

Analiza oraz wykorzystanie
podatności w zapytaniach
HTTP pomiędzy aplikacją
mobilną i API

API – co to jest?

API – Application Programming Interface – zestaw reguł definiujący komunikację pomiędzy systemami komputerowymi oraz pomiędzy systemem komputerowym a człowiekiem. API składa się z trzech głównych elementów:

- **Procedury** – nazywane również rutynami. Procedury odnoszą się do konkretnych zadań lub funkcji wykonywanych przez program. Na przykład Twitter dostarcza API dla programistów, dzięki któremu mają oni wgląd w dane w celach analitycznych.
- **Protokoły** – protokół jest formatem używanym do wymiany danych pomiędzy aplikacjami.
- **Narzędzia** – można je przyrównać do segmentów, z których da się tworzyć nowe programy.

API to kod, który kontroluje wszystkie punkty dostępu aplikacji lub serwera.

Do serwerów tych można wysłać zewnętrzne żądanie, które to API obsługuje jako tłumacz, który prześle zapytania do aplikacji oraz pozwoli odesłać jej informację zwrotną.

Podczas tworzenia REST API, do komunikacji wykorzystuje się metody HTTP, których łącznie jest 9:

- GET
- POST
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE
- PATCH
- HEAD

Cykl pracy z API możemy określić następująco:

- 1) Klient przygotowuje zapytanie w postaci odpowiedniego adresu (endpoint),
- 2) Klient wysyła przygotowane zapytanie (request),
- 3) System otrzymuje zapytanie klienta i przygotowuje odpowiedź (response),
- 4) System zwraca odpowiedź na zapytanie klienta,
- 5) Klient otrzymuje i przetwarza odpowiedź.

POST – służy tworzeniu i przesłaniu nowych danych. W tym przypadku konieczne jest już stworzenie ciała (body), w którym prześlemy dane do naszego REST API. W odpowiedzi powinien zostać zwrócony nagłówek HTTP Location z adresem do nowo utworzonego zasobu.

Najczęściej, jeśli chodzi o metodę POST to musimy zbudować body. Budowa body może też być przeniesiona do zmiennej ale czasem wygodniej jest zrobić tak:

```
Invoke-RestMethod http://localhost:7000/api/reservation/2 -Method  
POST -Body (@{ "UserId"=1; "Date"="2019-02-02" } | ConvertTo-Json) -  
ContentType "application/json"
```

O czym warto pamiętać? Tworząc obiekty używając symboli (, @ i { należy w body je przekonwertować na JSON-a, używając składni
| ConvertTo-Json

Endpoint – W architekturze REST punkt końcowy do którego można się odwołać. Częstym przypadkiem jest odwoływanie się do endpointa aby pobrać dane.

Przykładem może być wywołanie endpointa stanowiącego adres:

https://bawim.tk/get_app_version

parametry dla narzędzia curl (metoda POST):

<https://gist.github.com/subfuzion/08c5d85437d5d4f00e58>

`?_method=POST`

Jeśli administrator zablokuje np. endpoint POST, możemy sprawdzić czy na serwerze nie są zdefiniowane własne metody. Czasami przekazując metodę POST jako parametr GET (`?_method=POST`) uda się obejść blokowanie tej metody przez serwer.

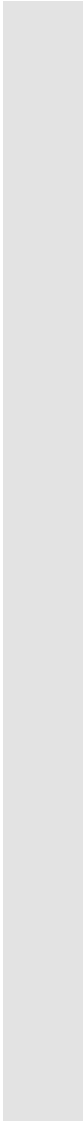

`curl https://bawim.tk/?_method=POST`

Aplikacja komunikuje się z serwerem przez REST API.

Posiada wadę projektową, ponieważ korzysta z jednego globalnego klucza API.

Aby przesłać klucz używając narzędzia curl do komendy należy dopisać:

```
-d key=Re2LPGUMEKa34ik1uhHOBMoc
```

Klucz ten może być ukryty w aplikacji poprzez zaciemnienie i obfuskację kodu, jednak będzie on widoczny przy analizie zapytań http i zrzutu pamięci programu.

Endpointy aplikacji

```
https://bawim.tk/get_app_version
https://bawim.tk/user/{user_id}
https://bawim.tk/user_data/{user_id}
https://bawim.tk/login/{username}
https://bawim.tk/reset_password/{user_id}
https://bawim.tk/set_password/{user_id}/{token}
https://bawim.tk/set_premium_account/{user_id}
```

W rzeczywistej sytuacji dokumentacja tego API nie byłaby dostępna. W takim wypadku, aby odnaleźć dostępne endpointy możemy:

- wyszukać napisy zawierające adresy API poleceniem strings w pliku wykonywalnym aplikacji (jeśli aplikacja została poddana obfuskacji to możemy nic nie znaleźć),
- wyszukać napisy zawierające adresy API poleceniem strings w pliku ze zrzutem pamięci programu,
- bezpośrednio analizować zapytania http do serwera używając:
 - emulatora w przypadku aplikacji na androida (httptoolkit - <https://httptoolkit.tech/blog/inspect-any-android-apps-http>),
 - burp suite w przypadku aplikacji webowej,
 - proxy w przypadku aplikacji desktopowej (fiddler - <https://www.telerik.com/fiddler>),
- skanować prawdopodobne ścieżki na serwerze metodą bruteforce.

<https://github.com/microsoft/avml>

```
m@m-VirtualBox:~/Desktop$ sudo strings cprogram
/lib64/ld-linux-x86-64.so.2
libc.so.6
__stack_chk_fail
sleep
__cxa_finalize
__libc_start_main
GLIBC_2.2.5
GLIBC_2.4
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u+UH
napif
napif
https://H
bawim.tkH
/get_appH
_inff
/user/{uH
ser_id}
dH34%(
[]A\A]A^A_
:*3$"
GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.8060
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
cprogram.c
__FRAME_END__
__init_array_end
_DYNAMIC
__init_array_start
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_

```

C cprogram.c X

home > m > Desktop > C cprogram.c > ...

```
1  #include <unistd.h>
2
3  int main() {
4      char napis1[] = "napis1";
5      char napis2[] = "napis2";
6      char napis3[] = "https://bawim.tk/get_app_info";
7      char napis4[] = "https://bawim.tk/user/{user_id}";
8      sleep(1000);
9      return 0;
10 }
11
12
13
14
```

```
m@m-VirtualBox: ~/Desktop
m@m-VirtualBox:~/Desktop$ ./cprogram

m@m-VirtualBox:~/Desktop$ sudo ./avml d.dump
m@m-VirtualBox:~/Desktop$
m@m-VirtualBox:~/Desktop$ sudo strings d.dump | grep https://bawim
char napis3[] = "https://bawim.tk/get_app_info";
char napis4[] = "https://bawim.tk/user/{user_id}";
char napis3[] = "https://bawim.tk/get_app_info";
char napis3[] = "https://bawim.tk/get_app_info";
char napis4[] = "https://bawim.tk/user/{user_id}";
char napis3[] = "https://bawim.tk/get_app_info";
char napis4[] = "https://bawim.tk/user/{user_id}";
char napis3[] = "https://bawim.tk/get_app_info";
char napis4[] = "https://bawim.tk/user/{user_id}";
https://bawim.tk/get_app_info
https://bawim.tk/user/{user_id}

```

```

m@m-VirtualBox:~/Desktop$ gcc cprogram.c -o cprogram
m@m-VirtualBox:~/Desktop$ sudo strings cprogram | grep napis1
m@m-VirtualBox:~/Desktop$ cat cprogram
@@@XX  PP--==`h--==888 XXXDDS+td888 P+td DDQ+tdR+td--==PP/lib64/ld-linux-x86-64.so.2GNU+GNU==E+g<E
                                                                 B\p+GNU

H+=2/H+=Y/H+R/H9+th/H+t+H+=)/H+5"/H)H+H+?H+H+H+th+.H+fD+=.u+UH+=.H+t+H+PTL+H+
                                                                 H+=.  ++++++.]++++
H+U+H+/get_appH+M+E+_inff+E+H+E+H+U+H+/user/{uH+ser_id}H+E+H+U++++H+u+dh34%(t<++++f.++AWL+=s+AVI++AUI++ATA++UH+-d
++\++++,++++4zRx
                ++/D$4++++0FJ
+
+?+:*3$"\++++t++++ +++++E+C
D+8++++F+I+E +E(+D0+H8+G@n8A0A(B BB+`++`
+++++O++++
+
+?0(+ +++++o8++++o++++&++++o+=0@GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.08X|++& 8
h
(
P`++ H +=+=+=?@+@+
                ++! 7+@F+=m`y+=++++L!++++=+=+=+= ++?+
                + w @3@+:Vu@+ @+ +0e++@{+/++@+i++@+ +"crtstuff.cderegister_tm
rame_dummy__frame_dummy_init_array_entrycprogram.c__FRAME_END____init_array_end_DYNAMIC__init_array_start__GNU_EH_FRAME_HD
_fail@@GLIBC_2.4__libc_start_main@@GLIBC_2.2.5__data_start__gmon_start____dso_handle_IO_stdin_used__libc_csu_init__bss_sta
_2.2.5.symtab.strtab.shstrtab.interp.note.gnu.property.note.gnu.build-id.note.ABI-tag.gnu.hash.dynsym.dynstr.gnu.version.g
.eh_frame.init_array.fini_array.dynamic.data.bss.comment#886XX$I|| W+++++a
+ + D+H +++++=-++?+@0
                ++i++++q++++o&&~++++o88+h+h+B((+ 0+PP+`` +++%+++
m@m-VirtualBox:~/Desktop$ sudo strings cprogram | grep bawim
bawim.tkh

```