

DNS

2008 and the new (old) nature of critical infrastructure

Dan Kaminsky
Director of Penetration Testing
IOActive, Inc.

What a year!

- Significant flaw found in DNS
 - You might have heard about it
- Pretty extensive simultaneous patching operation ensued
 - Microsoft
 - Linux / ISC
 - Sun
 - Cisco
 - All released patches on July 8th
- Expected patch rate: 50% of servers after

History

- I have never been a DNSSEC supporter.
- I've been researching DNS for many years, and I've been – at best – neutral about the technology.
 - I just didn't think it mattered, and the engineering effort never seemed to be going well.
- What changed?
 - Software engineering realities became too obvious to ignore.

The Hypothesis

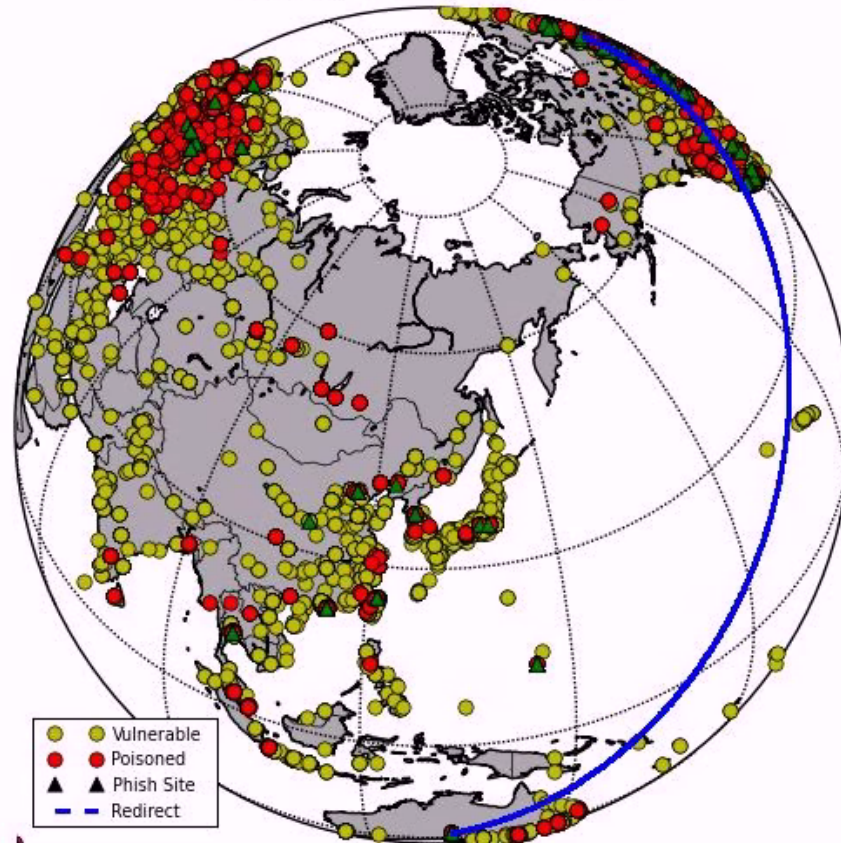
- DNS is the only real way to scale across organizational boundaries.
- Because DNS is insecure, its insecurity infects everything that uses it.
- Because DNS is insecure, security technology refuses to use it.
 - Security technology appears thus to have trouble scaling
- DNS is thus the common cause of security issues, and our inability to scalably fix them. Therefore, we need DNSSEC.
 - But is anyone actually out there, exploiting DNS, so that they can exploit all the things built on DNS?

Acute to Chronic

- We expected 50% patch rate after a year
- We got 66% patch rate after a month
 - Higher, if you consider exposure by user
- The Internet survived
 - It always survives, so that shouldn't be too surprising
- But things aren't perfect either
 - There's still a decent chunk of the network that can be easily poisoned
 - Is anyone actually doing it?
 - David Dagon, Manos Antonakakis, and Luo 'Daniel' Xiapu from Georgia Tech have been monitoring the situation closely

Attacks In The Real World

Jul 17, 2008, 05:33:00



Attacks are happening.

- It is difficult to detect poisoning attacks
 - The evidence is written in disappearing ink – you're poisoning a cache, which has a record expiring in some attacker controlled number of seconds
 - There are many, many caches
 - You can't remotely check all of them, but you can remotely poison all of them ☹️
- According to Dagon et al:
 - 1-3% of monitored unpatched nameservers have had a poisoning event detected
 - Confirmed phishing attacks have been found
 - The attackers are being sneaky

eBay in the wrong network (from Dagon et al)

Example Analysis: ebay.com resolved in ASN3462

ip	count	class	owner
...			
66.135.205.11	32	1	EBAY - eBay, Inc
66.135.221.10	13472	1	EBAY - eBay, Inc
66.135.221.11	13728	1	EBAY - eBay, Inc
66.135.223.137	20496	1	EBAY - eBay, Inc
66.208.160.87	9	4	KEC-NET - Kentucky Educational Computing Network
66.208.160.88	3	4	KEC-NET - Kentucky Educational Computing Network
66.209.160.87	6	4	ASN-CXA-ALL-CCI-22773-RDC - Cox Communications Inc.
66.209.160.88	2	4	ASN-CXA-ALL-CCI-22773-RDC - Cox Communications Inc.
66.211.160.87	8640	1	EBAY - eBay, Inc
66.211.160.88	8500	1	EBAY - eBay, Inc
192.12.94.30	12	1	FGTLD - VeriSign Global Registry Services
192.42.93.30	12	1	GGTLD - VeriSign Global Registry Services
...			

Sneaky, Sneaky Bastards (We see 'em hiding). (More from Dagon et al)

Example Analysis: Chameleon Poisoning

Fake	Real	Domain
66. 208 .168.209	66. 211 .168.209	paypal.com
66. 133 .221.10	66. 135 .221.10	ebay.com
157. 165 .224.26	157. 163 .224.26	cnn.com

The Flaw (1999 Edition)

- 1999: DJB says 16 bit transaction ID's on queries aren't enough – attacker can brute force and guess responses
 - DNS community responds: “There has to be a query waiting for a response, for an attacker to guess a response. The TTL – Time To Live – limits how rapidly an attacker can force new queries awaiting responses. So if the TTL is one day, an attack will take years!”
 - This *almost* became an RFC – “Forgery Resilience” – advocating long TTL's

The Flaw (2008 Edition)

- 2008: I point out that there are many, *many* ways to get around the TTL defense
 - Really, that's it.
 - *Maybe* I also found that since the attacker controls when the query occurs, he can reliably get hundreds of replies in before the real reply arrives
 - Without the TTL slowing down the attack, the attack takes seconds
 - The defense against DJB's attack didn't work
 - But then, it was 1999, most security in 1999 didn't work 😊

Nature Of My TTL Bypass (There are many others)

- 1) Force lookups for *sibling names* – 1.google.com, 2.google.com, etc. Since they're not cached, 1/65536 lottery for guessing correct TXID keeps getting hit
- 2) Pretend to be the legitimate name server, responding with:
 - 83.google.com IN NS www.google.com
www.google.com IN A 6.6.6.6
- 3) Since the server you're sending messages for is *in-bailiwick* to google.com, it's allowed to provide this new address for www.google.com while answering 83.google.com.
- There's code for this in Metasploit

Has Anyone Here Tested The Attack Code In Metasploit?

- It works:
 - Very reliably in testing
 - Against almost all name servers
 - Against almost all names
- It doesn't work:
 - Necessarily as well, or as quickly, in the field
 - Why?
 - This is a very interesting question.

A Question Of Trust

- BIND9 is a little more paranoid than many name servers
 - Nominum's pretty paranoid too
- If there is an answer in cache that came from the ANSWER section, the added data in ADDITIONAL cannot override it, even the new data comes from a source that's in-bailiwick
 - So this is why Metasploit's `bailiwicked_host` is so reliable on a test instance of BIND9 that's just been booted up, and less so on a server in the field
 - In the field, you have to wait for the cached record to expire

Not All Answers Are Found In The Same Place

- Many answers in a DNS cache were *originally acquired* via ADDITIONAL section
 - MX Records provide a list of mail servers, and *additionally* their IP addresses
 - CNAME Records provide the “Canonical Name” for a server, and *additionally* the IP of that server
 - CNAME may be returned for any type
 - Additional IP may show up in Answer section, unclear if treated as an Answer though
 - NS Records provide the next Name Server to delegate to, and *additionally* the IP address of that server
 - May also be returned for any type
 - NS comes in from AUTHORITY, and is thus not an ANSWER that’s difficult to budge
 - This is **by design** – NS’s are long TTL records, if they could not be overridden by anything you might see longer outages

These Imply A Series Of Attacks

- MX records don't get the ANSWER defense, so they're easy to hit
- CDN's cause major sites to be hosted via CNAMEs, so they're easy to hit
 - www.google.com, www.whitehouse.gov,
www.navy.mil
- Cached records need to expire *eventually*, so all names eventually fall to the NS attacks
 - Metasploit's Bailiwicked_domain is thus, in the long term, much more effective than Bailiwicked_host

And Just To Remind...

- Nonexistent subdomains can't already be cached, so they're easy to inject
 - NXDOMAIN replacement attacks on web security model from Jason Larsen and I, see http://www.doxpara.com/DMK_Neut_Toor.ppt
 - Attacks against Java's socket policy – most IP addresses don't have auth.4.3.2.1.in-addr.arpa style addresses registered
- Subdomains that naturally have low TTL's have their ANSWERs expire naturally
 - www.facebook.com
 - Also common for CDNs
 - Luis Grangeia's DNS Cache Snooping (querying the server with +norecurse / RD=0) lets attacker limit attacks to just when the target ANSWER is out of cache

To Be Clear

- This is why we were so insistent on deploying Source Port Randomization
- The rule with cache policy: There's always another hole!
 - Nicholas Weaver from UCB is trying to prove me wrong
 - He may very well 😊
- Does that mean every attack survived perfectly, given NXDOMAIN cache clearing?
 - BTF (Behind The Firewall) attacks are a little harder

Getting Our Universal Attack Working Against BIND again

- Ah, no ☺
- Florian Weimer discussed some very interesting NXDOMAIN semantics
 - NXDOMAIN means there are no records *of any type* for an entire domain – and if there any cached, all must be destroyed
 - There are actually 65,536 types
 - So:
 - 1) Poison NS for a given domain
 - 2) Flood DNS server with requests for incrementing types of the name you want to clear
 - 3) Flood with NXDOMAIN replies. You will eventually get one through
 - Can use Cache Snooping to verify
 - 4) Force a lookup to a sibling name. It will come to your NS, where your ADDITIONAL record for the target name will now have no ANSWER in its way.
- Florian has another trick where he CNAMEs off another type – doesn't trick BIND

Behind Enemy Lines

- BTF (Behind The Firewall) DNS attacks are more difficult, because you don't get to send queries to the victim server yourself
 - The victim server must look up 1.google.com, 2.google.com, etc, in order to be vulnerable to false replies for those names
 - However, there are many applications that will allow relatively untrusted people the ability to force a DNS lookup
 - Web Browsers
 - Mail Servers
 - See Black Ops 2008 Talk for details
 - These applications let you specify a *name*, but they don't let you specify a *type*, so you can't play the NXDOMAIN game
 - But do we really need it?

Hijacking Traffic From Behind The Firewall

- If you can force a mail server to look up an arbitrary record, do you force it to look up 1.google.com, 2.google.com, 3.google.com, and so on?
 - No! Because who knows when the application will get around to actually resolving those records? It could take thousands of milliseconds!
- Force the mail server to look up *your own* MX record
 - DNS delegates – so your reply to the MX request can force other requests
 - Including for out-of-bailiwick names like 1.google.com, 2.google.com, and so on
 - MX records can contain many names, and they'll all be resolved immediately (dozens of milliseconds)
 - MX records can also be given a short TTL, so when none of the attempted poison targets accept the mail, the MTA's retry will trigger a whole new cycle

What You Get

- Mail poisoning immediately
 - When you forge the fake NS for 83.google.com, you can override the ADDITIONAL mail records immediately, even on BIND
- A records eventually
 - Alas, cannot use NXDOMAIN cache clearing – no way to send a referral that changes the type

The Rub

- None of this should matter
 - No important systems should have been vulnerable
 - “I fail to understand the seriousness with which this bug is handled though. Anybody who uses the Internet has to assume that his gateway is owned.”
- What actually happened
 - Anybody != Halvar Flake

Not Repeating All The Slides, But...

- “Secure” systems are actually pretty rare in the field
 - Most things don’t even bother
 - Vast majority of the web
 - Email
 - Non-browser network applications
 - Those that try, mostly fail
 - 41% of SSL certs are self-signed
 - “Who are you encrypting to?” “I DON’T KNOW!”
 - Non-browser network applications that use SSL tend not to care if the cert is signed by anyone
 - There are some pretty scary implications
 - Automatic updaters are non-browser network applications that assume DNS is safe
 - SSL certificates depend on email to authenticate receivers
 - “Forgot My Password” systems bypass auth entirely
 - I don’t think people understand how serious that is

1) Find victim site

Drupal - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://frontend.doxpara.com/

foo

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source

Drupal

User login

Username: *

Password: *

- [Create new account](#)
- [Request new password](#)

The Fresh Prince Of Bel Air

Submitted by Little Bobby Tables on Tue, 12/23/2008 - 00:53.

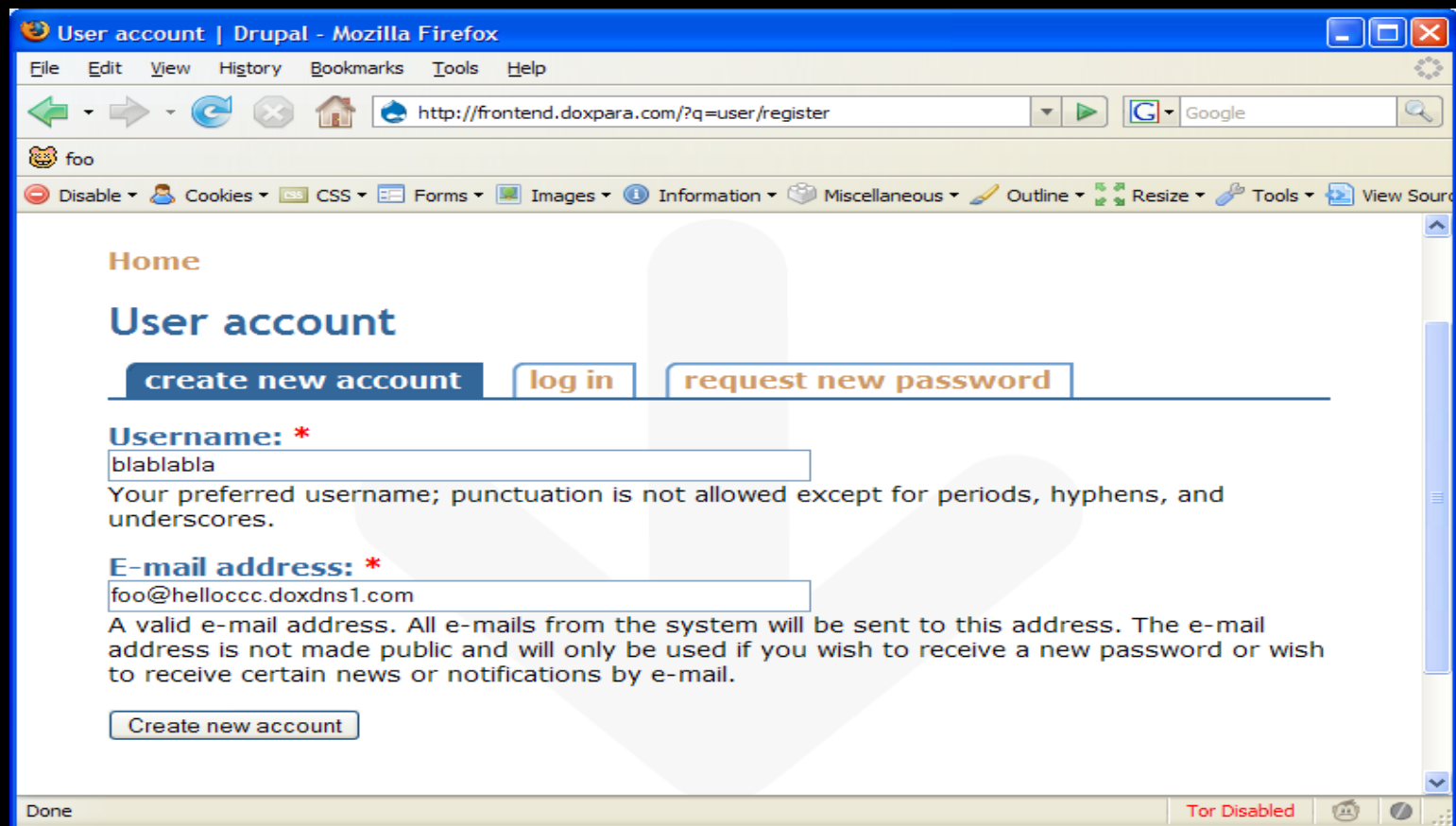
"Fresh Prince Of Bel-Air (Theme Song)"

Now, this is a story all about how
My life got flipped-turned upside down
And I liked to take a minute
Just sit right there
I'll tell you how I became the prince of a town called Bel Air

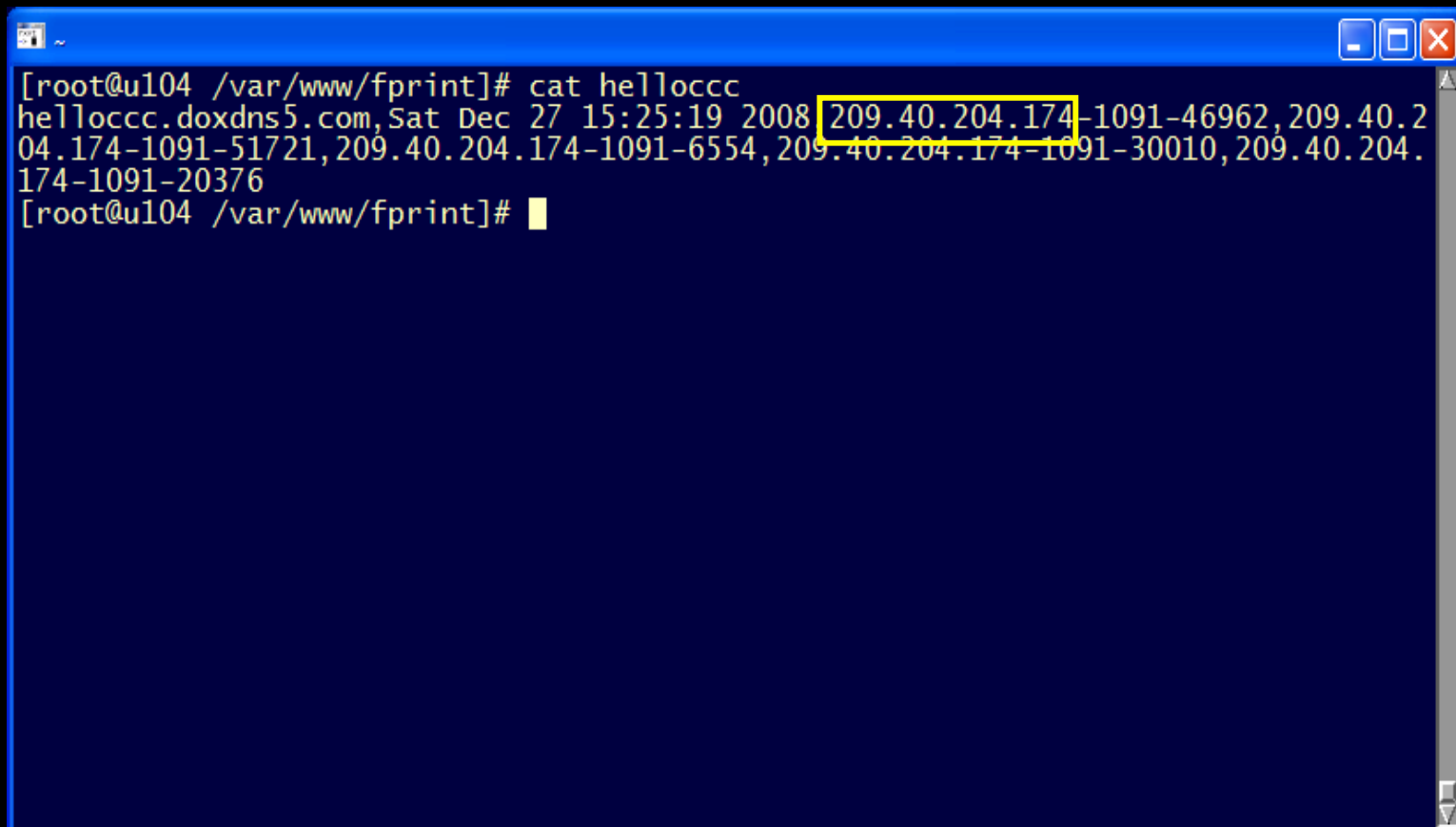
In west Philadelphia born and raised
On the playground was where I spent most of my days
Chillin' out maxin' relaxin' all cool
And all shootin some b-ball outside of the school
When a couple of guys

Done Tor Disabled

2) Force an email to be sent to a “test domain” (forces DNS lookup)



3) Check IP of DNS server used by mail server.



```
[root@u104 /var/www/fprint]# cat helloccc
helloccc.doxdns5.com,Sat Dec 27 15:25:19 2008 209.40.204.174-1091-46962,209.40.204.174-1091-51721,209.40.204.174-1091-6554,209.40.204.174-1091-30010,209.40.204.174-1091-20376
[root@u104 /var/www/fprint]#
```

4) Build name server that claims all addresses

```
[root@u104 ~]# dig @attacker.doxpara.com foo.com mx
; <<>> DiG 9.4.2 <<>> @attacker.doxpara.com foo.com mx
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52241
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;foo.com.                IN      MX

;; ANSWER SECTION:
foo.com.                10     IN      MX      10 mail.foo.com.

;; ADDITIONAL SECTION:
mail.foo.com.          10     IN      A       209.40.204.236

;; Query time: 24 msec
;; SERVER: 209.40.204.236#53(209.40.204.236)
;; WHEN: Sat Dec 27 14:54:25 2008
;; MSG SIZE rcvd: 62
```

5) Hijack to admin

```
~  
[*] Sent 2000 queries and 90000 spoofed responses...  
[*] Recalculating the number of spoofed replies to send per query...  
[*] race calc: 25 queries | min/max/avg time: 0.12/0.46/0.18 | min/max/avg rep  
lies: 0/47/27  
[*] Now sending 20 spoofed replies from each nameserver (2) for each query  
[*] Poisoning successful after 2750 queries and 120000 responses: doxpara.com. =  
= attacker.toorrr.com  
[*] Auxiliary module execution completed  
msf auxiliary(bailiwicked_domain) > set  
  
Global  
=====  
  
No entries in data store.  
  
Module: spoof/dns/bailiwicked_domain  
=====  
  
Name      Value  
----      -  
DOMAIN    doxpara.com  
NEWDNS    attacker.toorrr.com  
RECONS    208.67.222.222  
RHOST     209.40.204.174
```

6) Find Admin's Name

Drupal - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://frontend.doxpara.com/

foo

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source

Drupal

User login

Username: *

Password: *

Log in

- Create new account
- Request new password

The Fresh Prince Of Bel Air

Submitted by **Little Bobby Tables** on Tue, 12/23/2008 - 00:53.

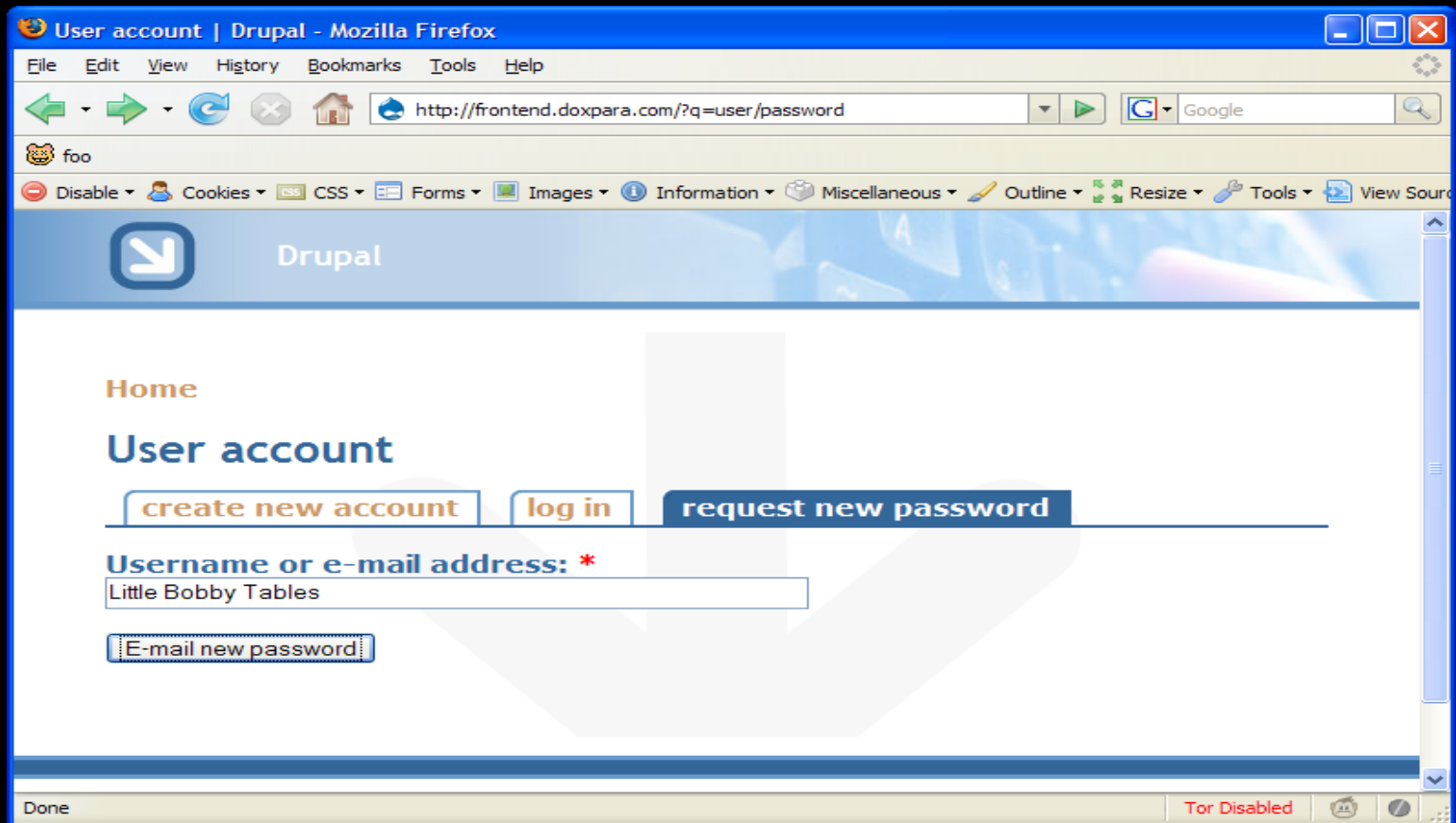
"Fresh Prince Of Bel-Air (Theme Song)"

Now, this is a story all about how
My life got flipped-turned upside down
And I liked to take a minute
Just sit right there
I'll tell you how I became the prince of a town called Bel Air

In west Philadelphia born and raised
On the playground was where I spent most of my days
Chillin' out maxin' relaxin' all cool
And all shootin some b-ball outside of the school
When a couple of guys

Done Tor Disabled

7) Forget Admin's Password



8) Click recovery link (wrote a small mail server)

```
root@attacker: /etc/mysql
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8; format=flowed
X-Mailer: Drupal
Message-Id: <20081227192309.A0EC7140D1@frontend>
Date: Sat, 27 Dec 2008 19:23:09 +0000 (UTC)
From: www-data@frontend.doxpara.com (www-data)
Content-Transfer-Encoding: quoted-printable

Little Bobby Tables,

A request to reset the password for your account has been made at Drupal.

You may now log in to frontend.doxpara.com by clicking on this link or copying a
nd pasting it in your browser:

http://frontend.doxpara.com/?q=user/reset/1/1230405786/c242bc17d80f026ab06786359
d2b54f3

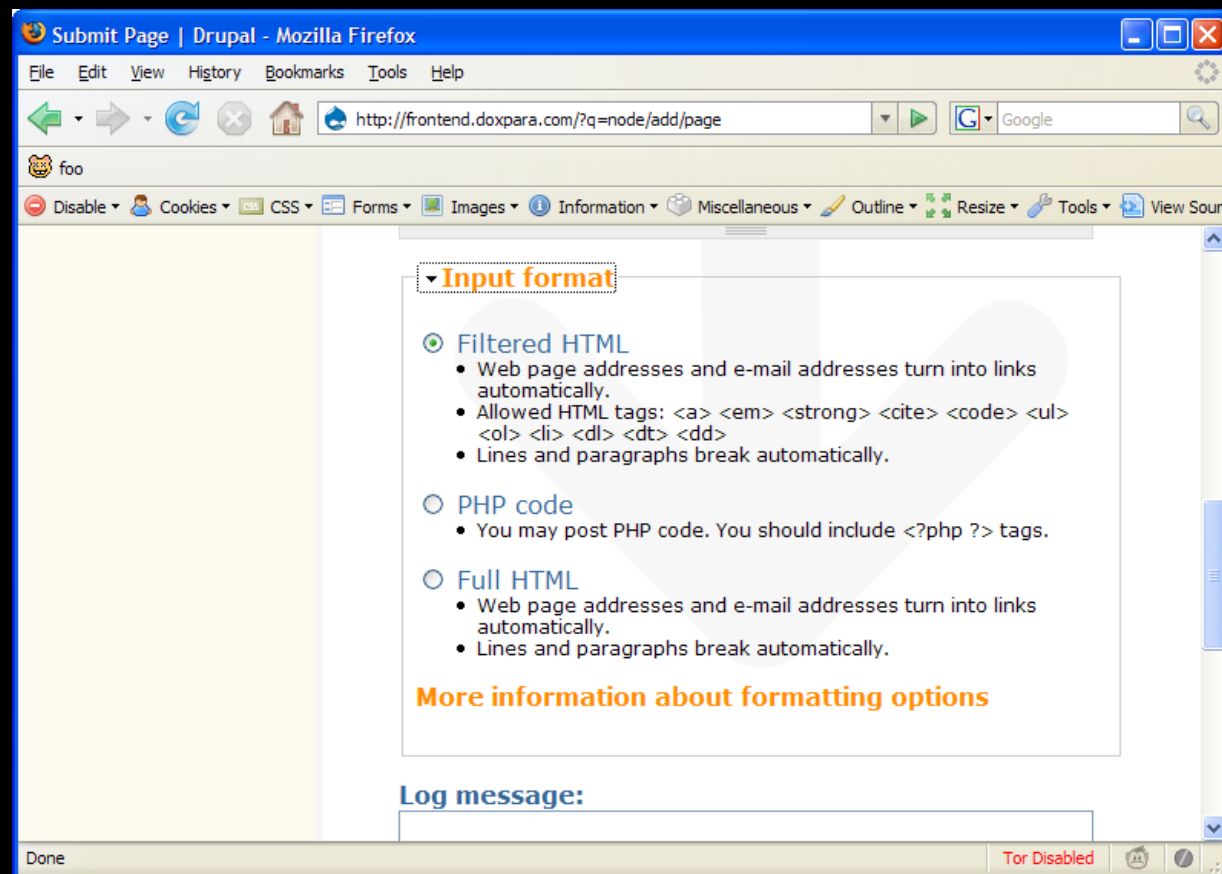
This is a one-time login, so it can be used only once. It expires after one day
and nothing will happen if it's not used.

After logging in, you will be redirected to http://frontend.doxpara.com/?q=user/
1/edit so you can change your password.
```

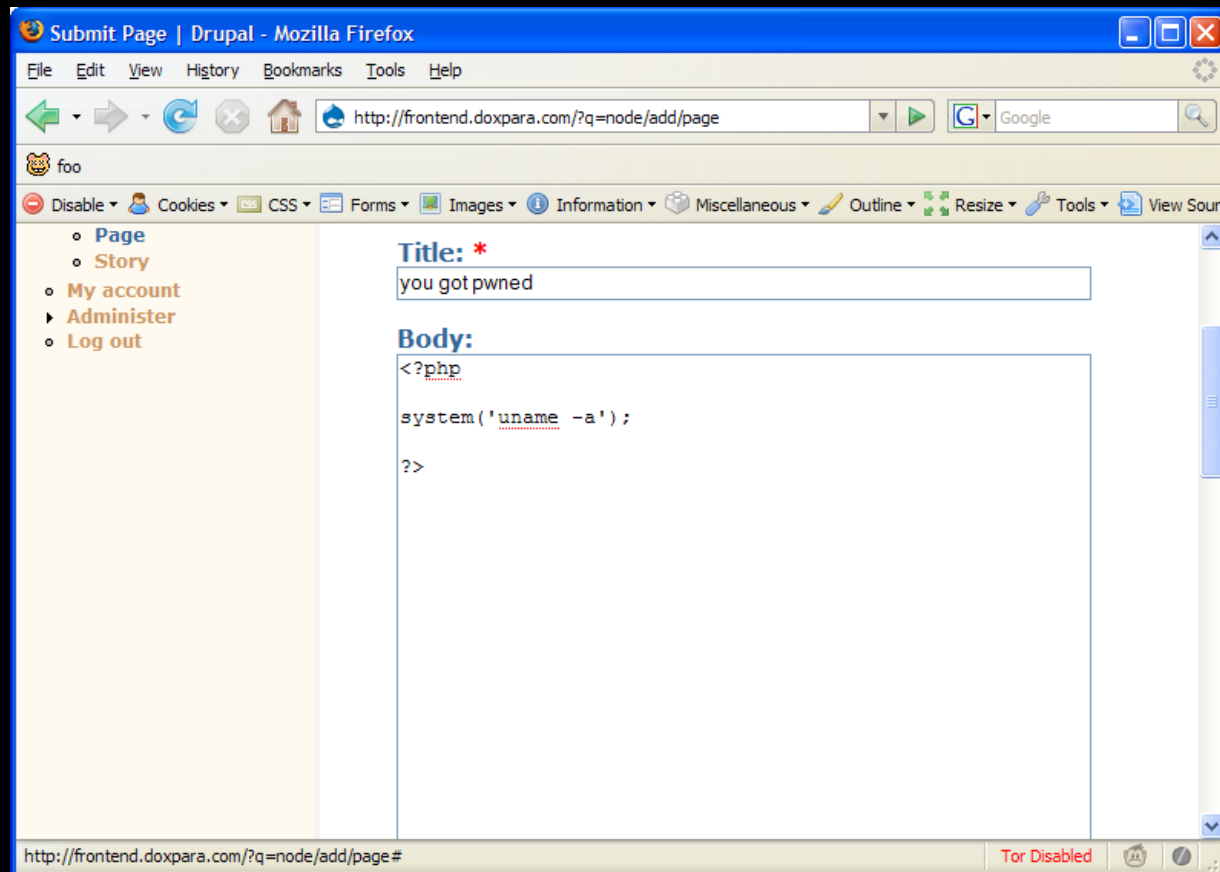

9) Enter Administrative Interface



10) Post content. Be sure to select “PHP Code”



11) Post PHP



12) Uh oh

Preview | Drupal - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://frontend.doxpara.com/?q=node/add/page

foo

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source

Drupal Edit primary links

Little Bobby Tables

- ▼ Create content
 - Page
 - Story
- My account
- ▶ Administer
- Log out

Home » Create content » Submit Page

Preview

you got pwned

Linux frontend 2.6.18-53.1.13.el5xen #1 SMP Tue Feb 12 14:04:18 EST 2008 i686 GNU/Linux

Title: *
you got pwned

Body:

Done Tor Disabled

What Just Happened?

- We can forget our passwords, and have them mailed to us.
 - Admins have passwords too.
 - Admins have code execution rights on pretty much every CMS web interface
 - Not just picking on Drupal here!
 - Working closely with them on building a test module in
 - this isn't a bug in their code, any more than a vulnerable TCP stack might be
 - You think this wouldn't work on almost every other real world CMS?
- We just received a code-execution equivalent token over email
 - "I fail to understand the seriousness with which this bug is handled though. Anybody who uses the Internet has to assume that his gateway is owned."
- Why did this work?
 - Ah, thus the subject of this talk.

Obviously, this is the fault of passwords!

- Without passwords, there would have been nothing to forget
- With nothing to forget, there would have been no need for a reminder email
- Without email, there would have been no dependency on DNS
- Without DNS, there would have been no exposure to cache poisoning
- So clearly, we need to stop using passwords and only use SSL client certificates!
 - Strong crypto
 - Global PKI
 - \$10 per user
- There are...costs.

Passwords Scale

- They are a fundamentally imperfect technology
- They also scale remarkably well
 - Nothing physically to lose
 - Nothing physically to leave inside a laptop
 - Nothing that will cause you to be locked out of a building because you left it in your laptop
 - String comparison is easy. Validation against a Certification Authority is not.
 - Especially cross-organizationally
 - User experience is easily customizable – no need for browser UI
- Given very strong mandates, extensive funding, and a well understood hierarchical authority, better can be done
 - For everybody else, **passwords scale.**
- **DNS scales too – like nothing else.**

Why DNS Works [0]

- DNS has first mover advantage, being built in 1983
 - Every IT shop has someone whose job it is to update the DNS
 - Why?
- DNS's centralized layer is very robust
 - Root and Com servers are necessarily some of the Internet's most reliable resources
 - They were there ten years ago
 - They will be there ten years from now
 - Lots of other things might change, but the roots will be there
 - *Do not underestimate how rare this is for anything in technology*
- DNS's decentralized layer is very hands off
 - No need to inform central authorities of every change
 - Delegation minimizes how much has to be centrally managed
 - Cross-organizational communication is *expensive*

Federation Is Hard.

- Definition of Federation: *the formation of a political unity, with a central government, by a number of separate states, each of which retains control of its own internal affairs.*
 - Put another way: Microsoft doesn't trust Google. Google doesn't trust Yahoo. Yahoo doesn't trust CNN. All share however a single namespace (the DNS), all control operations within their namespace
 - DNS provides a *canonical, federated, universally supported namespace. There are no others.*
- Federation is a hard problem
 - Requires technology
 - Synchronization of distributed databases is a classically hard problem
 - Requires *more* than just technology
 - Managing who is trusted to update what record there is as much a human problem, as it is a technical problem

Everyone Federates With DNS

- Email
 - To send a mail, check DNS to determine which server to initiate SMTP to
 - There's even a special record type -- MX
- The Web
 - "Same Origin Policy"
 - Arguably the largest advance in security technology in the last ten years
 - To determine whether one entity can access another, compare their DNS names
- SSL/x.509
 - Supposedly the *real* federated network
 - Not very reliably federated: Which root CA's do you or do you not trust?
 - Not very federated: Wildcard certs are difficult to acquire and unreliable, so constant cross-company interaction required
 - Not actually independent of DNS

Everyone federates with DNS

- Password resets use email, so that passwords only go to the user who owns the account
- OpenID uses the web and its Same Origin Policy, so that different sites can use the same authentication server safely
- SSL uses email, so that only the user that controls a domain can acquire a signed certificate for it

But There's A Problem

- DNS tells you how to get there, but it doesn't tell you what to expect when you arrive.
 - It's *the* worldwide, distributed, fully federated database that reasonably secures everything going *into* the database...but can't validate anything coming back out.
 - Public Key Infrastructure...without the keys
- Theory: Because DNS doesn't secure its content, nobody will treat its payloads as security critical
- Reality: It's the only thing that can scalably tell you where to go. People are using it anyway.

...and look:

- DNS tells you where to go, but not who to expect when you arrive.
- Email imports DNS. Email knows where to go, but not who not to deliver mail to.
- The web (HTTP) imports DNS. The web knows where to go, but not if an ISP has changed anything.
- Password resets import email, which imports DNS, know where to go, but not actually who they're being delivered to.
- DNS's inability to authenticate replies surfaces as a failure to authenticate in system after system after system
 - We can deny these systems exist
 - We can insult their authors
 - We can pat ourselves on the back
 - Or we can start dealing with our inability to authenticate.

Put Another Way...

- Stop arguing about whether DNS should be used for security.
- The ship has sailed. It is used for security, because it scales.
- The only thing that doesn't use DNS for security, is security technologies. How well do they scale?
 - Where's my secure email?

Commercial Realities Are A Crutch

- Have we been blaming the business guys for what's ultimately just poor engineering?
- The systems we are trying to build, to make up for the fact that DNS is insecure, are resource intensive and just do not scale
- We've spent the last year finding design bugs that break authentication.
 - Maybe there's something fundamentally missing, that keeps forcing these bugs in
- Perhaps DNS shouldn't be at the heart of authentication. But it is, and it's time we start treating it that way.

So what's it going to take?

- First, put out the immediate fire
 - What we just did
- Next, figure out how to make DNSSEC scale
 - It doesn't yet
- Finally, start migrating new applications to it
 - This adds its own layer of difficulty

A Few Thoughts on DNSSEC

- The present numbers say nothing.
 - DNSSEC, like *all* authoritative-server modifying solutions, needs the root signed for the solution to be meaningful
 - Otherwise, the attacker just attacks the parent
 - XQID thought they got around this. Bug me if you want to see the break in XQID.
 - The root has remained unsigned for far too long. That's apparently going to change.
 - We hope.

Why We Need The Root Signed

- A core element of *why DNS Works* is that *connectivity* can be bootstrapped with IP's that were there 10 years ago, and will be there 10 years from now
- We already have centralization of the bare minimum amount of data to tell us *where to go*
- We just need a little more information, so we can recognize what to expect *when we get there*
- This, of course, is the simple explanation.

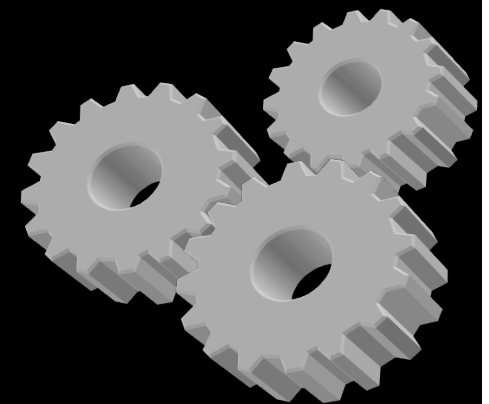
The Fundamental Difficulty Of Signing The Root: **PICK ANY TWO**



Politics



Security



Scalability

Security And Scalability: Sign the root!

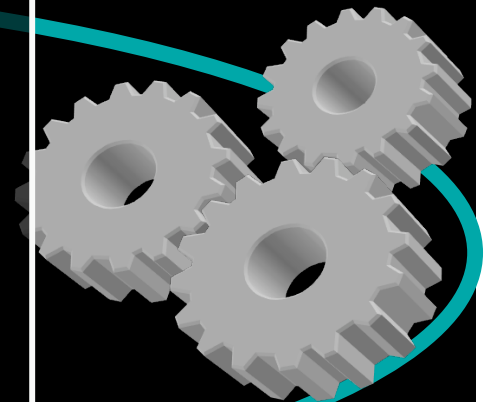


Politics

- Nameservers retrieve all their bootstrapping data from one set of servers
- Nameservers receive keying material at the same time they receive delegation material, making key acquisition as scalable as delegation acquisition
- US Department of Commerce cryptographically asserts the legitimacy of 187 countries...DNS



Security



Scalability

Politics and Scalability: Do nothing!

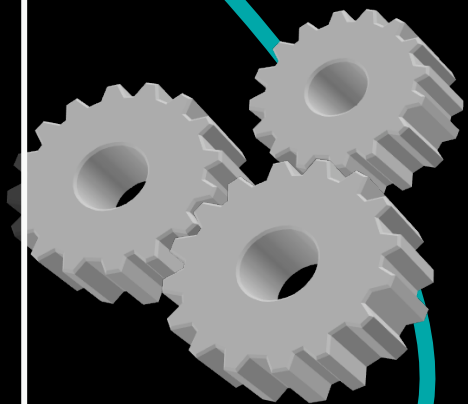


Politics

- Nameservers retrieve all their bootstrapping data from one set of servers
- US Department Of Commerce asserts the legitimacy of 187 countries DNS namespace, but there's already grandfathered détente so its ok
- Internet stays broken



Security

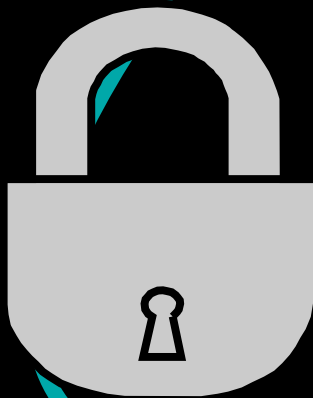


Scalability

Politics and Security: Force DNS Servers To Update Out-Of-Band from “Trust Anchor Repositories”



Politics



Security

- Private companies assert the legitimacy of 187 Countries DNS namespace
- Name servers acquire and maintain keying material for TLDs and other islands of trust for hundreds of different semi-private trust sources through complex, still somewhat undefined methods
- Fails catastrophically, leads to *islands of resolution* alongside *islands of trust*



Scalability

Where Things Are Going

- General IT community: Nowhere, this DNS thing has to work. (Scalability)
- Security: Politics is getting in our way more than Scalability, so...
 - Trust Anchor Repositories are popping up, to hopefully be consumed by implementations
- Yargh. **Let DNS be DNS!**

A Possible Solution?

- Sign the root, and everyone's TLDs
- Implementations allow *administrator opt-in* to local/national Trust Anchor Repositories
 - Russian name server admins can self-manage .ru
 - Finnish name server admins can self-manage .fi
 - American military server admins can self-manage .gov/.mil
- This probably requires little to no code modifications – with no root signed today, this is how trust anchors have to work already

The Other Side Of The Coin

- Signing the root (with potential local trust override) only addresses *how do we get recursive servers to recognize trust?*
- It does *not* solve the problem: How do we make this deployable on the authoritative servers that host the records in the first place?

NO MORE DEPLOYMENT GUIDES

- DNSSEC *must*, in order to scale, be far more automated than it is today
 - No manual key signing
 - No manual key updating
 - No risk that if you go on vacation for five days, DNS will break
 - No blaming the administrator for not knowing the magic invocations
 - We have to make most of DNSSEC automatic.

Automate, Automate, Automate

- Your server should sign records all by itself.
- Signing of records should happen either in the background, or on demand
- Signing as a proxy to a real backend name server should be possible
- **For DNSSEC to scale, it must be as straightforward to install as the Source Port Randomization patch**
 - That's not to say that patch was *easy*
 - Just that it was a one time operation that took care of itself (for the most part) after being deployed

Appliances?

- Appliances are a fantastic thing.
 - Paul Wouters has been pushing DNSSEC for a *long* time and has done some great work
 - Secure64 has apparently done some very good work as well
- For us to achieve a *change in the ecosystem*, the largest player in the ecosystem needs to be upgraded
 - Or else, you can't expect others to be able to validate your records, and you can expect others to have records you can validate!
- **I am trying to figure out how to make this happen for BIND. If you have suggestions, let me know.**

Integration With Registries and Registrars

- DNS is the only successful federated technology.
- DNSSEC solves the problem of getting data back out
 - The registries and registrars are the human/business factors that get data in
 - Easing the business load on them is as important as making DNSSEC manageable for the end administrator
 - We may need to explore alternate ways of populating key material at the registries.

The DDoS Amplification Problem

- We probably need to find a way to stop name servers from being an effective magnifier / obfuscator for DDoS attacks.
 - This is not going away.
 - This is in no way shape or form limited to just DNS – there are other protocols that amplify too
 - Hoping to work on this in 2009 as well
 - This is getting worse.

DNSSCurve?

- Regarding DNSSCurve, I think we have a lot to learn from it
 - DNSSCurve is DJB's concept for how to secure DNS
 - It's based on link-based crypto instead of anything that can be cached

DNSSCurve [1]

- What's Good
 - It posits online key signing
 - DNS material is far too dynamic, and admins are far too harried, for the old model of the offline keystore to make sense
 - Registrars don't have much to do – chaining is handled by the names of name servers

DNSSCurve [2]

- What's not so good
 - There's no code.
 - Um, that matters.
 - It requires new crypto.
 - ECC is standard, but the proposed curve is not.
 - “Optimized for speed” is not actually what you want to hear about a cryptosystem.
 - It's not actually that fast.

DNSSCurve and Performance

- DNSSEC was designed to require no per-query crypto operations on the servers, which may be heavily loaded
 - All operations may be done once, and cached
- DNSSCurve does a crypto operation per query
 - With DJB's sample code, a laptop that can do 15,000 DNS queries a second can do maybe 10,000 ECC operations per second. With 1 operation inbound and 1 operation outbound, that's 100% CPU on 1/3rd the traffic *before you've parsed a single DNS packet*
 - Could possible be optimized, but why?

The Big Problem

- There's no way to achieve end-to-end trust with DNSCurve.
 - With DNSSEC, eventually we can envision clients that do their own validation, using the name server infrastructure just to cache
 - DNSCurve offers a choice: Either abandon end-to-end trust (stub resolver doesn't talk to the real hierarchy), or abandon caching (stub resolver does talk to the hierarchy).
 - The DNS cannot absorb a 100x increase in load, even without added CPU hit from the crypto.
- **We cannot fix the applications of the future without end to end trust being a first class citizen in DNS security. Link based crypto cannot scalably achieve this.**

Nonetheless

- Again, DNSCurve has some really cool ideas for how to make DNS more secure.
- We have more to learn from DJB!

Conclusions

- 1) Fixing the DNS with SPR was necessary, due to the extensive set of attacks against it that were all mitigated severely with this one approach and the scale of systems that were threatened if DNS wasn't fixed.
- 2) People are using DNS because it solves their critical need for federation. In the choice between "doesn't work" and "doesn't work securely", more systems than we'd like to admit choose the latter.
- 3) Any fix to DNS, to make it secure, *needs to still work*.
- 4) Substantial work needs to be done with DNSSEC implementations to make them scale in the real world, even independent of the politics
- 5) Once we make DNS secure, an entire class of security problems may become possible to efficiently solve.

One More Thing...

- Remember when I polluted doxpara.com, so that I could collect the password from mail.doxpara.com?

I also polluted backend.doxpara.com. We REALLY need to fix DNS.

