RFC1918 updates

on servers

near M and F roots

Andre Broido, Marina Fomenkov, Young Hyun, kc claffy,
work in progress

C A I D A

CAIDA / SDSC / UCSD

www.caida.org

CAIDA–OARC Workshop

San Jose, 2005-07-25

Plan

- Background

- Windows traffic/sources' prevalence:
  - Application layer: DNS payload
  - Transport layer: TCP header
  - Network layer: IP header

- Conclusion

History

- 1996: RFC1918 reserves address blocks
  10/8, 172.16/12, 192.168/16 for private use
  People start using them for NATs

- 1997: RFC 2136 - dynamic DNS updates

- 2000: root servers see sharp increase
 in PTR updates for private addresses

Evi Nemeth starts looking into this and other problems,
suspects Windows

Remedy: AS 112 project

- Originally, Bill Manning introduced three servers authoritative for RFC1918 space

- Two servers process queries, one – updates

- prisoner.iana.org (192.175.48.1) – updates

- query servers: blackhole-1, blackhole-2

- in 2002 Paul Vixie and other operators distributed this service using anycast with origin AS 112

- In Jul.2004 12+ ASes provide this service
  - 40% Route Views peers see ISC
  - some peers see AS 7500 (WIDE)

- Our data consists of prisoner's nameserver logs (blackholes' traffic is not logged)

- We have Palo Alto and Osaka logs

RFC1918 DNS updates, May 2002--Dec 2004
Top: hazel (ISC). Bottom: Server near M-root
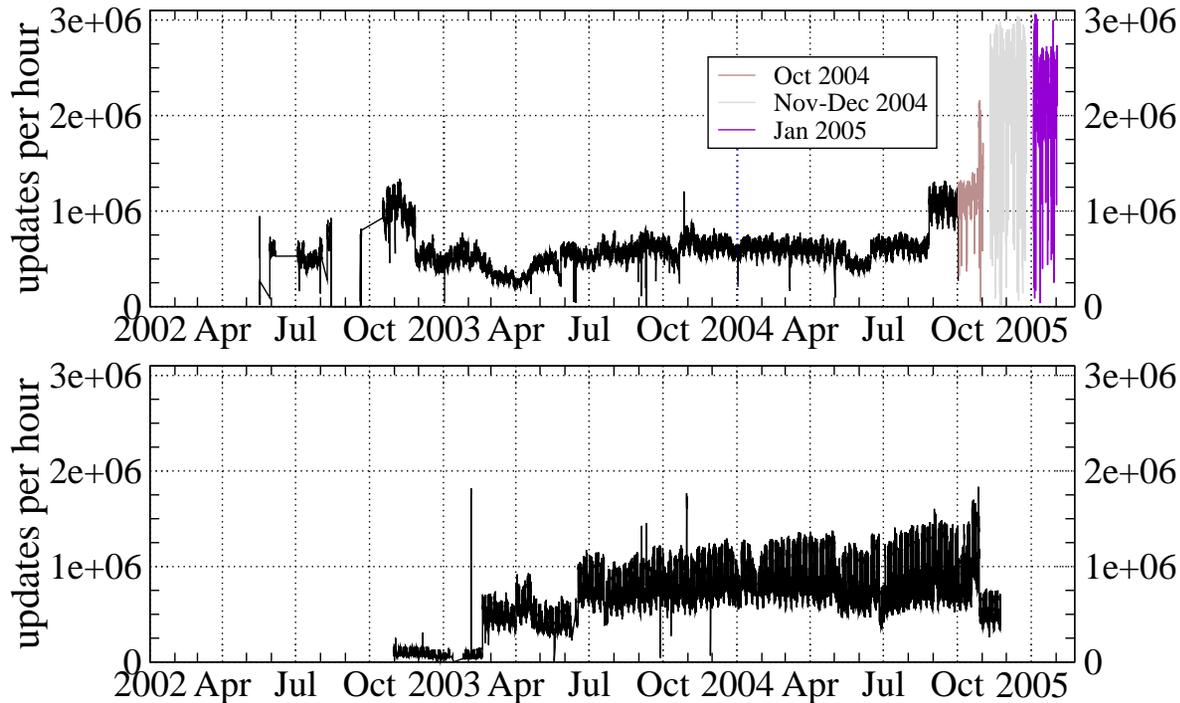
Server at Osaka (below) has less traffic, but higher spikes

The changes are abrupt, not long-term trends

Data is divesrse – 3 days of Palo Alto logs cover 16% prefixes, 30% ASes

Highest number of updates per second is 3889 in Apr 2004, Osaka

Why are leaking updates are bad?

- User's privacy is at stake

- Updates contain too much private data
  - machine name – often person's name
  - internal (RFC1918) IP address
  - global (ISP's) IP address
  - TCP reveals OS, service pack

- Useful for hacking, social engineering

- Queries reveal less (only IPs), no TCP

- In addition, operators need to install and maintain clusters of AS112 nodes all over the world

Recall that updates arrive at prisoner.iana.org
Will study prisoner's traffic, skip blackholes

Questions

- Which OSes are doing updates?

- What is the right way to do dynamic DNS updates for RFC1918 addresses?

We found these two statements to be true

- Most prisoner's packets come from Windows sources

- For most prisoner's source IPs, all observed transactions came from Windows hosts

Here 'most' means 96-98%

All our numbers are in that range

The situation is different for blackholes, but we do not study them here

Verification method: induction/closure

- Suppose that we established Windows origin for some packets

- Classify all related packets (such as packets with the same source IP and source port) as coming from Windows too

- Similar approach worked very well for p2p traffic

From the first glance, it might be possible to attribute all packets with the same source IP to Windows. However, NATs can multiplex systems with different OSes over the same source IP

How do we start OS fingerprinting?

- Need to find 'seed' packets for which we know their OSes. Will use:
    - Application layer: DNS payload data
    - Transport layer: TCP header
    - Network layer: IP header
    - Link/Network layer: Spectroscopy

- In some cases (e.g. root server traffic) can only use IP/UDP header

Summary of the proof

- Whatever layer we look at, we see evidence that packets came from Windows hosts.

- For any layer data that points to Windows constitutes 96-98% of packets, flows, IPs etc.

- Three independent 'witnesses' (application, transport and network layer) tell us the same story (97% of volume is Windows) with minor variations – the story must be true.

- This method comes from Bayesian Epistemology

Why do we need this study of updates' origin OS?

- Didn't we find with spectroscopy in 2002 that updates come from Windows?
  - Spectroscopy shows that many sources have the same periods as Windows machines (e.g one update each hour)
  - It does not guarantee that they are Windows
  - It does not give a precise fraction of Windows
  - We did not estimate traffic since nameserver logs don't show packet counts
  - Many systems go on and off and this destroys their periods

- Rationale: We still see the update volume growing, need to act

- Will make quantitative analysis with richer data source

Data sources

- Two 5-min AS112 packet traces taken at noon:

- Feb 28 2005 with 2.5 M packets, 114k source IPs

- Mar 17 2005 with 2.1 M packets, 94.5k source IPs

- packet rates of up to 6300 pps

- prisoner receives 75-77% packets (90% TCP, 10% UDP)

- blackholes receive 10-11%, only UDP packets

More workload info in the paper

Roadmap: You are here

- Application layer: DNS payload

- Transport layer: TCP header

- Network layer: IP header

Application layer

- 2002 lab study: Windows systems:
  - try secure update using Transaction Signature, RFC 2845
  - start by establishing secret keys with TKEY record, RFC 2930
  - do it by TCP three times in a row

DNS rarely uses TCP (except for AXFR/IXFR)
The fact of TCP use points to Windows

Components of a TKEY record

- Key name

- Algorithm name

- Data used for computing secret key

query: 910533066770-2. TKEY IN
answer: 910533066770-2. TKEY ANY TTL=0
gss.microsoft.com...(binary data)...
NTLMSSP...(binary data)...C27ALTEER

A TKEY record ends in abbreviated source domain, leaks private info

Microsoft name(s) in the DNS payload

- gss.microsoft.com: a key computation algorithm
- NTLMSSP = NT Lan Manager Security Support Provider
- TKEY record is misplaced
  (Answer instead of Additional)
- Rarely, gss-tsig algorithm, RFC 3645 by Microsoft
- Plus: key name for gss-tsig always contains '-ms-'
- for gss-tsig, TKEY record is in the right place

- Safe to assume Windows origin for both algorithms

Application layer results

- Assume that TCP flows with either of two algorithm names and NTLMSSP in TKEY are from Windows

- 98% TCP flows with 98.5% TCP packets are in this category

- Flows by packet, Syn and TKEY count (a triple):
  - 93% flows have (5,1,1) - 5 packets, 1 Syn, 1 TKEY
  - add retransmission-like triples (6,1,1), (6,2,1), (6,1,2)
  - add port reuse triple (10,2,2)

- these 5 groups of flows alone add up to 97% TCP flows and packets.

TCP packets make up about 90% of prisoner's traffic. so we classified a solid chunk of data.

We also studied transport layer approach (in the paper), which analyzes Syn packets with p0f tool and yields precisely the same results as app layer analysis.

Are we done?

- No – the above application- and transport-layer apply to TCP traffic, do not cover UDP

- Whenever RFC1918 addresses are involved, UDP and TCP can originate on different hosts, resolvers, forwarders, firewalls and NATs.

- We know that UDP updates and TKEY exchanges are part of one transaction

- Windows default: "Use secure updates when insecure update failed"

- Restoring this transaction is possible but complicated:
  - There is no common port, DNS ID, domain etc.
  - UDP update can come through a server, while TCP comes through a proxy.

We need to develop passive OS fingerprinting methods for UDP packets

Roadmap: You are here

- Application layer: DNS payload
- Transport layer: TCP header
- Network layer: IP header

New method

- Two previous methods (application and transport layer approaches) have parallels in our previous work (p2p traffic analysis and remote device fingerprinting)

- We now present a TTL-DF-ID method that we did not try before

- We aim at getting an estimate of the number of Windows packets, and an estimate of the error.

- We discuss this approach in greated detail than two other approaches because it is much more robust (uses only IP headers).

- We developed it with root server traffic analysis in mind

- This estimation technique is universal and applies outside DNS – for generic IP traffic

Network layer: IP TTL classification

- Studied packet traces for TTL vs. OS relation with p0f (turned off TTL in p0f logic)

- The answer: 98.8% Syns with TTL=128 are from Windows

- Also studied skitter replies to find common initial TTLs

- Answer: Powers of 2 – 32, 64, 128; 255, 120 plus trace amounts of 50, 100, 150, 155, 180, 200

Will assume TTL=65-128 to come from Windows. p0f predicts getting some false negatives, but only 1.2% false positives.

Skitter reply TTL density

skitter replies density. San Jose and M-root monitors, 2005-04-01..17
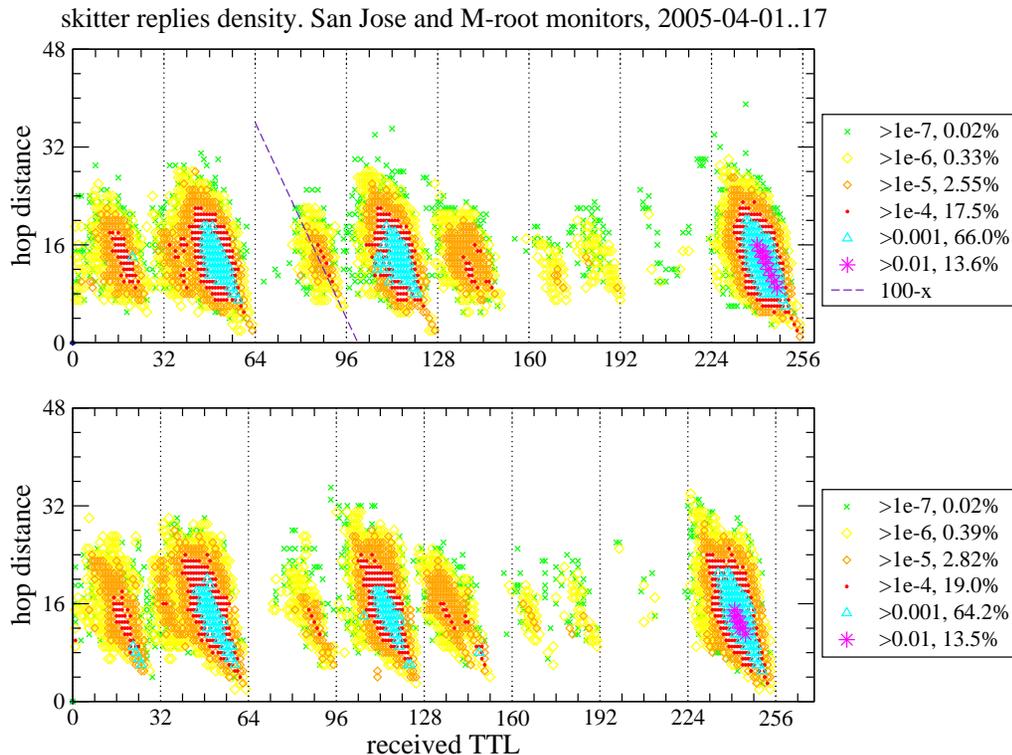
X axis: destination's reply packet TTL
Y axis: hop distance to the destination
Color: Density of reply packets

Top: San Jose monitor, April 01-17 2005, 5.5M replies from 432k dest. Bottom: Osaka, M-root, April 01-16 2005, 2.4M replies from 288k dest.

Estimating iTTL - initial TTL

- Example: We are 1 hop from the destination

- Replies come back with initial TTL:

- hop + TTL = iTTL + 1;

- This property holds whenever distance is symmetric

- (the distance can be symmetric even when the path is not)

- Will compute histograms for the sum

- Determine possible iTTLs from its maxima

- The rationale: there must be enough symmetric distances to create spikes

Estimating iTTL (cont'd)

initial TTL estimate by hop+rep.ttl-1. San Jose monitor, Apr 01-17, 2005
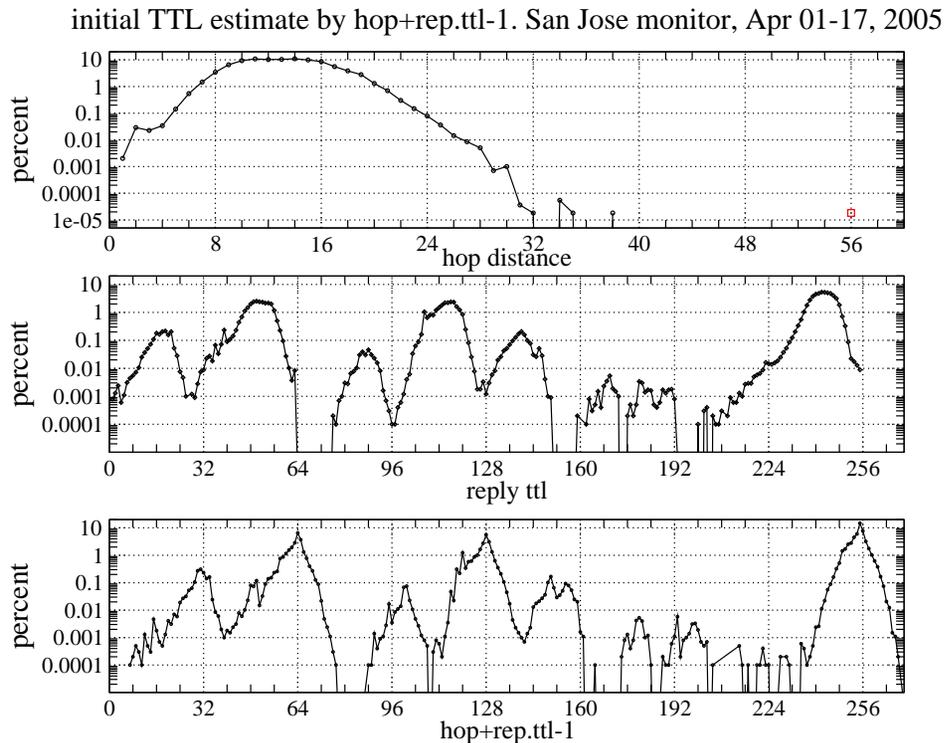
X axis: hop distance, reply's TTL, hop + reply TTL - 1.
Y axis: percent of skitter (SJ) replies at that value of X

Top: hop distance histogram
Middle: Reply TTL histogram
Bottom: hop distance + reply TTL - 1 histogram

TTL-related observations

- Unexpected iTTL values: 50, 100, 120, 150, 155, 180, 200.

- Common property: low 'Kolmogorov complexity' from human perspective

- However, only 120 really matters, with 2.24% replies.

- Lesson: 'diagonal' histogram has much better spike resolution

Estimating Windows' packet share by iTTL

- 98% packets are in iTTL=128 range

- This number is roughly the same (within 0.6%) for TCP, UDP and UDP-only sources

- With 98.8% true positives, 96.8% or more are Windows' packets

- Sounds right, given our TCP-based numbers

False negatives

A vendor's default iTTL of 128 can result in observed TTL under 64 or over 128 because:

- a user changed TTL setting by editing registry

- a firewall reset TTL or subtracted a large value

- TTL field got corrupted (e.g. bit 6 flipped from 1 to 0)

- packet came through a transient routing loop.

When these events produce false negatives they cause underestimation of Windows' share. We are mostly concerned about false positives

We take false positives rate (non-Windows machines in iTTL=128 group) from p0f's 1.2%. Since UDP packet count is small, error in this rate only affects Windows share in about 10% of total volume. Even a 10% rate would only change prisoner's total by 1%.

Network layer: TTL-DF-IPID signatures

- So far we mostly analyzed the volume measure

- There may be many non-Windows *sources* with small packet counts

- UDP-only sources make up 25-28% of source IPs.

- Need something comparable to p0f for UDP DNS packets

TTL-DF-IPID signatures: our machines

- We experimented with nslookup on several systems

- Made a table of TTL, DF bit and IP ID triples

- For Windows, DF=0 and IP ID is non-zero

- Signatures' mnemonic:
  - TTL: L=64, W=128, S=255
  - denote DF=0 by 'o', DF=1 by 'i'
  - IP ID = 0 by 'z', nonzero ID by 'n'

- Common systems' signatures:
  - Won - Windows
  - Lon - FreeBSD/MacX
  - Liz - Linux

TTL-DF-IPID signatures for UDP-only sources

- UDP-only sources send 3.7% packets in March trace

- Windows' signature Won is in 97% of them

- The source IPs with single signature Won make up 97.6% of all source IP

- The next most frequent signature is Lon (MacX or FreeBSD), with 1.4% sources, 1.8% packets.

- Woz is next, much more frequent than IP ID = 0 would naturally occur (firewalls zeroed IP ID to avoid Bellovin's attack?)

- Lon Won – Windows and MacX or FreeBSD sharing a source IP – comes fourth with 8 sources (0.04%) and 121 packets (0.02%)

- This statistics remains almost unchanged for all UDP sources and packets (details in the paper)

Conclusions

- Update rates are higher than in 2002

- Windows machines
  - send 97-98% of packets
  - make up over 96% of all sources

- while the share of systems that run Windows is 90%, i.e. it is exceeded

Suggested changes

- Vendor:
  - Send updates only to the hosts with RFC1918 address
  - Do not update PTR records by default (use manual configuration)

- User:
  - Do not run DHCP on Windows machines; home routers do DHCP without DNS updates
  - Change default config on your machine to not send updates