ETPLC

Un IDS pour vos logs proxy ou webserver

@Rmkml

http://etplc.org

http://sourceforge.net/projects/etplc/

Sommaire

-Présentation de l’auteur
-Pourquoi le projet ETPLC ?
-Pourquoi choisir d’essayer ?
-Les Menaces
-Exemples d’utilisations
-Options Category, Syslog, Debug
-Formats des logs reconnus
-Exemples de conversions de signatures
-Utilisations avancées
-Performances
-Nouveau “collecteur” permettant de récupérer les logs depuis Elasticsearch
-Questions ?
-Remerciements

Présentation de l’auteur

Rmkml

Consultant en Sécurité Informatique mais pas développeur ;)

Twitter: @Rmkml

Email: Rmkml@yahoo.fr

Pourquoi le projet ETPLC ?

-ETPLC signifie “Emerging Threats Proxy Log Checker”

-Ou comment trouver des menaces dans vos logs proxy ou webserver à partir des signatures réseaux Emerging Threats.

-Existe-t’il d’autres solutions permettant de rechercher des menaces/intrusions “indépendantes” dans les logs proxy/web ?

Projet Open Source sous licence GPL v2. Maj régulière
Presque 9000 signatures réseaux sont prises en charge dans ETPLC.
Principalement les signatures contenant: “to_server” ou “from_client” *HTTP*.

(Au total, Emerging Threats contient environ 18 000 signatures réseaux.)

Pourquoi choisir d’essayer ?

-Vous n’avez pas ou pas d’accès sur le port Span/Mirror de votre trafic Internet

-Vous n’avez pas d’IDS/IPS (oops, c’est un autre sujet)

-Vous avez un IPS mais pas d’IDS (et peut être très peu d’alertes?)

-Vous avez un IDS mais avec très peu d’alertes de “menaces” Botnet/Trojan/Malware… (Êtes-vous vraiment sécurisé ?)

-Faire une analyse “Post Mortem” de vos Logs

-Vous voulez suivre la Communauté de chercheurs en menaces + Emerging Threats

-Pour essayer, c’est gratuit ;)

Les Menaces

Dans la limite des capacités du projet et du contenu des logs:
-Détecter certaines variantes dans les EK/Malware/Trojan/APT comme TorrentLocker, Symmi, Neverquest, Gozi/UrSnif/Papras, Nuclear, SpyClicker, CryptoLocker, CryptoWall, Xsser, Flashpack, Job314, Infostealer, Tinba, SoftPulse, NewPosThings, Astrum, iOS/AppBuyer, JackPOS, DecebalPOS, Stobox, Kyle, Sweet Orange, Stan, Poweliks, OSX.XSLCmd, Bravix, Bapy, Android/Youmi, Waterspoot, Threebyte, Archie, ScanBox…

-Détecter les Injections / Remote Commandes / DirTraversal dans Bash (ShellShock), Twiki, Bugzilla, BossaBot, Webmin, WordPress…

-Les certificats SSL malicieux utilisés par Napolar, Upatre, Dyre...
-Tor, Sinkhole...

Exemples d’utilisations (1)

Sans argument pour avoir une aide d’utilisation: perl etplc.pl

PERL:

-Lecture d’un fichier via un “pipe” sous linux:

cat /var/log/messages | perl etplc.pl -f etplc.rules.gz

-Suivi d’une log “temps réel” sous linux:

tail -f /var/log/messages | perl etplc.pl -f etplc.rules.gz

PYTHON (v2): (même syntaxe en PYTHON v3)

cat /var/log/messages | python2 etplc.py2 -f etplc.rules.gz

tail -f /var/log/messages | python2 etplc.py2 -f etplc.rules.gz

Exemples d’utilisations (2)

Exemple de sortie:

cat /var/log/messages | perl etplc.pl -f etplc.rules.gz

ok trouvé: timestamp: 2013-11-22T22:01:49.577030+01:00, server_hostname_ip: mybox, client_hostname_ip: 192.168.1.1, client_http_method: POST, client_http_uri: /, client_http_useragent: Mozilla/, client_http_referer: -, client_http_cookie: cid=, http_reply_code: 200, etmsg: ET TROJAN Zeus POST Request to CnC - cookie variation, etmethod: POST, etagent: mozilla, etcookie: cid=, etpcreagent: ^Mozilla, etpcrecookie: ^cid=

-> Pour plus de détails sur cette signature, voir directement dans le fichier etplc.rules.gz ou http://rules.emergingthreats.net/open/snort-2.9.0/

Le “-” indique que le proxy web n’as pas de Referer HTTP.

Options disponibles

Pour voir toutes les options: *perl etplc.pl -h*
python2 etplc.py2 -h

-Envoi des “alertes” ETPLC via syslog avec l’option en ligne de cmd (-s) (par défaut sur le port 514/udp de la “localhost”)
Exemple: *tail -f /var/log/messages | perl etplc.pl -f etplc.rules.gz -s*

-Utilisation des “Catégories” pour limiter le nombre de signature (-c all ou proxy ou webserver)
Exemple: *tail -f /var/log/messages | perl etplc.pl -f etplc.rules.gz -c proxy*

-Mode Debug (-d) permettant d’afficher l’ensemble du fonctionnement interne

Format des logs reconnus

Nécessité d'avoir un parser de log "évolutif" pour gérer les multiples formats:

-Apache Web Server (je n'ai pas testé le module proxy) http://etplc.org/apache.html
-Squid Proxy http://etplc.org/squid.html
-BlueCoat SG Proxy
-Microsoft TMG/ForeFront Proxy
-McAfee WebGateway Proxy [mwg]

Importants: pour mieux détecter les menaces, il faut rajouter les champs suivants:

HTTP User-Agent # exemple: IE, Mozilla, Outlook, Nessus, Sqlmap...
HTTP Referer
HTTP Cookie
HTTP Remote IP # New depuis la version du 16 nov avec Squid

Exemple de conversions de signatures (1)

Exemple simple de conversion d'une signature au format Snort/Suricata:

alert tcp any any -> any 80 (msg:"exemple 1"; flow:to_server,established; content:"/test"; nocase; http_uri; sid:1; rev:1;)

->Cette signature détecte une requête web contenant une url contenant /test.

ETPLC reconnaît "nativement" cette signature.

Exemple de conversions de signatures (2)

Exemple de conversion d’une signature au format Snort/Suricata:

alert tcp any any -> any 80 (msg:”exemple 2”; flow:to_server,established; content:”Java/”; nocase; http_header; sid:2; rev:1;)

->Cette signature détecte une requête web contenant un User-Agent Java

ETPLC reconnaît “nativement” cette signature.

Exemple de conversions de signatures (3)

Exemple de conversion d'une signature au format Snort/Suricata:

alert tcp any any -> any 80 (msg:"exemple 3"; flow:to_server,established; content:"Referer|3a| http://dell"; nocase; http_header; sid:3; rev:1;)

->Cette signature détecte une requête web contenant un Referer commençant par "http://dell"...

ETPLC reconnaît "nativement" cette signature.

Exemple de conversions de signatures (4)

Exemple de conversion d'une signature au format Snort/Suricata:

alert tcp any any -> any 80 (msg:"exemple 4"; flow:to_server,established; content:".php"; nocase; http_uri; content:"Mozilla"; nocase; http_header; pcre:"/\/\d+\.php$/Ui"; sid:4; rev:1;)

->Cette signature détecte une requête web contenant un ".php" dans l'uri avec un User-Agent Mozilla et une expression régulière (pcre): "un_ou_plusieurs_chiffres" puis un "." puis "php" à la fin de l'uri.

ETPLC reconnaît "nativement" cette signature.

Exemple de conversions de signatures (5)

Depuis la version du 16 nov, ETPLC intègre la liste des IPs connus étant comme “Shadowserver C&C”, exemple de signature au format Snort/Suricata:

alert ip any any -> 130.13.232.232 any (msg:”Shadowserver C&C List: 130.13.232.232”; reference:url,rules.emergingthreats.net/fwrules/emerging-Block-IPs.txt; classtype:misc-activity; sid:9990001; rev:1;)

ETPLC reconnaît “nativement” cette signature.

Utilisations avancées

Un peu plus 4000 signatures réseaux réécrites pour le projet ETPLC.

-Toutes les signatures contenant un Host header HTTP ont été réécrites pour être intégrées dans le champ “http_uri”.
-Les nombreuses signatures contenant un domaine dns ont été réécrites, permettant de détecter ces domaines dans les URI (sans avoir besoin des logs dns).
-Récemment, Emerging Threats a intégré les certificats (SSL/TLS) provenant d’ Abuse.ch.
Elles ont été réécrites pour détecter les domaines malicieux dans les URI.
-Quelques signatures contenant un Cookie spécifique ont été réécrites

Performances

La version Perl utilise les Threads::Queue() permettant la gestion multicoeur.

Les versions Python v2/v3 utilisent le module multiprocessing.Pool()

La version Perl est ~15% plus rapide que la version Python v2, je pense principalement dû aux librairies Perl Regex et Python Regex (capture d'une centaine de paramètres dans les signatures).
La version Python v3 est ~10% moins rapide que la version Python v2.

ETPLC utilise l''ensemble des cores disponibles via une auto-détection (/proc/cpuinfo).

L'utilisation de l'option Categories (-c all,proxy,webserver) divise environ par deux le nombre de signatures, donc "augmente" par deux les performances!

Collecteur Elasticsearch

La première version du “Collecteur” qui permet de récupérer les logs depuis Elasticsearch. C’est un script Perl ;)

Exemple d’utilisation: *perl etplc_elasticsearch.pl | perl etplc.pl ...*
perl etplc_elasticsearch.pl | python2 etplc.py2 ...

Le collecteur va “chercher” les logs et les converti pour etplc au format Squid!

Avant d’utiliser le collecteur, il faut vérifier tout ses paramètres.
($timestampelasticsearch="now-30m", servers, size, index, sort…)

Questions ?

(Toutes les suggestions, modifications sont les bienvenues)

Remerciements

Ossir

La “Communauté” Sécurité

@EmergingTreats

@Elasticsearch

Nicolas Q

@Rmkml

http://etplc.org

http://sourceforge.net/projects/etplc/