

Trabajo Practico final: Deteccion de anomalias con Machine Learning

Agustin Miguel Payaslian (#96.885)
agustin.payaslian@gmail.com

Augusto Arturi (#97498)
turitoh@gmail.com

Estanislao Ledesma (#96622)
estanislaomledesma@gmail.com

2do. Cuatrimestre de 2018

66.69 Criptografia y Seguridad Informatica

Facultad de Ingeniería, Universidad de Buenos Aires



Índice

1. Introduccion	5
1.1. Random Forest	5
2. Primer caso	6
2.1. Analisis exploratorio	6
2.2. Entrenamiento	9
3. Segundo caso	10
3.1. Estudio del trafico	10
3.2. Entrenamiento	14
4. Repositorio	17
5. Conclusion	17

Resumen

En el siguiente trabajo practico se explicaran mediante dos casos practicos la utilizacion de algoritmos y metodologias de machine learning aplicadas a problemas de la seguridad informatica.

1. Introduccion

Se cree que las tecnologías de Machine Learning pueden aportar mucho valor al mundo de la seguridad. Una de las aplicaciones directas conocidas de estas técnicas es la **detección de anomalías de tráfico de red**. Esta hace referencia a comportamientos no esperables de acuerdo al funcionamiento habitual de la red. Por ejemplo, situaciones en las que se genere gran cantidad de tráfico repentino o muy diferente al habitual. El arte de estas técnicas consiste precisamente en determinar qué es diferente, habitual y disponer de los mecanismos adecuados para detectarlo.

1.1. Random Forest

Es un algoritmo de Machine Learning de aprendizaje supervisado. Es una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. Se caracteriza por:

- Ser uno de los algoritmos de aprendizaje más certeros que hay disponible. Para un set de datos lo suficientemente grande produce un clasificador muy certero.
- Correr eficientemente en bases de datos grandes.
- Manejar cientos de variables de entrada sin excluir ninguna.
- Dar estimaciones de qué variables son importantes en la clasificación.
- Tener un método eficaz para estimar datos perdidos y mantener la exactitud cuando una gran proporción de los datos está perdida.
- Sirve tanto para problemas de clasificación como de regresion
- Computar las proximidades entre los pares de casos que pueden usarse en los grupos, localizando valores atípicos, o (ascendiendo) dando vistas interesantes de los datos.
- Ofrecer un método experimental para detectar las interacciones de las variables.

Bagging en general implica aplicar el mismo clasificador n veces y luego promediar sus resultados para el obtener el resultado final. El problema es que aplicar el mismo clasificador n veces al mismo set de datos dara n veces el mismo resultado. Es por esto que bagging siempre se usa en conjunto con bootstrapping.

Bootstrapping consiste en tomar una muestra del set de entrenamiento del mismo tamaño del set de entrenamiento pero con reemplazo. Es decir que un dato puede estar varias veces en la muestra. Como cada clasificador no ve el total de registros del set de entrenamiento la técnica de bagging disminuye notablemente las posibilidades de caer en overfitting, ya que ninguno de los n clasificadores individuales puede sobreajustar al set de entrenamiento completo.

Bagging puede usarse con cualquier algoritmo de clasificación o regresión pero es muy popular con árboles de decisión, en donde da origen al algoritmo de Random Forests, el cual es uno de los algoritmos más populares en clasificación porque suele predecir buenos resultados para la mayoría de los set de datos, lo cual hemos comprobado a lo largo de este trabajo práctico.

En definitiva un RF es un conjunto de árboles de decisión en donde cada árbol usa un bootstrap del set de entrenamiento y un cierto conjunto de atributos tomados al azar. Los RF tienen, en general, dos hiperparámetros: la cantidad de árboles a crear y la cantidad de atributos de cada uno.

El primero (cantidad de árboles) no es muy complejo ya que en general a mayor cantidad de árboles mejor funciona el ensamble pero hay que tener en cuenta que llegado un punto la mejora es prácticamente inapreciable y perdemos performance, por esto en general usaremos una cantidad de árboles que podamos manejar y que nos den el mejor resultado posible o bien que no tengan una diferencia apreciable con el uso de mayor cantidad de árboles.

En la figura 5 observamos cómo a partir de los 20 árboles el error se va prácticamente a cero y la mejora a más árboles es inapreciable.

El segundo hiper-parámetro: la cantidad de atributos por árbol es el mas critico y necesita ser encontrado mediante la toma de subconjuntos de atributos tomados al azar lo cual ayuda a determinar cuales son los atributos que realmente importan para la clasificación promediando su precisión.

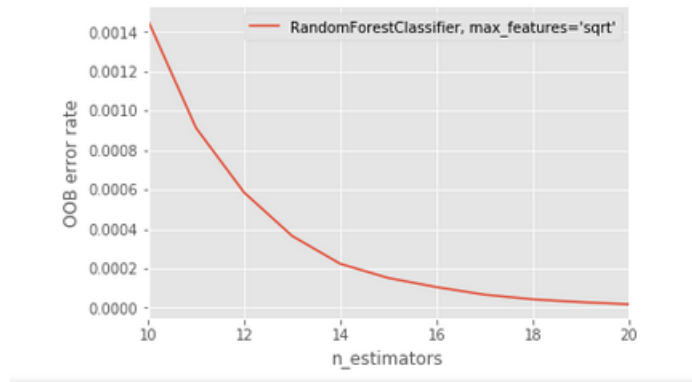


Figura 1

2. Primer caso

En este primer ejemplo práctico se utilizó el dataset **KDD99**. Este dataset fue creado para el “Third International Knowledge Discovery and Data Mining Tools Competition”. El objetivo de esta competencia era construir un detector de intrusiones de la red (network intrusion detector), es decir un modelo predictivo que sea capaz de distinguir entre conexiones “malas” y conexiones “buenas”. Este contiene una gran cantidad de tráfico, dentro del cual se encuentran intrusiones simuladas en un ambiente de red militar. Cada conexión tiene una categoría, ya sea normal, o el nombre específico del ataque.

Los ataques se encuentran dentro de estas categorías:

- DOS: denegación de servicio (Denial Of Service)
- R2L: acceso no autorizado desde un usuario remoto.
- U2R: acceso no autorizado a privilegios de superusuario.
- probing: vigilancia y otro tipo de sondeos. Ej: escaneo de puertos

2.1. Analisis exploratorio

Este dataset está dividido en 2 archivos. El archivo “10perfect” es el que se utiliza para obtener la información para entrenar el algoritmo y el archivo “corrected” que es el que se utiliza para verificar la precisión de la predicción.

Este provee información de valor por medio de sus features, los cuales serán trabajados como parte del preprocesado para luego entrenar al algoritmo deseado. La información básica provista por el mismo es la siguiente:

Feature name	Description
duration	length (number of seconds) of the connection
protocol_type	type of the protocol, e.g. tcp, udp, etc
service	network service on the destination, e.g., http, telnet, etc.
src_bytes	number of data bytes from source to destination
dst_bytes	number of data bytes from destination to source
flag	normal or error status of the connection
land	1 if connection is from/to the same host/port; 0 otherwise
wrong_fragment	number of “wrong” fragments
urgent	number of urgent packets
hot	number of “hot” indicators
num_failed_logins	number of failed login attempts

logged_in	1 if successfully logged in; 0 otherwise
num_compromised	number of “compromised” conditions
root_shell	1 if root shell is obtained; 0 otherwise
su_attempted	1 if “su root” command attempted; 0 otherwise
num_root	number of “root” accesses
num_file_creations	number of file creation operations
num_shells	number of shell prompts
num_access_files	number of operations on access control files
num_outbound_cmds	number of outbound commands in an ftp session
wrong_fragment	number of “wrong” fragments
is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise
is_guest_login	1 if the login is a “guest” login; 0 otherwise
count	number of connections to the same host as the current connection in the past two seconds
error_rate	% of connections that have “SYN” errors
reror_rate	% of connections that have “REJ” errors
same_srv_rate	% of connections to the same service
diff_srv_rate	% of connections to different services
srv_count	number of connections to the same service as the current connection in the past two seconds
srv_error_rate	% of connections that have “SYN” errors
srv_reror_rate	% of connections that have “REJ” errors
srv_diff_host_rate	% of connections to different hosts

El feature **label** corresponde al tipo de ataque al que pertenece el trafico y es uno de los mas importantes del set de datos, ya que este valor es el cual queremos clasificar.

En la Figura 1 se observa todos los tipos de ataques junto con su respectiva cantidad de apariciones.

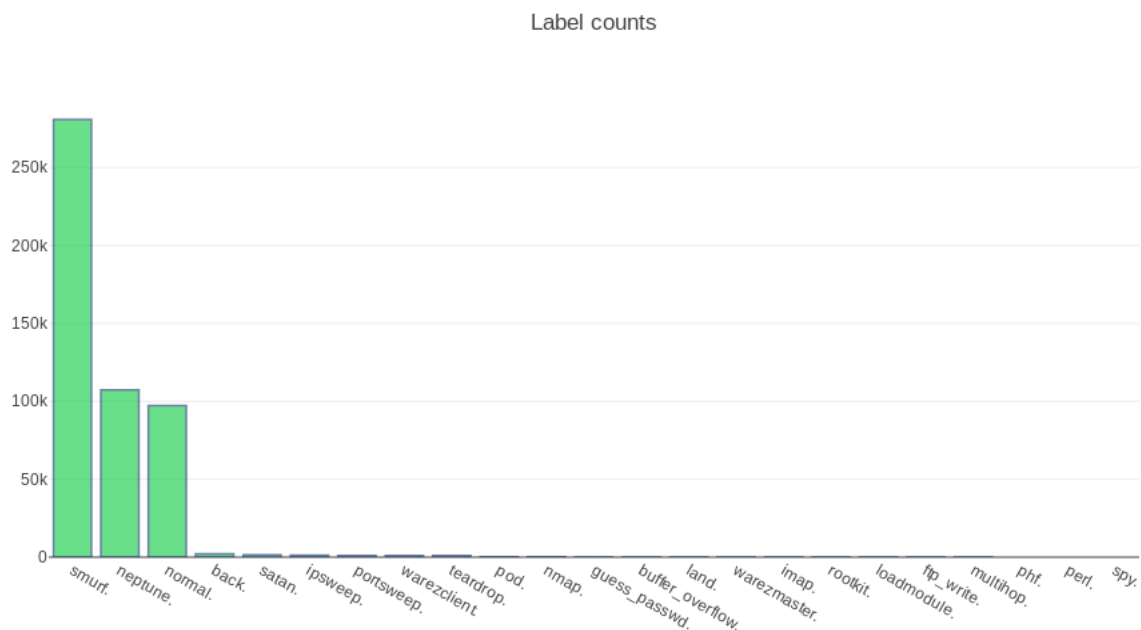


Figura 2

Para una mejor categorizacion, los ataques fueron agrupados por las 4 categorias mencionadas anteriormente

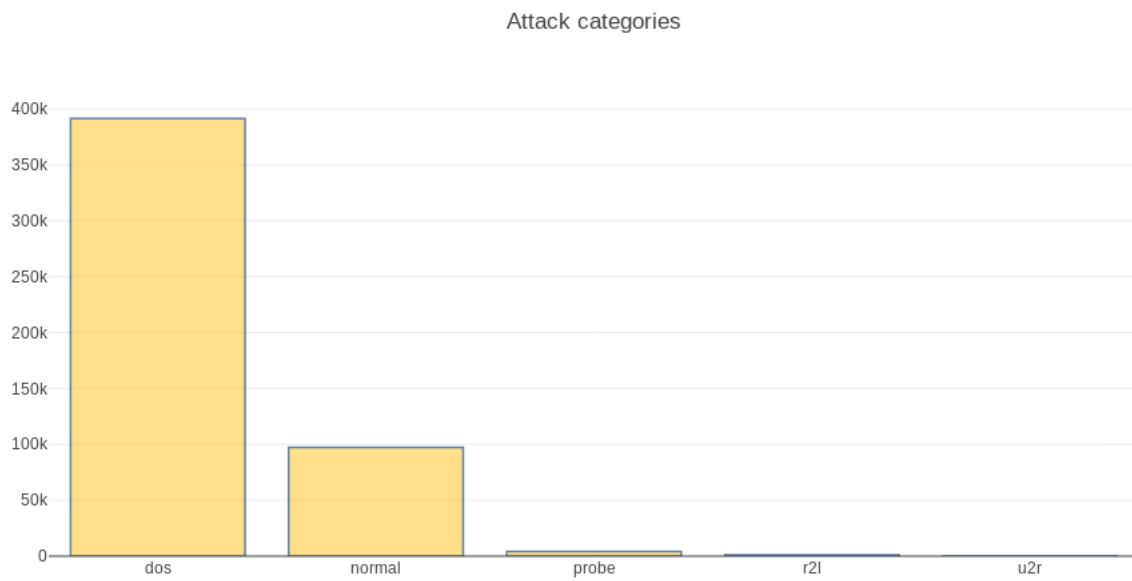


Figura 3

La solución que se plantea en este primer caso implica analizar el tráfico que pertenece a ataques de DOS y el tráfico normal.

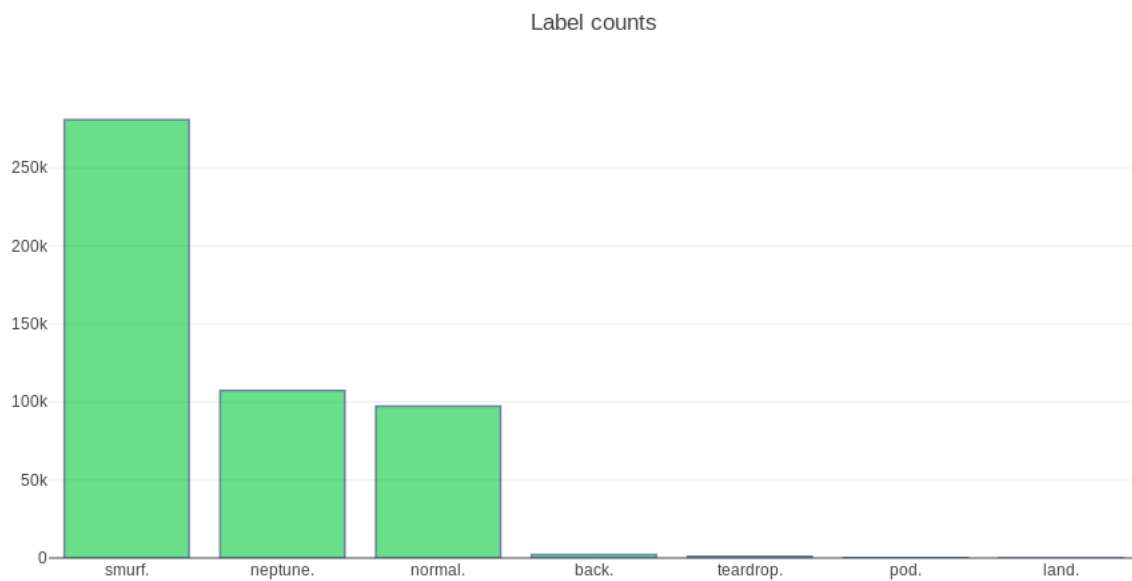


Figura 4

Listing 1: De todas formas, las filas eliminadas solo representan un 1 % del set de datos

```
1.0 - len(dfAnalysis.loc[(dfAnalysis.cat_attack == "dos") |
(dfAnalysis.cat_attack == "normal")]) / len(dfAnalysis)
```

Out: 0.01069792579667661

Finalmente para terminar el preprocesado se acomodo la informacion de una forma tal que le sea mas util a nuestro algoritmo para eso usamos funciones de normalizacion de los datos y funciones de *one hot encoding* la cual consiste endividir el atributo en tantas columnas como valores posibles puede tener y usar cada columna como un dato binario indicando si el atributo toma o no dicho valor.

2.2. Entrenamiento

Se trabajo con un set de entrenamiento y un set de test. La idea consisten en entrenar a nuestro algoritmo con el set de entrenamiento y luego lo aplicaremos al set de test. De esta forma, los datos para los cuales queremos probar el algoritmo nunca fueron vistos por el mismo, lo cual permite saber si el modelo fue capaz de generalizar correctamente. Para poder validar el modelo necesitamos dividir el set de entrenamiento original en dos: un set de entrenamiento y un set de validación. La idea es entrenar el modelo con el set de entrenamiento y luego probarlo con el set de validación a efectos de encontrar los mejores hiper-parámetros pertenecientes propios a cada algoritmo. El set de validación utilizado fue de un 25 % del set de entrenamiento.

Utilizamos el algoritmo de **random forest** para realizar la predicción. Para ello con la información obtenida del archivo "10perfect" entrenamos el algoritmo y luego predecimos los ataques para el archivo "corrected". Para verificar la eficacia del algoritmo comparamos los valores obtenidos por la predicción con los que ya se encontraban en el archivo. Al realizar la comparación obtuvimos el resultado siguiente:

Accuracy: 0.99469

Es decir el algoritmo diferenció con una precisión del 99% el trafico normal, del que se trataba de un ataque DOS.

Los features que mas importaron a la hora de la prediccion fueron los siguientes:

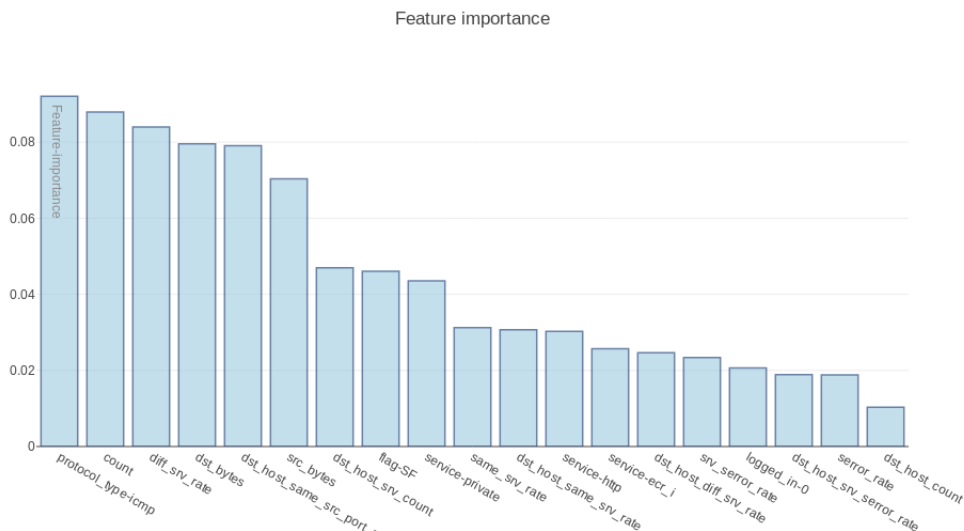


Figura 5

Es decir, que a partir de la figura anterior se puede interpretar que features aportan mas valor para generar el arbol de decision que clasifique al random forest.

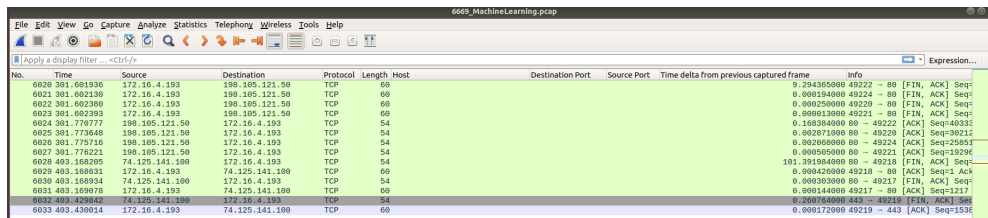
3. Segundo caso

En este segundo ejemplo práctico, se decidió trabajar sobre otro dataset, esta vez con un enfoque más realista el cual implica realizar un estudio previo de los datos y el problema para luego entrenar al algoritmo.

En primer lugar se tomo un ejemplo del sitio web ^[1] este resultado de gran interés ya que aquí se suben archivos de tráfico de red con malware, en el cual quien analiza el mismo debe encontrar en qué lugares existe una anomalía.

3.1. Estudio del trafico

A continuación se explica como fue estudiado el tráfico de red de este problema con la herramienta Wireshark.



No.	Time	Source	Destination	Protocol	Length	Host	Destination Port	Source Port	Time delta from previous captured frame	Info
6029	301.681336	172.16.4.193	198.105.121.50	TCP	60				0.294305000	49222 → 80 [FIN, ACK] Seq=
6021	301.682130	172.16.4.193	198.105.121.50	TCP	60				0.000194000	49224 → 80 [FIN, ACK] Seq=
6022	301.682380	172.16.4.193	198.105.121.50	TCP	60				0.000250000	49220 → 80 [FIN, ACK] Seq=
6023	301.682393	172.16.4.193	198.105.121.50	TCP	60				0.000033000	49221 → 80 [FIN, ACK] Seq=
6024	301.778777	198.105.121.50	172.16.4.193	TCP	54				0.168384000	80 → 49222 [ACK] Seq=40332
6025	301.773648	198.105.121.50	172.16.4.193	TCP	54				0.002671000	80 → 49220 [ACK] Seq=38211
6026	301.773716	198.105.121.50	172.16.4.193	TCP	54				0.002668000	80 → 49224 [ACK] Seq=75851
6027	301.776221	198.105.121.50	172.16.4.193	TCP	54				0.000505000	80 → 49221 [ACK] Seq=19296
6028	403.168205	74.125.141.100	172.16.4.193	TCP	54				191.391946000	80 → 49219 [FIN, ACK] Seq=
6029	403.168631	172.16.4.193	74.125.141.100	TCP	60				0.000426000	49218 → 80 [ACK] Seq=1 Ack=
6030	403.168934	74.125.141.100	172.16.4.193	TCP	54				0.000303000	80 → 49217 [FIN, ACK] Seq=
6031	403.169076	172.16.4.193	74.125.141.100	TCP	60				0.000144000	49217 → 80 [ACK] Seq=1217
6032	403.429842	74.125.141.100	172.16.4.193	TCP	54				0.268764800	443 → 49219 [FIN, ACK] Seq=
6033	403.430814	172.16.4.193	74.125.141.100	TCP	60				0.000172000	49219 → 443 [ACK] Seq=1535

Figura 6

En la figura simplemente se observa la cantidad de paquetes que se intercambio durante la toma de trafico, siendo esta 6033, como medida de dificultad para entender a que problema uno se enfrenta. Luego se utiliza el filtro **http.request** para verificar a que host se comunicó el cliente durante toda la conexión

No.	Time	Source	Destination	Protocol	Length	Host	Destination Port	Source Port	Time delta from previous captured frame	Info
141	0.209972	172.16.4.193	66.127.163.72	HTTP	151	www.msfnets.com			0.000246800	GET /ncs1.txt HTTP/1.1
157	0.215515	172.16.4.193	204.79.197.200	HTTP	559	www.bing.com			0.000245800	GET / HTTP/1.1
170	0.562039	172.16.4.193	204.79.197.200	HTTP	695	www.bing.com			0.126321800	GET /s/a/hp18.png HTTP/1.1
184	0.662372	172.16.4.193	204.79.197.200	HTTP	590	www.bing.com			0.026450000	GET /fd/s/a/hp/bing.svg HT
185	0.696347	172.16.4.193	204.79.197.200	HTTP	880	www.bing.com			0.027975000	GET /fd/1s/1716-19453800
196	0.768507	172.16.4.193	204.79.197.200	HTTP/XL	391	www.bing.com			0.000249000	POST /fd/1s/1isp.aspx? HTTP
201	0.773449	172.16.4.193	204.79.197.200	HTTP	1272	www.bing.com			0.000013000	GET /rns/RingCore_Bandier/
227	0.875215	172.16.4.193	204.79.197.200	HTTP	771	www.bing.com			0.020493000	GET /rns/rns328answers328
228	0.875937	172.16.4.193	204.79.197.200	HTTP	943	www.bing.com			0.000722000	GET /rns/Framework/c, n/
230	0.886308	172.16.4.193	204.79.197.200	HTTP	60	www.bing.com			0.000260000	POST /rewardsapp/rewarder/
258	0.958021	172.16.4.193	204.79.197.200	HTTP	80	www.bing.com			0.000243000	POST /rewardsapp/rewarder/
263	0.985500	172.16.4.193	204.79.197.200	HTTP	987	www.bing.com			0.000245000	GET /rns/Framework/c, n/
293	0.066384	172.16.4.193	204.79.197.200	HTTP	749	www.bing.com			0.000610000	GET /sz/hprichbg/rs/Villier
296	0.100648	172.16.4.193	204.79.197.200	HTTP	715	www.bing.com			0.033870000	GET /sa/R_1_2_016224/HPIn
298	0.126480	172.16.4.193	204.79.197.200	HTTP	824	www.bing.com			0.000021000	GET /sa/A/m/Chino.a.r.r.1e4

Figura 7

Se encontraron sitios como los mostrados en la siguiente Figura junto con su cantidad de apariciones.

Host	
1e46ba9c0151d4d34a7939daabd778ad.clo.footprintdns.com	2.0
2.bing.com	1.0
3a0849dbc3c36a673eb2ddd2fcf0494a.clo.footprintdns.com	2.0
40bbdaf00bf29a6114a5019e397a2a15.clo.footprintdns.com	2.0
6b8960d1b061131b015f93f32d0a56f4.clo.footprintdns.com	2.0
a4.bing.com	1.0
api.blockcypher.com	2.0
da6ab9a9cf82c8f939081a82c7d90031.clo.footprintdns.com	2.0
e623e8223493b6793a476840214720b1.clo.footprintdns.com	2.0
fdownload2.macromedia.com	1.0
p27dokhpz2n7nvgr.1jw2lx.top	24.0
report.footprintdns.com	2.0
retrotip.visionurbana.com.ve	1.0
spotsbill.com	2.0
tse1.mm.bing.net	4.0
tyu.benme.com	15.0
www.bing.com	78.0
www.google-analytics.com	4.0
www.homelmpovement.com	17.0
www.msftncsl.com	1.0

Figura 8: Lista de dominios generada con Python

Al observar estos dominios es importante la experiencia del analista, como primera medida verificamos las terminaciones, en la lista se identifica uno con final **.top**, googleando el dominio encontramos lo siguiente:

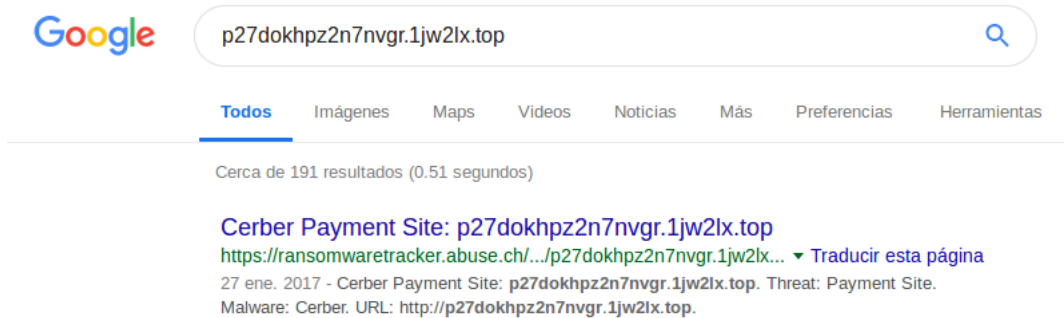


Figura 9

Aquí se identifica el malware, el cual se trata de un **Ransomware** del tipo **Cerber**, conociendo esto se podría prohibir el tráfico a este sitio, sin embargo además de ser algo relativamente simple, no produce ningún tipo de solución ya que el virus podría cambiar de página, por ende queda pendiente encontrar algún comportamiento anómalo en el tráfico, de esta manera se continúa con el análisis utilizando un plugin de Wireshark llamado Snort, de aquí se pretende obtener alguna información del exploit utilizado e información de la IP de la cual este proviene.

Src IP	SPort	Dst IP	DPort	Pr	Event Message
104.211.160.15	80	172.16.4.193	49190	6	GPL WEB_CLIENT web bug 0x0 gif attempt
138.91.83.37	80	172.16.4.193	49184	6	GPL WEB_CLIENT web bug 0x0 gif attempt
104.28.18.74	80	172.16.4.193	49195	6	ET CURRENT_EVENTS Evil Redirector Leading to EK Jul 12 2016
104.28.18.74	80	172.16.4.193	49195	6	ETPRO CURRENT_EVENTS Evil Redirect to RIG-v EK Oct 24 2016
172.16.4.193	49202	194.87.234.129	80	6	ET CURRENT_EVENTS RIG EK URI struct Oct 24 2016 (RIG-v)
194.87.234.129	80	172.16.4.193	49202	6	ETPRO CURRENT_EVENTS RIG EK Landing Pre-filter (Rig-v)
194.87.234.129	80	172.16.4.193	49202	6	ET CURRENT_EVENTS RIG EK Landing Sep 12 2016 T2
194.87.234.129	80	172.16.4.193	49202	6	ETPRO CURRENT_EVENTS RIG EK Landing Nov 30 2016 (RIG-v)
194.87.234.129	80	172.16.4.193	49209	6	ETPRO CURRENT_EVENTS RIG/Sundown/Xer EK Payload Jul 06 2016

Figura 10

Acorde a lo investigado RIG Exploit Kit es uno de los más activos y utilizados actualmente para realizar este tipo de ataques, tomamos la ip 194.87.234.129 y procedemos a buscarla en el tráfico con Wireshark mediante el filtro

No.	Time	Source	Destination	Protocol	Length	Host	Destination Port	Source Port	Time delta from previous captured frame	Info
2852	94.997940	172.16.4.193	194.87.234.129	HTTP	605	tyu.benne.com			0.000470800	GET /?ct=Vivaldi&biw=Viva
2896	95.005679	172.16.4.193	194.87.234.129	HTTP	593	tyu.benne.com			0.000798000	GET /?q=zn_QMvXcJwDQdofGMv
2936	95.370478	172.16.4.193	194.87.234.129	HTTP	611	tyu.benne.com			0.010705000	POST /?biw=Mozilla.102kd74
2937	95.380031	172.16.4.193	194.87.234.129	HTTP	612	tyu.benne.com			0.009553000	POST /?oq=CEh3h8_svK7pSP1
3108	116.415349	172.16.4.193	194.87.234.129	HTTP	754	tyu.benne.com			6.685430000	GET /?biw=SeaMonkey.105qj6
3109	116.480279	172.16.4.193	194.87.234.129	HTTP	735	tyu.benne.com			0.070930000	GET /?biw=Amaya.126qv100.4
3178	118.125512	172.16.4.193	194.87.234.129	HTTP	524	tyu.benne.com			0.000246000	GET /?ct=Mozilla&tuif=3375
3186	118.608947	172.16.4.193	194.87.234.129	HTTP	530	tyu.benne.com			0.000245000	GET /?yus=SeaMonkey.115uv6
5269	163.080771	172.16.4.193	194.87.234.129	HTTP	605	tyu.benne.com			0.000313000	GET /?ct=Vivaldi&biw=Viva
5271	163.081270	172.16.4.193	194.87.234.129	HTTP	593	tyu.benne.com			0.000249000	GET /?q=zn_QMvXcJwDQdofGMv
5297	163.438605	172.16.4.193	194.87.234.129	HTTP	632	tyu.benne.com			0.009512000	POST /?br_fl=3395&tuif=546
5298	163.445102	172.16.4.193	194.87.234.129	HTTP	632	tyu.benne.com			0.006497000	POST /?br_fl=1929&oq=ZaCm3
5520	184.817181	172.16.4.193	194.87.234.129	HTTP	765	tyu.benne.com			0.002059000	GET /?tuif=2138&br_fl=1786
5521	184.857556	172.16.4.193	194.87.234.129	HTTP	768	tyu.benne.com			0.040375000	GET /?oq=pLLYG0AS3jxbTfNg
5613	187.251521	172.16.4.193	194.87.234.129	HTTP	533	tyu.benne.com			0.000245000	GET /?br_fl=5844&tuif=5862

Identification: 0x05f5 (1525)
 Flags: 0x4000, Don't fragment
 Time to live: 128
 Protocol: TCP (6)
 Header checksum: 0x9509 [validation disabled]
 [Header checksum status: Unverified]
 Source: 172.16.4.193
 Destination: 194.87.234.129
 Transmission Control Protocol, Src Port: 49202, Dst Port: 80, Seq: 1, Ack: 1, Len: 551
 Hypertext Transfer Protocol
 [truncated]GET /?ct=Vivaldi&biw=Vivaldi.95ec76.40617c5k7&oq=h8fItKeRvawGjRaFwCwinyYdeAngQ8_qtiEKBzBKfgZ6D-hyMZh1z6LRVvQ42w&tuif=2320&q=wh7QMvXcJwDNFYbGMvrER6NbNknQA8KPxP2_drZdZqxKgn120b5UUSk6FqCEh
 Accept: text/html, application/xhtml+xml, */*
 Referer: http://www.homeimprovement.com/remodeling-your-kitchen-cabinets.html
 Accept-Language: en-US
 User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
 Accept-Encoding: gzip, deflate

Figura 11

Como conclusion, entendemos que el exploit es inyectado mediante el sitio www.homeimprovement.com el cual esta en el listado de la Figura 8.

Finalmente en la siguiente imagen se pueden ver algunos paquetes correspondientes al tráfico generado durante la infección. A partir de la línea 3977 el comportamiento no sigue los patrones de tráfico habituales en una red. Se pueden observar paquetes UDP que en un breve intervalo de tiempo (menos de 1 segundo) proceden de una misma IP de origen y que se dirige a diferentes (y en muchos casos consecutivas) IP de destino públicas. Además, los puertos origen y destino son fijos.

No.	Time	Source	Destination	Protocol	Length	Host	Destination Port	Source Port	Time delta from previous captured frame	Info
3977	129.352311	172.16.4.193	90.2.1.3	UDP	67		6892	58978	0.000013000	58978 → 6892 Len=25
3978	129.352323	172.16.4.193	90.2.1.4	UDP	67		6892	58978	0.000012000	58978 → 6892 Len=25
3979	129.352334	172.16.4.193	90.2.1.5	UDP	67		6892	58978	0.000011000	58978 → 6892 Len=25
3980	129.352345	172.16.4.193	90.2.1.6	UDP	67		6892	58978	0.000010000	58978 → 6892 Len=25
3981	129.352358	172.16.4.193	90.2.1.7	UDP	67		6892	58978	0.000013000	58978 → 6892 Len=25
3982	129.352371	172.16.4.193	90.2.1.8	UDP	67		6892	58978	0.000013000	58978 → 6892 Len=25
3983	129.352382	172.16.4.193	90.2.1.9	UDP	67		6892	58978	0.000011000	58978 → 6892 Len=25
3984	129.352394	172.16.4.193	90.2.1.10	UDP	67		6892	58978	0.000012000	58978 → 6892 Len=25
3985	129.352406	172.16.4.193	90.2.1.11	UDP	67		6892	58978	0.000012000	58978 → 6892 Len=25
3986	129.352417	172.16.4.193	90.2.1.12	UDP	67		6892	58978	0.000011000	58978 → 6892 Len=25
3987	129.352428	172.16.4.193	90.2.1.13	UDP	67		6892	58978	0.000011000	58978 → 6892 Len=25
3988	129.352440	172.16.4.193	90.2.1.14	UDP	67		6892	58978	0.000012000	58978 → 6892 Len=25
3989	129.352451	172.16.4.193	90.2.1.15	UDP	67		6892	58978	0.000011000	58978 → 6892 Len=25
3990	129.352465	172.16.4.193	90.2.1.16	UDP	67		6892	58978	0.000014000	58978 → 6892 Len=25
3991	129.352477	172.16.4.193	90.2.1.17	UDP	67		6892	58978	0.000012000	58978 → 6892 Len=25

Figura 12

Es este el cual se detecta cómo el comportamiento anómalo, el cual será trasladado al algoritmo de machine learning para que pueda prever este tipo de situaciones y notificarlo.

En conclusion del analisis del trafico:

IP y Puerto	Sitio Web	Info
104.28.18.74 port 80	www.homeimprovement.com	Sitio web comprometido
139.59.160.143 port 80	retrotip.visionurbana.com.ve	Puerta de redireccionamiento
194.87.234.129 port 80	tyu.benme.com	Rig EK
198.105.121.50 port 80	p27dokhpz2n7nvgr.1jw2lx.top	Cerber ransomware
90.2.1.0 to 90.2.1.31 (90.2.1.0/27) port 6892	-	Cerber post-infection UDP traffic
90.3.1.0 to 90.3.1.31 (90.3.1.0/27) port 6892	-	Cerber post-infection UDP traffic
91.239.24.0 to 91.239.25.255 (91.239.24.0/23) port 6892	-	Cerber post-infection UDP traffic

3.2. Entrenamiento

Basándose en esta simple observación definida anteriormente, se centra el funcionamiento del algoritmo en la creación de un feature o característica que cuenta el número de flujos en un intervalo de tiempo parametrizable en segundos, que desde una misma IP origen se conectan a diferentes IP destino públicas, con tráfico de un determinado protocolo y usando un cierto puerto destino. Lo que se espera es que para el tráfico normal este valor sea bajo y se consideran anomalías aquellos que tengan valores altos. Esto se trata tan sólo de un simple ejemplo y se podrían definir más características. De hecho, definir características acertadas (además de la adecuada elección del dataset) es el arte que puede llevar a que los algoritmos de aprendizaje sirvan realmente en el mundo real.

El primer desafío que se encontró en comparación del caso anterior, fue en la propia generación de los datos, esta vez solamente teníamos un archivo .pcap con el tráfico de red, el cual a su vez era escaso, ya que comentamos de la Figura, solo tenía 6033 líneas. De nuestro estudio en la temática sabemos lo importante que es tener datos de valor, y cuanto más grande sea el dataset aún mejor ya que brindarle más información al algoritmo implica más robustez y menos overfit, es decir, que la solución se vuelva muy particular y deje de generalizar para otro tipo de casos, lo cual queremos evitar. Por ende para ampliar el set de datos, se tomaron otros ejemplos de tráfico de ransomware, donde se detectó que el patrón seguía siendo el mismo, a su vez tener en el mismo dataset infecciones de fechas temporales distintas, con IPs (origen y destino) distintas de ataque, e incluso puertos distintos resulta valioso.

A su vez también debemos brindarle a este información de tráfico que sea “normal” sin malware, el cual es extremadamente útil para enfatizar cuando existe o no una anomalía, para generarlo lo hicimos con una de las propias computadoras de los integrantes del grupo capturando paquetes mediante Wireshark.

Realizado esto se prosiguió generando un dataframe que tenga el contenido de toda la información recolectada, para eso utilizamos **pandas** la cual es una librería de python para el análisis y creación de estructuras de datos, herramienta muy utilizada en el ámbito de machine learning.

A continuación, se creó un nuevo feature (columna) para categorizar al tráfico si este era como un ataque o como normal, esto fue realizado teniendo en cuenta todo el análisis explicado anteriormente. Este es muy necesario ya que será nuestro parámetro para más tarde establecer las comparaciones con las predicciones que hagamos a partir de nuestro entrenamiento con **Random Forest**.

El segundo paso fundamental para resolver el problema, fue trabajar la creación de un nuevo dataframe que contenga la información de valor en términos de flujos de datos, por esta razón es esencial el preprocesamiento del dataset, ya que cuando el algoritmo fue entrenado sin procesamiento alguno las predicciones acertadas fueron muy bajas. Para esto se decidió agrupar bajo las siguientes condiciones, las cuales fueron detectadas con el análisis del Ransomware:

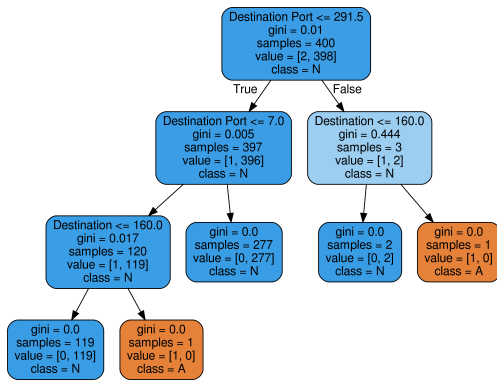
- Por tiempo, estimado en 1 segundo, esta parámetro puede ser variable para obtener resultado distintos, encontramos que una gran cantidad de paquetes UDP fueron enviados en este lapso de tiempo

- Por IP origen, se encontró un comportamiento anómalo cuando la IP source se mantenía constante pero la IP destino era cambiante en un breve lapso de tiempo, el cual es cuando actuaba Cerber.
- Por puertos, durante el ataque los puertos se mantenían constantes.

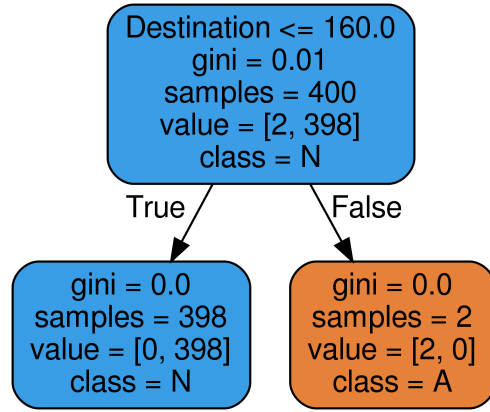
	Source	Protocol	label	Source Port	Destination Port	DateTime	Destination
0	10.1.19.102	UDP	A	62093	6892	2017-01-19 15:09:59	319
1	10.1.19.102	UDP	N	62093	6892	2017-01-19 15:10:00	1
2	10.1.19.103	UDP	A	63462	6892	2017-01-19 18:27:34	319
3	10.1.19.103	UDP	N	63462	6892	2017-01-19 18:27:35	1
4	172.16.4.1	DHCP	N	67	68	2017-01-27 19:53:08	1
5	172.16.4.1	DHCP	N	67	68	2017-01-27 19:53:11	1
6	172.16.4.1	DHCP	N	67	68	2017-01-27 19:55:12	1
7	172.16.4.1	DHCP	N	67	68	2017-01-27 19:56:20	1
8	172.16.4.1	DNS	N	53	50309	2017-01-27 19:53:56	1
9	172.16.4.1	DNS	N	53	50439	2017-01-27 19:54:41	1
10	172.16.4.1	DNS	N	53	51561	2017-01-27 19:54:10	1
11	172.16.4.1	DNS	N	53	52455	2017-01-27 19:53:56	1
12	172.16.4.1	DNS	N	53	52925	2017-01-27 19:54:09	1
13	172.16.4.1	DNS	N	53	54166	2017-01-27 19:54:43	1
14	172.16.4.1	DNS	N	53	54415	2017-01-27 19:54:09	1
15	172.16.4.1	DNS	N	53	56288	2017-01-27 19:55:27	1
16	172.16.4.1	DNS	N	53	56384	2017-01-27 19:54:28	1
17	172.16.4.1	DNS	N	53	56630	2017-01-27 19:54:12	1
18	172.16.4.1	DNS	N	53	56975	2017-01-27 19:53:52	1
19	172.16.4.1	DNS	N	53	57124	2017-01-27 19:55:27	1

Figura 13

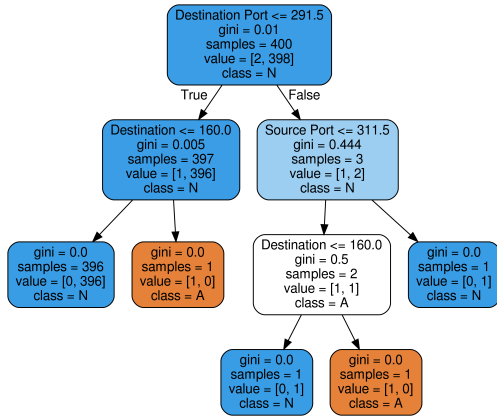
A continuación una serie de ejemplos de árboles de decisión generados por nuestro programa:



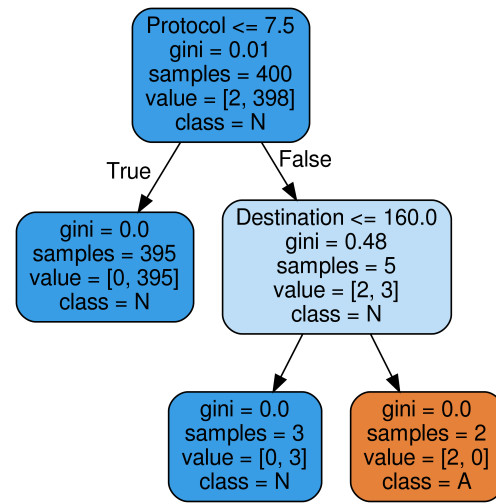
(a)



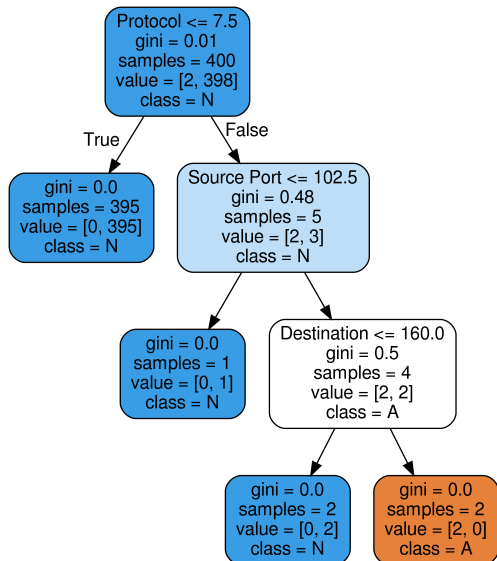
(b)



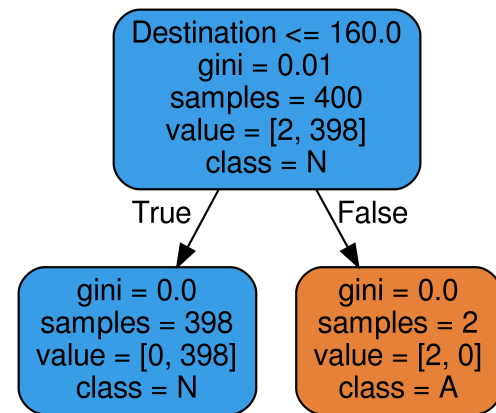
(c)



(d)



(e)



(f)

Figura 14: Árboles de decisión

Finalmente se logró clasificar por medio de este algoritmo a los datos con una precisión del **0.99253**

Para probar estos resultados se dividió nuestro set de datos en 60 % de entrenamiento y 40 % de prueba. Donde se encontró que las anomalías fueron detectadas todas correctamente lo cual era el objetivo principal de nuestro problema.

4. Repositorio

Link al repositorio: <https://github.com/AugustoArturi/Anomalias>

Archivos:

- Detail_ML_Solution.ipynb, **caso 1**
- Ransomware.ipynb, **caso 2**

5. Conclusion

Los algoritmos de machine learning pueden ser herramientas muy utiles para utilizar en el area de seguridad informática. En este trabajo se demostró que estos algoritmos pueden servir para detectar ataques que afectan a la disponibilidad del servicio (DOS), como también para los que afectan la integridad de los datos (Ransomware).

También hay que tener en cuenta que es muy importante para que este tipo de herramientas resulten útiles para su objetivo, es necesario hacer un análisis previo de los datos (preprocesamiento) para asegurarse de que los mismos puedan aplicarse correctamente.

Además, hay que considerar que cada set de datos , sobre todo de paquetes provenientes de internet, es diferente, con lo cual las métricas pueden funcionar bien para algunos casos y no para otros. A esto se suma que los ataques suelen evolucionar con el correr de los tiempos, con lo cual, los parámetros para detectarlos pueden quedar obsoletos en poco tiempo.

Queda para futuros trabajos investigar en que otros pilares de la seguridad informática se pueden usar estas herramientas, aplicando otros métodos y parámetros.

Referencias

- [1] <http://www.malware-traffic-analysis.net/>.
- [2] <https://scikit-learn.org/>
- [3] https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.html
- [4] <https://kdd.ics.uci.edu/databases/kddcup99/task.html>
- [5] Apunte Luis Argerich 75.06 Organización de Datos - FIUBA