



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání



jihořmoravský kraj

SPRÁVA A DOHLED NAD POČÍTAČOVOU SÍTÍ

Logy – ArcSight FlexConnector

Metodický list

Autor: Giovanni Guadagno, Metodik: Bc. Jaroslav Tihlařik

Recenzent: doc. Ing. Jaroslav Dočkal, CSc.

Rok vydání: 2023

Logy - ArcSight FlexConnector podléhá licenci CC BY-SA 4.0 International License (Offline use:

<http://creati-vecommons.org/licenses/by-nc-sa/4.0/>).



Obsah

Dovednosti	2
Pracovní prostředí	2
Průběh výuky	3
1 Příprava	3
2 Hledání vzorů logů & RegEx – normalizace.....	3
2.1 Hlavní vzor.....	3
2.1.1 Práce s poli hlavního vzoru	8
2.2 Práce se submessage	9
2.2.1 Nejednoznačné mapování	11
2.2.2 Custom pole	12
2.2.3 Povinné náležitosti každého patternu.....	12
2.2.4 Pole deviceEventClassId.....	12
2.2.5 Pole name.....	13
2.2.6 Určování severity	13
3 Kategorizace logu.....	14
3.1 Tvorba kategorizace.....	14
Shrnutí a závěr	15

Cíle

- Student popíše, co to je log
- Student vysvětlí přínosy logování
- Student popíše, jak lze logy zpracovávat a používat
- Student navrhne a napíše parser; určí rozdíl mezi normalizací a kategorizací

Dovednosti

- Student analyzuje raw log, najde různé vzory logů
- Student ovládá RegEx na úrovni potřebné pro vytvoření parseru
- Student umí používat jednotlivá pole formátu CEF
- Student zná základní funkce a metody pro psaní parseru
- Student umí log kategorizovat
- Student dokáže parser otestovat, analyzovat jej, identifikovat chyby a ty napravit

Pracovní prostředí

Úlohu lze realizovat v prostředí Cylab JCEKB

Pro práci budeme potřebovat následující:

- ArcSight Connector Regex tool
- Zdroj raw logů (v tomto případě Linux audit)
- Dokument ArcSight CommonEventFormatV25
- Dokument ArcSight Event Categorization
- Dokument ArcSight FlexConnectorGuide
- Dokument ArcSight SmartConnector for Linux Audit Syslog
- Nedílnou součástí cvičení je také výuková prezentace
- Doporučuje se externí nástroj pro tvorbu RegEx (*např. regex101.com*)

Průběh výuky

1 Příprava

- Diskuse na téma logy a zdroje (viz prezentace)
- Spuštění xxx v Cylabu
- Příkaz v terminálu – spuštění nástroje na tvorbu parseru

```
.../ArcSightSmartConnectors/current/bin/arcsight regex
```

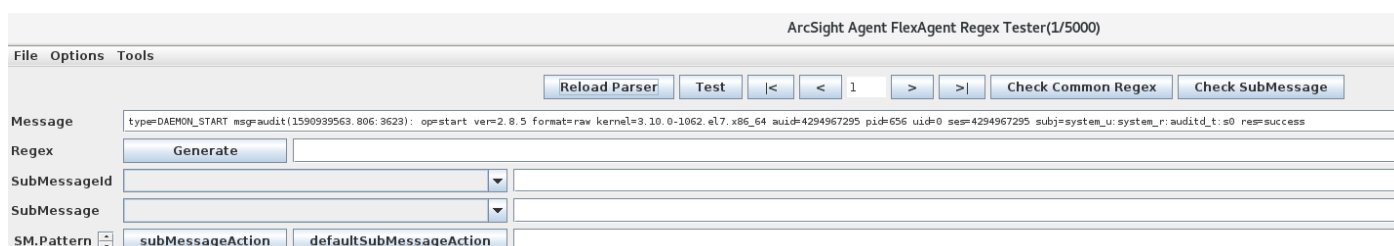
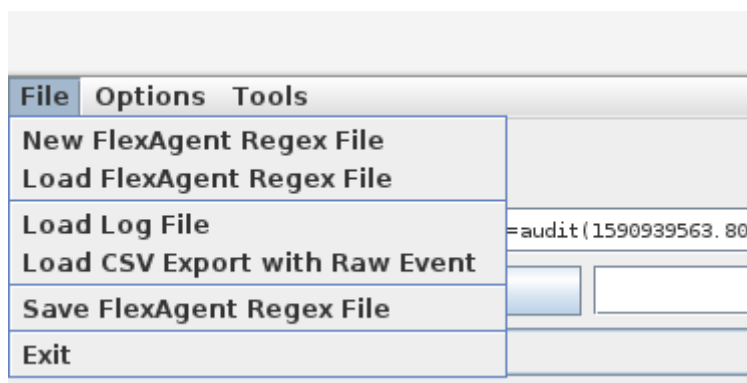
- Připravte si logy, které budou parsovány
- Příkazem v terminálu jako sudo vykopírujeme logy na vhodnější místo, například

```
cp var/log/audit/audit.log home/[username]
```

- Po vykopírování je vhodné změnit práva tak, aby se k souboru bylo možné nejen jako sudo

```
chmod 755 [absolutní cesta k souboru]
```

- Přes file a Load log file zvolím cestu, kde se nachází logy



2 Hledání vzorů logů & RegEx – normalizace

2.1 Hlavní vzor

- Celý parser v rámci normalizace může vypadat i jinak, zde je popsán jeden z možných přístupů
- Nejprve je nutné logy vizuálně analyzovat, pro tyto účely poslouží například Notepad++

- FlexConnector (parser) rozlišuje 3 různé vzorce logů
 - Hlavní – match všech logů procházených tímto parserem
 - Submessage – rozlišení různých vzorců logů následujících po hlavním (nepovinné)
 - Pattern – rozlišení vzorců logů v rámci jedné submessage (nepovinné, vždy svázáno s jednou submessage)
- V našem případě bylo zjištěno, že naprosto všechny logy začínají `type=[jméno]` a následují `msg=audit([epochtime])`, RegEx matchující tuto posloupnost znaků bude náš hlavní.

```

1 type=DAEMON_START msg=audit(1590939563.806:3623): op=start ver=2.8.5 format=raw kernel=3.10.0-10
2 type=CONFIG_CHANGE msg=audit(1590939564.067:5): audit_backlog_limit=8192 old=64 auid=429496729
3 type=CONFIG_CHANGE msg=audit(1590939564.067:6): audit_failure=1 old=1 auid=4294967295 ses=4294
4 type=SERVICE_START msg=audit(1590939564.069:7): pid=1 uid=0 auid=4294967295 ses=4294967295 su
5 type=SYSTEM_BOOT msg=audit(1590939564.074:8): pid=681 uid=0 auid=4294967295 ses=4294967295 s
6 type=SERVICE_START msg=audit(1590939564.077:9): pid=1 uid=0 auid=4294967295 ses=4294967295 su
7 type=SERVICE_START msg=audit(1590939564.096:10): pid=1 uid=0 auid=4294967295 ses=4294967295 s
8 type=SERVICE_START msg=audit(1590939564.109:11): pid=1 uid=0 auid=4294967295 ses=4294967295 s
9 type=SERVICE_START msg=audit(1590939566.648:12): pid=1 uid=0 auid=4294967295 ses=4294967295 s
10 type=SERVICE_START msg=audit(1590939566.673:13): pid=1 uid=0 auid=4294967295 ses=4294967295 s
11 type=SERVICE_START msg=audit(1590939566.675:14): pid=1 uid=0 auid=4294967295 ses=4294967295 s
12 type=SERVICE_START msg=audit(1590939566.676:15): pid=1 uid=0 auid=4294967295 ses=4294967295 s
13 type=SERVICE_START msg=audit(1590939566.677:16): pid=1 uid=0 auid=4294967295 ses=4294967295 s
14 type=SERVICE_START msg=audit(1590939566.686:17): pid=1 uid=0 auid=4294967295 ses=4294967295 s
15 type=SERVICE_START msg=audit(1590939566.688:18): pid=1 uid=0 auid=4294967295 ses=4294967295 s
16 type=SERVICE_START msg=audit(1590939566.690:19): pid=1 uid=0 auid=4294967295 ses=4294967295 s
17 type=SERVICE_START msg=audit(1590939566.696:20): pid=1 uid=0 auid=4294967295 ses=4294967295 s
18 type=SERVICE_START msg=audit(1590939566.706:21): pid=1 uid=0 auid=4294967295 ses=4294967295 s
19 type=SERVICE_START msg=audit(1590939566.791:22): pid=1 uid=0 auid=4294967295 ses=4294967295 s
20 type=SERVICE_STOP msg=audit(1590939566.791:23): pid=1 uid=0 auid=4294967295 ses=4294967295 su
21 type=SERVICE_START msg=audit(1590939567.570:24): pid=1 uid=0 auid=4294967295 ses=4294967295 s
22 type=SERVICE_START msg=audit(1590939567.572:25): pid=1 uid=0 auid=4294967295 ses=4294967295 s
23 type=SERVICE_START msg=audit(1590939567.575:26): pid=1 uid=0 auid=4294967295 ses=4294967295 s
24 type=SERVICE_START msg=audit(1590939567.576:27): pid=1 uid=0 auid=4294967295 ses=4294967295 s
25 type=SERVICE_START msg=audit(1590939567.577:28): pid=1 uid=0 auid=4294967295 ses=4294967295 s
26 type=SERVICE_START msg=audit(1590939568.053:29): pid=1 uid=0 auid=4294967295 ses=4294967295 s
27 type=SERVICE_START msg=audit(1590939568.159:30): pid=1 uid=0 auid=4294967295 ses=4294967295 s
28 type=SERVICE_START msg=audit(1590939568.806:31): pid=1 uid=0 auid=4294967295 ses=4294967295 s
29 type=SERVICE_START msg=audit(1590939569.105:32): pid=1 uid=0 auid=4294967295 ses=4294967295 s
30 type=SERVICE_START msg=audit(1590939569.142:33): pid=1 uid=0 auid=4294967295 ses=4294967295 s
31 type=SERVICE_START msg=audit(1590939569.600:34): pid=1 uid=0 auid=4294967295 ses=4294967295 s
32 type=SERVICE_START msg=audit(1590939570.171:35): pid=1 uid=0 auid=4294967295 ses=4294967295 s
33 type=SERVICE_START msg=audit(1590939571.813:36): pid=1 uid=0 auid=4294967295 ses=4294967295 s
34 type=SERVICE_START msg=audit(1590939572.781:37): pid=1 uid=0 auid=4294967295 ses=4294967295 s
35 type=NETFILTER_CFG msg=audit(1590939572.992:38): table=filter family=2 entries=0
36 type=NETFILTER_CFG msg=audit(1590939572.992:38): table=filter family=2 entries=0
37 type=NETFILTER_CFG msg=audit(1590939572.992:38): table=filter family=2 entries=0
38 type=SYSCALL msg=audit(1590939572.992:38): arch=c000003e syscall=175 success=yes exit=0 a0=227
39 type=PROCTITLE msg=audit(1590939572.992:38): proctitle=2F7362696E2F6D6F6470726F6265002D71002
40 type=NETFILTER_CFG msg=audit(1590939573.096:39): table=filter family=10 entries=0
41 type=NETFILTER_CFG msg=audit(1590939573.096:39): table=filter family=10 entries=0
42 type=NETFILTER_CFG msg=audit(1590939573.096:39): table=filter family=10 entries=0
43 type=SYSCALL msg=audit(1590939573.096:39): arch=c000003e syscall=175 success=yes exit=0 a0=1a6
44 type=PROCTITLE msg=audit(1590939573.096:39): proctitle=2F7362696E2F6D6F6470726F6265002D71002
45 type=SERVICE_START msg=audit(1590939573.327:40): pid=1 uid=0 auid=4294967295 ses=4294967295 s
46 type=NETFILTER_CFG msg=audit(1590939573.490:41): table=filter family=2 entries=0
47 type=NETFILTER_CFG msg=audit(1590939573.490:41): table=filter family=10 entries=0
48 type=SYSCALL msg=audit(1590939573.490:41): arch=c000003e syscall=272 success=yes exit=0 a0=400
49 type=PROCTITLE msg=audit(1590939573.490:41): proctitle="(ostnamed)"
50 type=SERVICE_START msg=audit(1590939573.617:42): pid=1 uid=0 auid=4294967295 ses=4294967295 s
51 type=SERVICE_START msg=audit(1590939573.730:43): pid=1 uid=0 auid=4294967295 ses=4294967295 s

```

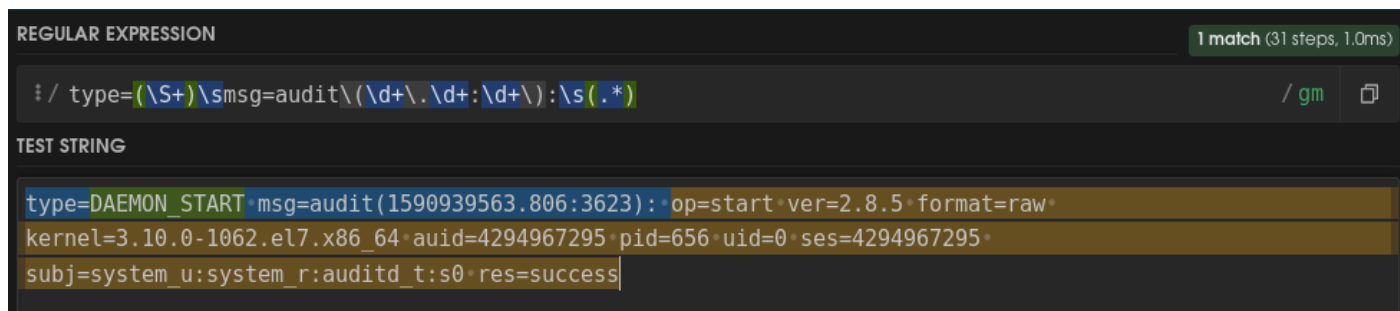
- Následující tabulka obsahuje tokeny, za pomoci kterých lze definovat určitý vzorec, posloupnost znaků
 - Není však třeba využívat jen tyto tokeny, je možné je kombinovat s „normálními slovy“

\s	Match whitespace
\S	Match non-whitespace
\d	Match číslice
\D	Match všeho kromě číslice
\w	Match písmen a číslic
\W	Match všeho kromě písmen a číslic
[xyz]	Match znaků x, y, z (Boolean OR)
[^xyz]	Match všeho kromě x, y, z (Boolean OR)
[A-Z]	Match znaků v rozmezí A-Z (case sensitive)
[^a-z]	Match všeho kromě znaků v rozmezí A-Z (case sensitive)
.	Match libovolného znaku
..	Match libovolných dvou znaků
?	Match nulakrát nebo jednou (např. \w?)
*	Match nulakrát nebo vícekrát
+	Match jednou nebo vícekrát
{5}	Match přesně pětkrát (např. \d{5})
{1,3}	Match jednou až třikrát (např. \{1,3} - vhodné pro IP adresy)
^	Začátek stringu
\$	Konec stringu
(...)	Match & capture (z chytnutého řetězce se stane token, možno dále pracovat)
(?:...)	Match, ale ne capture (token se nevytvoří) - non-capturing token
a b	Match a anebo b (není klasický Boolean OR)
\	Escape (např. \ (bude se závorkou pracovat jako s jiným znakem)

- Regex pro hlavní část logu může vypadat například takto

```
type=(\S+)\smmsg=audit\(\d+\.\d+:\d+\):\s(.*)
```

- Začíná type=, následuje alespoň jeden non-whitespace znak jako token (\S+), potom přesně jeden whitespace znak \s, po něm msg=audit, escape závorky \ (bere se jako obyčejný znak), potom alespoň jedna číslice \d+, space tečky \., alespoň jedna číslice \d+, znak dvojtečky, alespoň jedna číslice \d+, escape závorky \), znak dvojtečky, jeden whitespace \s a potom veškerý zbytek logu jako token, pokud nějaký zbyl (.*)
- S poslední tokenem budeme pracovat v submessage, resp. patternech
- Dle výše uvedeného je vhodné Regex psát v externím programu, například regex101.com, a to z důvodu mnohem lepší viditelnosti, nápovědy a případného debugu



- Modře je zde podbarven match celé posloupnosti znaků v logu, jinými barvami pak samotné tokeny, s nimiž budeme pracovat později
- Samotný Regex tool je pak vhodné používat při mapování na jednotlivá CEF pole

- Nyní již máme chytнутá nějaká data, s nimiž budeme moci dále pracovat; nejprve je však vhodné Regex otestovat – v našem případě budeme testovat hlavní, společný Regex, dáme proto „Check Common Regex“

- Log se podbarvil modře – to je chyba
- U některých logů není po časovém razítku dvojtečka

- Použijeme token, který umožní obě varianty logů – jak s dvojtečkou, tak bez

- Dvojtečka byla vložena do non-capturing tokenu a bude se chytat nulakrát, anebo jednou (?:)?
 - (?:něco) – non-capturing token, takový zápis vyžaduje znaky uvnitř, ale dál s nimi nepracuje
 - Kvantifikátor ? určuje, že předcházející token bude matchnut nulakrát, nebo jednou
 - Zápis (?:něco)? je tak vhodný na situace, kdy část logu obsahuje určitou posloupnost, kterou jiné logy neobsahují vůbec
- Upravíme Regex a znovu jej otestujeme

2.1.1 Práce s poli hlavního vzoru

- Nyní máme funkční Regex hlavního vzoru logu a můžeme s ním pracovat dále
- Nejprve nadefinujeme tokeny

Message: `type=DAEMON_START msg=audit(1590939563.806:3623): op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 aid=4294967295 pid=656 uid=0 ses=...`

Regex: `type=((\S+)\s)msg=audit((\d+\.\d+:\d+)\{(?:)?\}\s(.*)`

SubMessage: `op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 aid=4294967295 pid=656 uid=0`

Common Elements

Group	Label	Value
\$0	type	DAEMON_START
\$1	submessage	op=start ver=...

SubMessage Elements

Group	Value
-------	-------

```
# FlexAgent Regex Configuration File
do.unparsed.events=true

regex=type=((\S+)\s)msg=audit((\d+\.\d+:\d+)\{(?:)?\}\s(.*)

token.count=2

token[0].name=type
token[0].type=String

token[1].name=submessage
token[1].type=String
```

- Tokeny nejprve přejmenujeme dle libosti, v okně dole lze případně upravit datový typ
- Nyní tokeny namapujeme dle návodů
 - Token \$0 jako name
 - Token \$1 bude nést submessage – vždy mapujeme na pole message
 - Mapovat tokeny lze metodou „drag&drop“ tokenů z levé části doprostřed a následným vybráním příslušného pole z rozbalovací nabídky, nebo je možné psát přímo do okna dole

```
event.message=submessage //prvně definuji podle dle CEF a mapuji jej na jméno tokenu
event.name=type
```

Group	Label	Value	Field
\$0	type	DAEMON_START	event.message
\$1	submessage	op=start ver...	event.name
			event.generatorExternalID
			event.generatorURI
			event.globalEventId
			event.locality
			event.managerReceiptTime
			event.message
			event.modelConfidence
			event.name

- Mapování polí například na konstanty či používání funkcí (spojování) je však nutné přímo zapisovat do sponího okna
- V tomto případě nadefinujeme pole deviceVendor a deviceProduct na konstanty, a to následovně

```
event.deviceVendor=__stringConstant(„Unix“) //na pole dle CEF mapuji funkci (vždy začíná __)
event.deviceProduct=__stringConstant(„auditd“)
```

- Po uložení/přesunutí se na další log se změny propíší do přehledu nahoře

Field	Mapping
event.deviceVendor	stringConstant("Unix")
event.message	submessage
event.deviceProduct	stringConstant("auditd")
event.name	type

2.2 Práce se submessage

- Nyní máme nachystán hlavní vzor logu a je možné přistoupit k práci se submessages
- Každá submessage má svoje ID – to jsou znaky v hlavním vzoru, které definují submessage

```
21972 type=SERVICE_START msg=audit(1650817874.981:30): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=unbound-anchor comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21973 type=SERVICE_STOP msg=audit(1650817874.981:31): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=unbound-anchor comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21974 type=SERVICE_START msg=audit(1650817875.011:32): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=ModemManager comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21975 type=SERVICE_START msg=audit(1650817875.201:33): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=polkit comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21976 type=SERVICE_START msg=audit(1650817875.220:34): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=accounts-daemon comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21977 type=SERVICE_START msg=audit(1650817875.577:35): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=udisks2 comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21978 type=SERVICE_START msg=audit(1650817875.967:36): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=xdm comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21979 type=SERVICE_START msg=audit(1650817876.655:37): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=firewalld comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21980 type=NETFILTER_CFG msg=audit(1650817876.828:38): table=filter family=2 entries=0
21981 type=NETFILTER_CFG msg=audit(1650817876.828:38): table=filter family=2 entries=0
21982 type=NETFILTER_CFG msg=audit(1650817876.828:38): table=filter family=2 entries=0
21983 type=SYSCALL msg=audit(1650817876.828:38): arch=c000003e syscall=175 success=yes exit=0 a0=12c7340 a1=1db5 a2=41a96e a3=12c3300 items=0 ppid=633 pid=634 auid=4294967295 uid=0 gid=0 euid=0 fsuid=0 egid=0 sg
21984 type=PROCTITLE msg=audit(1650817876.828:38): proctitle=2F7362696E2F6D6F6470726F6265002D71002D2D0069707461626C655F66696C746572
21985 type=NETFILTER_CFG msg=audit(1650817876.886:39): table=filter family=10 entries=0
21986 type=NETFILTER_CFG msg=audit(1650817876.886:39): table=filter family=10 entries=0
21987 type=NETFILTER_CFG msg=audit(1650817876.886:39): table=filter family=10 entries=0
21988 type=SYSCALL msg=audit(1650817876.886:39): arch=c000003e syscall=175 success=yes exit=0 a0=20f3340 a1=1d05 a2=41a96e a3=20ef300 items=0 ppid=639 pid=640 auid=4294967295 uid=0 gid=0 euid=0 fsuid=0 egid=0 sg
21989 type=PROCTITLE msg=audit(1650817876.886:39): proctitle=2F7362696E2F6D6F6470726F6265002D71002D2D0069707461626C655F66696C746572
21990 type=SERVICE_START msg=audit(1650817876.977:40): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=NetworkManager comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21991 type=NETFILTER_CFG msg=audit(1650817877.021:41): table=filter family=2 entries=0
21992 type=NETFILTER_CFG msg=audit(1650817877.071:41): table=filter family=10 entries=0
21993 type=SYSCALL msg=audit(1650817877.071:41): arch=c000003e syscall=272 success=yes exit=0 a0=40000000 a1=7ff87c4ae20 a2=40000040 a3=22 items=0 ppid=1 pid=648 auid=4294967295 uid=0 gid=0 euid=0 fsuid=0 egid=0
21994 type=PROCTITLE msg=audit(1650817877.071:41): proctitle="(ostnamed)"
21995 type=SERVICE_START msg=audit(1650817877.172:42): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=systemd-hostnamed comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
21996 type=SERVICE_START msg=audit(1650817877.253:43): pid=1 uid=0 auid=4294967295 ses=4294967295 msg=unit=NetworkManager-dispatcher comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=suc
21997 type=NETFILTER_CFG msg=audit(1650817878.116:44): table=raw family=2 entries=0
21998 type=NETFILTER_CFG msg=audit(1650817878.116:44): table=raw family=2 entries=0
21999 type=NETFILTER_CFG msg=audit(1650817878.116:44): table=raw family=2 entries=0
22000 type=NETFILTER_CFG msg=audit(1650817878.116:44): table=raw family=2 entries=0
22001 type=SYSCALL msg=audit(1650817878.116:44): arch=c000003e syscall=175 success=yes exit=0 a0=a63390 a1=1a75 a2=41a96e a3=a5f300 items=0 ppid=701 pid=702 auid=4294967295 uid=0 gid=0 euid=0 fsuid=0 egid=0 sg
22002 type=PROCTITLE msg=audit(1650817878.116:44): proctitle=2F7362696E2F6D6F6470726F6265002D71002D2D0069707461626C655F66696C746572
22003 type=NETFILTER_CFG msg=audit(1650817878.127:45): table=security family=2 entries=0
22004 type=NETFILTER_CFG msg=audit(1650817878.127:45): table=security family=2 entries=0
22005 type=NETFILTER_CFG msg=audit(1650817878.127:45): table=security family=2 entries=0
22006 type=NETFILTER_CFG msg=audit(1650817878.127:45): table=security family=2 entries=0
22007 type=SYSCALL msg=audit(1650817878.127:45): arch=c000003e syscall=175 success=yes exit=0 a0=199a340 a1=1b3d a2=41a96e a3=1996300 items=0 ppid=704 pid=705 auid=4294967295 uid=0 gid=0 euid=0 fsuid=0 egid=0 sg
22008 type=PROCTITLE msg=audit(1650817878.127:45): proctitle=2F7362696E2F6D6F6470726F6265002D71002D2D0069707461626C655F7365637572697479
22009 type=NETFILTER_CFG msg=audit(1650817878.135:46): table=mangle family=2 entries=0
22010 type=NETFILTER_CFG msg=audit(1650817878.135:46): table=mangle family=2 entries=0
22011 type=NETFILTER_CFG msg=audit(1650817878.135:46): table=mangle family=2 entries=0
22012 type=NETFILTER_CFG msg=audit(1650817878.135:46): table=mangle family=2 entries=0
22013 type=SYSCALL msg=audit(1650817878.135:46): arch=c000003e syscall=175 success=yes exit=0 a0=1bdf340 a1=1c75 a2=41a96e a3=1bd9300 items=0 ppid=707 pid=708 auid=4294967295 uid=0 gid=0 euid=0 fsuid=0 egid=0 sg
22014 type=PROCTITLE msg=audit(1650817878.135:46): proctitle=2F7362696E2F6D6F6470726F6265002D71002D2D0069707461626C655F66696C746572
22015 type=NETFILTER_CFG msg=audit(1650817878.192:47): table=nat family=2 entries=0
22016 type=NETFILTER_CFG msg=audit(1650817878.192:47): table=nat family=2 entries=0
22017 type=NETFILTER_CFG msg=audit(1650817878.192:47): table=nat family=2 entries=0
22018 type=NETFILTER_CFG msg=audit(1650817878.192:47): table=nat family=2 entries=0
22019 type=SYSCALL msg=audit(1650817878.192:47): arch=c000003e syscall=175 success=yes exit=0 a0=1faf0 a1=20ed a2=41a96e a3=1fab300 items=0 ppid=710 pid=711 auid=4294967295 uid=0 gid=0 euid=0 fsuid=0 egid=0 sg
22020 type=PROCTITLE msg=audit(1650817878.192:47): proctitle=2F7362696E2F6D6F6470726F6265002D71002D2D0069707461626C655F66696C746572
22021 type=NETFILTER_CFG msg=audit(1650817878.207:48): table=raw family=10 entries=0
22022 type=NETFILTER_CFG msg=audit(1650817878.207:48): table=raw family=10 entries=0
```

- Nejprve nechme program, ať nám submessage vygeneruje včetně Regexu

Message `type=DAEMON_START msg=audit(1590939563.806:3623): op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 audit=4294967295 pid=656 uid=0 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=success`

Regex `type=(\S+)\smsg=audit\(\d+\.\d+:\d+\)\s\(\S+\)`

SubMessageId DAEMON_START

SubMessage op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 audit=4294967295 pid=656 uid=0 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=success

SM.Pattern

File Options Tools

Message `type=DAEMON_START msg=audit(1590939563.806:3623): op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 audit=4294967295 pid=656 uid=0 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=success`

Regex `type=(\S+)\smsg=audit\(\d+\.\d+:\d+\)\s\(\S+\)`

SubMessageId DAEMON_START

SubMessage op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 audit=4294967295 pid=656 uid=0 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=success

SM.Pattern[0] `op=start (\S+) format=raw (\S+) audit=(\S+) pid=(\S+) uid=(\d+) ses=(\S+) subj=system_u:system_r:auditd_t:(\S+) res=success`

Common Elements

Group	Label	Value	Field	Mapping	Value
\$0	type	DAEMON_START	event.deviceVendor	_stringConstant("Unix")	Unix
\$1	submessage	op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 audit=4294967295 pid=656 uid=0 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=success	event.message	submessage	op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 audit=4294967295 pid=656 uid=0 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=success
			event.deviceProduct	_stringConstant("Unix")	Unix
			event.name	type	DAEMON_START

SubMessage Elements

Group	Label	Value	Field	Mapping	Value
\$1	ver=2.8.5		event.name	\$1	ver=2.8.5
\$2	format=raw				
\$3	kernel=3.10.0-1062.el7.x86_64				
\$4	audit=4294967295				
\$5	pid=656				
\$6	uid=0				
\$7	ses=4294967295				
\$8	subj=system_u:system_r:auditd_t:s0				

```
#! On.filename.prefix=
submessage.count=1
submessage[0].messageid=DAEMON_START
submessage[0].pattern.count=1
submessage[0].pattern[0].regex=op=start (\S+) format=raw (\S+) audit=(\S+) pid=(\S+) uid=(\d+) ses=(\S+) subj=system_u:system_r:auditd_t:(\S+) res=success
submessage[0].pattern[0].fields=event.name
```

- Vlevo dole vidíme nové tokeny, které vyplývají z Regexu generovaného programem
- Takový Regex však není dostačující, ale je vhodný jako „odrazový můstek“ a navíc vygeneruje i náležitosti nutné ke tvorbě nové submessage
- Za pomoci návodů výše vytvoříme lepší Regex, který může vypadat například takto

REGULAR EXPRESSION 1 match (87 steps, 18.0ms)

```
?: / op=(\w+)\sver=(\S+)\sformat=(\w+)\skernel=(\S+)\sauid=(\d+)\spid=(\d+)\suid=(\d+)\sses= / gm
(\d+)\ssubj=(\S+)\sres=(\w+)
```

TEST STRING

```
op=start ver=2.8.5 format=raw kernel=3.10.0-1062.el7.x86_64 audit=4294967295 pid=656 uid=0 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=success
```

- V programu nyní můžeme pracovat s více daty, navíc i lépe namapovanými z hlediska Regexu
 - Jako ses nebo pid budou jen čísla
 - Odolnější vůči nechtěnému matchi – menší pravděpodobnost, že se vyskytne log, který bude mít vzor jako výše vygenerovaný, avšak bude například z jiné služby

SM.Pattern[0]			N/A	N/A	op=(\w+)\sver=(\^\s+)\sformat=(\w+)\skernel=(\^\s+)\saud=(\d+)\spid=(\d+)\suid=(\d+)\sses=(\d+)\ssubj=(\^\s+)\sres=(\w+)
Common Elements					
Group	Label	Value	Field	Mapping	
\$0	type	DAEMON_START	event.deviceVendor	stringConstant("Unix")	
\$1	submessage	op=start ver...	event.message	submessage	
			event.deviceProduct	stringConstant("Unix")	
			event.name	type	
SubMessage Elements					
Group	Value	Field	Mapping		
\$1	start	event.name	\$1		
\$2	2.8.5				
\$3	raw				
\$4	3.10.0-1062.el7.x86_64				
\$5	4294967295				
\$6	656				
\$7	0				
\$8	4294967295				
\$9	system_u:system_r:au...				
\$10	success				

- Dle návodu ArcSight SmartConnector for Linux Audit Syslog namapujeme jednotlivé tokeny na příslušná CEF pole
 - U tohoto typu patternu mapujeme dle návodu jak ver, tak kernel na jedno pole – deviceVersion
 - je možné použít funkci `__oneOf`, která namapuje může v případě absence jednoho namapovat druhé – to by ale nefungoval Regex, který jsme použili, počítá s tím, že je vše vyplněno
 - Některé části logu nejsou v návodu zmíněny – tyto je vždy vhodné nechat až na konec
 - U res je uvedeno jako mapování Event Outcome – jedná se o kategorizaci, s ní budeme pracovat později
 - S některými hodnotami dle návodu nepracujeme vůbec
 - Můžeme je namapovat na některé z custom polí, tuto činnost je vhodné provádět jako poslední

SubMessage Elements			Field	Mapping	Value
\$1	start		event.name	\$1	start
\$2	2.8.5		event.sourceProcessId	\$6	656
\$3	raw		event.sourceUserId	\$7	0
\$4	3.10.0-1062.el7.x86_64		event.deviceCustomNumber2	\$8	4294967295
\$5	4294967295		event.deviceCustomString5	\$9	system_u:system_r:auditd_t:s0
\$6	656				
\$7	0				
\$8	4294967295				
\$9	system_u:system_r:au...				
\$10	success				

```

#10n.filename.prefix=
submessage.count=1
submessage[0].messageid=DAEMON_START
submessage[0].pattern.count=1
submessage[0].pattern[0].regex=op=(\w+)\sver=(\^\s+)\sformat=(\w+)\skernel=(\^\s+)\saud=(\d+)\spid=(\d+)\suid=(\d+)\sses=(\d+)\ssubj=(\^\s+)\sres=(\w+)
submessage[0].pattern[0].fields=event.name,event.sourceProcessId,event.sourceUserId,event.deviceCustomNumber2,event.deviceCustomString5
submessage[0].pattern[0].mappings=$1$6$7$8$9

```

- Zde jsou dle návodu hodnoty, která mají jednoznačně nadefinované své pole

2.2.1 Nejednoznačné mapování

- Dle návodu se mapují hodnoty ver a kernel na jedno pole, a to deviceVersion – které z nich mapovat
 - Lze použít funkci `__oneOf(token)`, která zleva vybírá tokeny, které namapuje
 - Vhodné je ji použít, když se některá část logu občas nevyskytne, ale jinak se log neliší
 - S touto funkcí by měla přijít i úprava Regexu, a to použitím kvantifikátoru `?`, který zajistí, že jeden z tokenů může absentovat

- Namapování na custom
 - Jedna hodnota bude na poli dle návodu, druhá bude namapována na některém z custom polí včetně labelu, aby bylo jasné, o jakou hodnotu se jedná (viz níže)
- Nenamapování vůbec
 - Poslední varianta
 - Hodnota se nemapuje, pokud je naprosto zbytečná
 - ... jak definovat zbytečnou hodnotu?

2.2.2 Custom pole

- Některé hodnoty přímo v návodu byly namapovány na custom pole
- Na toto pole se mapují hodnoty, které nemají svoje pole nadefinované přímo v CEF
- U každého custom pole je nutné namapovat i jeho label – jako příklad poslouží subj a jeho hodnota
 - Dle návodu zadefinujeme hodnotu u subj na deviceCustomString5
 - Nyní namapujeme konstantu „audit“ na deviceCustomString5Label

```
submessage[0].pattern[0].fields=...,event.deviceCustomString5Label
submessage[0].pattern[0].mappings=...|stringConstant=__ („subj“)
```

- Domapujeme zbytek hodnot na pole, výsledek může vypadat například takto

Field	Mapping	Value
event.name	\$1	start
event.sourceProcessId	\$6	656
event.sourceUserId	\$7	0
event.deviceCustomNumber2	\$8	4294967295
event.deviceCustomString5	\$9	system_u:system_r:auditd_t:s0
event.deviceCustomString5Label	stringConstant("subj")	subj
event.destinationUserId	\$5	4294967295
event.deviceVersion	\$2	2.8.5
event.deviceCustomString1	\$4	3.10.0-1062.el7.x86_64
event.deviceCustomString1Label	stringConstant("kernel")	kernel

2.2.3 Povinné náležitosti každého patternu

- Každý pattern musí obsahovat name, deviceEventClassId, deviceSeverity
 - Name by měl obsahovat krátké, ale výstižné pojmenování logu
 - Podle deviceEventClassId bude následně určována kategorizace logu
 - Dle deviceSeverity určujeme závažnost logu

2.2.4 Pole deviceEventClassId

- Poslední token, který nebyl namapován, je token \$10 – udává hodnotu result, tedy, zda se operace povedla nebo ne
- Log lze kategorizovat již nyní následovně
 - categoryObject – jedná se o aplikaci, databázi, službu, síťový prvek...?
 - categoryBehavior – co vykonal objekt, nebo co se stalo ve spojitosti s ním?
 - categoryTechnique – co bylo použito za techniku u bezpečnostních událostí? DoS...?
 - categoryDeviceGroup – do jaké skupiny spadá objekt?

- categorySignificance – jedná se o pouhou informaci, stalo se něco, co vyžaduje naši pozornost?
- categoryOutcome – bylo to úspěšné, neúspěšné, nebo byl proveden jen pokus?
- Protože by to bylo značně zdlouhavé, je u každého patternu namapováno pole deviceEventClassId, podle něž se log následně kategorizuje
- Z výše uvedeného vyplývá, že deviceEventClassId by měl být jednoznačný identifikátor toho, co se stalo v rámci určování kategorizace
 - Například u HTML by to byly kódy – 200, 401, 404...
- Jak určit deviceEventClassId v tomto případě?
 - Každá submessage je definována hodnotou type, která je mapována v hlavním vzoru, a jedná se také o ID submessage – pro každou jednotlivou submessage je tato hodnota jedinečná
 - Krátce se navíc dozvíme, co se každý log „snaží říct“
 - Z výše uvedených možností kategorizace je zřejmé, že kategorizovat je možné i outcome
 - Kdyby bylo na deviceEventClassId namapována pouze hodnota type, nebylo by zřejmé, jestli to bylo úspěšné, či nikoliv – z hodnoty type to zřejmé není
 - Spojíme proto funkcí __concatenate() hodnotu type a res, jako separátor lze použít například dvojtečku, pipe, ale i mezeru
 - S tokeny v hlavním vzoru není možné pracovat v submessage, ale vzhledem k tomu, že víme, že pro každou submessage je tato hodnota jedinečná, můžeme jej nahradit konstantou

```
submessage[0].pattern[0].fields=...,event.deviceEventClassId
submessage[0].pattern[0].mappings=...|__concatenate(„DAEMON_START:“, $10)
```

SubMessage Elements		Field		Mapping	
Group	Value	Field	Mapping	Field	Value
\$1	start	event.name	\$1	start	
\$2	2.8.5	event.sourceProcessId	\$6	656	
\$3	raw	event.sourceUserId	\$7	0	
\$4	3.10.0-1062.el7.x86_64	event.deviceCustomNumber2	\$8	4294967295	
\$5	4294967295	event.deviceCustomString5	\$9	system_u:system_rauidid_t:s0	
\$6	656	event.deviceCustomStringLabel	_stringConstant("sub")	sub)	
\$7	0	event.destinationUserId	\$5	4294967295	
\$8	4294967295	event.deviceVersion	\$2	2.8.5	
\$9	system_u:system_rau...	event.deviceCustomString1	\$4	3.10.0-1062.el7.x86_64	
\$10	success	event.deviceCustomStringLabel	_stringConstant("kernel")	kernel	
		event.deviceEventClassId	_concatenate("DAEMON_START ", \$10)	DAEMON_START success	

```
submessage.count=1
submessage[0].messageid=DAEMON_START
submessage[0].pattern.count=1
submessage[0].pattern[0].regex=opt=([^\s]+)\sformat=([^\s]+)\skernel=([^\s]+)\saudit=([^\s]+)\suid=([^\s]+)\sres=([^\s]+)\sres=([^\s]+)
submessage[0].pattern[0].fields=event.name,event.sourceProcessId,event.sourceUserId,event.deviceCustomNumber2,event.deviceCustomString5,event.deviceCustomStringLabel,event.destinationUserId,event.deviceVersion,event.deviceCustomString1,event.deviceCustomStringLabel,event.deviceEventClassId
submessage[0].pattern[0].mappings=$1$6$7$8$9|_stringConstant("sub")$5$2$4|_stringConstant("kernel")_concatenate("DAEMON_START ", $10)
```

2.2.5 Pole name

- Obdobně z pole name by měl analytik pracující s Log management nástrojem zjistit, o jaký log se jedná
- V našem případě je pole name namapované již na hlavním vzoru
 - Pokud bychom jej namapovali znovu v jednotlivých patternech, pole přepíše to z hlavního vzoru
 - Je možné postupovat obdobně jako u deviceEventClassId – spojit hodnotu type a res

2.2.6 Určování severity

- Pole deviceSeverity určuje, jak závažná událost vygenerovala log
- Mapuje se jako low, medium, high, very high

- Lze mapovat rovnou jako agentSeverity, nebo jako deviceSeverity a následně na jejich hodnotách určovat agentSeverity
 - V našem případě budeme severitu mapovat podle hodnoty res
- Pro výukové účely budeme demonstrovat cestu přes deviceSeverity a určování na jejím základě
- Token \$10 nejprve namapujeme jako deviceSeverity

```
submessage[0].pattern[0].fields=...,event.deviceSeverity
submessage[0].pattern[0].mappings=...|$10
```

- Do části kódu, kde pracujeme s hlavním vzorem, vložíme následující kód

```
severity.map.high.if.deviceSeverity=failed
severity.map.low.if.deviceSeverity=success
```

- Když pole deviceSeverity bude nabývat hodnoty failed, bude agentSeverity nabývat hodnoty high, obdobně druhý řádek

3 Kategorizace logu

- Pro naše účely máme nyní normalizovaný log – hodnoty jsou namapovány na jednotlivá pole dle formátu CEF
- Každý pattern je nyní nutné kategorizovat na základě jeho deviceEventClassId
- Pro připomenutí lze log kategorizovat následovně
 - categoryObject – jedná se o aplikaci, databázi, službu, síťový prvek...?
 - categoryBehavior – co vykonal objekt, nebo co se stalo ve spojitosti s ním?
 - categoryTechnique – co bylo použito za techniku u bezpečnostních událostí? DoS...?
 - categoryDeviceGroup – do jaké skupiny spadá objekt?
 - categorySignificance – jedná se o pouhou informaci, stalo se něco, co vyžaduje naši pozornost?
 - categoryOutcome – bylo to úspěšné, neúspěšné, nebo byl proveden jen pokus?
- V našem případě již máme dvě deviceEventClassId – DAEMON_START:success a DAEMON_START:failed
- Kategorizace se píše do souborů csv, kdy jako hlavička je použito deviceEventClassId a jednotlivé kategorie
- Vytvoříme csv s názvem stejným jako deviceProduct
 - Pokud by pole v názvu obsahovalo například mezeru, dvojtečku, v názvu csv ji nahrazujeme středníkem
 - Náš kategorizační soubor se tedy bude jmenovat auditd.csv

3.1 Tvorba kategorizace

- K tvorbě kategorizace je vhodné používat soubor ArcSight Event Categorization
- Vytvoříme csv soubor a zadefinujeme hlavičku event.deviceEventClassId,set.event.categoryObject,...
- Začneme známým DAEMON_START:success – to je naše deviceEventClassId

- Dále namapujeme jednotlivé kategorie
 - Jako categoryObject /Host/Operating System
 - Jedná se o interní logování v operačním systému
 - Jako categoryBehavior /Execute/Start
 - Jedná se o start daemonu
 - Jako categoryTechnique
 - Toto pole necháme prázdné – nejedná se o žádný útok/kompromitaci
 - Jako categoryDeviceGroup /Operating System
 - Jako categorySignificance /Informational
 - Jedná se o informaci o startu daemonu
 - Jako categoryOutcome /Success
 - Spuštění proběhlo v pořádku
- Obdobně namapujeme DAEMON_START:failed
 - Změna nastane pouze u categoryOutcome /Failure

Shrnutí a závěr

V případě, že jsou normalizované veškeré patterny a všechny logy jsou kategorizovány, je možné přistoupit k otestování parseru a následnému nasazení.