Analisis de URL's malignas y benignas

Jose Luis Osorio Quintero & Melisa Garcia

3 de mayo de 2017

Introduccion

Las URL's contienen una cierta cantidad de características que nos permiten analizar y determinar si estas poseen contenido malicioso [1]. A continuación, se mostrarán datasets de URL's malignas y benignas, mostrando características de dos capas (red y aplicación). Se hará una limpieza, normalización, binarizacion, correlación y exploración de datos para cada uno de los datasets obtenidos. A continuación, se mostrarán los datasets y las características que cada uno posee.

Capa de aplicacion

- **longitud de la URL:** representa el tamaño total de la URL.
- **características especiales:** representa el total de características extrañas que contiene la URL, como lo son "/", "%", "#", "&", ".", "=", entre otros.
- **HTTPHeader_charset:** conjunto de caracteres con los que se identifica el sitio web, como lo son ANSI, ISO-8859-1, UTF8 entre otros.
- **HTTPHeader_server:** esta característica es el servidor en el que fue montada la URL.
- **HTTPHeader_content_length:** representa el tamaño del contenido de la cabecera http.
- **whois_regDate:** indica cuando el servidor del sitio web fue registrado
- **whois_Updated_date:** indica cuando el servidor fue actualizado
- **whois_country:** indica el país donde se encuentra el servidor del sitio web
- **whois_statePro:** indica el estado del sitio web
- **whois_Domain:** indica el dominio del sitio web

Capa de red

- **tcp_conversation_exchange:** cuenta la cantidad de paquetes que hay entre el honeypot y el sitio web para el protocolo TCP.
- **dist_remote_tcp_port:** número total de puertos distintos a los puertos TCP.
- **remote_ips:** número distinto de direcciones IP conectadas al honeypot.
- **pkt_without_dns:** almacena en un arreglo todos los paquetes que no son DNS.
- **app_bytes:** número de bytes de la capa de aplicación envía por el honeypot hacia el sitio web, no se incluyen los datos de los servidores DNS.
- **udp_packets:** número de paquetes UDP, no se incluyen los datos de los DNS.
- **tcp_urg_packet:** número de paquetes TCP con la bandera de URG.
- **source_app_packets:** número de paquetes enviados por el honeypot hacia el servidor remoto.
- **remote_app_packets:** volumen en bytes de la comunicación del servidor web al honeypot.
- **duration:** tiempo de duración de la página web.
- **avg_local_pkt_rate:** promedio de paquetes IP por segundo (paquetes enviados sobre la duración).
- **avg_remote_pkt_rate:** promedio de paquetes IP por segundo (paquetes enviados sobre la duración).
- **app_packets:** número de paquetes IP incluidos los del servidor DNS.
- **dns_query_times:** lista de capas de DNS queries.

Dataset de URL's malignas

Para obtener este dataset se examinaron 4 fuentes de URL las cuales fueron categorizadas por identificadores como M0, M1, M2, M3, M4 teniendo un total de 587, 3, 17, 75 y 35279 respectivamente. Posterior a su obtención y la verificación del estado de la URL, se procedió a pasar estos datos por algoritmos desarrollados en Python que cumplían la función de extraer las características necesarias según la investigación base [1, 2].

Matriz de aplicación

Para esta matriz se obtuvieron 12 variables donde se analizaron 2185 URL's entre los M0, M1, M2, M3 y M4 dando como resultado 2100 valores.

```
## # A tibble: 4 × 12
##     URL    A1    A2         A4                    A5       A6    A7    A8
##   <chr> <int> <int>      <chr>                 <chr>    <chr> <chr> <chr>
## 1  M0_9    43    10 ISO-8859-1              DOSarrest no-cache  None    us
## 2 M0_15    26     8      utf-8                  nginx     None  None  None
## 3 M0_16    37     9 ISO-8859-1                  nginx     None   162    CN
## 4 M0_21    28     9      UTF-8 Apache/2.2.15 (CentOS)     None  None    RU
## # ... with 4 more variables: A9 <chr>, A10 <chr>, A11 <chr>, A12 <chr>
```

Matriz de red

Para esta matriz se obtuvieron 16 variables donde se analizaron 1288 URL's entre los M0, M1, M2, M3 y M4 dando como resultado 755 valores.

```
## # A tibble: 4 × 16
##     URL    N1    N2    N3    N4    N5    N6    N7    N8    N9   N10
##   <chr> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1  M0_2     0     0     0     0     0     0     8    11  1434   648
## 2  M0_3     0     0     0     0     0     0     8    11  1406   620
## 3  M0_4     0     0     0     0     0     0     8    10  1316   592
## 4  M0_5     5     1     2   636     0     0    11    13  2095  1092
## # ... with 5 more variables: N11 <dbl>, N12 <chr>, N13 <chr>, N14 <int>,
## #   N15 <int>
```

Dataset de URL's benignas

Para obtener este dataset se examinó 1 fuente de URL's la cual fue categorizada por el identificador B0 teniendo un total de 28324 respectivamente. Posterior a su obtención y la verificación del estado de la URL, se procedió a pasar estos datos por algoritmos desarrollados en Python que cumplían la función de extraer las características necesarias según la investigación base [1,2].

Matriz de aplicación

Para esta matriz se obtuvieron 12 variables donde se analizaron 25171 URL's del dataset B0 dando como resultado 13794 valores.

```
## # A tibble: 4 × 12
##     URL    A1    A2         A4     A5                                  A6
##   <chr> <int> <int>      <chr>  <chr>                               <chr>
## 1  B0_1    56     8 iso-8859-1 Apache                               None
## 2  B0_3    63    12      UTF-8 Apache                               None
## 3  B0_4    54     9      UTF-8 Apache                               None
```

```
## 4  B0_7    63    10      UTF-8 Apache no-cache, must-revalidate, max-age=0
## # ... with 6 more variables: A7 <chr>, A8 <chr>, A9 <chr>, A10 <chr>,
## #   A11 <chr>, A12 <chr>
```

Matriz de red

Para esta matriz se obtuvieron 16 variables donde se analizaron 2323 URL's del dataset B0 dando como resultado 1742 valores.

```
## # A tibble: 4 × 16
##     URL    N1    N2    N3    N4    N5    N6    N7    N8    N9   N10    N11
##   <chr> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>  <int>
## 1  B0_1     0     0     0     0     0     0     0     2   124     0     NA
## 2  B0_2     2     0     2   128     0     0     8     9  1023   590     NA
## 3  B0_3    48     0     1  3840     0     0    52    51 52729  4156 680743
## 4  B0_4     4     1     3   280     0     0     6     7   542   422     NA
## # ... with 4 more variables: N12 <chr>, N13 <chr>, N14 <int>, N15 <chr>
```

Limpieza de datos

La limpieza de datos es una estandarización y eliminación de datos que proporcionan ruido a la investigación. En esta limpieza se hace un análisis exhaustivo de cada una de las columnas, se le proporciona el formato a cada una, se eliminan los valores 'NA' y se separan algunos valores en múltiples columnas.

Igualar matrices y sus valores

Como la extracción de características de las capas de red y aplicación son totalmente independientes en los Script de Python. Estos no van a arrojar los mismos resultados para las mismas URL's. Es por eso que se procede a igualar la información de ambas matrices con ayuda de los identificadores de cada URL. La matriz de aplicación y la matriz de red no tienen la misma cantidad de variables, es por eso que se procede a poner variables vacías en la matriz de aplicación con el fin de que no se pierda información al usar el algoritmo de comparación que provee la librería *compare*. Posterior a la comparación y tras obtener las matrices con iguales URL's se procede a borrar las columnas vacías.

```
library(compare)

# ASIGNACION DE COLUMNAS VACIAS PARA IGUALAR A LA MATRIZ DE RED
app_maligno$A13 <- 'NA'
app_maligno$A14 <- 'NA'
app_maligno$A15 <- 'NA'
app_maligno$A16 <- 'NA'

app_benigno$A13 <- 'NA'
app_benigno$A14 <- 'NA'
app_benigno$A15 <- 'NA'
app_benigno$A16 <- 'NA'

# COMPARAR MATRICES DE RED Y DE APP
comparison_maligno <- compare(app_maligno,red_maligno,allowAll=TRUE)
comparison_benigno <- compare(app_benigno,red_benigno,allowAll=TRUE)

# CREACION DE LAS MATRICES
```

```r
app_maligno <- data.frame(comparison_maligno$tM)
red_maligno <- data.frame(comparison_maligno$tC)

app_benigno <- data.frame(comparison_benigno$tM)
red_benigno <- data.frame(comparison_benigno$tC)

#ELIMINACION DE COLUMNAS NULAS DE APP
app_maligno <- app_maligno[0:12]
app_benigno <- app_benigno[0:12]
```

Asignación de nombres de las columnas

Se procede a poner el nombre a cada una de las matrices según lo mencionado en cada una de las capas.

```r
# Asignacion de nombres
names(app_maligno) <- c("URL",
                        "URL_LENGTH",
                        "NUMBER_SPECIAL_CHARACTERS",
                        "CHARSET",
                        "SERVER",
                        "CACHE_CONTROL",
                        "CONTENT_LENGTH",
                        "WHOIS_COUNTRY",
                        "WHOIS_STATEPROV",
                        "WHOIS_REGDATE",
                        "UPDATE_DATE",
                        "WHITIN_DOMAIN")

names(app_benigno) <- c("URL",
                        "URL_LENGTH",
                        "NUMBER_SPECIAL_CHARACTERS",
                        "CHARSET",
                        "SERVER",
                        "CACHE_CONTROL",
                        "CONTENT_LENGTH",
                        "WHOIS_COUNTRY",
                        "WHOIS_STATEPROV",
                        "WHOIS_REGDATE",
                        "UPDATE_DATE",
                        "WHITIN_DOMAIN")

names(red_maligno) <- c("URL",
                        "TCP_CONVERSATION_EXCHANGE",
                        "DIST_REMOTE_TCP_PORT",
                        "REMOTE_IPS",
                        "APP_BYTES",
                        "UDP_PACKETS",
                        "TCP_URG_PACKETS",
                        "SOURCE_APP_PACKETS",
                        "REMOTE_APP_PACKETS",
                        "SOURCE_APP_BYTES",
                        "REMOTE_APP_BYTES",
                        "DURATION",
```

```
                          "AVG_LOCAL_PKT_RATE",
                          "AVG_REMOTE_PKT_RATE",
                          "APP_PACKETS",
                          "DNS_QUERY_TIMES")

names(red_benigno) <- c("URL",
                          "TCP_CONVERSATION_EXCHANGE",
                          "DIST_REMOTE_TCP_PORT",
                          "REMOTE_IPS",
                          "APP_BYTES",
                          "UDP_PACKETS",
                          "TCP_URG_PACKETS",
                          "SOURCE_APP_PACKETS",
                          "REMOTE_APP_PACKETS",
                          "SOURCE_APP_BYTES",
                          "REMOTE_APP_BYTES",
                          "DURATION",
                          "AVG_LOCAL_PKT_RATE",
                          "AVG_REMOTE_PKT_RATE",
                          "APP_PACKETS",
                          "DNS_QUERY_TIMES")
```

Formato de las variables

Formato para matrices de red

```
red_benigno$URL <- as.factor(red_benigno$URL)
red_benigno$TCP_CONVERSATION_EXCHANGE <- as.numeric(red_benigno$TCP_CONVERSATION_EXCHANGE)
red_benigno$DIST_REMOTE_TCP_PORT <- as.numeric(red_benigno$DIST_REMOTE_TCP_PORT)
red_benigno$REMOTE_IPS <- as.numeric(red_benigno$REMOTE_IPS)
red_benigno$APP_BYTES <- as.numeric(red_benigno$APP_BYTES)
red_benigno$UDP_PACKETS <- as.numeric(red_benigno$UDP_PACKETS)
red_benigno$TCP_URG_PACKETS <- as.numeric(red_benigno$TCP_URG_PACKETS)
red_benigno$SOURCE_APP_PACKETS <- as.numeric(red_benigno$SOURCE_APP_PACKETS)
red_benigno$REMOTE_APP_PACKETS <- as.numeric(red_benigno$REMOTE_APP_PACKETS)
red_benigno$SOURCE_APP_BYTES <- as.numeric(red_benigno$SOURCE_APP_BYTES)
red_benigno$REMOTE_APP_BYTES <- as.numeric(red_benigno$REMOTE_APP_BYTES)
red_benigno$DURATION <- as.numeric(red_benigno$DURATION)
red_benigno$APP_PACKETS <- as.numeric(red_benigno$APP_PACKETS)
red_benigno$DNS_QUERY_TIMES <- as.numeric(red_benigno$DNS_QUERY_TIMES)

red_benigno <- red_benigno[,-13]
red_benigno <- red_benigno[,-13]

red_maligno$URL <- as.factor(red_maligno$URL)
red_maligno$TCP_CONVERSATION_EXCHANGE <- as.numeric(red_maligno$TCP_CONVERSATION_EXCHANGE)
red_maligno$DIST_REMOTE_TCP_PORT <- as.numeric(red_maligno$DIST_REMOTE_TCP_PORT)
red_maligno$REMOTE_IPS <- as.numeric(red_maligno$REMOTE_IPS)
red_maligno$APP_BYTES <- as.numeric(red_maligno$APP_BYTES)
red_maligno$UDP_PACKETS <- as.numeric(red_maligno$UDP_PACKETS)
red_maligno$TCP_URG_PACKETS <- as.numeric(red_maligno$TCP_URG_PACKETS)
red_maligno$SOURCE_APP_PACKETS <- as.numeric(red_maligno$SOURCE_APP_PACKETS)
red_maligno$REMOTE_APP_PACKETS <- as.numeric(red_maligno$REMOTE_APP_PACKETS)
```

```
red_maligno$SOURCE_APP_BYTES <- as.numeric(red_maligno$SOURCE_APP_BYTES)
red_maligno$REMOTE_APP_BYTES <- as.numeric(red_maligno$REMOTE_APP_BYTES)
red_maligno$DURATION <- as.numeric(red_maligno$DURATION)
red_maligno$APP_PACKETS <- as.numeric(red_maligno$APP_PACKETS)
red_maligno$DNS_QUERY_TIMES <- as.numeric(red_maligno$DNS_QUERY_TIMES)

red_maligno <- red_maligno[,-13]
red_maligno <- red_maligno[,-13]

app_benigno$CONTENT_LENGTH <- as.numeric(app_benigno$CONTENT_LENGTH)
app_maligno$CONTENT_LENGTH <- as.numeric(app_maligno$CONTENT_LENGTH)
```

Formato para matrices de aplicacion

Separación de valores de la columna cache_control (Matrices de aplicación)

Se separan los elementos de esta columna para una adecuada binarizacion de los valores.

```
library(reshape2)
library(splitstackshape)
library(dplyr)

ccb <- cSplit(melt(app_benigno[c(1,6)], id.vars ="URL"), "value", ",", "long")
head(ccb,5)
```

```
##        URL      variable     value
## 1:    B0_1 CACHE_CONTROL      NONE
## 2:   B0_10 CACHE_CONTROL      NONE
## 3:  B0_100 CACHE_CONTROL      NONE
## 4: B0_1000 CACHE_CONTROL   PRIVATE
## 5: B0_1000 CACHE_CONTROL MAX-AGE=0
```

```
ccm <- cSplit(melt(app_maligno[c(1,6)], id.vars ="URL"), "value", ",", "long")
head(ccm,5)
```

```
##       URL      variable           value
## 1: M0_100 CACHE_CONTROL        NO-CACHE
## 2: M0_100 CACHE_CONTROL MUST-REVALIDATE
## 3: M0_100 CACHE_CONTROL       MAX-AGE=0
## 4: M0_101 CACHE_CONTROL        NO-CACHE
## 5: M0_101 CACHE_CONTROL MUST-REVALIDATE
```

Posterior a la separación se aplica el proceso de binarizacion a estas nuevas variables.

```
library(data.table)
setDT(ccb)[, c(levels(ccb$value), "value") :=
            c(lapply(levels(value), function(x) as.integer(x == value)), .(NULL))]

ccb <- ccb[,-2]

setDT(ccm)[, c(levels(ccm$value), "value") :=
             c(lapply(levels(value), function(x) as.integer(x == value)), .(NULL))]

ccm <- ccm[,-2]
```

A continuación, se hace una agrupación de los valores por la URL para que estos queden contemplados en una sola fila.

```
ccb <- ccb[, lapply(.SD,sum), by=URL]

ccm <- ccm[, lapply(.SD,sum), by=URL]
```

Por último, se hace una unión de estos valores con las matrices de aplicación originales, obteniendo así la columna de cache control binarizada

```
app_benigno <- merge(app_benigno, ccb, all = TRUE)
app_benigno <- app_benigno[,-6]

app_maligno <- merge(app_maligno, ccm, all = TRUE)
app_maligno <- app_maligno[,-6]
```

Obtención del nombre de servidor

En esta parte solo se desea obtener el nombre del servidor para poder binarizar adecuadamente esta variable.

```
# SEPARACION DE SERVER (SOLO EL NOMBRE)
app_benigno$NAME_SERVER <- sapply(strsplit(app_benigno$SERVER,"/"), `[`, 1)
#app_benigno$OTHER_SERVER_ATTRIBUTES <- sapply(strsplit(app_benigno$SERVER,"/"), `[`, 2)
app_benigno <- app_benigno[,-5]

app_maligno$NAME_SERVER <- sapply(strsplit(app_maligno$SERVER,"/"), `[`, 1)
#app_maligno$OTHER_SERVER_ATTRIBUTES <- sapply(strsplit(app_maligno$SERVER,"/"), `[`, 2)
app_maligno <- app_maligno[,-5]
```

Normalización de matrices de aplicacion

La normalización permite que los valores puedan ser comparables unos a otros. En este caso se usa la librería scale para poder lograr normalizar los valores numéricos de las matrices. Pero, antes de la normalización se debe garantizar la eliminación de valores constantes dentro de las matrices.

Matriz de aplicacion benigna

```
# ELIMINACION DE CONSTANTES
library(caret)
app_benigno.n <- app_benigno

is.constant = function(x) all(x[1] == x)
constantes = sapply(app_benigno.n, is.constant)
app_benigno.n = app_benigno.n[,!constantes]

# NORMALIZACION DE DATOS NUMERICOS EN LA MATRIZ APP BENIGNO
app_benigno.n <- app_benigno
app_benigno.n[,c(2,3,5)] <- scale(app_benigno[,c(2,3,5)])
```

Matriz de aplicacion maligna

```
# ELIMINACION DE CONSTANTES
app_maligno.n <- app_maligno
constantes = sapply(app_benigno.n, is.constant)
app_benigno.n = app_benigno.n[,!constantes]

# NORMALIZACION DE DATOS NUMERICOS EN LA MATRIZ APP MALIGNO
app_maligno.n <- app_maligno
app_maligno.n[,c(2,3,5)] <- scale(app_maligno[,c(2,3,5)])
```

Binarización de matrices de aplicación

Las matrices de aplicación contienen valoren como el nombre del servidor, el charset y otro tipo de variables no numéricas que impiden un entrenamiento adecuado para los algoritmos.

Matriz de aplicación benigna

```
# BINARIZAR VARIABLES CATEGORICAS
dummifica2 = dummyVars( ~ ., data = app_benigno.n[2:ncol(app_benigno.n)])
app_benigno.n = predict(dummifica2, newdata = app_benigno.n)
URL <- app_benigno$URL

app_benigno.n <- data.frame(URL,app_benigno.n)
```

Matriz de aplicación maligna

```
# BINARIZACION DE VARIABLES CATEGORICAS
dummifica2 = dummyVars( ~ ., data = app_maligno.n[2:ncol(app_maligno.n)])
app_maligno.n = predict(dummifica2, newdata = app_maligno.n)
URL <- app_maligno$URL

app_maligno.n <- data.frame(URL,app_maligno.n)
```

Normalización de matrices de red

```
# NORMALIZACION DE DATOS DE RED MALIGNOS Y BENGNINOS
red_benigno.n <- red_benigno
red_benigno.n[,2:ncol(red_benigno)] <- scale(red_benigno.n[,2:ncol(red_benigno)])

red_maligno.n <- red_maligno
red_maligno.n[,2:ncol(red_benigno)] <- scale(red_benigno.n[,2:ncol(red_benigno)])
```