να περιγράφουμε τα χαρακτηριστικά εκείνα τα οποία ταυτοποιούν μια γνωστή απειλή, ή την μεθοδολογία ενός επιτιθέμενου.

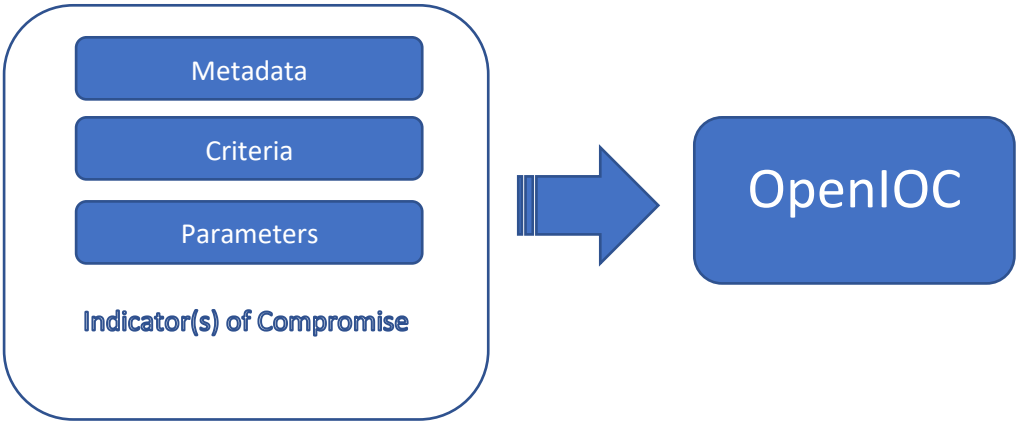
[10] Η εταιρεία Mandiant έχει εξαγοραστεί πλέον από την εταιρεία FireEye.

3.1.10.1 Δομή της γλώσσας OpenIOC

Ως υποκείμενη γλώσσα (*underlying language*), για την περιγραφή των παραπάνω, χρησιμοποιείται η γλώσσα XML (*eXtensible Markup Language*). Έτσι, είναι εύκολο να δημιουργηθούν προγραμματιστικά εργαλεία (*utilities*), με τα οποία θα μπορεί η OpenIOC, να μετατραπεί σε άλλα formats, ή να αναλυθεί (*parsed*) περαιτέρω (Mandiant, 2016).

Κάθε δείκτης επικινδυνότητας (*IoC*), αποτελείται από 3 διακριτές περιοχές (*sections*) (Mandiant, 2013):

- Μετα-δεδομένα (*Metadata*)
- Κριτήρια (*Criteria*)
- Παράμετροι (*Parameters*)



Εικόνα 10. Δομή της γλώσσας OpenIOC

Αναλυτικότερα, έχουμε τα εξής:

1. Μετα-δεδομένα (Metadata): Εμπεριέχει μετα-δεδομένα[11] σχετικά με τον δείκτη επικινδυνότητας (*Indicator of Compromise - IoC*).
2. Κριτήρια (Criteria): Πρόκειται για την περιοχή εκείνη στο XML schema, όπου γίνεται η αντιστοίχιση μέσω λογικής boolean έκφρασης, δηλαδή εκεί συμβαίνει η λογική αξιολόγηση (*evaluation*), και όπου γίνεται η διαπίστωση εάν έχει βρεθεί κάτι κατακριτέο (*evil*). Η OpenIOC επίσης υποστηρίζει και την χρήση κανονικών εκφράσεων – *regular expressions*.

[11] Μετα-δεδομένα: Δεδομένα αναφορικά με άλλα δεδομένα.

3. Παράμετροι (Parameters): Είναι μετα-δεδομένα, εφαρμοστέα στα κριτήρια στην παραπάνω περιοχή (*section*). Η χρήση των παραμέτρων υποβοηθάει στο να εξειδικεύονται τα αντικείμενα των κριτηρίων.

3.1.10.2 Χαρακτηριστικά της γλώσσας OpenIOC

Η OpenIOC εμπεριέχει περί τους 601 όρους Indicators of Compromise, (*OpenIOC terms*), όπως μπορούμε να διαπιστώσουμε από το XML σχήμα (*schema*) της, στην έκδοσή της 1.1, έτσι καλύπτοντας ένα μεγάλο εύρος κυβερνο-απειλών. Επιπλέον, έχοντας επεκτάσιμο και προσαρμόσιμο χαρακτήρα, μπορεί ο οποιοσδήποτε να προσθέσει τους δικούς του όρους IoC, στην γλώσσα.

Πλέον, η εταιρεία Cisco, χρησιμοποιεί το πρότυπο OpenIOC, ως μέρος της πλατφόρμας της, "*Advanced Malware Protection - AMP*", επιτρέποντας στους χρήστες αυτής της πλατφόρμας να υποβάλλουν (*submit*), τους δικούς τους Δείκτες Επικινδυνότητας (*IoC*), για να μπορέσουν να συλλάβουν στοχευμένες επιθέσεις (*targeted attacks*), καθιστώντας παράλληλα την πλατφόρμα AMP ιδιαιτέρως αποτελεσματική στο σταμάτημα εισερχόμενου κακόβουλου λογισμικού (*incoming malware block*), σε ένα ποσοστό που φτάνει το 99% (Dominguez, 2015), (Cisco, 2016).

3.1.10.3 Εργαλεία για την γλώσσα OpenIOC

Τα 2 κυρίως εργαλεία για την OpenIOC, είναι τα εξής:

1. IOC Editor: Με αυτό το εργαλείο μπορούμε να δημιουργήσουμε / εισάγουμε με δομημένο τρόπο τους δείκτες επικινδυνότητας (IOC), μέσα από το γραφικό περιβάλλον (*Graphical User Interface - GUI*) του προγράμματος.
2. IOC Finder: Εφ'όσον δημιουργήσουμε τους δείκτες επικινδυνότητας, τότε με αυτό το εργαλείο μπορούμε να συγκρίνουμε με αυτούς τους δείκτες επικινδυνότητας που έχουμε συλλέξει από κάποιο σύστημα (*host*).

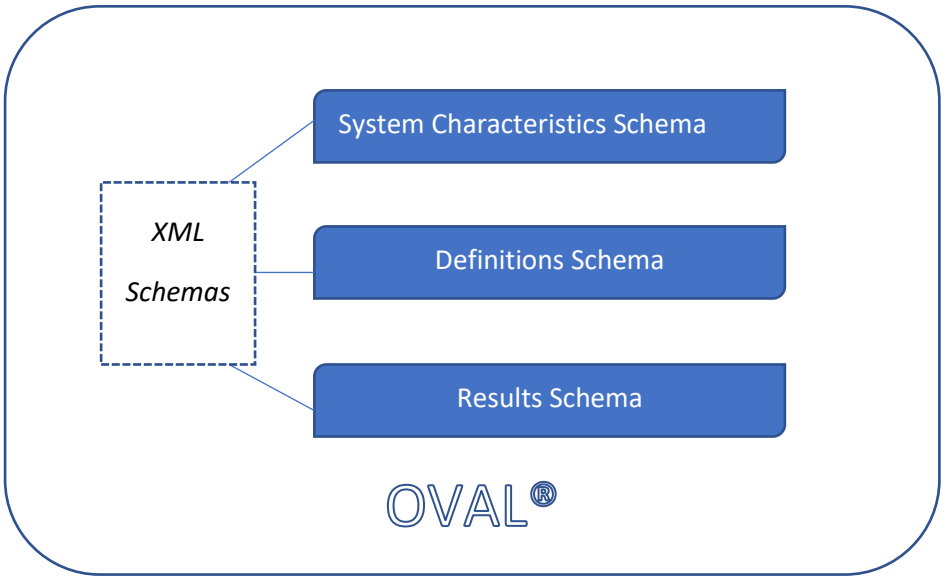
3.1.11 OVAL®

Η γλώσσα OVAL® (Open Vulnerability and Assessment Language), δημιουργήθηκε από την εταιρεία Mitre Corporation. Μεταξύ άλλων περιπτώσεων χρήσης (*use cases*), χρησιμοποιείται και για την ανίχνευση κακόβουλου λογισμικού. Αποτελεί ένα ελεύθερο και ανοιχτό πρότυπο (*open standard*) (Open Vulnerability and Assessment Language, 2016).

3.1.11.1 Αρχιτεκτονική / Δομή της γλώσσας OVAL®

Η υποκείμενη γλώσσα της OVAL® είναι η γλώσσα XML. Έτσι, η γλώσσα OVAL αποτελείται από 3 σχήματα XML (*XML Schemas*), τα οποία και αποτελούν τον σκελετό της (*framework*) (Mitre Corporation, 2017):

1. Σχήμα Χαρακτηριστικών Συστήματος (*System Characteristics Schema*): Είναι το σχήμα XML εκείνο το οποίο αντιπροσωπεύει την κατάσταση του συστήματος.
2. Σχήμα Ορισμών (*Definitions Schema*): Είναι το σχήμα XML εκείνο το οποίο εμπεριέχει τις εκφράσεις της κατάστασης του συστήματος (*machine state*).
3. Σχήμα Αποτελεσμάτων (*Results Schema*): Είναι το σχήμα XML με το οποίο γίνεται η αναφορά (*reporting*) των αποτελεσμάτων.



Εικόνα 11. Δομή της γλώσσας OVAL

Οι ορισμοί (*definitions*) της OVAL, είναι τυποποιημένοι έλεγχοι (*standardized tests*), οι οποίοι μπορούν να διαβαστούν από τον υπολογιστή (*machine readable*), κι έτσι να ελέγξουν το υπολογιστικό σύστημα για την παρουσία / ύπαρξη ευπαθειών λογισμικού (*software vulnerabilities*), θεμάτων διαμόρφωσης (*configuration issues*), και προγραμμάτων. Ως τέτοιοι, οι ορισμοί (*definitions*), αποτελούνται από 3 ενότητες: τα μετα-δεδομένα (*meta-data*), την υψηλού επιπέδου περίληψη (*high-level summary*), και τον αναλυτικό ορισμό (Mitre Corporation, 2017).

3.1.11.2 Εργαλεία για την γλώσσα OVAL®

Το εργαλείο που παρέχεται για την γλώσσα OVAL®, είναι ο διερμηνέας OVAL (*OVAL interpreter*), με τις κάτωθι δυνατότητες (OVAL Interpreter, 2018):

- Έχει δημιουργηθεί για την επίδειξη (*demonstration*) της χρησιμότητας των ορισμών της γλώσσας OVAL®, και να εξασφαλίσει την σωστή σύνταξη τους (*syntax*).
- Πρόκειται για απλοϊκή εφαρμογή, διεπαφής γραμμής εντολών (*command-line interface, CLI*), της οποίας η εκτέλεση λαμβάνει χώρα στο σύστημα του χρήστη.

3.1.12 RBS

Η γλώσσα RBS (Raw Behavior Specification), είναι μία γλώσσα προσδιορισμών (*specification language*), η οποία αποθηκεύει την ολοκληρωμένη διαφαινόμενη συμπεριφορά κάποιου δείγματος κακόβουλου λογισμικού, για την μεταγενέστερη ανάλυσή της. Έχει σχεδιαστεί έχοντας 2 στόχους: Να είναι πλήρης (*complete*), και αποτελεσματική (*efficient*) ως προς την ανίχνευση του κακόβουλου λογισμικού (Deschamps, 2008).

Ως γλώσσα, αντιπροσωπεύει όλη εκείνη την πληροφορία η οποία γίνεται διαθέσιμη μετά από την ανάλυση του κακόβουλου λογισμικού (Deschamps, 2008):

1. Κλήσεις συστήματος (*system calls*),
2. Κλήσεις βιβλιοθηκών (*library calls*),
3. Κίνηση δικτύου (*network traffic*),
4. Ροή πληροφοριών (*information flow*).

Επίσης, έχει δημιουργηθεί με σκοπό να είναι μια γλώσσα με ελάχιστο ίχνος μνήμης (*minimum memory footprint*), καθώς και ακριβής (*concise*) στην περιγραφή της.

3.1.12.1 Δομή της γλώσσας RBS

Η γλώσσα Raw Behavior Specification, αποτελείται κυρίως από 2 αρχεία:

1. Καταγραφή κίνησης δικτύου (*network traffic dump*): Πρόκειται για αρχείο σε μορφή PCAP[12], το οποίο αποθηκεύει την δικτυακή κίνηση του δυαδικού (*binary*) αρχείου. Αυτό το αρχείο, εν συνεχεία μπορεί να αναγνωστεί από κάποιο λογισμικό ανάγνωσης αρχείων PCAP, όπως είναι το δημοφιλές λογισμικό Wireshark.
2. Ίχνος εκτέλεσης (*execution trace*): Το χρονολογημένο (*time-stamped*) αρχείο αυτό, καταγράφει τις κλήσεις συστήματος και βιβλιοθηκών, και είναι σε ASCII μορφή κειμένου.

[12] PCAP: "Packet Capture", μορφή (format) η οποία καταγράφει την κίνηση του δικτύου.

3.1.13 VERIS

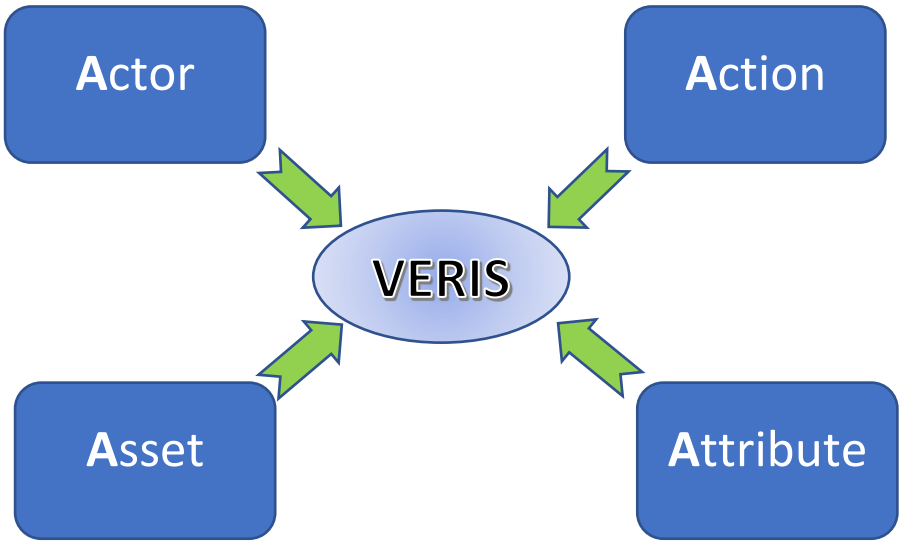
Η VERIS (Vocabulary for Event Recording and Incident Sharing), αποτελεί ένα πρότυπο/σκελετό (framework), λεξιλόγιο (vocabulary), το οποίο έχει δημιουργηθεί από την εταιρεία Verizon Corporation.

Αποτελείται από ένα σύνολο μετρικών (*metrics*) / στοιχείων (*elements*), με τα οποία μπορούμε να περιγράψουμε συμβάντα ασφαλείας (*security incidents*) σε μια δομημένη μορφή (*structured format*). Το πρότυπο VERIS επιτρέπει τον ανώνυμο διαμοιρασμό (*anonymous sharing*) αυτής της πληροφορίας των συμβάντων ασφάλειας, και είναι ανοικτού κώδικα (*open source*) (VERIS Framework, 2018).

Κατά τους Burger *et al.* (Burger, Goodman, Kampanakis, & Zhu, 2014), η γλώσσα VERIS, είναι ένας χαρακτηρισμός των συμβάντων κυβερνο-ασφάλειας (*cyber incidents*), αφού αυτά έχουν συμβεί, με σκοπό την στρατηγική ανάλυση των τάσεων που αυτά αντιπροσωπεύουν (*strategic trend analysis*).

Τα συμβάντα (*events*) στην γλώσσα VERIS, εμπεριέχουν εκείνη την πληροφορία σχετικά με τους διενεργητές (*actors*) και τις πράξεις τους (*actions*), τα αγαθά (*assets*) που επηρεάζονται, και τα χαρακτηριστικά τους (*attributes*) (Kampanakis P. , 2014).

3.1.13.1 Δομή της γλώσσας VERIS



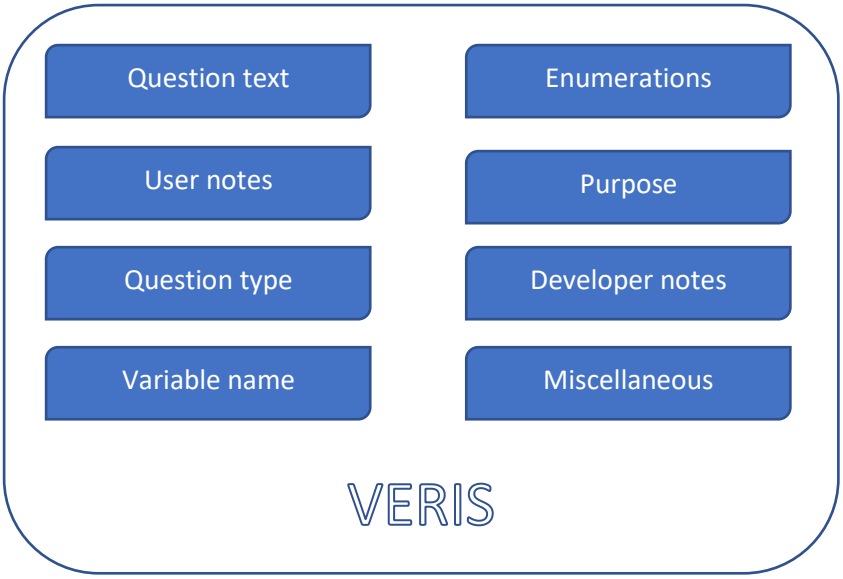
Εικόνα 12. Οι βάσεις της γλώσσας VERIS - 4A

Η γλώσσα με την οποία εκφράζεται η γλώσσα VERIS (v.1) είναι η γλώσσα JSON (Kampanakis P. , 2014). Επί του παρόντος στην έκδοση (version) του προτύπου 1.3.2, η βάση της γλώσσας VERIS είναι τα «4**A**», ονομαστικά "**A**ctor, **A**ction, **A**sset, **A**ttribute":

- ***A**ctor*: *Ποιός* κρύβεται πίσω από την επίθεση,
- ***A**ction*: *Ποιά* μέθοδο χρησιμοποίησε,
- ***A**sset*: *Τί* είδους μηχανήματα / συσκευές (*devices*) επηρέασε,
- ***A**ttribute*: *Πώς* αυτά -τα μηχανήματα- επηρεάστηκαν.

Η μορφοποίηση του κάθε στοιχείου (*element*) της γλώσσας, διαμορφώνεται ως εξής (VERIS Framework, 2018):

- Question text (Κείμενο ερώτησης): Ο προτεινόμενος τρόπος διατύπωσης για τις ερωτήσεις, σε μία εφαρμογή βασισμένη πάνω στην VERIS.
- User notes (Σημειώσεις χρήστη): Οποιεσδήποτε βοηθητικές εφαρμογές ή συμβουλές μιας εφαρμογής βασισμένης πάνω στην VERIS.
- Question type (Τύπος ερωτήσεως): Ο τύπος της ερώτησης, δηλαδή είτε πεδίο κειμένου, ή απαριθμημένη (*enumerated*) λίστα.
- Variable name (Όνομα μεταβλητής): Τα ονόματα των μεταβλητών που λαμβάνουν μέρος στο σχήμα (*schema*).
- Enumerations (Απαριθμήσεις): Οι απαριθμήσεις οι οποίες απαντώνται στα έγγραφα JSON ή XML της VERIS, και οι οποίες συσχετίζονται με τις μεταβλητές.
- Purpose (Σκοπός): Η εξήγηση του λόγου ύπαρξης του στοιχείου (*element*) στην γλώσσα VERIS.
- Developer notes (Σημειώσεις συντάκτη/προγραμματιστή): Βοηθητικές πληροφορίες ή συμβουλές προς τους συντάκτες (*developers*) των εφαρμογών βασισμένων πάνω στην VERIS.
- Miscellaneous (Διάφορα): Οποιαδήποτε άλλη χρήσιμη πληροφορία για το στοιχείο (*element*).



Εικόνα 13. Μορφοποίηση στοιχείου της γλώσσας VERIS

Πρέπει να σημειωθεί, ότι ο αρχικός οραματισμός για την γλώσσα VERIS, ήταν μόνο για αναφορές και ανάλυση (*reporting & analysis*), και όχι για τον διαμοιρασμό πληροφορίας (*information sharing*) (Kampanakis P. , 2014).

3.1.14 YARA

Η YARA (Yet Another Recursive Acronym) είναι εργαλείο, γραμμένο σε γλώσσα προγραμματισμού Python, το οποίο βασίζεται σε υπογραφές (*signatures*) γραμμένες σε γλώσσα περιγραφής δεδομένων JSON, και σκοπός του οποίου είναι να βοηθήσει τους αναλυτές κακόβουλου λογισμικού, να ταυτοποιήσουν και να ταξινομήσουν το κακόβουλο λογισμικό (YARA, 2018).

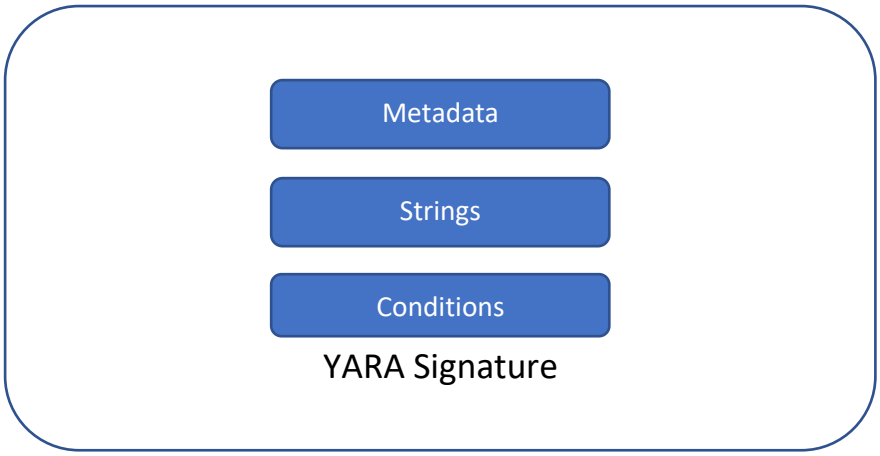
Είναι μια απλή γλώσσα περιγραφής κακόβουλου λογισμικού, η οποία δύναται να περιγράψει το κακόβουλο λογισμικό (*malware*), ή και *packet captures*, αλλά είναι περιορισμένη ως προς το εύρος του τί μπορεί να περιγράψει (Kampanakis P. , 2014).

3.1.14.1 Δομή της γλώσσας YARA

Επί του παρόντος (2018), η γλώσσα YARA βρίσκεται στην έκδοσή της 3.7.1. Βασίζεται στην δημιουργία περιγραφών - κανόνων (*rules*), οι οποίοι με την σειρά τους βασίζονται σε κειμενικά (*textual*) ή δυαδικά (*binary*) μοτίβα (*patterns*).

Κάθε κανόνας αποτελείται από ένα σύνολο συμβολοσειρών (*strings*), κι από μία λογική έκφραση (*Boolean expression* / *condition*), η οποία αποτελεί και την λογική του κανόνα. Έτσι, η υπογραφή YARA αποτελείται από 3 βασικές ενότητες (Marak, 2015):

1. Μετα-δεδομένα (Meta): Αυτή η ενότητα εμπεριέχει τα μετα-δεδομένα ή το κείμενο εκείνο (ως ζεύγη ονομάτων-τιμών), που προσδίδουν επιπρόσθετες πληροφορίες στον χρήστη.
2. Συμβολοσειρές (Strings) (προαιρετική ενότητα): Η ενότητα αυτή περιέχει τα κειμενικά (*textual*) καθώς και τα δεκαεξαδικά (*hexadecimal*) στοιχεία, ως μία βάση δεδομένων για τους κανόνες. Εκτός από τους δύο αυτούς προαναφερόμενους τύπους συμβολοσειράς, υπάρχει και η δυνατότητα για χρήση κανονικών εκφράσεων (*regular expressions*) (Van Impe, 2015).
3. Συνθήκες (Conditions): Η ενότητα αυτή περιέχει τις λογικές (*Boolean*) συνθήκες οι οποίες λαμβάνουν ως όρισμα (ή ορίσματα) τα δεδομένα που παρέχονται στην ενότητα Strings, και δίνουν το τελικό αποτέλεσμα στον κανόνα YARA.



Εικόνα 14. Δομή της υπογραφής YARA

Ένα πολύ απλό παράδειγμα χρήσης κανόνα -σε JSON μορφή-, είναι και το εξής (YARA Documentation, 2018):

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        thread_level = 3
        in_the_wild = true
    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
    condition:
        $a or $b or $c
}
```

Εικόνα 15. Κανόνας (rule) της γλώσσας YARA (YARA Documentation, 2018)

, δηλαδή, εάν επιβεβαιώνεται η ύπαρξη είτε της συμβολοσειράς `$a`, ή της `$b`, ή της `$c`, τότε ο κανόνας *επιτυγχάνει*, και το αρχείο αναφέρεται ως "`silent_banker`".

Η ανίχνευση κακόβουλου λογισμικού με την γλώσσα YARA καθίσταται εύκολη και απλή διαδικασία, παρ'όλο που έχει τους περιορισμούς της, όντας μέθοδος ανίχνευσης βασισμένη σε υπογραφές (*signatures*) εκτελέσιμων αρχείων – αυτό αποτελεί και βασικό μειονέκτημα της χρήσης της. Στα πλεονεκτήματά της, συμπεριλαμβάνεται ότι είναι γραμμένη στην δημοφιλή γλώσσα Python, και ότι μπορεί να τρέχει σε διάφορες πλατφόρμες (*multi-platform*) (Van Impe, 2015).

4 Κριτήρια Σύγκρισης

Όπως έχουμε διαπιστώσει στο προηγούμενο κεφάλαιο, όπου παραθέσαμε τα ιδιαίτερα χαρακτηριστικά κάθε γλώσσας περιγραφής-χαρακτηρισμού του κακόβουλου λογισμικού, πολλές από αυτές τις γλώσσες -αλλά όχι όλες-, είναι XML-based ή JSON-based, δηλαδή βασισμένες είτε στην γλώσσα XML ή στην γλώσσα JSON.

4.1 Γενικά Χαρακτηριστικά

Κάποια από τα κοινά χαρακτηριστικά (attributes) τα οποία συναντώνται στις περισσότερες από τις προ-αναφερθείσες γλώσσες/πρότυπα περιγραφής, είναι τα παρακάτω:

- Τα διάφορα χαρακτηριστικά, όπως για παράδειγμα οι δείκτες επικινδυνότητας (*Indicators of Compromise*, "*IoC*"), συνήθως περιγράφονται σε κάποια γλώσσα περιγραφής δεδομένων, όπως για παράδειγμα, XML ή JSON.
- Πρόκειται για γλώσσες συγκεκριμένου τομέα (*Domain-Specific Languages*), ή πιο συγκεκριμένα "Domain-Specific *Markup* Languages", δηλαδή γλώσσες οι οποίες περιγράφουν τις γλώσσες χαρακτηρισμού του κακόβουλου λογισμικού ως δομημένα δεδομένα (Domain-specific language, 2018).
- Συνήθως είναι frameworks, δηλαδή πλαίσια / σκελετοί, που εμπεριέχουν μορφοποιημένα-κατηγοριοποιημένα δεδομένα.

4.2 Κριτήρια σύγκρισης

Πρωτευόντως, κάποια από τα κριτήρια σύγκρισης των γλωσσών περιγραφής του κακόβουλου λογισμικού, έχουν ως εξής:

1. Είναι ικανή η γλώσσα περιγραφής να αναχαιτίσει κάποια προχωρημένης μορφής επίθεση (Προηγμένη Επίμονη Απειλή - *Advanced Persistent Threat*, "*APT*") στο εν λόγω σύστημα;
2. Μπορεί να περιγράψει με την απαιτούμενη ακρίβεια το κακόβουλο λογισμικό, ούτως ώστε να μπορεί να το αναχαιτίσει;
3. Έχει εκείνα τα χαρακτηριστικά, τα οποία απαιτούνται για την ικανή διάκριση του καλοήθους (*benign*) από το κακόβουλο (*malicious*) λογισμικό;

4. Είναι λιτή και κατανοητή προς τον άνθρωπο – αναλυτή κακόβουλου λογισμικού, η γλώσσα περιγραφής αυτή;
5. Από ποιούς έχει μελετηθεί και έχει περαιτέρω αναπτυχθεί; Υπάρχει κάποια εταιρεία πίσω από την οργανωμένη ανάπτυξη της γλώσσας περιγραφής, ή πρόκειται για μεμονωμένη προσπάθεια κάποιων αναλυτών κυβερνο-ασφάλειας;
6. Έχει τα αναμενόμενα αποτελέσματα ως γλώσσας περιγραφής, ως προς την ανίχνευση και ως προς την αναχαίτιση του κακόβουλου λογισμικού;
7. Πρόκειται για μία καθαρά ακαδημαϊκή γλώσσα, πιθανόν χωρίς μεγάλο πρακτικό χαρακτήρα, ή έχει πιο εφαρμοσμένο στην πράξη χαρακτήρα;
8. Είναι τυποποιημένο (*standardized*) πρότυπο;
9. Έχει επεκτάσιμο (*extendable*) και (*customizable*) χαρακτήρα;
10. Έχει την δυνατότητα για ανίχνευση του κακόβουλου λογισμικού σε πραγματικό χρόνο (*real-time detection*);
11. Προσφέρεται για τον διαμοιρασμό συμβάντων (*event sharing*);

Δευτερευόντως, και ως προς την βάση χρηστών της γλώσσας, θα μπορούσαμε να διακρίνουμε τα εξής παρακάτω κριτήρια:

1. Είναι δημοφιλής η γλώσσα, μέσα στα πλαίσια της κοινότητας των χρηστών της;
2. Πόσο μεγάλη είναι η βάση χρηστών που την χρησιμοποιεί;
3. Είναι υποστηριζόμενη από την κοινότητα (*community driven*);
4. Τί είδους άδεια υπάρχει για την χρήση της; Πρόκειται για ελεύθερη άδεια χρήσης, ή δεν είναι ελεύθερη η άδεια χρήσης της, υπάρχει δηλαδή copyright που καθιστά την άδεια πιο περιορισμένη;

Αναλυτικότερα:

Η γλώσσες MAEC™, CAPEC™, CybOX™, και OVAL®, έχουν δημιουργηθεί από την εταιρεία Mitre Corporation, με τις γλώσσες OpenIOC να έχει δημιουργηθεί από την εταιρεία Mandiant, και την γλώσσα VERIS από την εταιρεία Verizon.

Οι υπόλοιπες γλώσσες έχουν δημιουργηθεί στα πλαίσια ακαδημαϊκού περιβάλλοντος, όπως η ενδιάμεση γλώσσα MAIL, η οντολογία MBO, η μετα-γλώσσα

MIST, και οι γλώσσες προσδιορισμού AMBL, BSL, και RBS, ενώ η γλώσσα IODEF είναι γλώσσα που δημιουργήθηκε από μέλη της ομάδας IETF.

Από τις βασισμένες σε XML γλώσσες (XML-based languages), η MAEC™ έχει αποκλειστικό σκοπό την περιγραφή του κακόβουλου λογισμικού. Οι γλώσσες CAPEC™, CybOX™, OVAL®, και OpenIOC, συμπεριλαμβάνουν και άλλες περιπτώσεις χρήσης, γενικότερα για τις κυβερνο-επιθέσεις, εκτός του χαρακτηρισμού του κακόβουλου λογισμικού.

Λόγω της ανομοιογένειας των προαναφερόμενων μελετηθεισών γλωσσών / προτύπων, θα συγκρίνουμε τις γλώσσες αυτές σύμφωνα με τα επιμέρους χαρακτηριστικά τους.

(Σ.σ.: Στις παρακάτω ενότητες, στους πίνακες το σύμβολο [●], σημαίνει ότι η γλώσσα περιγραφής *έχει* το συγκεκριμένο χαρακτηριστικό, ενώ η απουσία του συμβόλου υποδηλώνει είτε ότι δεν υπάρχει το χαρακτηριστικό αυτό, ή ότι υπάρχουν ελλιπή δεδομένα).

4.2.1 Κριτήριο υποκείμενης γλώσσας

Όσον αφορά το κριτήριο ως προς την γλώσσα που χρησιμοποιείται:

Πίνακας 2. Κριτήριο υποκείμενης γλώσσας

Όνομα γλώσσας	***Βασισμένη στην γλώσσα XML/JSON;***	***Οντολογία;***	***Γλώσσα προσδιο-ρισμών;***
AMBL			●
BSL			●
CAPEC™			
CybOX™	●		
IODEF	●		
MAEC™	●		
MAIL			
MBO		●	
MIST			
OpenIOC	●		
OVAL®	●		
RBS			●
VERIS			
YARA	●		

Οι γλώσσες CybOX™, IODEF, MAEC™, OpenIOC, OVAL®, και YARA, είναι βασισμένες στην γλώσσα XML, είτε στην γλώσσα JSON. Οι γλώσσες που χρησιμοποιούν την XML, τείνουν να είναι περισσότερο βερμπαλιστικές (*verbose*).

Από την άλλη, ως οντολογία ορίζουμε, «*εκείνο το κοινό λεξιλόγιο (vocabulary) το οποίο χρησιμοποιείται από τους ερευνητές, προς διαμοιρασμό της πληροφορίας σε κάποιον συγκεκριμένο τομέα (domain)*» (Noy & McGuinness, 2001). Από τον παραπάνω πίνακα διαπιστώνουμε ότι υπάρχει τουλάχιστον μία οντολογία η οποία διαπραγματεύεται την ανίχνευση του κακόβουλου λογισμικού, η οντολογία MBO.

Οι δε γλώσσες AMBL, BSL, και RBS, είναι γλώσσες προσδιορισμών. Η γλώσσα MAIL είναι ενδιάμεση γλώσσα μηχανής (*assembly*), η γλώσσα MIST είναι

μετα-γλώσσα, και οι γλώσσες CAPEC™ και VERIS βασίζονται σε κειμενικά δεδομένα (*textual data*).

4.2.2 Κριτήριο επεκτασιμότητας / προσαρμοστικότητας

Το κριτήριο αυτό ασχολείται με το εάν έχει η γλώσσα την δυνατότητα να είναι επεκτάσιμη, ή και, προσαρμόσιμη:

Πίνακας 3. Κριτήριο επεκτασιμότητας / προσαρμοστικότητας

Όνομα γλώσσας	***Είναι επεκτάσιμη;***	***Είναι προσαρμόσιμη;***
AMBL		
BSL		
CAPEC™		
CybOX™	●	●
IODEF		
MAEC™	●	
MAIL		
MBO		
MIST		
OpenIOC	●	●
OVAL®		
RBS		
VERIS		
YARA		

Οι γλώσσες CybOX™, MAEC™, OpenIOC, είναι επεκτάσιμες (*extensible*), δηλαδή μπορούν να επεκταθούν περαιτέρω με νέες δυνατότητες και λειτουργίες – όπως είδαμε στην προηγούμενη ενότητα είναι βασισμένες στις γλώσσες XML/JSON. Από αυτές τις τρεις γλώσσες, η CybOX™, και η OpenIOC, είναι και προσαρμόσιμες (*adaptable*), δηλαδή η γλώσσα μπορεί να προσαρμοστεί σε νέα δεδομένα.

4.2.3 Κριτήριο απλότητας στην περιγραφή

Οι γλώσσες περιγραφής, για να είναι αναγνώσιμες (*human readable*), θα πρέπει να χρησιμοποιούν κάποια υποκείμενη γλώσσα όπως είτε οι XML/JSON (οι οποίες είναι εκφραστικές γλώσσες), ή να εκφράζονται ως κείμενο (*text*). Έτσι, ως προς απλότητα στην περιγραφή, οι γλώσσες χαρακτηρισμού μπορούν να ταξινομηθούν ως εξής:

Πίνακας 4. Κριτήριο απλότητας στην περιγραφή

Όνομα γλώσσας	***Είναι λιτή/αναγνώσιμη γλώσσα;***
AMBL	
BSL	
CAPEC™	●
CybOX™	●
IODEF	●
MAEC™	●
MAIL	
MBO	
MIST	
OpenIOC	●
OVAL®	●
RBS	
VERIS	●
YARA	●

Οι γλώσσες CAPEC™, CybOX™, IODEF, MAEC™, OpenIOC, OVAL®, VERIS, και YARA, επιτυγχάνουν σε αυτό το κριτήριο, με τις υπόλοιπες γλώσσες να μην επιτυγχάνουν διότι είτε είναι οντολογίες, όπως η MBO, ή πρόκειται για δυαδικές γλώσσες (*binary languages*), όπως για παράδειγμα η γλώσσα MAIL.

4.2.4 Κριτήριο τυποποιημένης γλώσσας

Ένα άλλο κριτήριο χαρακτηρισμού των γλωσσών περιγραφής, είναι και το εάν αυτές είναι τυποποιημένες, δηλαδή εάν είναι συμμορφωμένες σε κάποιο πρότυπο (*standard*):

Πίνακας 5. Κριτήριο τυποποιημένης γλώσσας

Όνομα γλώσσας	***Είναι τυποποιημένη;***
AMBL	
BSL	
CAPEC™	●
CybOX™	●
IODEF	●
MAEC™	●
MAIL	
MBO	
MIST	
OpenIOC	●
OVAL®	●
RBS	
VERIS	●
YARA	●

Οι γλώσσες CAPEC™, CybOX™, IODEF, MAEC™, OpenIOC, OVAL®, VERIS, και YARA, είναι τυποποιημένες (*standardized*), οπότε κι επιτυγχάνουν σε αυτό το κριτήριο.

4.2.5 Κριτήριο υποστήριξης διαμοιρασμού πληροφορίας

Το κριτήριο αυτό, ασχολείται με το εάν υπάρχει η δυνατότητα διαμοιρασμού πληροφορίας (*information sharing*) μεταξύ των ενδιαφερόμενων οντοτήτων (*entities*), όπως για παράδειγμα οι επαγγελματίες της κυβερνο-ασφάλειας, ούτως ώστε να είναι δυνατή η αποτελεσματικότερη επικοινωνία μεταξύ τους.

Πίνακας 6. Κριτήριο υποστήριξης διαμοιρασμού πληροφορίας

Όνομα γλώσσας	***Είναι διαμοιραζόμενη;***
AMBL	
BSL	
CAPEC™	
CybOX™	●
IODEF	●
MAEC™	
MAIL	
MBO	
MIST	
OpenIOC	
OVAL®	
RBS	
VERIS	●
YARA	

Στις περιπτώσεις των γλωσσών CybOX™, IODEF, και VERIS, υπάρχει αυτή η δυνατότητα διαμοιρασμού της πληροφορίας. Για τις υπόλοιπες γλώσσες, δεν υπάρχει αναφορά στη βιβλιογραφία για το συγκεκριμένο κριτήριο.

4.2.6 Κριτήριο αντιμετώπισης Προηγμένης Επίμονης Απειλής

Αυτό το κριτήριο, αναφέρεται ως προς την αποτελεσματικότητα της αντιμετώπισης (*mitigation*) κάποιας Προηγμένης Επίμονης Απειλής (*Advanced Persistent Threat*):

Πίνακας 7. Κριτήριο αντιμετώπισης Προηγμένης Επίμονης Απειλής

Όνομα γλώσσας	***Προηγμένη Επίμονη Απειλή;***
AMBL	
BSL	
CAPEC™	
CybOX™	
IODEF	
MAEC™	
MAIL	
MBO	
MIST	
OpenIOC	
OVAL®	
RBS	
VERIS	
YARA	

Στην βιβλιογραφία υπάρχουν ελλιπή στοιχεία / δεδομένα για το εάν κάποια από τις παραπάνω γλώσσες είναι ικανή να αντιμετωπίσει μια Προηγμένη Επίμονη Απειλή.

4.2.7 Κριτήριο ακρίβειας περιγραφής κακόβουλου λογισμικού

Ο αντικειμενικός σκοπός κάθε γλώσσας περιγραφής, είναι να περιγράψει με τον δυνατόν καλύτερο τρόπο τα επιμέρους χαρακτηριστικά του κακόβουλου λογισμικού. Το κριτήριο αυτό, ασχολείται με το εάν έχει την δυνατότητα η γλώσσα περιγραφής να περιγράψει με μεγάλη ακρίβεια το κακόβουλο λογισμικό, ούτως ώστε να έχει υψηλά ποσοστά αναχαίτισής του.

Πίνακας 8. Κριτήριο ακρίβειας περιγραφής κακόβουλου λογισμικού

Όνομα γλώσσας	***Ακρίβεια περιγραφής***
AMBL	●
BSL	●
CAPEC™	
CybOX™	●
IODEF	
MAEC™	●
MAIL	●
MBO	●
MIST	●
OpenIOC	●
OVAL®	●
RBS	●
VERIS	
YARA	●

Οι γλώσσες χαρακτηρισμού με την μεγαλύτερη ακρίβεια στην περιγραφή του κακόβουλου λογισμικού -ώστε να είναι πιο αποτελεσματικές στην αντιμετώπιση και αναχαίτιση του κακόβουλου λογισμικού-, είναι οι εξής: AMBL, BSL, CybOX™, MAEC™, MAIL, MBO, MIST, OpenIOC, OVAL®, RBS, YARA. Οι γλώσσες CAPEC™, IODEF, VERIS δεν έχουν αυτήν τη δυνατότητα, καθώς δεν περιγράφουν μόνο κακόβουλο λογισμικό, αλλά και άλλες περιπτώσεις χρήσης.

4.2.8 Κριτήριο ικανότητας διάκρισης καλοήθους από κακόβουλο λογισμικό

Το κριτήριο αυτό, έχει να κάνει με το πόσο ικανή είναι μια γλώσσα περιγραφής να διακρίνει το καλοήθες (benign) από το κακόβουλο λογισμικό, κατά την διαδικασία ανίχνευσης.

Πίνακας 9. Κριτήριο ικανότητας διάκρισης καλοήθους από κακόβουλο λογισμικό

Όνομα γλώσσας	***Ικανότητα διάκρισης***
AMBL	
BSL	
CAPEC™	
CybOX™	
IODEF	
MAEC™	
MAIL	●
MBO	
MIST	
OpenIOC	
OVAL®	
RBS	
VERIS	
YARA	

Στην βιβλιογραφία υπάρχουν ελλιπή στοιχεία / δεδομένα για το εάν κάποια από τις γλώσσες είναι ικανή να διακρίνει το καλοήθες από το κακόβουλο λογισμικό, εκτός από την γλώσσα MAIL, για την οποία και αναφέρεται αυτή η δυνατότητα.

4.2.9 Κριτήριο αποτελεσματικότητας στην αναχαίτιση του κακόβουλου λογισμικού

Το κριτήριο αυτό, ασχολείται με το κατά πόσο είναι μια γλώσσα περιγραφής αποτελεσματική (*effective*) ως προς την αναχαίτιση του κακόβουλου λογισμικού:

Πίνακας 10. Κριτήριο αποτελεσματικότητας στην αναχαίτιση κακόβουλου λογισμικού

Όνομα γλώσσας	***Είναι αποτελεσματική;***
AMBL	●
BSL	●
CAPEC™	
CybOX™	
IODEF	
MAEC™	
MAIL	●
MBO	●
MIST	
OpenIOC	●
OVAL®	
RBS	●
VERIS	
YARA	

Η γλώσσα MAIL αναφέρει υψηλά ποσοστά ανίχνευσης κακόβουλου λογισμικού, και για πρότινος άγνωστο, καθώς και για γνωστό κακόβουλο λογισμικό (Alam, 2014). Επίσης, ιδιαίτερα αποτελεσματικές είναι οι γλώσσες προσδιορισμών (*specification languages*) AMBL και RBS (Deschamps, 2008). Η δε γλώσσα OpenIOC, μέσω της πλατφόρμας Cisco AMP, έχει σχεδόν άριστα αποτελέσματα στην αναχαίτιση του κακόβουλου λογισμικού (Cisco, 2016).

Η οντολογία MBO έχει την δυνατότητα να ανιχνεύει άγνωστο κακόβουλο λογισμικό, παρ'όλο που στην παρούσα φάση δεν μπορεί να το ανιχνεύσει σε πραγματικό χρόνο (*real-time detection*) (Grégio, Bonacin, & Nabuco, 2016),

(Mavroeidis & Bromander, 2017). Η γλώσσα BSL, επίσης έχει ικανή δυνατότητα ανίχνευσης (Martignoni, Stinson, Fredrikson, Jha, & Mitchell, 2008).

Για τις υπόλοιπες γλώσσες έχουμε ελλιπή δεδομένα στην βιβλιογραφία, ως προς την αποτελεσματικότητά τους στην ανίχνευση, ή δεν έχουν υψηλά ποσοστά ανίχνευσης.

4.2.10 Κριτήριο ύπαρξης εργαλείων γλώσσας περιγραφής

Ως προς την ύπαρξη ή μη εξειδικευμένων εργαλείων για την χρήση της γλώσσας περιγραφής, οι γλώσσες κατηγοριοποιούνται ως εξής:

Πίνακας 11. Κριτήριο ύπαρξης εργαλείων γλώσσας περιγραφής

Όνομα γλώσσας	***Είναι αποτελεσματική;***
AMBL	
BSL	
CAPEC™	
CybOX™	
IODEF	
MAEC™	
MAIL	
MBO	
MIST	
OpenIOC	●
OVAL®	●
RBS	
VERIS	
YARA	

Οι γλώσσες OpenIOC, και OVAL®, έχουν δικά τους εργαλεία για την χρήση τους. Μέσω τέτοιων εργαλείων, μπορεί ο χρήστης να εισάγει δικούς του συμπεριφορικούς όρους (*behavioral terms*) για το κακόβουλο λογισμικό, κι έτσι να κάνει αποτελεσματικότερη χρήση της γλώσσας περιγραφής. Για τις υπόλοιπες γλώσσες, δεν αναφέρεται η ύπαρξη εξειδικευμένων εργαλείων στην βιβλιογραφία.

4.3 Συγκριτική αξιολόγηση

Συγκεντρωτικά, τα κριτήρια σύγκρισης, όπως αξιολογήθηκαν στις προηγούμενες ενότητες, διαφαίνονται στον παρακάτω πίνακα:

Πίνακας 12. Συγκεντρωτικός πίνακας κριτηρίων (Α)

Όνομα γλώσσας	**Αντιμετώπιση APT**	**XML/JSON**	**Οντολογία**	**Γλώσσα προσδιορισμών**	**Επεκτάσιμη**	**Δομημένη**	**Προσαρμόσιμη**	**Τυποποιημένη**	**Διαμοιραζόμενη**	**Αποτελεσματική**
AMBL				●		●				●
BSL				●		●				●
CAPEC™						●		●		
CybOX™		●			●	●	●	●	●	
IODEF		●				●		●	●	
MAEC™		●			●	●		●		
MAIL						●				●
MBO			●			●				●
MIST						●				
OpenIOC		●			●	●	●	●		●
OVAL®		●				●		●		
RBS				●		●				●
VERIS						●		●	●	
YARA		●				●		●		

[●]: Η γλώσσα περιγραφής *έχει* το συγκεκριμένο χαρακτηριστικό.

Πίνακας 13. Συγκεντρωτικός πίνακας κριτηρίων (Β)

Όνομα γλώσσας	**Λιτή / Αναγνώσιμη**	**Ικανότητα διάκρισης**	**Ακρίβεια περιγραφής**	**Ύπαρξη εργαλείων**	**Ακαδημαϊκή**	**Εταιρική**	**Ανοικτού κώδικα**	**Υποστηριζόμενη**
AMBL			●		●			
BSL			●		●			
CAPEC™	●					●		
CybOX™	●		●			●		
IODEF	●							
MAEC™	●		●			●		
MAIL		●	●		●			
MBO			●		●			
MIST			●		●			
OpenIOC	●		●	●		●	●	●
OVAL®	●		●	●		●	●	
RBS			●		●			
VERIS	●					●		
YARA	●		●				●	●

[●]: Η γλώσσα περιγραφής *έχει* το συγκεκριμένο χαρακτηριστικό.

Θα πρέπει να αναφέρουμε σύμφωνα με τον παραπάνω πίνακα, ότι όλες οι μελετηθείσες γλώσσες είναι δομημένες, έχουν συγκεκριμένα κάποια δομή ως προς την περιγραφή τους. Επίσης, δεν υπάρχουν πληροφορίες στην βιβλιογραφία για τις άδειες χρήσης για τις παραπάνω γλώσσες, εκτός των γλωσσών OpenIOC, OVAL® και YARA, οι οποίες είναι ανοικτού κώδικα. Επιπλέον, οι γλώσσες OpenIOC και YARA είναι υποστηριζόμενες από την κοινότητα.

5 Συμπεράσματα

Στην βιβλιογραφία αναφέρεται ένας ικανός αριθμός γλωσσών-προτύπων περιγραφής συμβάντων (*events*) κυβερνο-ασφάλειας. Μελετήσαμε, και δώσαμε μία περίληψη των ικανοτήτων της κάθε γλώσσας που συναντήσαμε στην βιβλιογραφία, όπως επίσης και των ειδικών δομικών χαρακτηριστικών της κάθε γλώσσας περιγραφής.

Οι γλώσσες περιγραφής έχουν ετερογενή χαρακτηριστικά, κάτι που καθιστά αρχικά, δύσκολη μία σύγκρισή μεταξύ τους. Παρ'όλα αυτά, συγκρίνοντας τα επιμέρους χαρακτηριστικά τους, μπορούμε να εξάγουμε τα ανάλογα συμπεράσματα.

Οι μισές περίπου από τις μελετηθείσες γλώσσες στην βιβλιογραφία (CybOX™, IODEF, MAEC™, OpenIOC, OVAL®, YARA), εκφράζονται μέσω των γλωσσών περιγραφής δεδομένων XML/JSON, ενώ αναφέρονται μία οντολογία, και τρεις γλώσσες προσδιορισμών. Έτσι τείνουν να είναι αναγνώσιμες γλώσσες (*human-readable languages*), και περαιτέρω, επιτρέπουν τον διαμοιρασμό της πληροφορίας μεταξύ των ενδιαφερόμενων οντοτήτων (*entities*).

Επιπλέον, οι γλώσσες περιγραφής που βασίζονται στις XML/JSON, είναι εταιρικής προέλευσης, με την οντολογία MBO, τις γλώσσες προσδιορισμών (*specification languages*), AMBL, BSL, και RBS, και τις γλώσσες MAIL, MIST, να έχουν δημιουργηθεί στα πλαίσια ακαδημαϊκού επιπέδου.

Όλες οι γλώσσες είναι δομημένες, και οι περισσότερες από αυτές διακρίνονται για την ακρίβειά τους στην περιγραφή του κακόβουλου λογισμικού, με προεξάρχουσες γλώσσες τις, AMBL, BSL, CybOX™, MAEC™, MAIL, MBO, MIST, OpenIOC, OVAL®, RBS, και YARA. Η ακρίβεια στην περιγραφή του κακόβουλου λογισμικού, είναι ένας βασικός παράγοντας, ούτως ώστε να έχει την δυνατότητα η κάθε γλώσσα να μπορεί να ανιχνεύει το κακόβουλο λογισμικό με μεγάλα ποσοστά επιτυχίας.

Η ανίχνευση κυρίως του άγνωστου κακόβουλου λογισμικού είναι ένα δύσκολο πρόβλημα, λόγω της πολυμορφικότητάς του, αλλά οι γλώσσες περιγραφής, καθώς είναι βασισμένες σε συμπεριφορικά χαρακτηριστικά (*behavioral attributes*), δείχνουν να μπορούν να αντιμετωπίσουν αυτό το πρόβλημα, και πολλές φορές με πολύ υψηλό ποσοστό επιτυχίας, που κάποιες φορές μπορεί και να αγγίζει το 99-100% (όπως στην περίπτωση της γλώσσας MAIL, ή της OpenIOC μέσω της πλατφόρμας "Cisco AMP").

Ως προς την αποτελεσματικότητα στην ανίχνευση (*detection*), οι γλώσσες οι οποίες είναι αποτελεσματικές στην ανίχνευση του κακόβουλου λογισμικού, είναι οι AMBL, BSL, MAIL, MBO, OpenIOC, RBS.

Η γλώσσα OpenIOC συγκεντρώνει τα περισσότερα ποιοτικά επιμέρους χαρακτηριστικά από όλες τις γλώσσες περιγραφής κακόβουλου λογισμικού, ακολουθούμενη από την γλώσσα CybOX™, όπως μπορούμε να συμπεράνουμε διαβάζοντας τους πίνακες 12 & 13.

Επίσης, συμπερασματικά, μπορούμε να αναφέρουμε ότι υπάρχουν ελλιπή δεδομένα στην βιβλιογραφία, σχετικά με το εάν οι γλώσσες περιγραφής, ή έστω κάποιες από αυτές, μπορούν να αντιμετωπίσουν με επιτυχία μία Προηγμένη Επίμονη Απειλή (*Advanced Persistent Threat* – "*APT"*).

Οι γλώσσες περιγραφής-χαρακτηρισμού του κακόβουλου λογισμικού, είναι βασισμένες στα συμπεριφορικά χαρακτηριστικά του κακόβουλου λογισμικού, κι έτσι μπορούν να περιγράψουν με μεγαλύτερη ακρίβεια την συμπεριφορά του, κι επίσης να διακρίνουν με μεγαλύτερη ακρίβεια το καλοήθες από το κακόβουλο λογισμικό, προσδίδοντας έτσι μεγαλύτερα ποσοστά ανίχνευσης, κάτι που επιτυγχάνεται από την γλώσσα MAIL.

Η μελέτη δεν θα μπορούσε να είναι διεξοδική, καθώς αναφέρεται στην βιβλιογραφία μία πληθώρα τέτοιων γλωσσών. Αυτό που στάθηκε δυνατόν, είναι να μελετηθούν αναλυτικά και εις βάθος οι προεξάρχουσες / πιο επιφανείς γλώσσες περιγραφής, είτε είναι αυτές ανοιχτού κώδικα (*open source*), ή είναι δημιουργήματα μεγάλων εταιρειών του χώρου της κυβερνο-ασφάλειας, όπως η εταιρεία *MITRE* ή η εταιρεία *FireEye*.

6 Βιβλιογραφία - Πηγές

Accenture. (2017). *Cost of Cyber Crime Study*. Retrieved from https://www.accenture.com/t20170926T072837Z__w__/us-en/_acnmedia/PDF-61/Accenture-2017-CostCyberCrimeStudy.pdf

Advanced persistent threat. (2018, December 5). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Advanced_persistent_threat

Alam, S. (2014). MAIL: Malware Analysis Intermediate Language.

Andress, J. (2015, May). Working with Indicators of Compromise. *ISSA Journal*. Retrieved from https://c.ymcdn.com/sites/www.issa.org/resource/resmgr/journalpdfs/feature0515.pdf

Aycock, J. (2006). *Computer Viruses and Malware.* Springer.

Aycock, J., & Barker, K. (2004). Creating a Secure Computer virus Laboratory (Case Study). *EICAR*.

Barnum, S., Martin, R., Worrell, B., & Kirillov, I. (2012, April 13). *The CybOX™ Language Specification - Version 1.0 (draft).* Retrieved from CybOX™ - MITRE Corporation: https://cybox.mitre.org/language/specifications/CybOX_Language_Core_Specification_v1.0.pdf

Burger, E. W., Goodman, M. D., Kampanakis, P., & Zhu, K. A. (2014, November 3). Taxonomy Model for Cyber Threat Intelligence Information. *WISCS'14*. doi:http://dx.doi.org/10.1145/2663876.2663883

CAPEC™. (2018). Retrieved from Common Attack Pattern Enumeration and Classification: http://capec.mitre.org

Cisco. (2016, April 12). *Cisco Advanced Malware Protection Solution Overview.* Retrieved from Cisco: https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/advanced-malware-protection/solution-overview-c22-734228.html

Cuckoo Sandbox. (2017). Retrieved from http://cuckoosandbox.org

CybOX™ Design: Object Hierarchy Structuring. (2018). Retrieved from GitHub: https://github.com/CybOXProject/schemas/wiki/CybOX-Design:-Object-Hierarchy-Structuring

Danilyw, R., Meijer, J., & Demchenko, Y. (2007, December). The Incident Object Description Exchange Format. (IETF, Ed.) Retrieved from https://tools.ietf.org/html/rfc5070

Deeg, M., Nerz, S., & Sauder, D. (2014). *Outsmarted - Why Malware Works in face of Antivirus Software.* SySS GmbH.

Deschamps, N. (2008). D08 (D4.1) Specification language for code behavior. *Wombat Project*, (pp. 23-36). Retrieved from http://wombat-project.eu/WP4/FP7-ICT-216026-Wombat_WP4_D08_V01_Specification_language_for_code_behaviour.pdf

Domain-specific language. (2018, October 17). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Domain-specific_language

Dominguez, J. (2015, April 8). *Cisco AMP Just Got Better – Enhancements for Continuous Breach Detection, Response, and Remediation.* Retrieved from Cisco: https://blogs.cisco.com/security/cisco-amp-just-got-better-enhancements-for-continuous-breach-detection-response-and-remediation

Elisan, C. (2015). *Advanced Malware Analysis.* McGraw Hill.

FireEye. (2018). *Anatomy of Advanced Persistent Threats.* Retrieved 2018, from FireEye: https://www.fireeye.com/current-threats/anatomy-of-a-cyber-attack.html

Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware Analysis and Classification: A Survey. *Journal of Information Security*(5), 56-64.

Grégio, A., Bonacin, R., & Nabuco, O. (2016, March 22). An Ontology of Suspicious Software Behavior.

Hernandez-Ardieta, J. L., & Tapiador, J. E. (2013). Information Sharing Models for Cooperative Cyber Defense. *5th International Conference on Cyber Conflict*, 60-85.

Idika, N., & Mathur, A. P. (2007). *A Survey of Malware Detection Techniques.* West Lafayette, IN, USA: Purdue University.

Incident Object Description Exchange Format. (2018). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Incident_Object_Description_Exchange_Format

Indicator of Compromise. (2018). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Indicator_of_compromise

IODEF. (2018). Retrieved from IODEF - Incident Object Description and Exchange Format: http://xml.coverpages.org/iodef.html

Kampanakis, P. (2014, September/October). Security Automation and Threat Information-Sharing Options. *CSIRTS*.

Kampanakis, P., & Suzuki, M. (2017, November). Incident Object Description Exchange Format Usage Guidance. (IETF, Ed.) Retrieved from https://tools.ietf.org/html/rfc8274

Kirillov, I. A., Beck, D. A., Chase, M. P., & Martin, R. A. (2010). The Concepts of the Malware Attribute Enumeration and Characterization (MAEC) Effort.

Kirillov, I., Beck, D., Chase, P., & Martin, R. (2014). MAEC - Malware Attribute Enumeration and Characterization.

Koen, E. (2017, February 10). *Indicators of Compromise and where to find them.* Retrieved from Cisco: https://blogs.cisco.com/security/indicators-of-compromise-and-where-to-find-them

Lee, A., Varadharajan, V., & Tupakula, U. (2013). On Malware Characterization and Attack Classification. *Proceedings of the First Australasian Web Conference (AWC).*

Liao, X., Yuan, K., Wang, X., Li, Z., Xing, L., & Beyah, R. (2016, October 24-28). Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence. *CCS'16,* 755-766. doi:10.1145/2976749.2978315

Mandiant. (2013). *OpenIOC 1.1 DRAFT.* Retrieved from GitHub: https://github.com/mandiant/OpenIOC_1.1/

Mandiant. (2016). *Sophisticated Indicators for the Modern Threat Landscape: An Introduction to OpenIOC.* Mandiant. Retrieved April 2018, from http://docplayer.net/11673966-Sophisticated-indicators-for-the-modern-threat-landscape-an-introduction-to-openioc.html

Mandiant. (2017). Retrieved from OpenIOC: https://web.archive.org/web/20171016095119/http://www.openioc.org:80/

Marak, V. (2015). *Windows Malware Analysis Essentials.* Packt Publishing.

Martignoni, L., Stinson, E., Fredrikson, M., Jha, S., & Mitchell, J. C. (2008). A Layered Architecture for Detecting Malicious Behaviors. *11th International Symposium on Recent Advances in Intrusion Detection.*

Mavroeidis, V., & Bromander, S. (2017). Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence. *Proceedings of the European Intelligence and Security Informatics Conference.* Attica, Greece.

Mitre Corporation. (2017). *Common Attack Pattern Enumeration and classification - CAPEC™ - A community knowledge resource for building secure software.* MITRE Corporation. Retrieved from http://makingsecuritymeasurable.mitre.org/docs/capec-intro-handout.pdf

Mitre Corporation. (2017). *CybOX™ - Cyber Observable eXpression.* Retrieved from Github: https://cyboxproject.github.io/

Mundie, D. A., & McIntire, D. M. (2013). An Ontology for Malware Analysis. *International Conference on Availability, Reliability, and Security,* 556-558. doi:10.1109/ARES.2013.73

Noy, N. F., & McGuinness, D. L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology.* Retrieved from Stanford University: https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

Obrst, L., Chase, P., & Markeloff, R. (2012). Developing an Ontology for the Cyber Security Domain. *STIDS*, 49-56.

Oktavianto, D., & Murhadianto, I. (2013). *Cuckoo Malware Analysis.* Packt Publishing.

Ontology (information science). (2018). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Ontology_(information_science)

Open Vulnerability and Assessment Language. (2016, February 9). Retrieved from OVAL® - Open Vulnerability and Assessment Language: https://oval.mitre.org/index.html

OVAL Interpreter. (2018). Retrieved from Sourceforge: https://sourceforge.net/projects/ovaldi/

Pfleeger, C. P., Pfleeger, S., & Margulies, J. (2015). *Security in Computing* (5th ed.). Prentice Hall.

Rattan, A., Kaur, N., Chamotra, S., & Bhushan, S. (2017). Attack Data Usability and Challenges in its Capturing and Sharing. *International Journal of Advanced Studies in Computer Science and Engineering, 6*(9), 35-42.

Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software.* No Starch Press.

Swimmer, M. (2008, January 27). Towards an Ontology of Malware Classes. *John Jay College of Criminal Justice*.

Trinius, P., Willems, C., Holz, T., & Rieck, K. (2009, December 17). A Malware Instruction Set for Behavior-Based Analysis.

Van Impe, K. (2015, June 24). *Signature-Based Detection with YARA.* Retrieved from https://securityintelligence.com/signature-based-detection-with-yara/

VERIS Framework. (2018). Retrieved from VERIS Community: http://veriscommunity.net/

YARA. (2018). Retrieved from GitHub: https://github.com/VirusTotal/yara

YARA Documentation. (2018). Retrieved from Read_The_Docs: https://yara.readthedocs.io/en/v3.7.0/

Yu, B., Fang, Y., Yang, Q., Tang, Y., & Liu, L. (2017, February 21). A survey of malware behavior description and analysis. doi:http://dx.doi.org/10.1631/FITEE.1601745

Zeltser, L. (2015, February 9). *Virtualized Network Isolation for a Malware Analysis Lab.* Retrieved from Information Security in Business: https://zeltser.com/vmware-network-isolation-for-malware-analysis/