Black Ops 2006
pattern recognition
Dan Kaminsky
DoxPara Research
Who Am I?

- Coauthor of several book series
  - Hack Proofing Your Network
  - Stealing The Network
- Formerly of Cisco and Avaya
  - Presently partnering with IOActive
  - One of the “Blue Hat Hackers” that has been auditing Windows Vista
- Sixth Year Speaking At Black Hat!
  - TCP/IP, DNS, MD5, SSH, etc.
What Are We Here To Do Today?

- Enforce Network Neutrality
- Gaze Horrified Upon 2.4 Million SSL Servers
- Fix Online Banking (just a little)
- Fix the security hole I put in OpenSSH
- Make entropy recognizable
  - Useful for cryptosystems (like SSH)
  - *Really* useful for fuzzing
- Pretty, pretty pictures.
  - New for this year: USEFUL pretty, pretty pictures
    - Even if they’re +100Mpix
Making Use of 100+ Megapixels: Visual Bindiff
Enforce Network Neutrality?

• Telecom Companies have essentially stated – they wish to spy upon and selectively censor traffic, so as to maximize revenue from those who will pay the most to see their traffic pass unhindered.

• This devolves down to a common refrain in Crypto: “Alice and Bob are in prison, and are attempting to communicate without the Warden interfering”
  – Don’t believe the premise?
Internet Isolationism: $1140 A Year To Check Your Email

• “To accommodate the needs of our customers who do choose to operate VPN, Comcast offers the Comcast @Home Professional product. @Home Pro is designed to meet the needs of the ever growing population of small office/home office customers and telecommuters that need to take advantage of protocols such as VPN. This product will cost $95 per month, and afford you with standards which differ from the standard residential product.”

  – What, you didn’t actually think the war against Network Neutrality had anything to do with video, did you?
What It’s Really About

• It’s all about $1100+ a year per telecommuter
  – 40M telecommuters in 2004 * $1140 a year = $45.6B
  – How many telecommuters if the US has to cut back on oil consumption, by saying every Friday is a telecommute-to-work day?
• As people realize what’s coming, the question will stop being, “Should the network be neutral”, and will become, “Is it possible to detect non-neutral networks?”
  – The answer is yes. Yes it is.
TCP Bandwidth Estimation: An Elegant Weapon, For A More Civilized Age

• TCP automatically determines the amount of available bandwidth between any two points
  – Multiple TCP streams sharing the same communication channel do not send packets to one another
  – All communication happens implicitly, via dropped packets
  – Dropped packets are a source of information about the amount of bandwidth available on a given channel
    • If more packets show up, then a particular line is willing to route, then some will be dropped, and TCP will quickly notice.
  – Can we figure out who’s causing our packets to drop?
Active Network Probing, or how TTLs just never go out of style

• Suppose you can only send data to someone at 5k/sec, and you’re curious, why so slow?
  – What this means is – you get dropped packets whenever you try to send faster than 5k/sec.

• Experiment: Send more data alongside the session, but TTL limit the transmissions until you figure out which hop causes packet drops in the primary.
  – Too much data…one hop…no effect on 5k/sec stream.
  – Too much data…two hops…no effect on 5k/sec stream.
  – Too much data…three hops…5k/sec stream stops. Third hop is your limiting node.
  – Demo
What Can You Detect?

• Source Preference
  – Spoof the source IP for your extra packets. If Viacom can send extra data, but random_blackhole_ip can’t, then you know Viacom has preference.
  • Possible to detect this even if full TCP sessions are required, by controlling the client (Google Desktop) and having it send the requisite series of fake SYNs and ACKs, TTL limited to prevent the real site from responding. Ask me later if you want more details.

• Content Preference
  – Spoof particular payloads for your extra packets. If encrypted traffic causes TCP to detect dropped packets, but unencrypted traffic gets through just fine, you get signal.
Of Course They’d Block Crypto

• 1) Precedent
  – Comcast already tried to knock out IPsec
• 2) Proxy Avoidance
  – “The Open Internet” is still out there – you just need to route to it, via SSH, SSL, IPsec, DNS…
    • Bouncing through proxies is a standard passtime in some lands
  – Encryption keeps them from being able to see that you’re not stealing service, therefore Encryption = Theft of Service
• 3) Profit Capture
  – Who uses encryption?
    • Workplaces that make money from their employees at home
    • E-Commerce sites that make money from consumers at home
    • Money made = increased ability to pay
• As security professionals, it’s hard enough deploying secure solutions without wondering if/when the telco’s going to block traffic for it being encrypted.
On Deploying SSL

- SSL/TLS: Standard Internet protocol for certificate-based authentication of otherwise unknown parties
  - Has a couple of basic rules for deployment:
    - Do not put anything secret into an SSL cert; there’s a reason they’re called public keys
    - Do not put the same key on two different boxes. SSL lacks Perfect Forward Secrecy, so not only will Alice be able to impersonate Bob, but Alice will be able to passively monitor all of Bob’s traffic.
- I have a high speed scanning node called Deluvian, with which I found 2.4M SSL hosts (specifically, HTTPS)
  - Weirdest results of any scan I’ve ever done – enough that I’m not going to discuss all my results, they’re too weird
Total Mysterious Statement

• IF YOU ARE THE SORT OF SITE THAT DOES NOT WANT PEOPLE KNOWING ALL YOUR INTERNAL DNS NAMES, BE VERY CAREFUL WHAT SSL CERTS YOU LET THE PUBLIC SCAN FOR
  – Side note:  You might not want to put this on your honeypot:
  – '/C=JP/ST=TOKYO/O=XXXXXX/OU=IT Division/CN=honeypot.xxxxxx.com/emailAddres s=nw-admin@xxxxxxx.com'
What Appears To Be The Case

• What DID the numbers say?
  – Good: 90% of keys on only one box
  – Bad: 10% of keys were everywhere, enough that only one out of three boxes found had a unique key.
• Theory: No two devices are supposed to have the same key
  – Reality: A depressing number of VPN concentrators and embedded devices had SSL keys pre-burned into them at ship.
  – Depressing Reality: It vaguely appears like a group that really should know better has deployed tens of thousands of machines with the same cert
• Caveat: Absolute numbers are really sketchy. Only half of IP addresses that respond to TCP/443 actually had anything there, and a fair number of those addresses actually changed what key they were hosting when tested.
  – Someone in the audience probably knows WTF 😊
  – In the mean time, there is a very obvious SSL flaw…
“Why Is This Secure”
The World’s Most Depressing Google Search

- Everything here is delivered over HTTP. So an attacker can just replace https with http and hijack your login.
- 26% of the Top 50 banks operate insecurely; all but one use a picture of a lock to assure users the link is safe.
We’re Going To Need A Bigger Boat

• People have been complaining about this for quite some time – believe me, I’m not the first to notice
• Choices seem to be:
  – 1) Force everyone at the home page to go to SSL
    • Too expensive to send everyone to SSL, so that’s out
  – 2) Force everyone at the home page to click through to a login page
    • Confuses users = still too expensive. Users might call up instead, and who wants to talk to users?
  – 3) Allow people to log in directly through the home page
    • *crickets*

• Is it possible for users on online applications to use a home page login screen securely?
Another Option

- Web pages aren’t static – they can recode themselves in response to user input
- `<IFRAME>` is a mechanism for putting a “mini-window” of another site in a page.
  - Known: IFRAMEs are useful for precaching entire web pages
  - Not Known: IFRAMEs can contain https links
- Solution: When the user first interacts with the Username field, `document.write` an IFRAME to your SSL site. **This initializes SSL, and starts precaching site content.** When they shift focus into the password field, immediately redirect the window to the https site.
  - Demo
Example(HTML)

- Create a username and password field, plus a SPAN to inject an IFRAME into

  `<td>Username: <input name="login" id="username" type="text" onKeyUp="precache();"></td>`

  Password: <input name="password" id="username" type="text" onFocus="window.location.href='https://login.yahoo.com';"></td>`

  `<hr>`

  `<span id="TextDisplay"></span>`
Example (JS)

- Add an iframe, once, if precache is called.
- `<script>
  var changed=0;
  function precache() {
    if(changed) {return 0;}
    changed=1;
    var divel=document.getElementById("TextDisplay");
    divel.innerHTML='<iframe height=400 width=400 SECURITY="restricted" src="https://login.yahoo.com"></iframe>';
  }
</script>`
3) **Immediate redirect to** [https://login.yahoo.com](https://login.yahoo.com) **upon entry into password field.**

- **How to make users understand the quick screen flicker?** **Use an animated GIF of a lock closing.**
- **You will need to move username from http to https, w/o XSS please.**
As Long As We’re Talking About Bugs In Cryptosystems…

- 2001: Found that SSH can be turned into an extremely flexible VPN solution. Problem is…when used as one, it will in many instances leak DNS resolution requests required for remote use, onto the Local LAN. Who knows who will answer, or with what?
  - Mozilla can use SOCKS5 support via hidden settings; sends full DNS name upstream
  - IE6/7 does not support SOCKS5

- Is it possible to fix this problem w/o changing client code?

Solution:

**Dynamic Forwarding w/ SOCKS**

- ssh user@host -D1080
- SOCKS4/5: An in-band protocol header, nothing more, that allows the client to very quickly tell a proxy server where its actual destination was
- SOCKS4 is extraordinarily simple
  - ~9 bytes from Client, 8 byte response, and the client has informed the “proxy” where it actually wants to go!
  - “Library Preloads” are excessive
- The idea: Run a trivial SOCKS daemon in the ssh client; use it to redirect the destination of each channel.
Why Not Forward DNS over SSH?

- DNS is a UDP protocol, and SSH only moves TCP.
  - Could put a big huge translation layer into SSH, whereby it converted UDP requests into TCP, decapsulated them back to UDP, and sent them off to some UDP server...
  - Or we could just tell the local DNS client that whenever they request something over UDP, the response is just too big...better retry over TCP 😊
- Put up a server that does nothing but set the truncation bit to one
- Tell SSH to do a Local Port Forward as normal
- Set system to use 127.0.0.1 as system DNS server.
- This is a general purpose strategy for anything that only moves TCP (Tor, some SSL-VPN clients).
Sadly, I Am Going To Hell

- 2006: DNS->SSH
- 2004: SSH->DNS
- DNS->SSH->DNS == DNS->DNS
- Malkovich Malkovich Malkovich
SSH’s Wetware Bug

• $ ssh dan@blah
The authenticity of host 'blah (1.2.3.4)' can't be established.
Are you sure you want to continue connecting (yes/no)?

• 09:a9:b1…am I supposed to do something with this?
  – Yes. According to SSH’s design, you’re supposed to reject the proposed fingerprint if it looks unfamiliar. (Seriously.)
• The “Two Billion SSH Key” attack (by ADM) just comes up with 2B keys and emits the visibly closest key. It works.
Cryptomnemonicics

• There are three classes of memory, at least to the degree as is useful in cryptography
  – Rejection: “I’ve never seen that before”
  – Recognition: “It’s that one, not that other one”
  – Recollection: “Let me describe it to you.”
• SSH just requires rejection – “What? That’s new.”
• Hex domain clearly does not work. What else is available?
Other Attempts

• Abstract Art via déjà vu
• Calculated faces via Passfaces
• Both have attempted to address limited capacity for recollection by moving authentication to a recognition problem
• But recognition offers only a limited number of bits: $9^5 = 59049 < 2^{16}$
  – This is OK, since Passfaces is online and thus can lock a user out before 59K attempts are up
  – We are not online – but we only need to reject, not recognize and certainly not recollect
Betcha Didn’t Think I Could Make A DNS Reference

- Humans do not remember arbitrary strings of characters effectively.
- Humans do remember stories well, but stories can morph over time. The most stable element of any story, though, are the names of its participants.
- We do seem to have “hardware acceleration” for names.
- What if we represent “rejection proposals” as a short series of names?
#DEFINE
BIZARRE_BUT_EFFECTIVE

• 2) Noting that there are more unique female names than male names, and way more last names than either, find:
  – 512 Male names (9 bits)
  – 1024 Female names (10 bits)
  – 8192 Last names (13 bits)
  – Use an Edit Distance metric (Perl’s String::Similarity, Python’s Levenshtein, C’s fstrcmp) to prevent two names from going on the final list that may be confused for one another. May also use acoustic measures, like Soundex or Metaphone.
• 3) Split the 160 bit hash rejection proposal from OpenSSH into 32 bit chunks. Male+Female+Last=9+10+13=32 bits, so you’ll get five couples.
Demo

• $ ssh dan@blah

Key Data:
  julio and epifania dezzutti
  luther and rolande doornbos
  manual and twyla imbesi
  dirk and cuc kolopajlo
  omar and jeana hymel

The authenticity of host 'blah (1.2.3.4)' can't be established.
Are you sure you want to continue connecting (yes/no)?

• It is critical that the Key Data be shown every time there’s a connection. The user must become familiar with the “characters” in the “story”.
  – This actually seems to work.
Speaking of broken representations of Entropy

• $ od -t x1 foo
• 00000000 6a ac 06 2d f2 86 76 4c a3 b6 d4 29 26 45 ef 9c
• 00000200 40 07 42 8b e3 de d3 9e 67 c8 8f fa 80 86 32 72
• 00000400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
• *
• 00002400 40 56 7c 5a 84 25 6c c8 8a 26 57 7d 50 b9 16 df
• 00002600 5c b4 72 ec 5e 44 ff e8 37 54 7c 53 f9 77 96 e3
• 00003000

• There has got to be a better way to represent complex file signals than “um, here’s some hex bytes, and there’s a big section of zeros right here”
  – “Yeah! Add ASCII!” I mean, more than that.

• Better entropy representations needed for:
  – Data analysis (first view of new protocols)
  – Fuzzing
Fuzzing A Midpoint

- “Dumb Fuzzing”: Take a file, flip some bits, see what happens
- “Smart Fuzzing”: Take a file, understand its internal structure, fuzz the structure, see what happens
- Understanding requires skill, potentially non-existent documentation, time.
  - Dumb fuzzing requires none of these things
  - Can we increase the intelligence of dumb fuzzers?
    - Well, we’ve got this this that’ll find structure in anything...
N’est’ce pas Non Sequitur

• **Sequitur: Linear Time Pattern Finder**
  – Creates hierarchal Context Free Grammars from arbitrary input

```java
switch (c) {
    case 1: value = 2
    case 2: value = 3
    case 3: value = 4
    case 4: value = 5
}
```

```plaintext
S → switch (c) {A1B2A2B3A3B4A4B5...}
A → defcase
B → : value =
```

• Compression Algorithm in which you can “look under the covers” to see what’s going on
• Created by Craig Neville-Manning as his PhD thesis a decade ago
  – He’s now Chief Research Scientist at Google
Syntax Highlighting For Hex Dumps

- Trivial Algorithm: In a hierarchical grammar, each byte requires traversing to a certain depth in order to recover the raw literal.
- Color each byte by how deep in the tree you have to go.
- Can we do more?
Setting Up For The CFG9000

- Turns code on left into symbolic set on right; it's easy then to link the symbols together as per the graph.
- This works for non-textual data
- Sequitur imputes meaningful symbols from arbitrary input data

```
a switch (c) {
  case 1: value = 2
  case 2: value = 3
  case 3: value = 4
  case 4: value = 5
}
```

```
b S → switch (c) {A1B2A2B3A3B4A4B5...}
A → case
B → : value =
```
Context Free Grammar Fuzzer: THE CFG9000

- Reduce input data to a stream of symbols
- Fuzz data at the symbol level, rather than at pure bytes
  - Shuffle
  - Drop
  - Repeat
- Sequitur is not necessarily the best way to generate a grammar. In fact, Suffix Trees are probably the appropriate mathematical construct.
  - Sequitur may scale better (100MB input)
Sample CFG9000 Output

- `calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(calculate_rule_usage(p->rule()))))))))))))))`
TODO

• Create “Requitur”; Sequitur implementation optimized for fuzzer use
  – Generate larger symbols
    • No two byte symbols please; we’re not trying to compress, we’re trying to elucidate structure
  – Eliminate redundant symbols
    • Keiffer-Yang optimization in ~2001: If symbol (x) == symbol (y), then delete (y) and set all instances of (y) to (x)
    • Need to do this to actually consistently fuzz all instances of a particular trope
  – Possibly remove in-memory grammar requirement
    • Use mechanisms from Ray, a out-of-memory variant
  – Add foreign grammar capability

• Sequitur is really cool, but not yet where we need it…
Another Approach: DotPlots

• Remembered an old paper, entitled *Visualizing Music And Audio Using Self-Similarity*
  – Jonathan Foote from Xerox

• Brute Force solution – compare songs to themselves, splitting them into tiny chunks and marking light for similar and dark for dissimilar
  – Disassociated Audio will do this for you
Day Tripper from the Beatles…
can we get something similar from fuzz targets?
Pirate Baby MPEG Says Yes
What Exactly Are We Doing

• Jonathan Helman’s “DotPlot Patterns: A Literal Look at Pattern Languages” offers an introduction

• Instead of “to, be, not” etc, we use chunks of data from arbitrary files
  – The same similarity metric used to disambiguate names for the SSH hack, is used to measure similarity here 😊
History

• Extensive history in bioinformatics world (talk about legacy code)
• Can’t find any reference to it being used to guide security research
• What would we want:
  – 1) Global view of section boundaries
    • Can I separate out clearly different sections?
  – 2) Local view of what exactly is going on
    • Can I get some idea of exactly what’s happening, given certain visible patterns?
Java Class Files
.NET Assemblies
CNN’s Home Page
SMBT Torture Traffic
(Packets!)
Kernel32.dll
Chromosome 22
The Legend Of Zelda
Autocorrelation Dotplots Appear Helpful

• Tool being released shortly (hardcorr) calculates these images
  – Hacking: IMAX Style (100Mpix images are very common)

• Global goal clearly achieved
  – Fuzzing is a combinatorial game
  – Uniquely identifying self-similar sections gives us finite regions to analyze and comprehend

• Can we get any local knowledge?
From The Paper

a) Squares.
b) Diagonals.
c) Shuffle.
d) Light Cross.
e) Broken Diagonals.
f) Reordered Diagonals.
g) Reordered Squares.
h) Density Variation.
i) Dark Cross.
j) Palindrome.
We have those patterns, but we have some pretty weird stuff too…
More Research To Do

• Determine meaning of various visual tropes that are evolving from the data
• Create interactive tools for dotplot evaluation
  – Data Microscopy 😊
• Colorize
  – Use different similarity metrics to evoke different colors
  – Did build a generic similarity construction out of bzip2/gzip; it works but finds too many similarities
• Better Symbol Selection
  – X86 aware, jump target normalization, integrate Sequitur CFG, reimplement Halvar
• That’s what I might do. What, you don’t think I’m done, do you?
If autocorrelation is interesting…

- Cross-Correlation is where the real fun lies
  - Autocorrelation: Compare A to A
  - Cross-Correlation: Compare A to B

- Most files are sufficiently dissimilar that not very interesting structure shows up
  - Notable exception: Different versions of the same binary
Visual Bindiff!
MSVCR70.DLL v. MSVCR71.DLL
In Summary

• Your VPN is under threat – tell your boss!
  – If steroids aren’t illegal, a test isn’t useful
• Check your devices for generic SSL certs
  – Especially you guys.
• Fix any application that submits to HTTPS from HTTP, it’s easy
• Use apps that support SOCKS5 for SSH Dynamic Forwarding if you can, or reset your system DNS as described if you can’t
• Stop expecting users to remember long strings of hex characters
• FUZZ YOUR FILE FORMATS, seriously
• Take a look at your data, you might be surprised at what you find.