POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki Teoretycznej
i Systemów Informacyjno-Pomiarowych
Zakład Elektrotechniki Teoretycznej
i Informatyki Stosowanej

PRACA DYPLOMOWA INŻYNIERSKA

na kierunku Informatyka
w specjalności: Inżynieria oprogramowania

Wykorzystanie protokołu HTTP/2 do budowy szybkiej aplikacji internetowej

Piotr Szklanko
nr albumu 244145

promotor
mgr inż. Bartosz Chaber

Warszawa, 2017

Wykorzystanie protokołu HTTP/2.0 do budowy szybkiej aplikacji internetowej

Streszczenie

Praca składa się z krótkiego wstępu jasno i wyczerpująco opisującego oraz uzasadniającego cel pracy, trzech rozdziałów (2-4) zawierających opis istniejących podobnych rozwiązań, komponentów rozpatrywanychjako kandydaci do tworzonego systemu i wreszcie zagadnień wydajności wirtualnych rozwiązań. Piąty rozdział to opis środowiska obejmujący opis konfiguracji środowiska oraz przykładowe ćwiczenia laboratoryjne. Ostatni rozdział pracy to opis możliwości dalszego rozwoju projektu.

Słowa kluczowe: praca dyplomowa, LaTeX, jakość

THESIS TITLE

Abstract

This thesis presents a novel way of using a novel algorithm to solve complex problems of filter design. In the first chapter the fundamentals of filter design are presented. The second chapter describes an original algorithm invented by the authors. Is is based on evolution strategy, but uses an original method of filter description similar to artificial neural network. In the third chapter the implementation of the algorithm in C programming language is presented. The fifth chapter contains results of tests which prove high efficiency and enormous accuracy of the program. Finally some posibilities of further development of the invented algoriths are proposed.

Keywords: thesis, LaTeX, quality

Warszawa, 1 lutego 2017

POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa inżynierska pt. Wykorzystanie protokołu HTTP/2 do budowy szybkiej aplikacji internetowej:

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Piotr Szklanko.....

Spis treści

Rozdział 1

Wstęp

Moim celem jest przeprowadzenie testów protokołu HTTP w najnowszej wersji 2.0. Obecnie powszechnie stosowana jest wersja 1.1, która została wprowadzona w roku 1999. Jednakże szybki rozwój technologii internetowych sprawia, że wprowadzony osiemnaście lat temu protokół przestaje powoli spełniać swoje zadanie. Obecnie bez wykorzystania serwerów CDN czy cache przeglądarki oraz innych sposobów nie jest możliwe stworzenie płynnie działającej strony internetowej. Za pomocą własnoręcznie stworzonej aplikacji chcę przekonać się, czy wprowadzone funkcje faktycznie mają tak ogromny wpływ na szybkość komunikacji pomiędzy klientem i serwerem.

Swoją aplikację stworzyłem wykorzystując zestaw oprogramowania MEAN – MongoDB, Express.js, Angular i Node.js.

- MongoDB – baza danych NoSQL (cos o mongoose? dodać dokumentację do wszystkich punktów),
- Express.js – framework Node.js do tworzenia aplikacji sieciowych od strony serwera,
- Angular – framework JavaScript służący do budowy dynamicznej aplikacji internetowej od strony użytkownika,
- Node.js – środowisko uruchumieniowe języka JavaScript, które pozwala wystartować serwer.

Zdecydowałem się na to rozwiązanie z kilku powodów:

- po przejrzeniu dostępnych w sieci informacji doszedłem do wniosku, że implementacja protokołu HTTP/2 jest najlepiej opisana oraz wspierana przez środowisko związane z JavaScriptem,
- dobra znajomość języka JavaScript oraz jednoczesna chęć rozwoju umiejętności tworzenia aplikacji w tym języku,

- chęć poszerzenia wiedzy dotyczącej budowania aplikacji internetowych za pomocą technologii javascriptowych,
- nie ukrywam, że znaczący wpływ na moją decyzję miała również popularność języka JavaScript na rynku pracy.

Rozdział 2

HTTP/2

2.1 Historia

Pracę nad zmianami w protokole zapoczątkowała w 2009 roku firma Google ze swoim projektem SPDY. Zdecydowali się oni na stworzenie protokołu, który miał usprawnić działanie aplikacji oraz stron internetowych rozwiązując ograniczenia nałożone przez HTTP/1.1. Z biegiem czasu coraz więcej przeglądarek oraz stron internetowych, zarówno tych dużych jak i tych małych, zaczęło wspierać SPDY, co zainteresowało osoby pracujące nad protokołem HTTP. Zdecydowali się oni wykorzystać dokumentację protokołu SPDY jako początek prac nad własnym protokołem – HTTP/2. Od tego momentu aż do roku 2015, kiedy to standard HTTP/2 został oficjalnie zaakceptowany (**TODO**odnośnik RFC 7540 i może 7541), projekty były rozwijane równolegle. SPDY było wykorzystywane do testów nowych funkcjonalności, które miały zostać wprowadzone do nowego protokołu HTTP. Niedługo po oficjalnym zaakceptowaniu HTTP/2 ogłoszono, że SPDY nie będzie dalej wspierane.

W kilku poniższych akapitach postaram się przybliżyć zmiany, które zostały wprowadzone do protokołu HTTP.

2.2 Protokół binarny

Kluczową zmianą, która determinuje brak wstecznej kompatybilności z HTTP/1.1, jest przejście na kodowanie binarne przesyłanych wiadomości. Jest to rozwiązanie dużo bardziej kompaktowe i łatwiejsze w implementacji, niż przesyłanie zwykłego tekstu. Dzięki temu zabiegowi w ramach jednego połączenia TCP z serwerem może zostać utworzonych wiele dwukierunkowych strumieni danych przesyłających wiadomości HTTP. Taka wiadomość

to w rzeczywistości zapytanie od klienta lub odpowiedź serwera składające się z ramek. Każda ramka natomiast musi przynajmniej posiadać nagłówek z informacją, do którego strumienia danych należy. Kodowanie binarne nie ma wpływu na składnie zawartości ramki – wszystkie nagłówki czy zapytania HTTP/1.1 pozostawiono bez zmian. **TODO**schemat ramki i połączenia

2.3 Multiplexing

W poprzedniej wersji protokołu, pomimo, że istniała możliwość przesyłania wielu zapytań w ramach jednego połączenia, nie można było wykonywać ich równolegle. Każde zapytanie musiało być rozpatrywane i odesłane przez serwer do klienta zgodnie z kolejnością nadania, co powodowało efekt HOL (head-of-line blocking ODNOSNIK PO PL?). Aby wykonywać zapytania równolegle należało utworzyć kilka zapytań TCP, co obciąża serwer oraz jest czasochłonne. Protokuł HTTP/2 umożlwia przesyłanie oraz odbieranie wielu wiadomości jednocześnie. Są one rozbijane na pojedyncze ramki, przesyłane, a następnie odczytywane i składane po stronie odbiorcy. Dzięki temu nie jest już konieczne uciekanie się do takich zabiegów jak:

- scalanie plików (WEBPACK),
- wykorzystywanie spritów,
- domain sharding (DOCZYTAC).

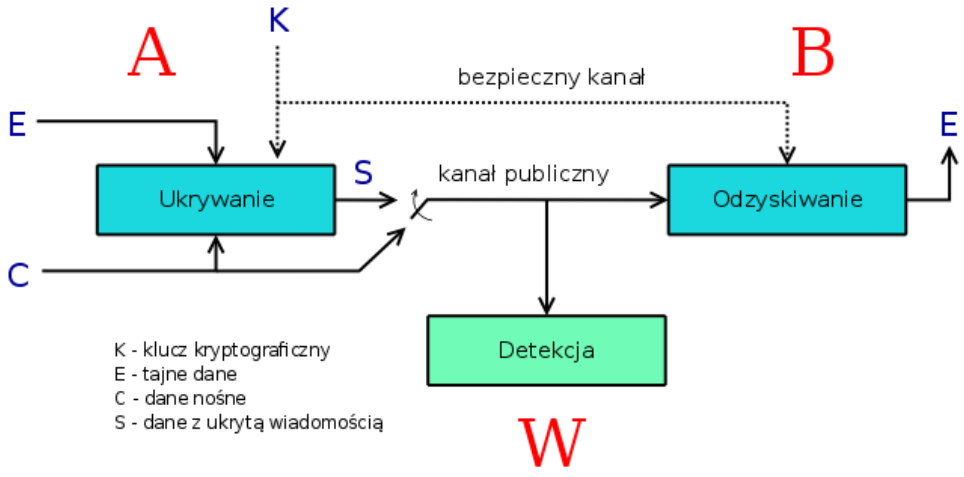
2.4 Prioretyzacja

2.5 Multiplexing

Podstawowy scenariusz, powszechny w literaturze na temat steganografii, odnosi się do sytuacji opisanej w [4]. Dwóch więźniów (w naszym przypadku Alicja(`A`) i Bob(`B`)) zamknięci są w dwóch odrębnych celach. Mogą się ze sobą kontaktować, jednak ich cała korespondencja przechodzi przez ręce Wartownika (`W`). Ma on pełen wgląd do przekazywanych informacji, więc może przechwycić wszelkie przekazywane tajemnice, a dodatkowo w razie podejrzeń może nie dopuścić do komunikacji[1]. W takim przypadku w celu przekazania ważnych informacji `A` i `B` muszą posłużyć się pewnego rodzaju podstępem. Muszą tak sformułować treść przekazu, aby `W` nie rozróżnił „niegroźnej" wiadomości od wiadomości z ukrytym przekazem. Dlatego też przekazują wia-

[1] podejrzana informacja jest tu analogią do stosowania kryptografii przez więźniów

domość, w której prawdziwa treść możliwa jest do odczytania po złożeniu kolejno każdej np. drugiej litery z każdego wyrazu.



Rysunek 2.1: Schemat komunikacji steganograficznej

Przedstawioną tak sytuację pokazuje rysunek 2.1[2]. `A` próbuje przesłać tajną informację `E` do `B`. Cała komunikacja odbywa się przez kanał publiczny, kontrolowany przez `W`. W celu ukrycia faktu komunikacji `A` stara się ukryć tajny przekaz w informacji `C`. W celu uzyskania skutecznej steganografii `W` nie może rozróżnić informacji poprawnej, nie zawierającej tajnych danych, od informacji `S`, która zawiera tajną informację. W celu dodatkowego zabezpieczenia przekazu, `A` i `B` mogą korzystać z funkcji kryptograficznej zabezpieczającej przekazywane informacje. Można tu wykorzystać metody kryptografii symetrycznej (ustalony klucz kryptograficzny `K`) lub niesymetrycznej (klucz publiczny K_{pub} i klucz prywatny K_{pryw}).

Stosowanie technik kryptograficznych wpływa na poprawę bezpieczeństwa przesyłanej informacji, jednak należy pamiętać o nieporządnych cechach jakie mogą one wywołać. W większości przypadków umieszczenie tajnej informacji steganograficznej w przekazie wiąże się z zamianą istniejącej już nieważnej części informacji. Jednak każda porcja usuniętej informacji może mieć pewną charakterystyczną postać lub specyficzny histogram. Zastosowanie funkcji kryptograficznej w stosunku do tajnej informacji zmienia ją, a wynikowy rozkład bitów jest nieprzewidywalny i w większości przypadków różny od standardowych histogramów określonych dla podmienianych części wiadomości.

[2]sporządzony na podstawie [7], rysunek 1, strona 3

2.6 Pipelining

Przesłanie danych za pomocą przekazu steganograficznego wiąże się w większości przypadków z umieszczeniem dodatkowej informacji w wiadomości. Odbywa się to za pomocą podmiany tej części wiadomości (nagłówka TCP/IP), która wykazuje cechy nadmiarowości lub której (kontrolowana) zmiana nie prowadzi do przerwania transmisji. Pewną podgrupą może być w tym przypadku wykorzystanie pól oryginalnie pustych (zerowych) lub niewykorzystywanych w istniejących implementacjach.

Kanały steganograficzne można podzielić na dwa zasadniczne typy[8]:

- kanał pojemnościowy (ang. storage channel) - informacja zawarta w częściach wiadomości, polach nagłówka,
- kanał czasowy (ang. timing channel) - informacja zawarta w czasach wystąpienia danych zdarzeń, np. przesłania pakietu TCP/IP.

W przypadku sieci pakietowych można także połączyć dwa typu kanałów steganograficznych, tworząc kanał mieszany, w którym jeden z typów (np. pojemnościowy) będzie wykorzystywany do przekazywania informacji, a drugi (np. czasowy) do sygnalizacji tego zdarzenia.

Większość opracowanych programów służących do przesyłania danych z wykorzystaniem steganografii opiera się na kanałach pojemnościowych. Wynika to z faktu, że kanały czasowe narzucają pewne ograniczenia na generację pakietów TCP/IP przez co ich wykrycie staje się prostsze.

Dodatkowo należy zauważyć, że w sieciach pakietowych można skonstruować abstrakcyjny kanał steganograficzny, w którym do przesyłania tajnych danych lub/i obsługi protokołu steganograficznego wykorzystywane są różne pola nagłówka. Zmiana wykorzystania danego pola może być dynamiczna, zależna od wymaganej przepustowości lub w celu zminimalizowania wykrycia kanału steganograficznego.

2.7 Kompresja danych

2.8 Steganografia w obiektach multimedialnych

Rozdział 3

Wnioski

Protokół TCP/IP jest najbardziej rozpowszechnionym i używanym protokołem komunikacji między systemami w sieci Internet oraz w sieciach intranet. Niestety został on opracowany na początku lat siedemdziesiątych, gdy problemy bezpieczeństwa informacji nie stały na pierwszym miejscu. Ciągły wzrost działań przestępczych w sieci Internet, w tym wymiana nielegalnych treści, prowadzi do stosowania coraz to nowszych technik zabezpieczających. Z tego względu obserwuje się działania mające na celu wprowadzenie tajnej komunikacji między przejętymi systemami, tak aby nie wzbudzić ostrzeżeń w analizatorach sieciowych. Taka ukryta komunikacja odbywa się z wykorzystaniem steganografii.

Wprowadzenie steganografii do niskich warstwach stosu TCP/IP umożliwia obejście wielu filtrów nałożonych na warstwy wyższe. Większość sieci oparta jest na protokołach rodziny TCP/IP, przez co nie można zabronić ich używania. Możliwa jest jedynie kontrola poprawności semantyki protokołów TCP/IP, a także ewentualna ingerencja w przekazywane wartości, z uwzględnieniem stanowości niektórych pól.

Opracowany schemat generacji początkowych numerów sekwencyjnych w jak najlepszy sposób odzwierciedla oryginalny proces zachodzący w stosie sieciowym systemu Linux. W większości przypadków występujących w rzeczywistych sieciach i systemach, numery wygenerowane przy pomocy `Shushi` nie byłyby rozróżnialne od numerów wygenerowanych przez stos sieciowy systemu.

Jeżeli proces generacji wartości użytych do przekazania danych steganograficznych zostanie oparty o oryginalne mechanizmy używane do ich generacji, to pasywny analizator sieciowy nie będzie w stanie wykryć istnienia anomalii. Różnice możliwe są do zaobserwowania w przypadku zaistnienia specyficznych sytuacji występujących dla danej implementacji protokołu. W przypadku zastosowania pasywnego analizatora wymaga to jednak oczekiwa-

nia na taką sytuację. Z przeprowadzonych testów wynika, że lepszym podejściem jest zastosowanie analizatorów aktywnych, które posiadają wiedzę na temat testowanych systemów oraz ich chrakterystycznych cech implementacji. Skonstruowanie takiego analizatora jest zadaniem stosunkowo prostym a daje bardzo wysoką skuteczność.

Z przeprowadzonych testów wynika, że celowe jest prowadzenie dalszych prac w następujących obszarach:

- dokładniejszy mechanizm generacji wartości mikrosekund
- wprowadzenie algorytmów zdolnych wykryć i uniemożliwić działanie analizatora aktywnego

Jeżeli powyższe punkty nie zostaną spełnione, analizatory aktywne będą w stanie wykryć istnienie modułu steganograficznego opartego na początkowych numerach sekwencyjnych.

pierwsza kolumna	druga	trzecia
1	2	3
a	b	c

$$E = mc^2 \tag{3.1}$$

Rozwój opracowanego rozwiązania steganograficznego jest możliwy poprzez wprowadzenie elementów – patrz wzór (3.1) – jak:

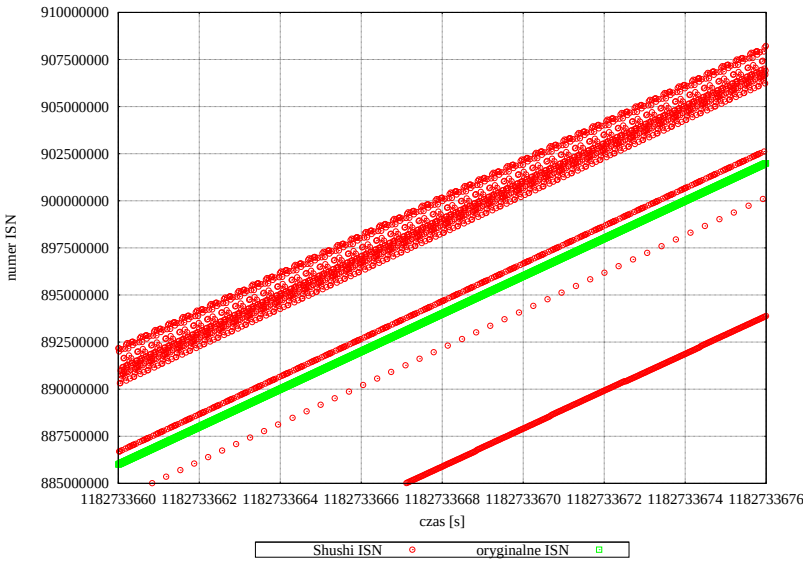
- obsługa innych, przyszłościowych protokołów sieciowych, takich jak SCTP (ang. Stream Control Transmission Protocol)[12]
- zapewnienie dwustronnej komunikacji z wykorzystaniem numerów potwierdzenia `ACK`
- przeniesienie implementacji do innych systemów operacyjnych

Wraz ze wzrostem przepustowości urządzeń sieciowych (obecnie 10Gb/s i więcej) wzrasta problem analizy przepływających danych w czasie rzeczywistym. Analizatory sieciowe muszą w coraz krótszym czasie zbadać coraz większy strumień danych (miliony pakietów na sekundę). Jednak problem wzrostu prędkości sieci utrudnia zadanie także osobom implementującym kanały steganograficzne w protokole TCP/IP. Coraz więcej operacji wyższych warstw stosu sieciowego przenoszonych jest do układów scalonych interfejsów sieciowych. Taka technologia znana jest pod skrótem TOE (ang. TCP Offload Engine) i odnosi się przede wszystkim do sprzętowej generacji sum kontrolnych oraz mechanizmu TSO (ang. TCP segmentation offload). W następnych latach spodziewane jest przenoszenie kolejnych elementów stosu sieciowego TCP/IP do implementacji sprzętowych.

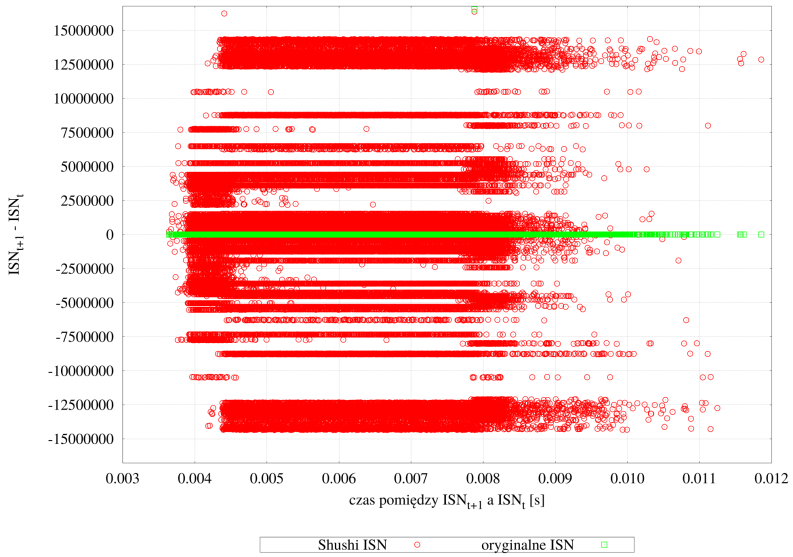
Ze względu na rozwój systemów zabezpieczających ruch sieciowy oraz wzrost bezpieczeństwa systemów operacyjnych, w kolejnych latach wzrośnie także wykorzystanie technik steganograficznych przez grupy przestępcze działające w ramach Internetu. Z tego powodu poznanie technik steganograficznych oraz wypracowanie metod obrony i wykrywania takiej komunikacji jest bardzo ważne.

Dodatek A

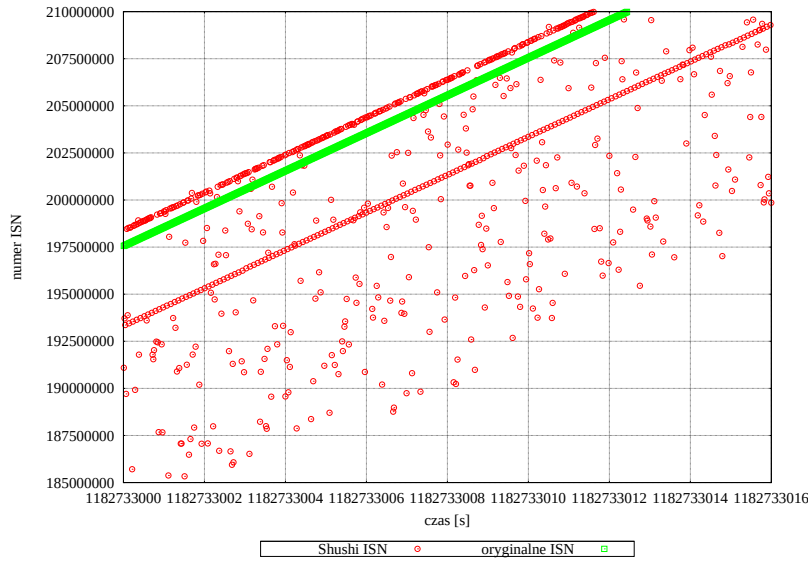
Porównanie numerów ISN jądra Linux i modułu Shushi



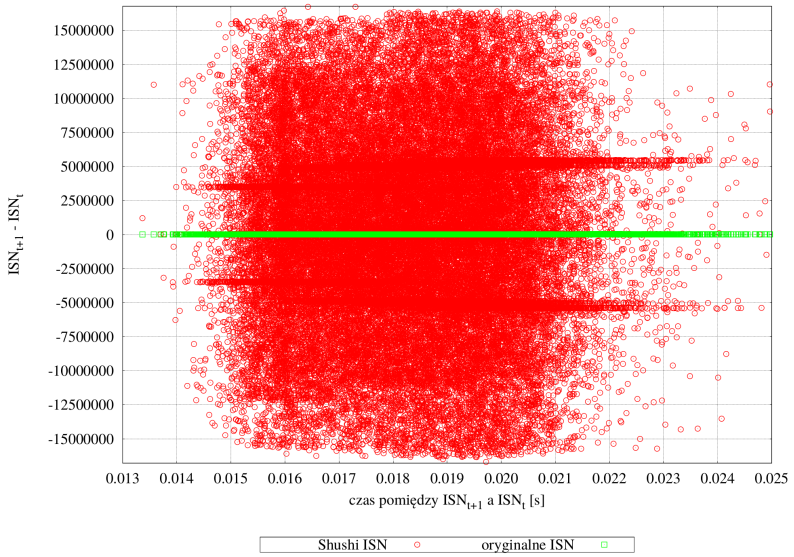
Rysunek A.1: Numery ISN wygenerowane przez jądro oraz `Shushi`, stałe numery IP oraz porty TCP, stałe dane dla `Shushi`, serie po około 2800 próbek.



Rysunek A.2: Różnice pomiędzy kolejnymi numerami ISN wygenerowanymi przez jądro oraz `Shushi`, stałe numery IP oraz porty TCP, stałe dane dla `Shushi`, serie po około 60000 próbek.



Rysunek A.3: Numery ISN wygenerowane przez jądro oraz `Shushi`, stałe numery IP oraz porty TCP, losowe dane dla `Shushi`, serie po około 860 próbek.



Rysunek A.4: Różnice pomiędzy kolejnymi numerami ISN wygenerowanymi przez jądro oraz `Shushi`, stałe numery IP oraz porty TCP, losowe dane dla `Shushi`, serie po około 60000 próbek.

Bibliografia

[1] W. R. Stevens, G. R. Wright, „Biblia TCP/IP tom 1", RM, 1998.

[2] U. S. Department Of Defense, „Trusted Computer System Evaluation Criteria", 1985.

[3] B. W. Lampson, „A note on the confinment problem", w „Proc. of the Communications of the ACM", październik 1973, numer 16:10, strony 613-615.

[4] G. J. Simmons, „The prisoners' problem and the subliminal channel", w „Advances in Cryptology: Proceedings of Crypto 83 (D. Chaum, ed.)", strony 51-67, Plenum Press, 1984.

[5] A. Kerckhoffs, „La Cryptographie Militaire (Military Cryptography)", J. Sciences Militaires, luty 1883.

[6] A. Havill, „The Spy Who Stayed Out In The Cold: The Secret Life of Double Agent Robert Hanssen", St. Martin's Press, 2001.

[7] C.Cachin, „An Information-Theoretic Model for Steganography", w „Information and Computation", 4 marzec 2004.

[8] S.Chauhan, „Embedding Covert Channels into TCP/IP", 7th Information Hiding Workshop, czerwiec 2005.

[9] Information Sciences Institute, University of Southern California, „Transmission Control Protocol", RFC793, wrzesień 1981.

[10] V. Jacobson, R. Braden, D. Borman, „TCP extensions for high performance", RFC1323, maj 1992.

[11] S. Bellovin, „Defending against sequence number attacks.", RFC1948, IETF, 1996.

[12] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, „Stream Control Transmission Protocol", RFC2960, Network Working Group, październik 2000.

[13] C. H. Rowland, „Covert Channels in the TCP/IP Protocol Suite", First Monday, 1997.
http://www.firstmonday.dk/issues/issue2_5/rowland/

[14] Alhambra, daemon9, „Project Loki: ICMP Tunneling", Phrack Magazine, Issue 49. http://phrack.org

[15] daemon9, „LOKI2", Phrack Magazine, Issue 51. http://phrack.org

[16] van Hauser, Reverse WWW Shell, THC, The Hacker's Choice.
www.thc.org

[17] T. Sohn, J. Seo, J. Moon, „A Study on the Covert Channel Detection of TCP/IP Header Using Support Vector Machine", Volume 2836 of Lecture Notes in Computer Science., Springer-Verlag (2003) 313-324.

[18] T. Sohn, T. Noh, J. Moon, „Support Vector Machine Based ICMP Covert Channel Attack Detection", Volume 2836 of Lecture Notes in Computer Science., Springer-Verlag, 2003, strony 461-464.

[19] J. Giffin, R. Greenstadt, P. Litwack, R. Tibbetts, „Covert messaging in TCP", w Dingledine, Privacy Enhancing Technologies. Volume 2482 of Lecture Notes in Computer Science., Springer-Verlag (2002) 194-208.
http://www.mit.edu/~gif/covert-channel/

[20] G. Fisk, M. Fisk, Ch. Papadopoulos, J. Neil, „Eliminating Steganography in Internet Traffic with Active Wardens", 5th International Workshop on Information Hiding, październik 2002.

[21] J. Rutkowska, „The Implementation of Passive Covert Channels in Linux Kernel", Chaos Communication Congress, grudzień 2004.

[22] Ch. Benvenuti, „Understanding Linux Network Internals", O'Reilly, grudzień 2005.

[23] kossak, „Building Into The Linux Network Layer", Phrack Magazine, Issue 55. http://phrack.org

[24] Steven J.Murdoch and Stephen Lewis, „Embedding Covert Channels into TCP/IP", University of Cambridge, Computer Laboratory, 29 lipec 2005.

[25] Eugene Tumoian, Maxim Anikeev, „Detecting NUSHU Covert Channels Using Neural Networks", Taganrog State University of Radio Engineering, Department of Information Security.

[26] mayhem, „IA32 Advanced Function Hooking", Phrack Magazine, Issue 58. http://phrack.org

[27] bioforge, „Hacking the Linux Kernel Network Stack", Phrack Magazine, Issue 61. http://phrack.org

[28] Robert Love, „Kernel Korner - Allocating Memory in the Kernel", 1 grudzień 2003.