Fakulta informačních technologií
Vysoké učení technické v Brně

Přenos dat, počítačové sítě a protokoly

L2 MitM

23. dubna 2017 Daniel Dušek

Obsah

1 Úvod

Cílem projektu bylo naimplementovat sadu aplikací jejichž použitím lze demonstrovat možnost provedení úspěšného útoku typu MitM na L2 vrstvě. Tyto aplikace se konkrétně soustředí na provedení MitM útoku s využitím protokolů ARP a NDP.

Tato dokumentace obsahuje zjednodušený popis zmíněných protokolů, se speciální pozorností věnovanou způsobům jak využít jejich zprávy ke zmatení zařízení připojených k síti a realizovat s jejich pomocí MitM útok.

Dále je zde popsána implementace jednotlivých aplikací a v závěru i demonstrace použití aplikací v prostředí domácí virtualizaované sítě.

2 Útoky typu Man in the Middle

Útoky typu MitM - *Man-in-the-Middle* - jsou útoky typické tím, že se na síti kromě normálních komunikujících aktérů vyskytuje ještě jeden, či více dalších aktérů - tzv. *útočníci*, kteří nepozorovaně zasahují do komunikace. Typickou motivací útočníků může být zisk dat posílaných po síti, která nejsou určena pro ně, a to, pokud možno, nepozorovaně. Další motivací pak může být například narušení provozu na síti.

Pro zisk těchto informací je ve většině případů potřeba nějakým způsobem zmást komunikující stanice na síti - například tak, že jim útočící stanice tvrdí, že je někdo jiný. Díky tomu pak může přebírat komunikaci těchto stanic a provádět nad ní škodlivou činnost (odposlech, modifikace, zahození a další).

Stanice útočníka realizuje matení stanic na síti tak, že podvrhuje - zpravidla opakovaně po časovém intervalu - zprávy, které mezi sebou stanice využívají k tomu, aby se v síti dokázaly lokalizovat a posléze spojit. Často zneužívanými protokoly jsou potom *ARP* a *NDP*, jejichž využití je popsáno níže v této práci.

3 ARP - Address Resolution Protocol

Protokol ARP je využíván k překladu L3 adres (například IP) na L2 adresy (typicky MAC). Stanice v IPv4 síti využívají tohoto protokolu k zjištění, kde se nachází ostatní stanice, se kterými mají zájem komunikovat.

Protokol obsahuje celkem 4 typy zpráv - ARP request (dotaz na L2 adresu stanice jejíž L3 adresu stanice, která požadavek posílá zná), ARP reply (odpověď na ARP request obsahující L3 adresu a její L2 umístění), a další dvě zprávy pro reverzní rezoluci, které nejsou pro kontext této dokumentace důležité. Protokol nepodporuje žádnou vestavěnou formu ověření původu zpráv, čehož implementované aplikace využívají.

3.1 ARP Tabulka

Uvažme scénář, kdy stanice S1 chce komunikovat se stanicí S2, ale zná pouze její L3 adresu, například `169.254.179.143`. S1 pošle ARP request na MAC broadcast adresu `FF:FF:FF:FF:FF:FF`, ve které se ptá *„Kdo má 168.254.179.143“*. Dostane se jí odpovědi, kde S2 odpovídá, že se nachází na L2 adrese `08:00:27:56:97:53`. S1 si ukládá tuto informaci do své lokální cache, zvané ARP tabulka. Při příštím pokusu o komunikaci se stanicí na L3 adrese bude stanice vycházet z informace, kterou má uloženou v cache paměti (je-li stále přítomna) a bude vědět, že má komunikovat s konkrétní L2 adresou.

Obsah lokální ARP cache se dá vypsat (min. pro Linux a Windows) pomocí příkazu `arp -a`, vypsaný obsah tabulky může vypadat například takto:

```
pds2 (169.245.74.244) at 08:00:27:a9:1b:db [ether] on eth1
pds3 (169.245.174.244) at 08:00:27:cc:dd:db [ether] on eth1
```

3.2 Otrávení ARP Tabulky

Otrávení tabulky je způsob útoku, kterým můžeme donutit stanici S1, aby při příštím pokusu o komunikaci se stanicí S2, jejíž hodnotu má uloženou ve své lokální ARP tabulce, začala komunikovat se stanicí útočníka. Otrávení tabulky bude probíhat jako kontinuální činnost prováděná útočící stanicí - útočící stanice bude generovat periodicky ARP reply zprávy ve které bude informovat, že L3 adresa stanice S2 se nachází pod L2 adresou stanice útočníka. Při příští komunikaci stanice S1 se z lokální ARP cache tabulky vybere otrávený záznam odpovídající L3 adresy stanice S2 a stanice S1 zahájí komunikaci se stanicí útočníka.

3.3 Intercept - Přeposílání provozu

Tato sekce je poplatná i pro útok otrávení NDP cache.

Po otrávení ARP/NDP cache tabulky a donucení odpovídajících stanic - tj. takový stanic, v jejichž komunikaci má útočící stanice zájem se angažovat - komunikovat s útočící stanicí je třeba zajistit, aby tyto stanice nic nepoznaly. Útočící stanice, hrající roli *Man-in-the-Middle* to zajistí tak, že přijatá data od stanice S1 přepošle stanici S2 a obráceně. Samotné přeposílání je realizováno záměnou cílové L2 adresy zprávy, kterou obdržela útočící stanice, za adresu odposlouchávané stanice za kterou nás stanice co odesílala zprávu považuje.

4 NDP - Neighbor Discovery Protocol

Pro stanice v IPv6 sítích již ARP protokol není dostačující a proto byl zaveden protokol NDP, který (mimo jiné) umožňuje stanicím v síti se vzájemně lokalizovat. Tato dokumentace se bude zabývat pouze podmnožinou zpráv tohoto protokolu a to takových, které jsou vhodné pro vykonání MitM útoku NDP cache poisoning. Jsou to zprávy `Neightbor Solicitation` a `Neighbor Advertisement`.

Zpráva `Neighbor Solicitation` slouží k zjištění L2 adresy stanice v síti na základě její L3 adresy. Zpráva `Neighbor Advertisement` slouží k oznámení kde se v rámci L2 sítě stanice na síti nachází. Tato zpráva může být zaslána jako `solicited` resp. `unsolicited` v závislosti na tom, zda je, resp. není, poslána jako odpověď na zprávu typu `NS`.

4.1 NDP - Lokální cache tabulka

Opět, podobně jako v případě ARP, si každá stanice v IPv6 síti udržuje svou lokální tabulku o umístění ostatních stanic v rámci L2 sítě, zvanou NDP cache. I tato tabulka jde otrávit a implementovaná aplikace *PDS-Spoof* toto realizuje. Princip je velmi podobný jako u otrávení ARP cache a je blíže popsán v

sekci Implementace.

Obsah lokální NDP cache jde zobrazit na referenčním obrazu ISA2015 pomocí příkazu `ip -6 neighbor show`, vypsaný obsah pak může vypadat například takto:

```
fe80::c628:cac5:2252:265a    dev    eth1    lladdr    08:00:27:a9:1b:db
fe80::e770:cac5:2252:d323    dev    eth1    lladdr    08:00:27:b6:a9:8c
```

5 Implementace

Jako implementační jazyk byl zvolen `C++11`, přeložitelný na poskytnutém referenčním stroji. Implementace částí projektu je rozdělena do jednotlivých souborů s názvy `pds-spoof`, pds-scan, vždy s příponami *.cpp a *.h.

5.1 Implementace nástroje PDS-Scanner

Z implementačního hlediska jsou v nástroji přítomny dvě hlavní větve - větev pro skenování IPv4 sítě a větev pro skenování IPv6 sítě, v kódu zapouzdřeny do funkcí `discoverDevicesARP()` a `discoverDevicesNDP()`. Funkce jsou provedeny sekvenčně - nejprve je oskenována IPv4 síť a až následně síť IPv6. Implementace jednotlivých funkcí však využívají více samostatných procesů, viz. níže. Skenovací aplikace se po proskenování sítě sama ukončí a vygeneruje výstupní soubor, dle požadavků ze zadání. Pokud je aplikace ukončena násilně signálem `SIGINT`, skenování je ukončeno a soubor je i přesto vygenerován.

discoverDevicesARP() implementuje funkcionalitu skenování sítě skrze množství generovaných zpráv typu ARP Request, na všechny adresy v síti. Před vykonáním funkce je volána funkce `extractAddressesForInterface()`, která získá do interní struktury lokální MAC, IPv4, IPv4 masku sítě a IPv6 adresy. Lokální IPv4 adresa a maska sítě jsou použity k vypočítání rozsahu segmentu sítě, který funkce skenuje. Dále je pak funkce rozdělena na rodičovský a potomkovský uzel, kde právě potomek počítá možné adresy v síti a posílá na ně ARP request. Rodič mezitím čeká na odpovědi, které zapracuje a uloží do vnitřní struktury existujících zařízení v IPv4 síti. Nepřijdou-li po dobu 2 sekund žádné odpovědi, rodič se taktéž ukončí a program pokračuje ve větvi skenující IPv6 síť.

discoverDevicesARP() implementuje skenování IPv6 sítě. Pro initiální průzkum sítě je program ihned rozdělen na rodiče a potomka, kde potomek opět posílá požadavek a rodič přijímá odpovědi. Potomek vytvoří ICMPv6 paket simulující zprávu `ICMPv6 ECHO` a tento paket zašle na multicastovou adressu `ff02::1`, která odpovídá všem uzlům v síti *(pozn. autora: a dalo by se ji považovat za maskovaný broadcast)*, následně se ukončí. Uzly připojené na multicastovou adresu „všechny uzly“ se pak ozvou s ICMPv6 zprávou ECHO reply. Rodič si ozývající se uzly pochytá a jejich IP adresy si uloží do vnitřní struktury objevených IPv6 adres.

Dále se rodič opět rozpoltí a vytvoří nový potomkovský proces. Potomek převezme strukturu naplněnou ozývajícími se IPv6 adresami, zkonstruuje NDP zprávu typu `Neighbor Solicitation`, spočítá si multicastovou adresu cílového uzlu a tuto zprávu na ni odešle. Výpočet multicastové adresy probíhá tak, že se vezme 104 prvních bitů z „všechny-uzly prefixu“ a 24 posledních bitů z adresy cílového uzlu a spojí se dohromady. Oslovené uzly se ozvou zpět se zprávou `Neighbor Advertisement` a rodič tuto zprávu chytí a nalezené zařízení uloží do své vnitřní struktury nalezených zařízení. Rodič čeká vždy 1 sekundu na příchozí zprávu a nedojde-li, sníží počítačku „zbývajících pokusů“ (nastavena na 5 při inicializaci) o jedničku. Přijde-li (očekávaná) zpráva, počítačku opět nastaví na 5. Nedostane-li 5x za sebou žádnou zprávu, ukončí se.

Není-li program v době svého běhu násilně ukončen (a tedy je vyvoláno generování souboru), vygeneruje soubor a dobrovolně se ukončí.

5.2 Implementace nástroje PDS-Spoof

Po zpracování parametrů je možné se již ve vstupním bodě aplikace rozhodnout, jestli budeme otravovat ARP nebo NDP cache a tedy jsou opět k dispozici dvě větve, kterými se program může vydat - `poisonARPCache` a `poisonNDPCache`. Po spuštění se program sám od sebe neukončí a čeká na své násilné ukončení pomocí signálu `SIGINT`. Dojde-li k tomuto násilnému ukončení, je volána v závilosti na parametry specifikovaném protokolu buď funkce `antidoteARPCache()` nebo `antidoteNDPCache()`, které se postarají o navrácení stavu cache otravovaných zařízení do původního stavu.

poisonARPCache() implementuje process otravování cache zařízení specifikovaných skrze parametry spuštění programu. Na základě těchto parametrů jsou tedy vytvořeny podvržené ARP pakety typu `reply`, které obsahují vždy IP adresu jedné z obětí a MAC adresu útočící stanice. Tyto pakety jsou pak odesílány odpovídajícím stanicím v nekonečném cyklu (s odpovídajícím, parametry specifikovaným intervalem) stále dokola čímž dochází k otravování cache. Funkce sama od sebe nikdy neskončí a čeká, až bude ukončena signálem SIGINT.

poisonNDPCache() funkce je implementována velmi podobným způsobem - ze strukturálního pohledu - jako funkce `poisonARPCache()`. Rozdílem je zde však způsob otravování cache. Jsou zde místo ARP paketů vytvořeny NDP pakety typu `Neighbor Advertisement`, kde jsou opět odpovídajícím a analogickým způsobem prohozené IP a MAC adresy. Tento paket je ještě rozdílný oproti normálním NA paketům tím, že v jeho části pro „flags“ je nastaven `override` bit, který zaručuje, že při přijetí zprávy přepíše hodnotu v cache na cílové stanici. Další změnou je pak to, že tato NA zpráva není posílána jako „solicited“, ale jako „unsolicited“ - tzn. není přímou odpovědí na ničí solicitation dotaz.

antidoteARPCache(), antidoteNDPCache() funkce implementují „protilátku“ podanou stanicím v moment co je aplikace PDS-Spoof ukončena. Realizovány jsou v podstatě stejným způsobem: Pošlou cílovým stanicím dvojice zpráv se správnými IP a MAC adresami. Tyto zprávy jsou poslány pro jistotu vícekrát (20x) a před jejich odesíláním se program uspí na stejný moment, jako je interval mezi odesíláním otrávených paketů - tím je dosaženo toho, že i při zahlcení linky je cache uvedena do správného stavu.

6 Demonstrace činnosti

Aplikace byly implementovány a testovány v prostředí třech propojených virtuálních strojů referenčního stroje ISA2015. Každý z virtuálních strojů byl pak dále označován síťovým aliasem pdsX, kde X náleží do z obou stran uzavřeného intervalu od 1 do 3. Útočící stanicí byl vždy uvažován stroj pds1. Virtuální síť je možné popsat takto:

```
eth1 pds1 MAC: 08:00:27:81:B7:C8  IPv4: 169.254.167.236 IPv6: fe80::ee94:42ac:df70:d323
eth1 pds2 MAC: 08:00:27:A9:1B:DB  IPv4: 169.254.179.143 IPv6: fe80::e779:30b6:9b28:5b9f
eth1 pds3 MAC: 08:00:27:1D:D9:84  IPv4: 169.254.145.246 IPv6: fe80::af08:e6e4:72:b59a
Maska sítě 255.255.0.0
```

6.1 Demonstrace činnosti aplikace PDS Scanner

Po přeložení aplikace PDS-Scanner a jejím spuštění příkazem `./PDS-Scanner -i eth1 -f out.xml` se jako výstupní soubor vygeneruje následující výstup:

```
<?xml version="1.0" encoding="UTF-8"?>
<devices>
    <host mac="0800.271d.d984">
        <ipv4>169.254.145.246</ipv4>
        <ipv6>fe80::af08:e6e4:72:b59a</ipv6>
    </host>
    <host mac="0800.27a9.1bdb">
        <ipv4>169.254.179.143</ipv4>
        <ipv6>fe80::e779:30b6:9b28:5b9f</ipv6>
    </host>
</devices>
```

Kromě výstupu do souboru se navíc na obrazovce objeví textová informace o dokončení skenovací procedury.

6.2 Demonstrace činnosti aplikace PDS Spoof

Nejprve se na obětních stanicích podíváme do ARP cache `arp -a` a zjistíme, že záznamy pro všechny se zdají býti platné. Při použití příkazu `ping` na druhou napadanou stanici se ping vrací.

Po přeložení a spuštění aplikace na útočící pds1 stanici příkazem:

```
./pds-spoof -i eth1 -t 1000 -p arp
-victim1ip 169.254.179.143 -victim1mac 0800.27a9.1bdb
```

`-victim2ip 169.254.145.246 -victim2mac 0800.271d.d984`

Je možné pomocí příkazu `arp -a` zkontrolovat na obětních stanicích, že jejich arp záznam pro druhou obětní stanici změnil svou MAC adresu. Příkaz `ping` nefunguje, neboť provoz není přeposílán. Zastavíme-li nyní program signálem SIGINT (stisk CTRL+C) a znovu si prohlédneme výstup příkazu `arp -a`, uvidíme, že vše bylo vráceno do původního stavu a příkaz `ping` začne opět fungovat.

Stejným způsobem a dvojicí příkazů `ip -6 neighbor show` a `ping6` můžeme demonstrovat i pro NDP.

Literatura

[1] DeSanti, C., Carlson, C., and R. Nixon, "Transmission of IPv6, IPv4, and Address Resolution Protocol (ARP) Packets over Fibre Channel", RFC 4338, DOI 10.17487/RFC4338, January 2006, ¡http://www.rfc-editor.org/info/rfc4338¿.

[2] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 1970, DOI 10.17487/RFC1970, August 1996, ¡http://www.rfc-editor.org/info/rfc1970¿.

[3] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)", RFC 1885, DOI 10.17487/RFC1885, December 1995, ¡http://www.rfc-editor.org/info/rfc1885¿.