

**Black Ops 2008:
It's The End Of The Cache
As We Know It
Or: "64K Should Be Good Enough For
Anyone"**

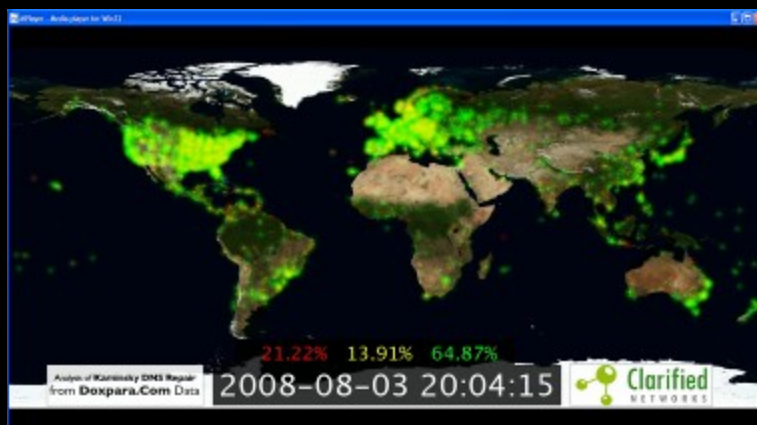
TOORCON KEYNOTE REMIX

**Dan Kaminsky
Director of Penetration Testing
IOActive, Inc.**

Hi guys!

- So, this is my first keynote.
 - Ever!
- Purpose of a keynote: To talk about the overarching themes of a conference
- Purpose of my talks: To play with toys
- We have 90 minutes. Can we do both?
 - Lets find out.

What Just Happened



- There was a *really big bug* that hit DNS
- Industry responded pretty awesomely
 - Microsoft
 - ISC
 - Cisco
 - Nominum
- Hundreds of millions of users were protected

Intro to DNS

- System on Internet which maps names (that humans understand) to numbers/ “IP Addresses” (that the Internet can deal with)
 - Just like 411 information, or the White Pages
 - Numbers change so frequently on the Net, that it’s easier to just keep looking them up
 - Almost everything on the Internet depends on DNS returning the right number for the right request
 - More than you’d think
 - Foreshadowing!

DNS is distributed

- Three possible answers to any question
 - “Here’s your answer”
 - “Go away”
 - “I don’t know, ask that guy over there”
 - This is delegation. You start with a request, and then get bounced around all over the place.
 - 13 root servers: “www.foo.com? I don’t know, go ask the com server, it’s at 1.2.3.4”
 - Com server: “www.foo.com? I don’t know, go ask the foo.com server, it’s at 2.3.4.5”
 - Foo.com server: “www.foo.com? Yeah, that’s at 3.4.5.6.”
- Dealing with “ask that guy” (“Delegation”) a lot of work, so DNS infrastructure divided into Servers (that run around) and Clients, or “Stub Resolvers”, that either do or don’t get an answer
 - BIND = Name Server
 - Your Desktop = Stub Resolver

What about bad guys?

- If everything depends on receiving the right number for the right name, wouldn't a bad guy want his number returned instead?
 - Yup
- So when the name server asks ns1.foo.com for www.foo.com, couldn't the bad guy reply first, with his own number?
 - Yup
- What's supposed to prevent this?
 - Transaction ID – “random” number between 0 and 65535. The real name server knows the number, because it was contained in the request. The bad guy doesn't know – at best, he can guess

The Guessing Game

- Good guy – the real name server – has a 65,536 to 1 advantage over the bad guy
 - Those are “long” odds for the bad guy
 - Those are ridiculously *short* odds by 2008 standards
 - Most web session IDs in 2008 are 2^{112} times more secure
 - Too bad they leak, Mike Perry ☺
- When the good guy gets his reply in – “wins the race” – he can say how long until the next “race”, via something called the TTL, or “Time To Live”
 - 1 minute
 - 1 hour
 - 1 day
 - This is how long a given number is “valid” for a particular name.
- $1 \text{ day} * 65,536 \text{ races} / 2 = 84.5 \text{ years}$ for 50% chance
 - Good luck on that.

And thus, Forgery Resilience

- Document being assembled by Bert Hubert, author of PowerDNS
 - Was soon to be an Internet RFC
- Basic concept: Long TTL = High Security, Low TTL = Low Security
 - 65,535 minutes / 2 = 22 days for 50% chance
- The basic concept is wrong, *very very wrong*
 - Quote from my Black Hat 2007 talk: “TTL’s are not a security feature”
 - The concept implies its opposite, i.e. that the bug I found must exist, because there’s no way something not intended to be a security feature would ever stand up to attack
 - So Bert delayed his RFC while we fixed the bug
- However, I had no idea this was under development when I found the flaw
 - So what’s the bug?
 - There are three issues – first two were kind of known, the last is what’s new

First: If it's a race, between who can reply with the correct TXID first, the bad guy has the starter pistol

- Bad guy can force the name server to go run to the good guy and look something up
 - It takes time to get the real request (with random number) to the good guy
 - It takes more time to get the real response back from the good guy
 - It takes no time for the bad guy to immediately follow up a request with a fake response
 - Might have the wrong random number, but it'll definitely arrive first

Second, who said the bad guy can only reply once

- Winner of the race is the first person to show up with the correct random number
- Nowhere does it say the bad guy can't try lots of random numbers
 - He has time – he doesn't need to wait for anything to reach him, because nothing ever will
- If the bad guy can reply 100 times before the good guy returns, that 65536 to 1 advantage drops to 655 to 1.
 - Alas...still long odds. And when he loses, he has to wait the TTL. That could be 655 days – almost 2 years!
 - Or maybe not.

Finally, the bad guy doesn't actually need to wait to try again.

- If the bad guy asks the name server to look up www.foo.com ten times, there will only be one race with the good guy
 - The first race will be lost (most likely), and then the other nine will be suppressed by the TTL
 - No new races on this name for one more day! Here, use the answer from a while ago
 - So, can we race on other names?
- If the bad guy asks the name server to look up 1.foo.com, 2.foo.com, 3.foo.com, and so on, for ten names, there will be 10 races with the good guy
 - TTL only stops repeated races for the same name!
- Eventually, the bad guy will guess the right TXID before the good guy shows up with it
 - And now...the bad guy is the proud spoofer of ... 83.foo.com
 - So? He didn't *want* to poison 83.foo.com. He wanted www.foo.com

Bait and Switch

- Is it possible for a bad guy, who has won the race for 83.foo.com, to end up stealing [www.foo.com](#) as well?
 - He has three possible replies that can be associated with correctly guessed TXID
 - 1) “Here’s your answer for 83.foo.com – it’s 6.6.6.6”
 - 2) “I don’t know the answer for 83.foo.com.”
 - 3) “83.foo.com? I don’t know, go ask the [www.foo.com](#) server, it’s at 6.6.6.6”
 - This has to work – it’s just another delegation
 - 13 root servers: “83.foo.com? I don’t know, go ask the com server, it’s at 1.2.3.4”
 - Com server: “83.foo.com? I don’t know, go ask the foo.com server, it’s at 2.3.4.5”
 - Foo.com server: “83.foo.com? I don’t know, go ask the [www.foo.com](#) server, it’s at 6.6.6.6”

Enter The DNSRake

- Named after a common method for lockpicking
- 1) Send a query to a nameserver, for \$RANDOM.foo.com
 - The bad guy has the starter pistol
- 2) Send 200 fake replies to that nameserver, with TXID 0-200
 - The bad guy can reply multiple times
- 3) Send replies containing nameserver redirections to www.foo.com
 - \$RANDOMwww.foo.com IN NS www.foo.com
 - www.foo.com IN A 6.6.6.6
 - If this works, it works
 - If it fails, return to step 1

What's it look like?

- 1 0.000000 1.2.3.4-> 66.240.226.139 DNS Standard query ANY 2465786792ask-dan-at-foo-com.foo.com
- 2 0.000669 1.2.3.4-> 66.240.226.139 DNS Standard query response NS ask-dan-at-foo-com.foo.com A 6.6.6.6
- 3 0.001008 1.2.3.4-> 66.240.226.139 DNS Standard query response NS ask-dan-at-foo-com.foo.com A 6.6.6.6
- 4 0.001304 1.2.3.4-> 66.240.226.139 DNS Standard query response NS ask-dan-at-foo-com.foo.com A 6.6.6.6
 - 5-201 are like 4
 - 202 repeats back to 1

Running the attack...

- `dnsrake 66.240.226.139 1.2.3.4 ask-dan-at-foo-com.foo.com 63752 6.6.6.6 200`
 - 1) IP of my name server, mail.doxpara.com (BIND9, but it'll work against anyone)
 - 2) IP of ns*.foo.com
 - Repeat command for each ns*
 - 3) Name I'd like to pollute
 - 4) Fixed source port of my server, leaked by having it look up something off one of my own domains
 - 5) IP I want to force people to use
 - 6) Ratio of random requests to spoofed responses

Validating the attack

- # dig @mail.doxpara.com
ask-dan-at-foo-com.foo.com
; <<>> DiG 9.2.5 <<>> @mail.doxpara.com
ask-dan-at-foo-com.foo.com
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status:
NOERROR, id: 59212;; flags: qr rd ra; QUERY: 1,
ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 5
;; QUESTION SECTION:
;ask-dan-at-foo-com.foo.com. IN A
;; ANSWER SECTION:
ask-dan-at-foo-com.foo.com. 86279 IN A 6.6.6.6

Extending The Attacks

- So that works against pretty much everything in wide deployment
 - BIND8/9
 - MSDNS
 - Nominum (with some tweaks)
 - Doesn't work against DJBDNS, PowerDNS, MaraDNS
- Most commonly offered defense: "Our DNS servers don't accept queries from the outside world. They must be safe!"
 - Can someone ask them to look up www.doxpara.com, will they return 157.22.245.20?
 - If so, don't be so sure

On Bailiwicks

- 1.foo.com is able to return a reply for www.foo.com for a reason
 - “In bailiwick”
 - The root servers can return any record
 - The com servers can return any record...for com
 - The foo.com servers can return any record for foo.com
 - It wasn't always this way, but then Eugene Kashpureff wanted his own TLD (com, net, etc)
 - He just added additional records in every reply for foo.com, declaring his own TLD existed
 - Everyone accepted it
 - So the bailiwick system was invented to prevent foo.com from declaring anything about com, or some other new TLD
 - (This was 1997, the last time we had a bug this bad.)
 - 2002, Vagner Sacramento's Birthday Attacks, couldn't override cache
 - 2007, Amit Klein's TXID prediction, couldn't override cache

Out Of Bailiwick Referrals, or How To Attack Name Servers Behind Firewalls

- DNS doesn't stop working when you get a referral into another bailiwick
 - If foo.com says “Ask that guy over there, here's his address”, and that guy is bar.com, the name server goes back to the root and asks: “Heh, I hear I need to look up something from bar.com, but I can't trust the guy who told me to go there. Where's bar.com?”
- This means any lookup can spawn any other arbitrary lookup, on demand
 - 1. Force a lookup to 1.badguy.com
 - 2. Reply with a referral (NS or CNAME) to 1.foo.com
 - This *immediately* causes a request to be sent to the foo.com name server
 - 3. Follow the reply with an immediate stream of fake replies from the foo.com name server
- There are many many ways to do #1

The Many Starter Pistols Of Mr. Bad Guy

- Web Browsers will look up what the bad guy wants
 - Any link, any image, any ad, anything can cause a DNS lookup
 - No Javascript required, though h0h0h0 it helps
- Mail Servers will look up what the bad guy wants
 - On first greeting: HELO
 - On first learning who they're talking to: MAIL FROM
 - On SPAM check
 - Get worried now.
 - When trying to deliver a bounce
 - When trying to deliver a newsletter (Lyris, ahem, plz patch)
 - When trying to deliver an actual response from an actual employee

GetHostByName() Considered Harmful

- Web log resolution
 - Reverse DNS – given a connection from 6.6.6.6, PTR lookup to www.badguy.com
 - Return a CNAME (alias) with a 0 TTL, to anyone else's name
 - Each record will now repeatedly look up the attacker controlled name, even though the target aliased into has a longer lifespan
- “Web Bugs” in documents
 - File formats that “call home” to their authors upon reading
 - They're not just about privacy violation anymore
- Lots and lots of things in Web 2.0
 - URL attachments
 - Keep getting more worried

Takeaway #1: Protocols Cannot Be Understood In Isolation

- Theory: If a name server only resolves names for trusted hosts, it will be safe.
- Reality: “Trusted hosts” resolve names for untrusted hosts *all the time*, from other protocols
- **It is not protocols that are under attack, but systems. If you are not aware of how all the protocols in your systems interact, you will miss vulnerabilities that are obvious to an attacker.**

GetHostByAddr() ain't doing too well either

- IDS/IPS
 - Scan a large network, especially with a lame attack, and something will automatically try to figure out who you are
 - When they try to figure out who you are, they will do a Reverse DNS lookup on your address, to look up your name
 - Attacker can control the address-to-name mapping
 - Attacker can return a CNAME in response to an address-to-name request
 - So:
 - IDS sees a Slammer attack from 6.6.6.6.
 - IDS resolves 6.6.6.6 back to a name
 - Attacker returns a CNAME to 1.foo.com, TTL=0
 - While IDS DNS server tries to look up 1.foo.com, Attacker replies with false replies for 1.foo.com
 - If he wins, great. If not, go send another Slammer attack ☺
 - Technically, you don't even need to scan with your own IP – you know, Slammer from any IP may cause a DNS lookup for that IP.

Takeaway #2: Everything You Do Can Be Used Against You

- There is a cost to every action you take, in response to an attacker
- You are effectively granting the attacker special control over some subset of your network
 - Every response you make is another degree of freedom for an attacker
 - This *can* be a worthwhile tradeoff, but always recognize this as a tradeoff!

Another Stunt: Roy Arends' Trick

- The Microsoft nameserver, when it sent a query to the outside world, would accept queries back on that particular socket
 - So, you answer a question with a question
 - Roy found this, mentioned it to the dev, it was fixed in an unrelated codepath and the fix was absorbed with the overall update
 - Nice find, Roy!

Note: People still firewall with router ACLs

- The “right” way to firewall: Keep track of outbound requests, only accept legitimate responses
 - This requires expensive gear if you want to protect high-traffic servers, because you have to keep track of lots of state
- The “wrong” way to firewall: Statelessly pattern match IP packets, reject those that come from the “wrong addresses”
 - Much easier to do this, *especially if there is asymmetric routing anywhere*
 - Free (already in whatever routers are providing network access)
- We’ve been saying this for six or seven years!

Alas...

- The wrong way to firewall *actually works pretty well*
- Router ACL firewalls are usually vulnerable to someone spoofing their source IP
 - Yes, it's sometimes possible to pay attention to which interface a packet comes in on, but not always
- But most interesting services are hosted via TCP, which is relatively spoof-resistant
 - SEQ#/ACK# prevent blind spoofing (well, now they do)

Why DNS is particularly vulnerable to bad Router ACLs

- DNS is special
 - Hosted via UDP – pure request/response
 - Recurses – a request can spawn another request, meaning you don't care where the response goes because you saw a request
- So, if you have a nameserver that doesn't want to respond to a DNS attack, because it's only doing lookups for a particular IP range...you can spoof the range
 - Usually just spoof a neighbor, far enough away to be on a different subnet, but close enough to be in the same organization
- Note: This works against all rules on DNS servers themselves (allow-recursion in BIND, etc)
 - No per-interface data!

Takeaway #3: If It's Stupid And It Scales, It Isn't Stupid

- A *lot* of things are being done, “*wrong*”, because they actually work
 - People can nod their head all day at solutions that are *supposed* to be right, but if the “technically secure” way barely scales, while the “hideously insecure” alternate Just Works, it’s the latter you’re going to find in the field
- To secure the network, you must secure the actual network, not a theoretical infinite-resource RFC-compliant spherical cow 😊
 - So how did we fix this particular bug?

The “Fix”, As Per DJB: Source Port Randomization

- Before: 65536 to 1 odds
- After: Between 163,840,000 to 1 and 2,147,483,648 to 1 odds
- This is an improvement
 - *That’s a lot of traffic to go unnoticed, and undefended by secondary measures*
 - Not necessarily too much
- So why not go with something “perfect”?

This is not a simple attack.

- One reason there were so many questions about exactly what went wrong, is because there were so many *variants*
 - Probably a good 15 ways to run this attack
 - Family 1: Pure TTL Bypasses
 - Override glue w/ NS record (the “attack”)
 - Override glue w/ CNAME
 - Override glue w/ DNAME
 - Override glue w/ extra in-bailiwick glue
 - Family 2: Prevent the authoritative server from populating the cache
 - Ask for a nonexistent query type
 - Ask for a nonexistent query class
 - Ask for a nonexistent sibling name in bailiwick (cnn.com)
 - Just flood it with some huge amount of traffic

Florian Weimer / Brian Dowling's new PowerDNS attack

- In short, PowerDNS does not respond to certain queries it considers malformed. This in itself is not a problem, and was even thought of as a security measure. Brian and Florian, independently I think, have discovered that not answering a query for an invalid DNS record within a valid domain allows for a larger spoofing window of the valid domain. Because of the Kaminsky-discovery, this has become bad. For a sophisticated attacker, this provides no benefit. However, such a long window allows unsophisticated hackers to achieve better results.
 - Bert Hubert, PowerDNS
- Basically, recursive resolvers would pass a query to PowerDNS, which it authoritatively would ignore. This meant that there was an infinite window – the other guy wasn't even in the race

And Keep Going...

- Other methods
 - Any query type whose response is not cached
 - Anything that causes the cache to clear
 - Abusing naturally low-TTL records
 - Facebook: TTL=30
 - Google Analytics: TTL=300
 - Abusing IDS systems that block if they see an attack (2005 Black Ops)

And then there's the problem of sibling names

- Some people are trying to bring TTL's back
 - to assert that, if the TTL is 2000, then an attack will not work for 2000 seconds
- This has a fatal weakness
 - 1.google.com
 - 2.google.com
 - 3.google.com
- Can we ignore that fatal weakness?

no

- Toorcon Seattle: Used malicious injected subdomains to compromise main site
 - Instead of injecting subdomains via ad servers, now we just inject via 1.google.com etc.



Takeaway #4: It's not enough to solve 99% of the problem, if the last 1% is *really really important*

- You don't have to be perfect, you just have to be good enough
 - But if you don't secure the web, you aren't actually good enough

So what are the ground rules?

- 1) We must secure all names, not just “most”
- 2) We must secure all authoritative name servers, not just those that opt in
 - This *eventually* gets relaxed with DNSSEC, but that’s not a solution for today
 - This *does* mean that we must secure the link to the root servers, and the com servers, without actually requiring root or com to update anything
 - If root or com is busted, then we never get routed to our “secure” name servers
- 3) We must not alter the semantics of DNS

The “One Character” Patch

- One character patch was proffered, which would lock the NS record for a given name for however long the TTL was set
 - “What’s the downside to my patch ? I guess we are now holding an authoritative server to the promise not to change the NS record for the duration of the TTL, which is kinda what the TTL is for in the first place 😊”
- Problem: This changed the semantics of DNS
 - *Before, we’d get an updated nameserver for Google.*
 - *Now, it will take 95 hours.*
 - If you think any IT infrastructure would ever deploy *anything* that threatened a 95 hour outage, you’re simply wrong.

Takeaway #5: It is more important to work, than to be secure.

- You don't have to like it.
- Maybe it's right, maybe it's wrong.
- At minimum, you'll get *vastly* increased patch delays.
- Realistically, you'll get vastly decreased patch rates.
 - Attacks from bad guys might happen.
 - Failures in every day operation *will* happen.
 - Any failure that might take a month to show up is *worse* than a failure that can be tested and proved non-existent immediately.

Can we do better?

- There is indeed some demand for a fix better than port randomization, but short of DNSSEC
 - It's theoretically possible to receive the hundreds of millions to billions of packets necessary to still attack through the patch
 - Most attackers don't have GigE on the LAN, but bandwidth is only getting higher
 - DNSSEC breaks the ground rule of working against arbitrary authoritative names
 - Still interesting.
- What can we do, that still meets the ground rules?

Comprehensive Options

- Attack Mode
 - The name server can detect the gross imbalance between requests sent and responses received, and apply protections specifically to the name servers that are under attack
 - If someone is under attack, apply “extra validation” to names from their IP during the attack
- 0x20
 - DNS is case-ignoring but case-preserving, meaning a name like www.google.com can be represented also as wWw.GooGle.cOM w/ 12 bits of extra entropy
 - Only accept responses with the correct

Exception Handling

- Alas, there are exceptions.
 - If you fail to handle the exceptions, you fail the ground rule on changing the semantics.
- Attack mode exceptions
 - What does it *mean* to apply “extra validation”?
 - Double querying *usually* works, but nothing actually forces a name server to reply with the same name twice in a row
 - Akamai, Google, Facebook often won't
 - It's not enough to secure most names 😊
 - There are some possible other solutions but they're in progress

More Exception Handling

- 0x20 Exceptions
 - Some authoritative servers aren't case-preserving after all
 - Some names cannot be protected with 0x20
 - 1.a111111111 gets only 1 bit of protection
- Any solution beyond port randomization will require the mixing of multiple approaches

Takeaway #6: Elegance is less important than coverage.

- One should be as elegant as possible, but no more.
 - If the choice is between elegance, and supporting real-world scenarios that customers require, elegance will lose every single time

What of the client?

- Name servers got all the attention, but DNS clients had the same bug too
 - Fixed / predictable source ports
- Hasn't been the focus of the remediation efforts
 - Has received two MSRC fixes in the last six months:
 - 1) TXID fix, for Amit Klein's research
 - 2) Source Port fix, for my research
 - Why?

To Attack A Client

- In order to attack a client, you must know:
 - Port to send packets to
 - TXID to send packets to
- Amit Klein discovered that if you knew TXID 1, 2, 3, you could predict TXID 4 on many platform, including Windows clients
 - Most people thought this didn't matter – if you were in a position to sniff TXID 1, 2, and 3, you wouldn't need to *predict* TXID 4, because you could *sniff* TXID 4 and simply reply on it first
- Turns out Amit was right.

Port Discovery

- Three possible broken port selection algorithms
 - Fixed/Known: Always sends on 53
 - Easy – if you know the client, just use its port!
 - Fixed/Startup: Grabs a socket on boot, stays up
 - Relatively easy – always within a small range, easy to UDP port scan for
 - Sequential: First packet
 - Annoying but doable -- Hold the port open with a query to a name you control, and then you UDP port scan for that port. Eventually you'll find a port that's no longer responsive
 - Use all the arbitrary DNS query generators from earlier, which by their very nature involve a *client* making queries to a name server
 - Much easier – Let TCP leak UDP's port # 😊
 - On Windows, pre-MSRC, TCP and UDP shared the same sequential counter
 - An HTTP connection to a bad guy on port 10000, would be followed by a DNS connection to the internal guy on port 10001!

TXID Discovery

- In my bug, you don't *need* to predict TXID, since you're just going to flood 'em all.
- Amit's bug works by discovering – without sniffing – TXID 1, 2, and 3 – so that TXID 4 can be predicted and ultimately spoofed
 - How can we do that?
 - We can guess TXID *for names we control*

You Have The Right To Remain Silent

- Browsers can be made to resolve [1.victim.com](#), or [1.attacker.com](#)
- If the browser queries for [1.attacker.com](#), the Local Name Server will ask the attacker for a record.
- *When* the attacker replies to the Local Name Server, the Local Name Server will almost immediately send a packet to the browser
 - The browser will almost immediately issue an HTTP query, to the address contained in the DNS reply
- If the attacker does not reply to the Local Name Server, the browser will wait, and wait, and wait, and send no HTTP query
 - ...unless the attacker is spoofing the Local Name Server, flooding with incorrect TXIDs until he finally guesses the right one!
 - Only 64K, remember?
 - Attacker normally wouldn't know that he guessed correctly...but for that the HTTP connection went through
 - Cannot analyze protocols in isolation!

The Chain

- Client browses to a website controlled by badguy
- Client looks up after.badguy.com against Local Name Server
- Local Name Server goes to ns1.badguy.com, asking for after.badguy.com
- ns1.badguy.com **doesn't reply** to Local Name Server.
- Local Name Server thus **doesn't reply** to Client.
- Client sits around twiddling its thumbs.
 - Open port
 - Open TXID
- Ns1.badguy.com pretends to be Local Name Server, replying with all possible TXIDs
 - Knows what port from the before.badguy.com connection
 - Doesn't know what TXID – but will, on average, guess every 32K packets
 - If doesn't guess in time, oh well, try again
 - Will eventually guess correctly

Signals

- When ns1.badguy.com guesses the TXID correctly:
 - 1) A connection will go to whatever address ns1.badguy.com declared for after.badguy.com
 - 2) This will happen immediately
- Strategies
 - 1) Timing: *When* the HTTP connection arrives, see what we were sending a few milliseconds ago.
 - 100ms or so of accuracy, on a 3 second window, yields ~5 bits of data
 - Can retry
 - 2) Direction (thank you Florian Weimer): The more addresses the bad guy has, the more information he can get via which address “wins”.
 - If he has 1024 addresses, he gets 10 of 16 bits
 - If he has 65536 addresses (a class B), he wins
 - Without IPv6, that’s not happening..

Shared Signals

- There are lots of hosts out there
- Can an attacker borrow some of them?
 - Return the IP addresses of other hosts, then find out which one was “visited”?
- Alternate approaches:
 - 1) Idle Scanning
 - Find 64K boxes that don't have any traffic
 - Forge replies across all TXIDs, each with a different address destination
 - Check all boxes for sudden traffic spikes
 - Doesn't really work anymore, too many people scanning the Internet 😊

Another Path

- 2) PTR pollution
 - Find 64K networks that reverse lookup everything, and put it into a publicly visible name server
 - Forge replies, then see what's in the name server cache
 - Too noisy, can't be run multiple times in short succession

Nobody ever expects The Billy Hoffman Option

- 3) Return 64K different web servers, then read via the DOM which one you actually hit
 - Whoever it is, is going to be a host in your own domain
 - DNS rebinding to discover DNS TXID
 - Huzzah!
 - Can also find 64K sites with different cookies, and then analyze document.cookie to see which one was hit
- This goes back to Takeaway #1: You can't analyze protocols in isolation.

Of course, much easier with my attack

- Don't bother predicting the next TXID – just force repeated lookups of the target name
 - May have to play some games to clear the client cache, if there is one
- “Sniper Rifle” vs. a Nuke
 - Attack one host? Or many?
- This is already weaponized
 - * - Initiate sequence to trigger a dns lookup by the adns resolver. Send * the same range of spoofed DNS ids in a constant flood spoofed as the * primary DNS server for the host. Even a local DNS request will take * long enough to allow some amount of the spoofed DNS responses through * before the primary DNS responds. Since the resolver does not cache * results, the dns lookups can be triggered until the DNS id is * incremented within the DNS id range being spoofed.
 - h0dns spoof.c - zmda - saik0pod@yahoo.com

So, is that all?

- No. That's HOW to attack DNS. More interesting question: WHY to attack DNS.

We Start With The TLDs

- It is indeed possible to pollute com, net, org, etc.
 - Directly: com NS
 - Indirectly: A.GTLD-SERVERS.NET, B.GTLD-SERVERS.NET., C.GTLD-SERVERS.NET...
- When the bad guy poisons com, he gets all requests
 - Even requests he didn't know in advance he wanted!
 - He gets to decide:
 - What he'll poison forever (response, long TTL)
 - What he never wants to see again (delegation, real NS)
 - What he'll check out for a little while (response, short TTL)

MX Intercept: It's Not Just For the NSA Anymore

- “Remember how pissed you were when you found out the NSA had rooms where they could read everything? That’s every kid right now.” –Brad Hill
- Mail is special – has its own type of record
 - MX – Mail Exchange
- Attacker who owns com, can see who’s sending mails to who, and can pick off any he likes
 - Can silently intercept, then let the mail run off to its correct destination
 - Give himself top priority, fail to fully accept a message, then let the message fall through to the next server

Message Pollution

- 1/3rd of attacks come from direct user action
 - Loading a document
 - Downloading and installing malware
- Attacker can also accept a message, infect attachments with malware, and forward it along
 - DOC -> Infected Doc
 - EXE -> Infected Exe
 - ZIP with Password containing EXE -> ZIP with Password containing Infected EXE
 - Attacker can read 😊
 - Link to EXE -> Link to infected EXE
 - Attacker can either change link, or poison link in destination

Takeaway #7: Never Forget The Human Factor

- Again, 1/3rd of attacks come from direct user action
 - Users are asked to escalate privilege *all the time*
 - Worse, attempts to prevent users from escalating privilege lead only to users abandoning all security measures entirely
 - Block all .PL files -> Flood of password-encrypted ZIP files
- Defenders cannot optimize away the users, much as they'd like to
- Attackers will not ignore the users, not with their 1/3 failure rate
 - Anything that can make the success rate even higher will be jumped on

Shouldn't The SPAM Filter Stop This?

- SPF should notice the wrong IP
 - SPF comes from DNS
 - All SPAM filtering comes from DNS
 - Can actually hijack SPAM filters –
attacker ends up controlling mail
reception entirely

Not going there, but...

- SIP ain't looking too great either
 - SRV records are easily detectable
 - SIP INVITE/REGISTER messages look like they can contain DNS names – triggering a lookup in target networks
 - If you have an environment that explicitly uses DNS name contacts, you might even be able to choose your intercepts
- Thanks to Zane Lackey

Spidey Sense

- Obviously the entire web is affected, for a client behind a corrupted DNS server
 - Can directly poison via com corruption
 - Requires rebinding to read actual site contents
 - Can indirectly poison a single site via its subdomain library dependencies
 - Prototype.js
 - CSS scripts
 - Can indirectly poison the entire web via google-analytics.com, ad.doubleclick.net, sitemeter, or any other codebase commonly loaded via an external `<script src>` tag.
 - Hope you're not downloading any executables from the web...
- (But SSL will save us!)

The Internet is more than the Web; HTTP is more than the Browser

- Welcome to the third age of hacking
 - 1st age: Servers
 - FTP, Telnet, Mail, Web, Time
 - These were the things that consumed bytes from a bad guy
 - These are the things that got locked down
 - 2nd Age: Browsers
 - Javascript, ActiveX, Java, Image Formats, DOMs
 - These are the things that are getting locked down
 - Slowly
 - Incompletely (ActiveX Sitelock to http:// doesn't work too well right now)
 - 3rd Age: EVERYTHING ELSE
 - Check out this desktop from an Internet Cafe





We're no longer in
browserland anymore...



Remember Sidebar from Last Year?

The screenshot shows a Microsoft Internet Explorer browser window displaying a Microsoft Security Bulletin page. The browser's address bar shows the URL: <http://www.microsoft.com/technet/security/Bulletin/MS07-048.msp>. The page title is "Microsoft Security Bulletin MS07-048 - Important: Vulnerabilities in Windows Gadgets Could Allow Remote Code Execution (938123)". The page content includes a search bar, a navigation menu, and a sidebar with a search function. The sidebar contains a search box and a "Go" button, and a list of links: TechNet Security, Security Bulletin Search, Library, Learn, Downloads, Support, and Community. The main content area displays the bulletin title, publication date (August 14, 2007), version (1.0), and a "General Information" section with an "Executive Summary".

Microsoft Security Bulletin MS07-048 - Important: Vulnerabilities in Windows Gadgets Could Allow Remote Code Execution (938123)

Published: August 14, 2007

Version: 1.0

General Information

Executive Summary

This important security update resolves two privately reported vulnerabilities in addition to other vulnerabilities identified during the course of the investigation. These vulnerabilities could allow an anonymous remote attacker to run code with the privileges of the logged on user. If a user subscribed to a malicious RSS feed in the Feed Headlines Gadget or added a malicious contacts file in the Contacts Gadget or a user clicked on a malicious link in the Weather Gadget an attacker could potentially run code on the system. In all attack vectors, users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.

This is not an exception

- Browsers are *really really good client code*
 - Relatively 😊
 - They're so much more complex than anything we ever put on the server
 - We've been trying to secure them for far longer
- What do you think happens when you fuzz weak clients?

Ilja van Sprundel, dumb fuzzing IRC with ircfuzz.c

- * ircfuzz v 0.3 by Ilja van Sprundel. * so far this broke: - BitchX (1.1-final) * - mlIRC (6.16) * - xchat (2.4.1) * - kvirc (3.2.0) * - ircii (ircii-20040820) * - eggdrop (1.6.17) * - epic-4 (2.2) * - ninja (1.5.9pre12) * - emech (2.8.5.1) * - Virc (2.0 rc5) * - TurboIRC (6) * - leafchat (1.761) * - iRC (0.16) * - conversation (2.14) * - colloquy (2.0 (2D16)) * - snak (5.0.2) * - ircle (3.1.2) * - ircat (2.0.3) * - darkbot (7f3) * - bersirc (2.2.13) * - Scrollz (1.9.5) * - IM2 * - pirch98 * - trillian (3.1) * - microsoft comic chat (2.5) * - icechat (5.50) * - centericq (4.20.0) * - uirc (1.3) * - weechat (0.1.3) * - rhapsody (0.25b) * - kmyirc (0.2.9) * - bnirc (0.2.9) * - bobot++ (2.1.8) * - kwirc (0.1.0) * - nwirc (0.7.8) * - kopete (0.9.2)
- Things are a *little* better now
 - Not much
 - *You really really don't want to be talking to a malicious IRC server*
 - Lets not even talk about netsplits

Lets not forget about the biggest, most extensive clients out there

- Games
 - Gaming - The Next Overlooked Security Hole
 - Ferdinand Schober,
Security Researcher
 - We're now seeing those unifying technologies the web, and monolithic engines making their way in to these games. Automatic updates, electronic publishing systems, in-game advertisements, pay-for-item MMORPG systems all of these represent structural weaknesses that more and more people should be exploiting. Given the expectation of today's gamers a far as graphics, physics, and other frivolous crap, smaller developers have to purchase someone else's engine to get started and all of the bugs that come with it.

How do you know what to attack?

- You know where it's going
 - It tells you via DNS
 - That often tells you who it is
- But if you need more, and it's speaking HTTP...
 - Host header
 - User-Agent
 - Extra Headers

Who needs an exploit? Lured by design, upgraded by design

- Francisco Amato's EvilGrade
 - Implemented modules: ----- - Java plugin - Winzip - Winamp - MacOS - OpenOffices - iTunes - LinkedIn Toolbar - DAP [Download Accelerator] - notepad++ - speedbit
 - Bigger companies than I thought. But otherwise, yeah, we knew this was going to be a problem
 - Actually warned LinkedIn in advance

Autoupgrade Is Hard

- To succeed, your update package must be:
 - Signed.
 - Signed by you.
 - Signed by you, using the right ECU (Extended Key Usage)
 - Signed from an unrevoked signature
 - Be the same product
 - Be a new version
- Or you could use SSL, but ZOMG PERFORMANCE
- Translation: Must be Windows Update, or you're hosed 😊
 - Maybe Adobe. MAYBE.
 - See also: **“Secure Software Updates: Disappointments and New Challenges”**, Bellissimo, Burgess, Fu

Takeaway #8: Code that's never been attacked, is usually remarkably fragile.

- Web browsers got hit, web browsers got fixed
- Web servers got hit, web servers got fixed
- Exploitation appears to cause an increase in code quality, directly in the code that's attacked, and indirectly in all peer software
- *Most code has never been attacked, therefore most code is pretty fragile*
 - Depends on having never been exposed to attack
 - But exposure changes over time.

But what of SSL?

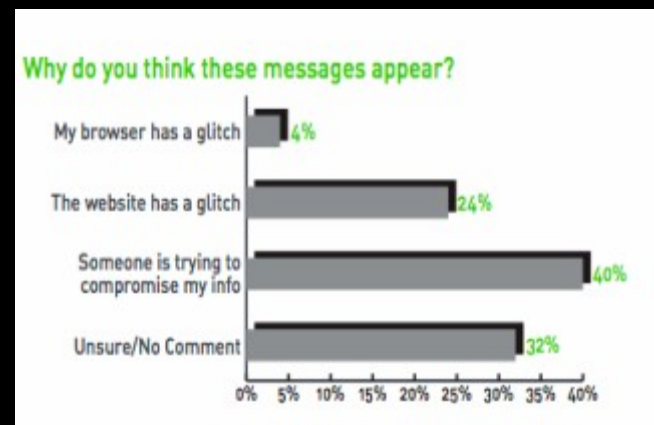
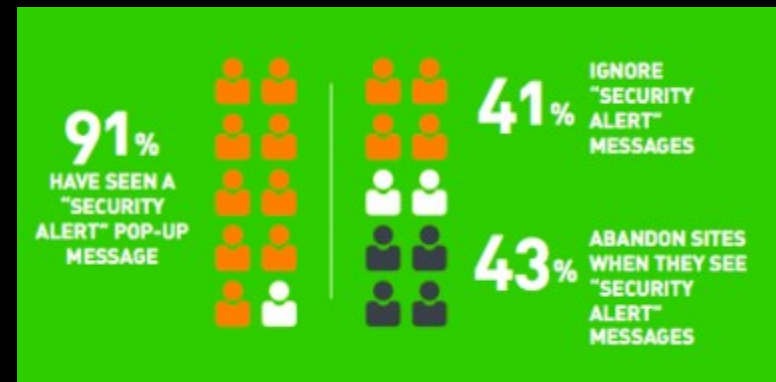
- Theoretically, DNS poisoning shouldn't matter, because everything important is protected by SSL.
 - Nothing big should ever touch HTTP!
- This is the first big test of SSL
- Has it stood up because of the strength of its crypto?
- Or has it stood up because nobody had the opportunity to play MiTM?

SSL Problem #1: It's Not Particularly Widely Deployed

- Pretty much only financial sites support 100% SSL operation
 - Takeaway #3: If it's stupid and it scales, it isn't stupid
 - We're *really really* good at making the web scale without SSL
 - **SSL needs work to make it scale better!**
 - **How did we end up without virtual hosting support, by default, mandatory, in TLS?**
- We won't bank without SSL, but we will download executables in the clear
 - This of course makes no sense.
 - Takeaway #7: Never forget the human factor.

SSL Problem #2: Users Ignore Errors

- Again, Takeaway #7: Never forget the human factor
- 41% of users *admit* to ignoring “security alert” messages
 - Source: Consumer Reports
- Actual data: When a major online bank in New Zealand had its cert expire, 99.5% of users still entered their credentials.
 - DNS-based attacker has a remarkably high success rate



SSL Problem #3: Even when we will use SSL, we arrive at it insecurely

- <http://www.paypal.com> redirects to <https://www.paypal.com>
- <http://www.bankofamerica.com> redirects to <https://www.bankofamerica.com>
- <http://www.e-commerce-site.com> redirects to <https://checkout.e-commerce-site.com>
- If you're an attacker, and you control the HTTP version of a site, where will you send users?
 - Probably not to the version of the site that cryptographically authenticates.

SSL Problem #4: Most sites leak credentials otherwise acquired via SSL.

- Mike Perry's research with Cookie Monster
- Summary: HTTP is stateless, even when run over SSL. But nobody wants to type in their password on every query, so we get a "cookie" that represents our identity.
 - If the cookie is marked "secure", it will only be transmitted to SSL (read: not DNS poisoned) endpoints of sites
 - Are enough sites protected?
 - No
 - <http://fscked.org/blog/incomplete-list-alleged-vulnerable-sites>
 - Scary, scary list
- Note, even sites *with* secure cookies can have those cookies overridden by an attacker, since HTTP can write HTTPS secure cookies (!)
 - Adam Barth / Collin Jackson's research

SSL Problem #5: Many Non-Browsers Don't Actually Validate Certs!

- Again, Takeaway #8: Code that's never been attacked, is usually remarkably fragile.
 - The Internet is more than just the Web – but SSL is generic!
 - This was *supposed* to be a strength
 - But it turns out that many of SSL's strengths are only enforced via previously exploited web browsers.
- 327,467 SSL certificates were scanned
 - 140,355 SSL certificates were *self-signed* – 42%!
 - That's not even saying the other 58% are signed by a trusted CA!
- So who's using these self-signed certs?

Mike Zusman on SSL VPN's

- The only purpose of an SSL VPN is to prevent bad guys from getting access to network traffic. Do they? Mike:
 - “Yes, you will see many SSL VPN servers on the Internet serving invalid certificates.

Cert validation varies product to product. One particular SSL VPN client I've worked with is hardcoded to only accept valid, trusted certs. If it is not signed by a trusted CA, the cert needs to be added to the local trust store. Another one allows you to turn validation on and off.

None of the clients I've seen do any caching/white listing (like an SSH client). This way once you have the SSL VPN client installed, you can connect to ANY server you need to seamlessly. Great for re-purposing attacks.”

- In other news, go read Mike Zusman's notes on SSL VPN's. He is doing some very important work.

SSL Problem #6: The Certificates Are Still Signed With MD5

- It's 2008, and we still base the security of SSL on an algorithm that was:
 - Academically discredited in 1996 by Hans Dobbertin
 - Federally discredited in 1998 by NIST
 - Publicly collided in 2005 by Xiaoyun Wang
- And still, *everyone's going to be so surprised and unprepared when it finally breaks completely.*

SSL Problem #7: Revocation Is A Myth, Especially For The Debian Case

- The browsers barely check for certificate revocation
 - Takeaway #5: It is more important to work, than to be secure.
 - Revocation “works”, but is still too slow.
- Non-browsers outside of .GOV environments don't even pretend to check revocation.

The Debian Problem is particularly worrisome

- Nobody has an efficient solution for Luciano Bello's Debian bug, where a few badly generated keys are spread across some huge number of Certificate Authority signed names
 - Bloom filters across all the vulnerable moduli are still too large ☹
- Anyone who thought to collect all the certificates that were badly generated, can impersonate all those sites, and will be able to for approximately the next five years.

Takeaway #9: There is no bug so good, that another bug cannot make it better.

- People want there to be some sort of competition between bugs, as if attackers could only choose one so they better choose wisely.
 - The reality is that attackers can blend whatever they like, and *have*
 - Nimda, *from 2001*, combined email, share pollution, IIS exploitation, and browser bugs!
- So many of the attacks described thus far – from Zusman's to Perry's, from Wang's to Bello's -- are made far more problematic by having a real world MiTM vector.

You'd think that would be obvious, but...

Package Managers As Achilles Heel

Posted by [timothy](#) on Thursday July 10, @06:53PM
from the [dip-your-computer-in-the-styx](#) dept.

An anonymous reader writes

"Researchers from the University of Arizona have released a study that takes a look at the [security of ten popular package managers](#). They were able to show all ten were vulnerable to attacks from a mirror or man-in-the-middle that allow an attacker to (along with other things) crash the system or obtain root access. Furthermore, the researchers created a fictitious administrator and company name and were able to lease a server and get it listed as an official mirror for all the distributions they tried (Ubuntu, Debian, Fedora, CentOS, and OpenSUSE). This raised the question: What keeps you up at night, the thought of attacks on your package manager or previously discussed and patched [vulnerability in DNS?](#)"

[justin samuel](#) (one of the Arizona researchers) also points out a [synopsis on CERT's blog](#).



- What is this “OR” you speak of?
- What, I can only use one bug at a time?
- DNS **provides** the Man-In-The-Middle that breaks Package Managers!
- It's not a “bug competition” not because of some ethical limitation, but because *it blinds you to actual vulnerabilities when bugs are combined*

SSL Problem #8: Certificate Acquisition Itself Depends On DNS

- Why do you think SSL certificates are valuable?
 - Anyone can buy one
 - Anyone can generate bits
 - What prevents anyone in the audience from getting a cert for www.microsoft.com?
- CA's sell bits
 - But there's some meaning applied to those bits
 - an assertion that an identity has been validated, at least to some level
 - How are identities validated by most CA's?

Say Hello To My Little Friend

- Domain Validation: How SSL Certificate Authorities use DNS to determine whether you get a certificate
 - Look up the domain in WHOIS
 - DNS address lookup
 - Send an email to the mail address on file
 - DNS MX record lookup
 - Visit the web page and look for a file
 - DNS A record lookup
- **Guess how secure that is in the face of a DNS attack?**

Hello My Little Friend

- Actually, we're doing OK
 - For some reason, every CA scrambled during the month of July to make sure that they were patched
 - Thank you, Tom Albertson, Kelvin Yiu, Zot O'Connor of Microsoft 😊
 - Thank you Verisign, Comodo, Digicert, Trustwave, everyone who kept this matter secret
- www.SSLShopper.com figured it out...but they think the answer is to buy EV certs. Unfortunately...

And what about EV?

- EV is a *display* mechanism, not a *code security* mechanism
 - Extended Validation. The browser's scripting policy does not distinguish between HTTPS connections that use an Extended Validation (EV) certificates from those that use non-EV certificates. For example, PayPal serves `https://www.paypal.com/` using an EV certificate, but a principal who has a non-EV certificate for `www.paypal.com` can inject script into the PayPal login page without disrupting the browser's Extended Validation security indicators; see Figure 2.
 - “Beware of Finer Grained Origins”, Collin Jackson, Adam Barth
- So, no. EV does nothing in the face of also-extant Domain-Validated Certificate

What Else Is Interesting?

- CA's have web interfaces to manage previously issued certs...
 - ...web interfaces you have to sign into.

When I said The Web was broken, I wasn't talking about just its clients.

(confused?)

Account login

Email address

PayPal password

Log In

Forgot your [email address](#) or [password](#)?

New to PayPal? [Sign up](#).

Email:

Password:

Remember me

Login

[Forgot Password?](#)

Log In | **Sign Up!**

Email:

Password:

Remember Me

Log In

[Forgot your password?](#)

Windows Live ID:

Password:

[Forgot your password?](#)

Remember me on this computer (?)
 Remember my password (?)

Sign in

Sign in to Yahoo!

Are you protected?
Create your sign-in seal.
(Why?)

Yahoo! ID:

Password:

Keep me signed in
for 2 weeks unless I sign out. [Info](#)
[Uncheck if on a shared computer]

Sign In

[Forgot your ID or password?](#) | [Help](#)

Sign in to your account

Back for more fun? Sign in now to buy, bid and sell, or to manage your account.

User ID
[I forgot my user ID](#)

Password
[I forgot my password](#)

Sign in to Gmail with your Google Account

Username:

Password:

Remember me on this computer.

Sign in

[I cannot access my account](#)

Forgot My Password Modes

- This is a generic lost credential technique
 - Generally, a fully automated way to get into an account without the password
 - Near-universally deployed
 - Three modes seen in the field
 - 1) Password Leak: Just mails you your password. Somewhat uncommon.
 - 2) Reset Password: Mails you a link that resets your password. Guarantees detection of attack. Most common.
 - 3) Reset w/ Additional Protections: Mails you a link, and makes you jump through hoops. Somewhat common on high-value sites.
 - #1 and #2 are trivial to pop (though #2 has side effects).

Attacking Forgot My Password systems

- It's just an email, meaning, it forces a lookup to an attacker controlled name
 - What did we need, to pollute com?
 - *This means any lookup can spawn any other arbitrary lookup, on demand*
 - 1. Force a lookup to 1.badguy.com
 - 2. Reply with a referral (NS or CNAME) to 1.foo.com
 - 3. Follow the reply with an immediate stream of fake replies from the foo.com name server
- Not complicated. After poisoning, request password for arbitrary account
 - It will do an MX lookup
 - You will see the MX lookup
 - Game over ☹

News

- Fixed (beyond just the CA's)
 - Google
 - Live
 - Yahoo
 - Paypal
 - eBay
 - MySpace
 - Facebook
 - LinkedIn
 - Bebo
 - Craigslist
 - LiveJournal
 - Hi5
 - Citrix (GoToMyPC)
- This is very, very cool. Thank you to all the companies who worked with me on this!
 - Ow my cell phone bill 😊

Reality Check

- No way we got everyone

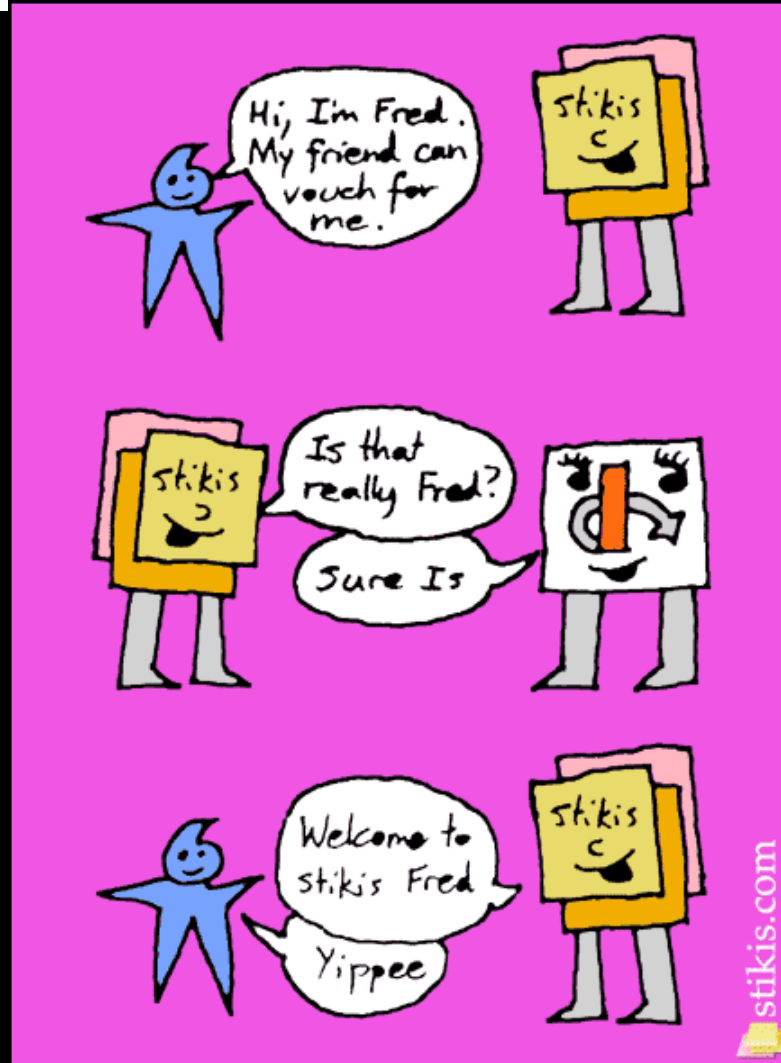
Results 1 - 10 of about 8,670,000 for "forgot my password". (0.21 seconds)

Results 1 - 10 of about 88,000,000 for "forgot your password". (0.29 seconds)

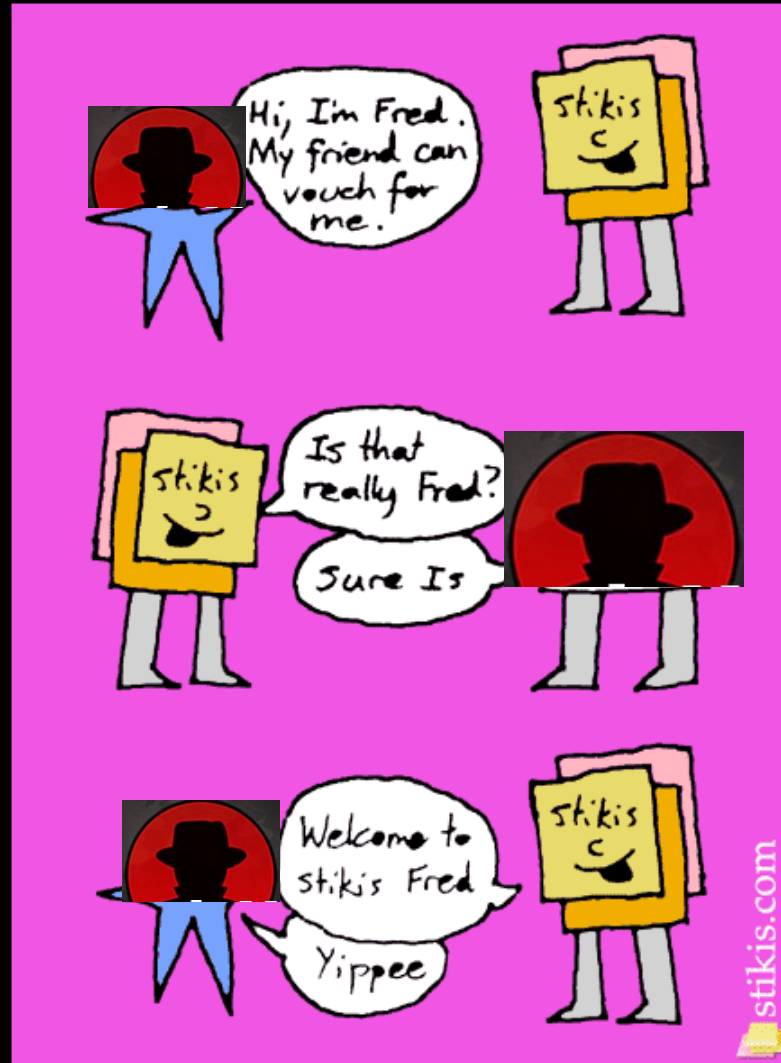
Results 1 - 10 of about 69,600,000 for "forgot password". (0.23 seconds)

- But we did ok.

Would OpenID have helped?



How did Stikis find the “friend”? Hint: DNS



What of OpenID with HTTPS?

- Should be fine...
 - Alas: Ben Laurie found that a couple major OpenID providers were using HTTPS...
 - But with a Debian misgenerated certificate
 - DNS + OpenID + Debian NRNG = WIN
- **glorious**

Takeaway #10: Flawed Authentication Is The Unifying Theme Of 2008's Major Bugs

- Specifically:
 - **Weaknesses in authentication and encryption, some which have been known to at least some degree for quite some time and many of which are sourced in the core design of the system, continue to pose a threat to the Internet infrastructure at large, both by corrupting routing, and making those corrupted routes problematic.**

Going Down The Line

- My DNS: Failure to correctly authenticate DNS reply.
- Perry's Cookie Monster: Failure to deliver authentication blob to secure endpoint.
- Zusman's SSL-VPN's: Failure to authenticate foreign endpoint
- Bello's NRNG: Failure to synthesize unique authentication material.
- Laurie's OpenID: Failure to synthesize unique authentication material *on a centralized authentication platform*
- Amato's Evilgrade: Failure to authenticate update packages
- University of Arizona's Package Manager flaws: MORE failure to authenticate update packages
- Pilosof's BGP: Failure to authenticate the data supplied by an authenticated BGP peer
- ...and what about Hardakar's SNMPv3 bug, probably the single coolest auth bug in years?

On Hardakar's SNMPv3 Flaw

- SNMPv3 – Simple Network Management Protocol, Version 3
 - Useful mechanism for monitoring and maintaining infrastructure
- SNMPv3 uses a fairly standard challenge-response authentication system
 - Server provides the challenge
 - Client provides a response, via HMAC
 - All good so far...
- Client can declare how many bytes his response needs to get correct.
 - Client can declare he only needs to get 1 byte right.
 - There's only 256 possibilities...
- So yes -- *yet another auth bug* – but this one chains with DNS in interesting ways
 - *Lots of infrastructure is behind firewalls* – can't break SNMPv3 without getting past them.
 - Can DNS help?

Let Us Discuss The Inconvenient Matter Of Reverse DNS

- You know, we *also own in-addr.arpa*
 - This is the space that, when you look up 1.2.3.4, returns “a.b.c.d.com”
- What can you do with this?
 - Obvious: Spoof log entries in Apache
 - Apache Double-Reverse Lookup Log Entry Spoofing Vulnerability
 - Martin Kraemer, 2002
 - Apache will log the name that reverse DNS provides, *if* that name resolves back to the same IP
 - Well, we control both forward and reverse DNS, so heh
 - May even be able to fake numeric TLDs...
 - 6.6.6.6 IN PTR 1.2.3.4
 - 1.2.3.4 IN A 6.6.6.6
 - Possibly (probably) stopped by client side APIs
 - » Never assume an API is every smarter than it had to be to ship

Lets Party Like It's 2007

- Black Ops 2007: Possible to use browser plugins to connect to internal resources behind firewalls
 - You browse to my site, I get TCP, maybe UDP, to your site
 - Flash
 - Java
 - Flash secured this with crossdomain.xml, per IP address
 - Java secured this with...reverse DNS
 - Which we own.
 - I can has 1.0.0.10.in-addr.arpa!
 - Full TCP and UDP from any browser in your org, to any host behind the firewall, if you don't patch DNS
 - And have Java
 - Bonus: IPsec!
 - Note: You don't get 127.0.0.1, because 1.0.0.127.in-addr.arpa has an authoritative record on most servers.
 - MAY get per-interface bind though...
 - To get 127.0.0.1 out of Java, see John Heasman's talk

And thus, SNMPv3

- If you can spoof arbitrary UDP packets, behind the firewall, you can spoof arbitrary SNMPv3 packets too
 - Have *you* patched against the SNMPv3 bug?
- Takeaway #9: There is no bug so good, that another bug cannot make it better 😊

Spreading The Phun

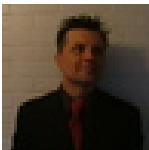
- If you have arbitrary socket access...
- ...then you can spew packets on Port 53...
- ...then you can scan for more name servers behind the corporate firewall...
- ...and you can poison Internet DNS.
 - Maybe.

Difficulty: Cannot poison authoritative on servers...

- ...with DNSRake, anyway.
- Can still poison authoritative with client attacks
- Not everyone is talking directly to primaries/secondaries
 - Isolated Split Brain is *very* safe
 - No external DNS to desktop + fake roots = attacker never sees any internal traffic
 - AD is heavy on primaries – also safe
 - No cache to corrupt
 - Split brain environments are heavy on forwarders – somewhat safe
 - Attacker may not be able to spoof destination of forwarder
 - If internal servers are talking to a normal name server, that's recursing both to the Internet and to internal names: *Game On*

When Internal DNS Goes Bad

- *So much bad behavior behind the firewall, all directed via DNS!*
 - Telnet
 - SNMP – queries and traps
 - Auth servers (RADIUS/TACACS)
 - Backup/Restore
 - SOA architectures
 - Resolve back to *names*, DNS determines *addresses*
 - Backend Databases



[shawnmoyer](#): [@hevnsnt](#) I think something about DSNs. Apparently [@dakami](#) found a way to hijack everyone's **ODBC** connections, or something. Plz advise.

about 16 hours ago · [Reply](#) · [View Tweet](#)

Even if *internal* DNS is hard to hit, external dependencies are fair game

- *So many connections* between companies
 - DNS controls how the servers find each other
 - The link *might* be secured
 - If SSL is used – is anyone actually checking the certificates?
 - Are you sure?
 - If IPsec is used – is it tied to destination subnet?
 - **DNS changes destination subnet, therefore DNS can change IPsec rules.**

The ultimate external dependencies

- Payment processing / Offsite backup
 - *Now is not a good time to have an insecure link to your offsite stores*
 - I'm not saying anyone does. But if there's a scintilla of a chance, *patch patch patch*
- SNMP against the Internet
 - If you are using SNMP to log into machines on the Internet, you're probably using DNS to find them
 - Interesting interactions with SNMPv3 bugs?
- Search Engine Population
 - There's Search Engine Optimization, and there's this
- CDN population – CDN's are populated by:
 - Providing the CDN a URL
 - Uses DNS, pulls data
 - Treating the CDN as a proxy
 - Uses DNS, pulls data
 - *It would be really, really bad if Akamai etc. DNS went bad*

Summary

- DNS servers had a core bug, that allows arbitrary cache poisoning
 - The bug works even when the host is behind a firewall
 - There are enough variants of the bug that we needed a stopgap before working on something more complete
- Industry rallied pretty ridiculously to do something about this, with hundreds of millions protected
- DNS clients are at risk, in certain circumstances
- We are entering (or, perhaps, holding back a little longer) a third age of security research, where all networked apps are “fair game”
 - Autoupdate in particular is a mess, broken by design (except for Microsoft)
- SSL is not the panacea it would seem to be
 - In fact, SSL certs are themselves dependent on DNS
- DNS bugs ended up creating something of a “skeleton key” across almost all major websites, despite independent implementations
- Internal networks are not at all safe, both from the effects of Java, and from the fact that internal routing could be influenced by external activity
 - The whole concept of the fully internal network may be broken – there are just so many business relationships – and, between IPsec not triggering and SSL not being cert-validated, these relationships may not be secure
 - We’re not even populating CDN’s securely!

Meta-Summary

- Takeaway #1: Protocols Cannot Be Understood In Isolation
- Takeaway #2: Everything You Do Can Be Used Against You
- Takeaway #3: If It's Stupid And It Scales, It Isn't Stupid
- Takeaway #4: It's not enough to solve 99% of the problem, if the last 1% is really really important
- Takeaway #5: It is more important to work, than to be secure.
- Takeaway #6: Elegance is less important than coverage.
- Takeaway #7: Never Forget The Human Factor
- Takeaway #8: Code that's never been attacked, is usually remarkably fragile.
- Takeaway #9: There is no bug so good, that another bug cannot make it better.
- Takeaway #10: Flawed Authentication Is The Unifying Theme Of 2008's Major Bugs.

Lessons Learned

- We have to get better at fixing infrastructure.
 - We got lucky with this bug.
 - The next one will not be so “smooth”
 - Disaster recovery planning needs to include how to handle the discovery in a flaw *in any mission critical code anywhere*
 - Servicability needs to start becoming a more important purchasing metric.
 - Servicability is, ultimately, the measure of software flaw survivability.
- Cooperation across competitors, and researchers, can indeed be very productive
 - 120M users from just one infrastructure provider
- A lot of people just do not realize the degree to which security best practices have been ignored for years
 - DNS should not have been capable of this much damage.
 - It was. Why?

Bottom Line

- We are doing a lot of things insecurely.
 - Even with DNS fixed, there are other scenarios in which unencrypted IP traffic is lost to an attacker
 - That attacker is capable of way more than he should be.
 - More than I've even said here.