TLP:White

ETPLC

« Un IDS pour vos logs proxy ou webserver »

@Rmkml

http://etplc.org
https://sourceforge.net/projects/etplc/
https://github.com/rmkml/etplc

(Etplc: Emerging Threats Proxy Logs Checker)

Sommaire:

-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 Splunk
-"Collecteur" permettant de récupérer les logs depuis Elastic(search)
-Questions ?
-Remerciements

Pourquoi le projet ETPLC ?:

-ETPLC signifie “Emerging Threats Proxy Log Checker”

-Outil permettant de 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 crée en 2013.
- Mises à jour régulières quasi hebdomadaire des signatures
- Plus de **9000 signatures réseaux** prises en charge (principalement les signatures contenant: “to_server” ou “from_client” *HTTP*).

(Emerging Threats contient au total plus de 21 000 signatures réseaux.)

Pourquoi choisir d’essayer ?:

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

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

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

-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 J

Les Menaces:

Dans la limite des capacités du projet et du contenu des logs:

- Recherche de menaces très vastes:

- Détecter certaines variantes dans les EK (Exploit Kit) / Malware / Trojan / APT comme Mirai, Symmi, SunOrcal, DeepEnd, RootX, Gozi/UrSnif/Papras, Nuclear, SpyClicker, CryptoLocker, CryptoWall, Flashpack, Job314, Infostealer, Tinba, SoftPulse, NewPosThings, Astrum, iOS/AppBuyer, JackPOS, DecebalPOS, Stobox, Kyle, Sweet Orange, Stan, Poweliks, OSX.XSLCmd, Bapy, Android/Youmi, Waterspoot, Threebyte, Archie…

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

-Les certificats SSL malicieux utilisés par Brazilian Banke, Napolar, Upatre, Dyre...

-Tor, Sinkhole...

Exemples d'utilisations (1):

PERL:

-Lecture d'un fichier via un "pipe" sous linux:

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

=> Si vos logs contiennent des menaces, alors vous aurez une sortie « colorisé » permettant d'identifier rapidement les « menaces » ou des «signaux faibles »

-Suivi d'une log "temps réel" sous linux:

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

PYTHON (v2 ou v3):

tail -F /var/log/messages | python etplc.py -f etplc.rules.gz

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=

Contenu de la signature au format Snort/Suricata:

alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"ET TROJAN Zeus POST Request to CnC - cookie variation"; flow:established,to_server; content:"POST"; nocase; http_method; content:"Cookie|3a 20|cid="; http_header; content:"User-Agent|3a 20|Mozilla"; http_header; reference:url,zeustracker.abuse.ch/monitor.php?search=209.59.216.103; classtype:trojan-activity; sid:2014107; rev:3;)

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

Options disponibles:

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

python etplc.py -h

-Filtre sur les signatures avec la nouvelle option « ***-y*** » permettant de choisir la ou les années de création, exemple: « -y 2017,2016 »

-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 -Squid Proxy
-BlueCoat SG Proxy -Nginx
-Microsoft TMG/ForeFront Proxy -Microsoft IIS
-McAfee WebGateway Proxy [mwg] -WebSense (ForcePoint now)

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

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

Exemple simple de conversion d'une signature au format Snort/Suricata: (1)

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

```
alert ou drop ou pass…          # « réaction » de l'IDS/IPS
tcp ou udp ou icmp ou ip…       # Protocole
any any                  # IP source et Port source
any 80                   # IP destination et Port destination
msg                      # Message si détection
flow…                    # connexion établie ou pas et sens de la cnx
content…                 # critère de recherche
nocase                   # non sensible à la casse
http_uri                 # limitation dans la recherche uniquement dans l'uri
sid rev                  # Snort ID et Révision
```

->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 5000 signatures réseaux réécrites pour le projet ETPLC.

-Toutes les signatures contenant un Host header HTTP ont été réécrites

-Les nombreuses signatures contenant un domaine dns ont été réécrites, permettant de détecter ces domaines dans les URI (sans avoir besoin des logs dns).

-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 des multicoeurs.

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 (nécessité de la 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 simple auto-détection dans /proc/cpuinfo.

->L'utilisation de la nouvelle option « year » (-y 2017) divise environ par cinq le nombre de signatures, donc multiplie par cinq les performances!
->L'utilisation de l'option « Categories » (-c all,proxy,webserver) divise environ par deux le nombre de signatures, donc multiplie par deux les performances!

Collecteur Splunk:

La première version du “Collecteur” qui permet de récupérer les logs depuis Splunk via la « REST API » au format CIM (Common Information Model=normalisation des champs). C’est une boucle Perl J

Exemple d’utilisation: *perl etplc_splunk.pl … | perl etplc.pl ...*
perl etplc_splunk.pl … | python etplc.py ...

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

Avant d’utiliser ce collecteur, il faut vérifier tout ses paramètres.

Pour plus d’infos: http://etplc.org/splunk.html

Collecteur Elastic(search):

La première version du “Collecteur” qui permet de récupérer les logs depuis Elastic(search) via la « REST API ». C’est également une boucle Perl J

Exemple d’utilisation: *perl etplc_elasticsearch.pl | perl etplc.pl ...*
perl etplc_elasticsearch.pl | python etplc.py ...

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

Plus d’infos sur: http://etplc.org/elasticsearch.html

Questions ?

(Toutes les suggestions sont les bienvenues)

Remerciements:

L'OSSIR
La "Communauté" Sécurité
@EmergingThreats
@Splunk
@Elastic(search)
Nicolas Q.
Julien S.
@Rmkml
http://etplc.org
https://sourceforge.net/projects/etplc/
https://github.com/rmkml/etplc