IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

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
- This talk is *not* going to be a repeat of the DNS hacking talk
  - If you want that, the video and slides are online
  - Or you're welcome to ask me
- This talk has a somewhat different focus

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.

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 ☺

The Rub

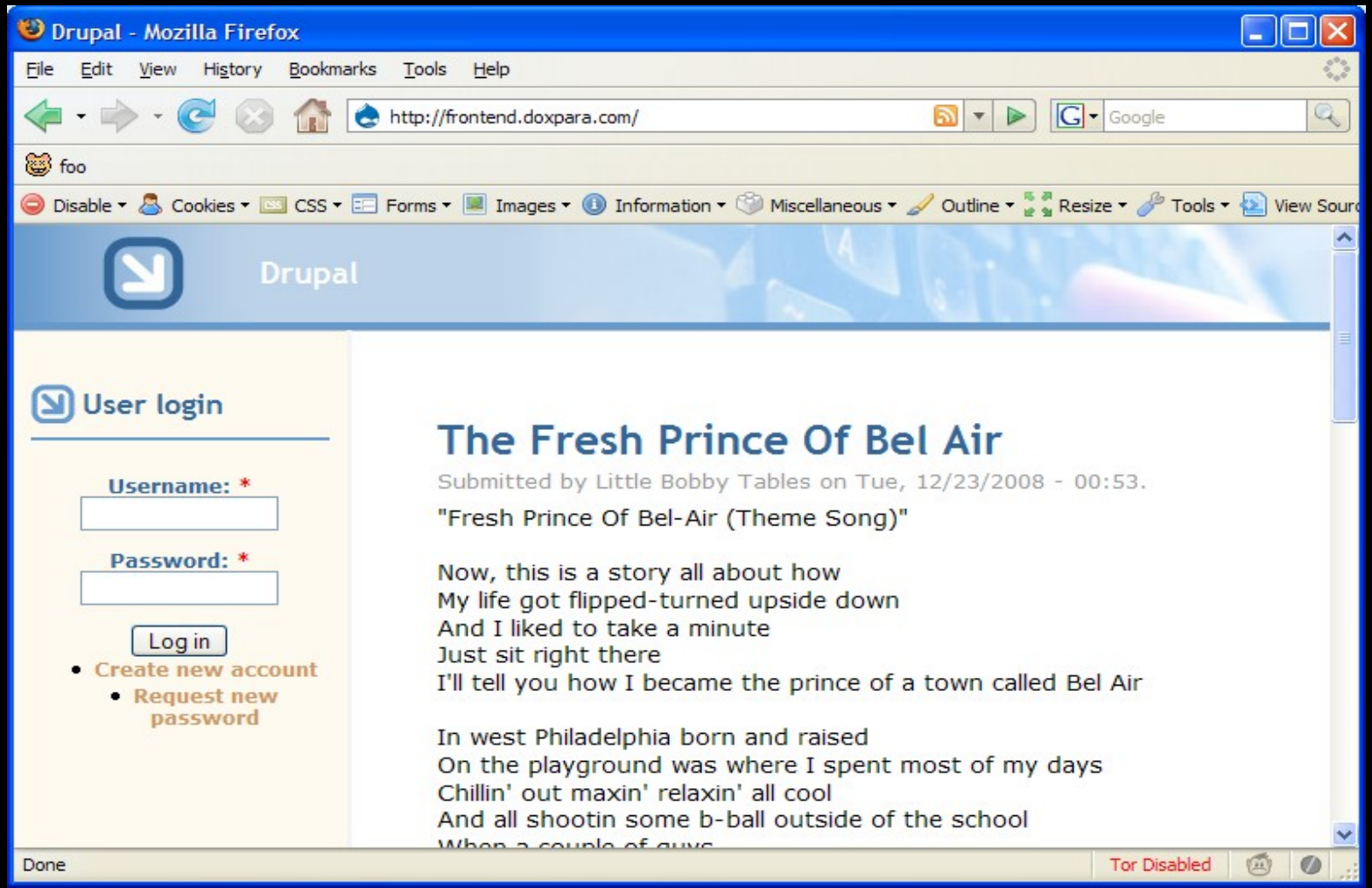
- What should have happened
  - No important systems should have been vulnerable
    - “I fail to understand the seriousness with which this bug is handled though. Anybody who uses the Internet has to assume that his gateway is owned.”
- What actually happened
  - Anybody != Halvar Flake

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

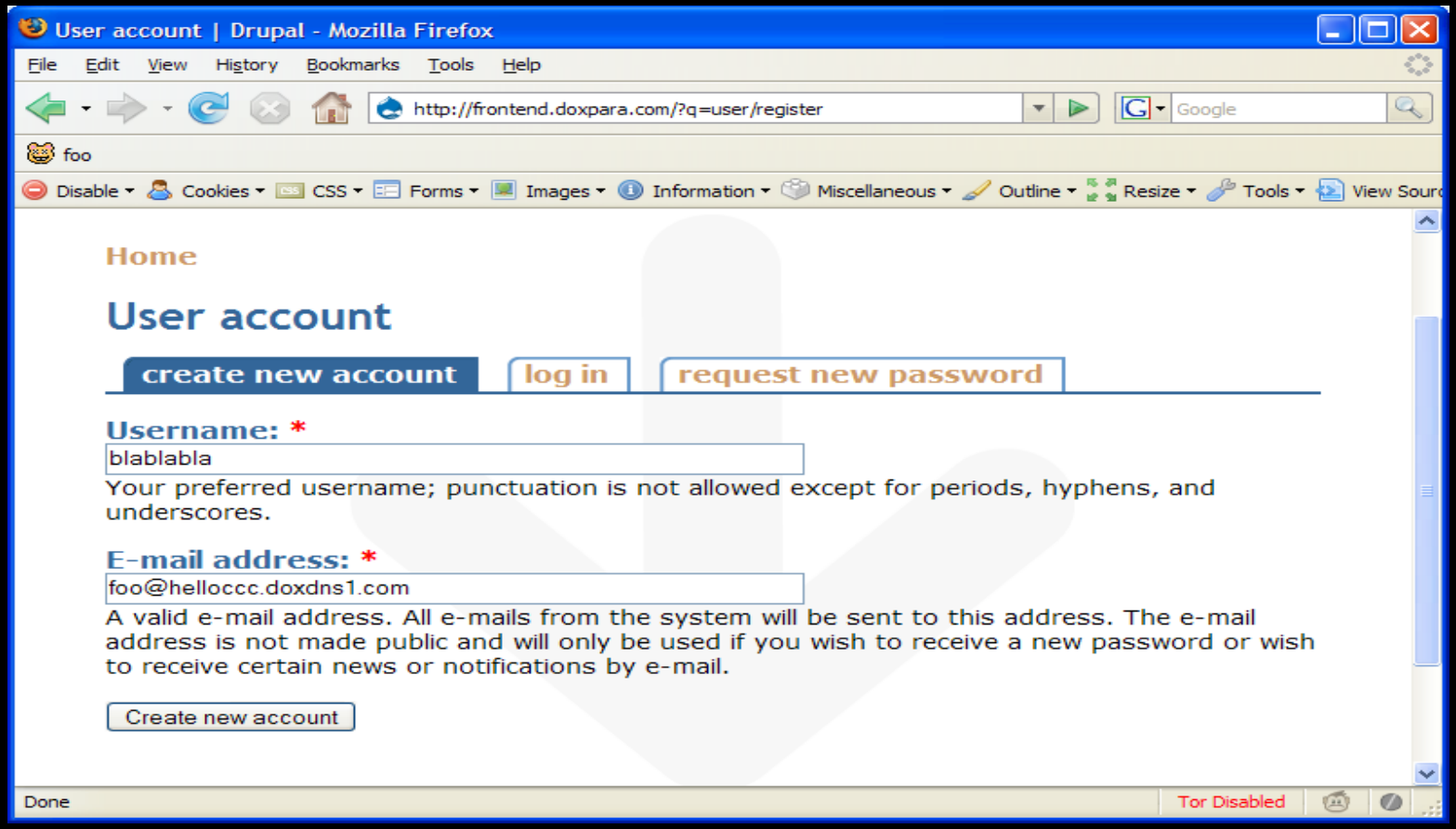
IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

1) Find victim site



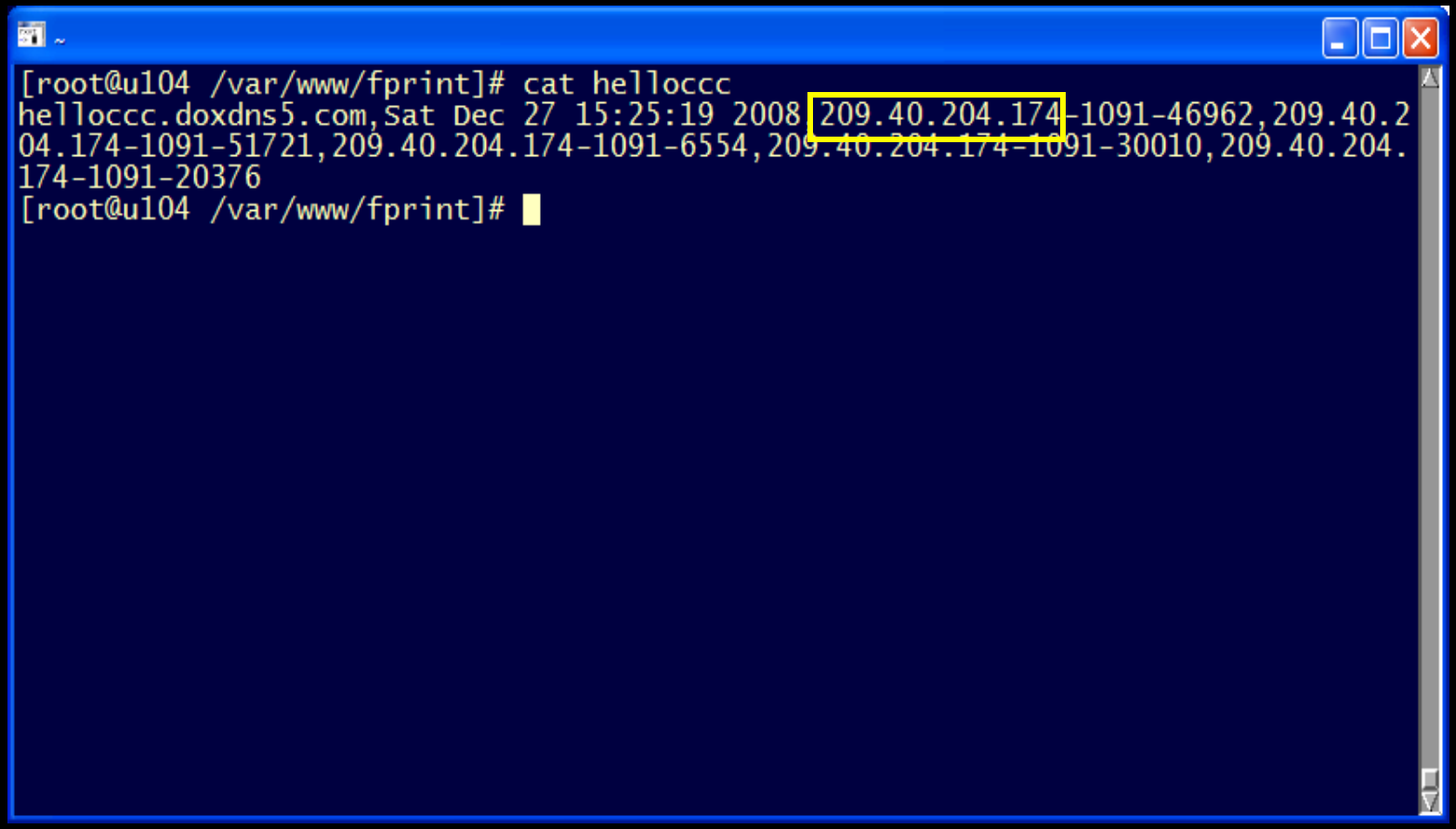
IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

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

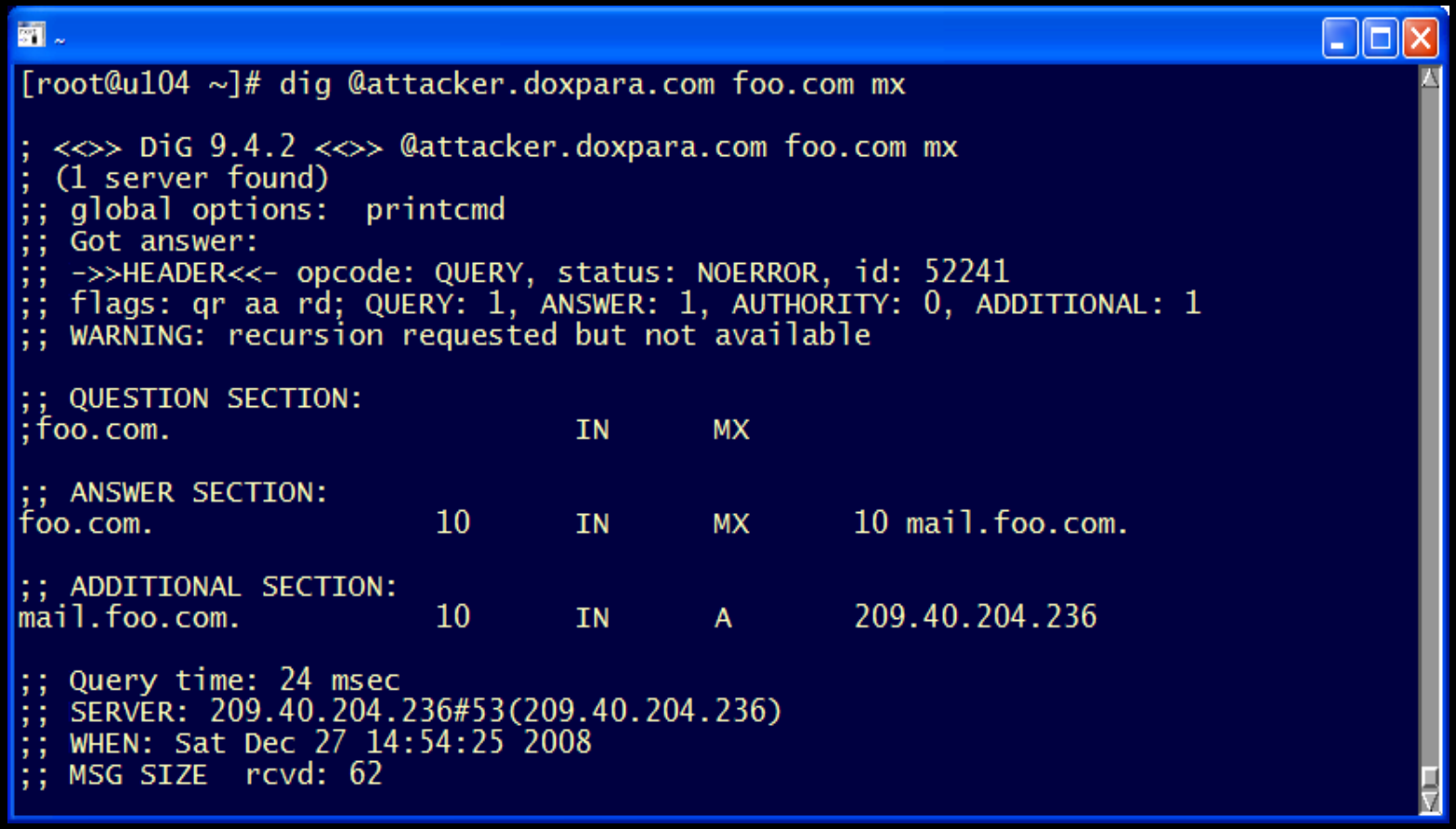


IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

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

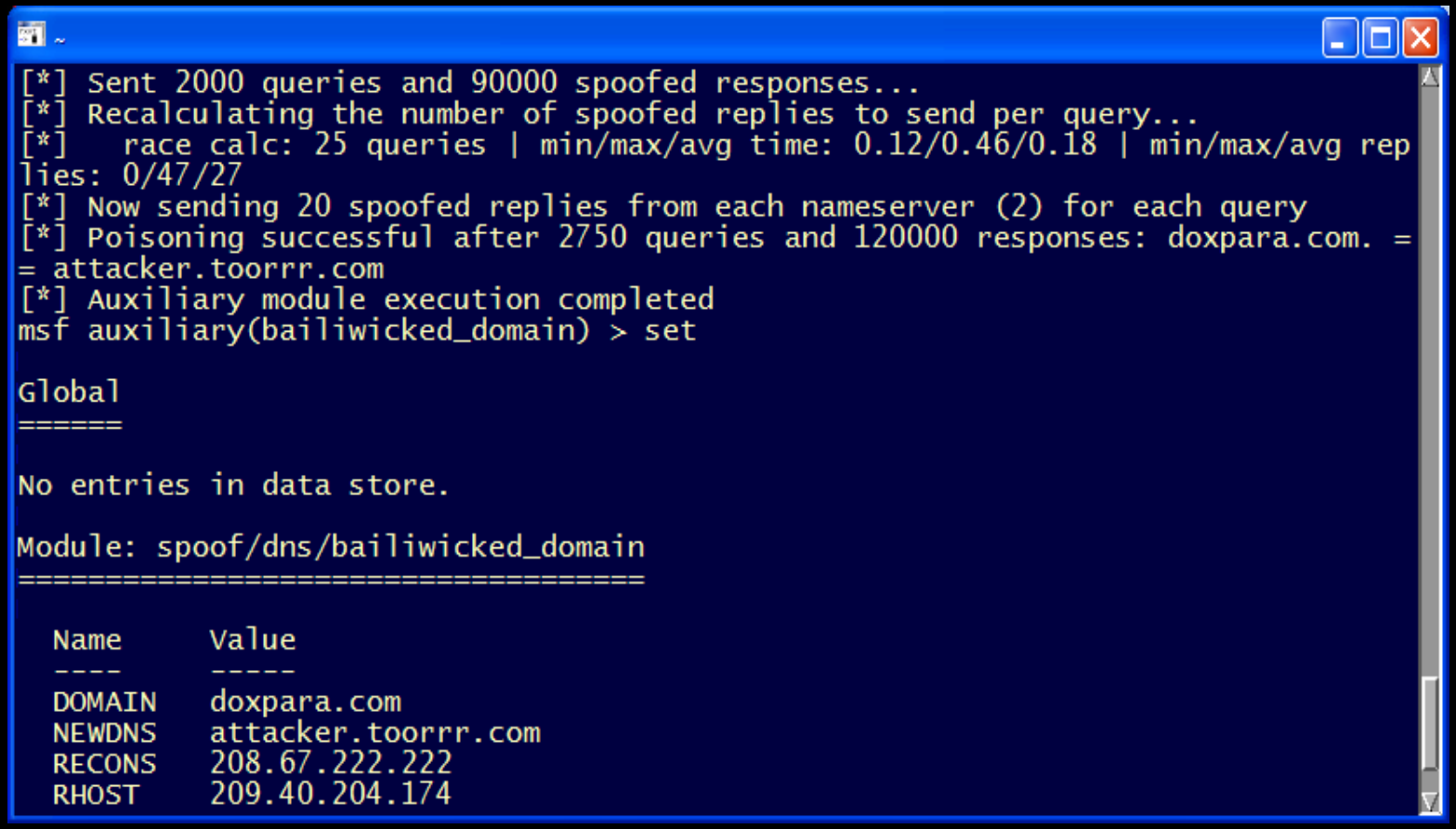


4) Build name server that claims all addresses



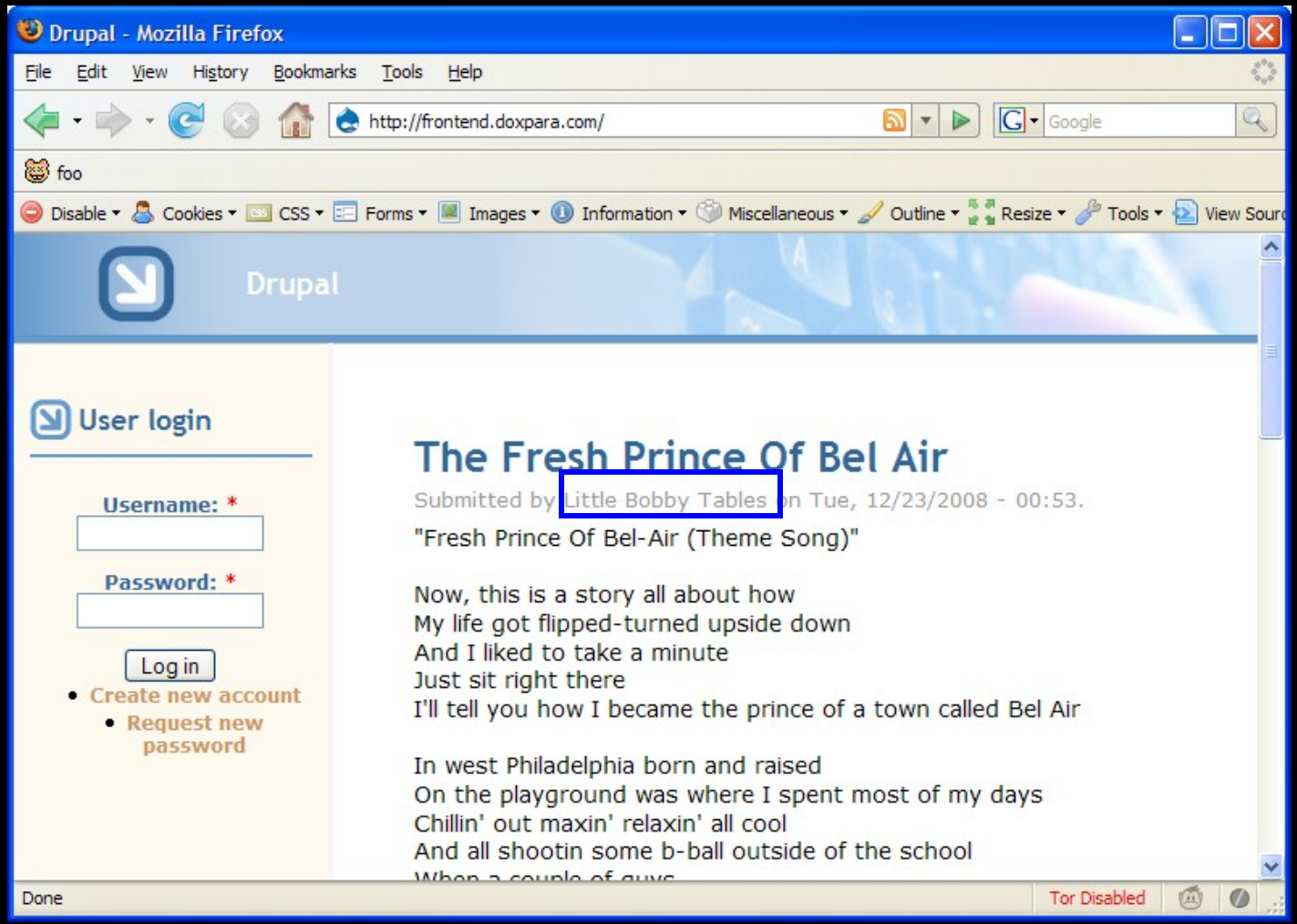
IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

5) Hijack to admin



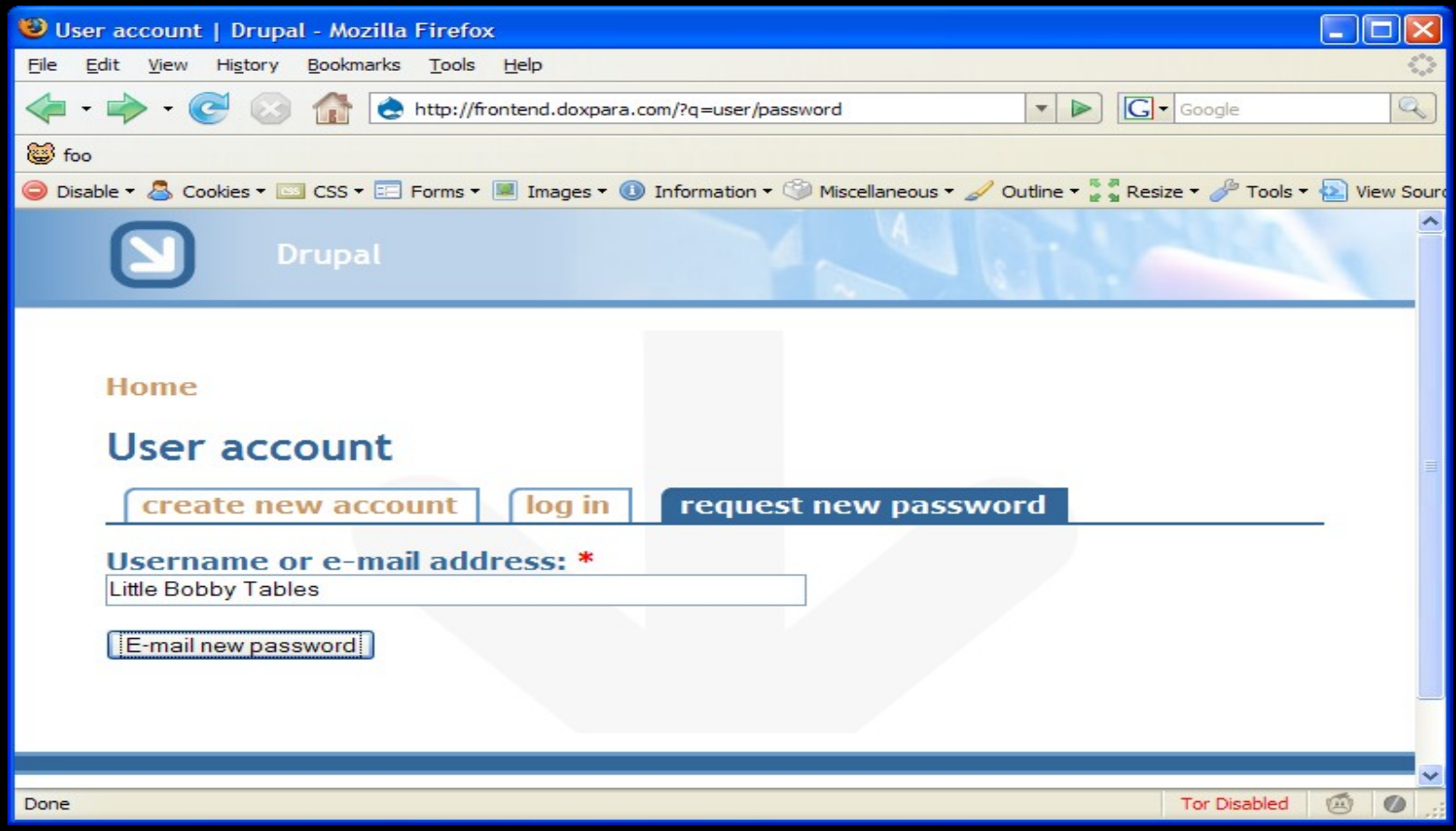
IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

6) Find Admin's Name

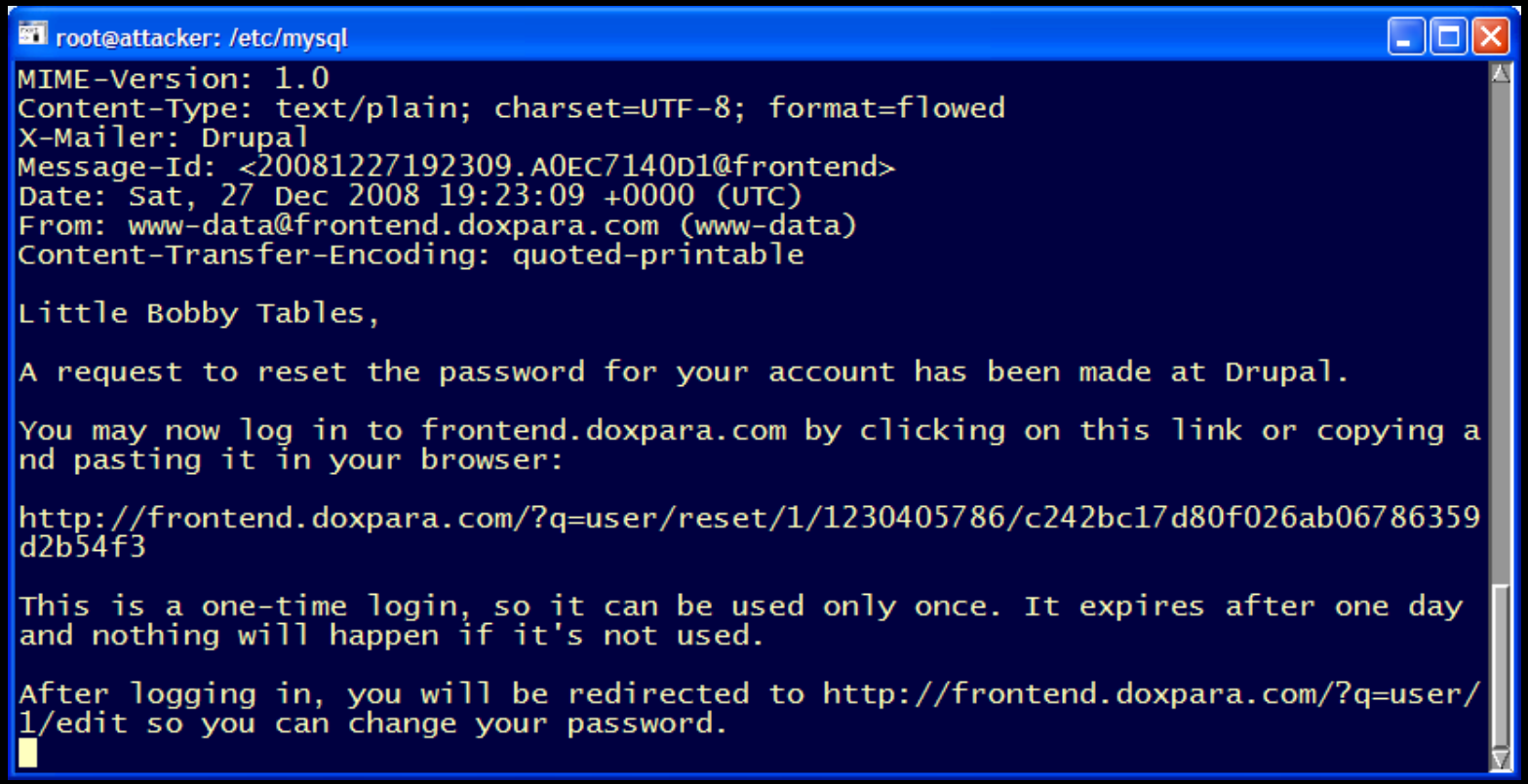


IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

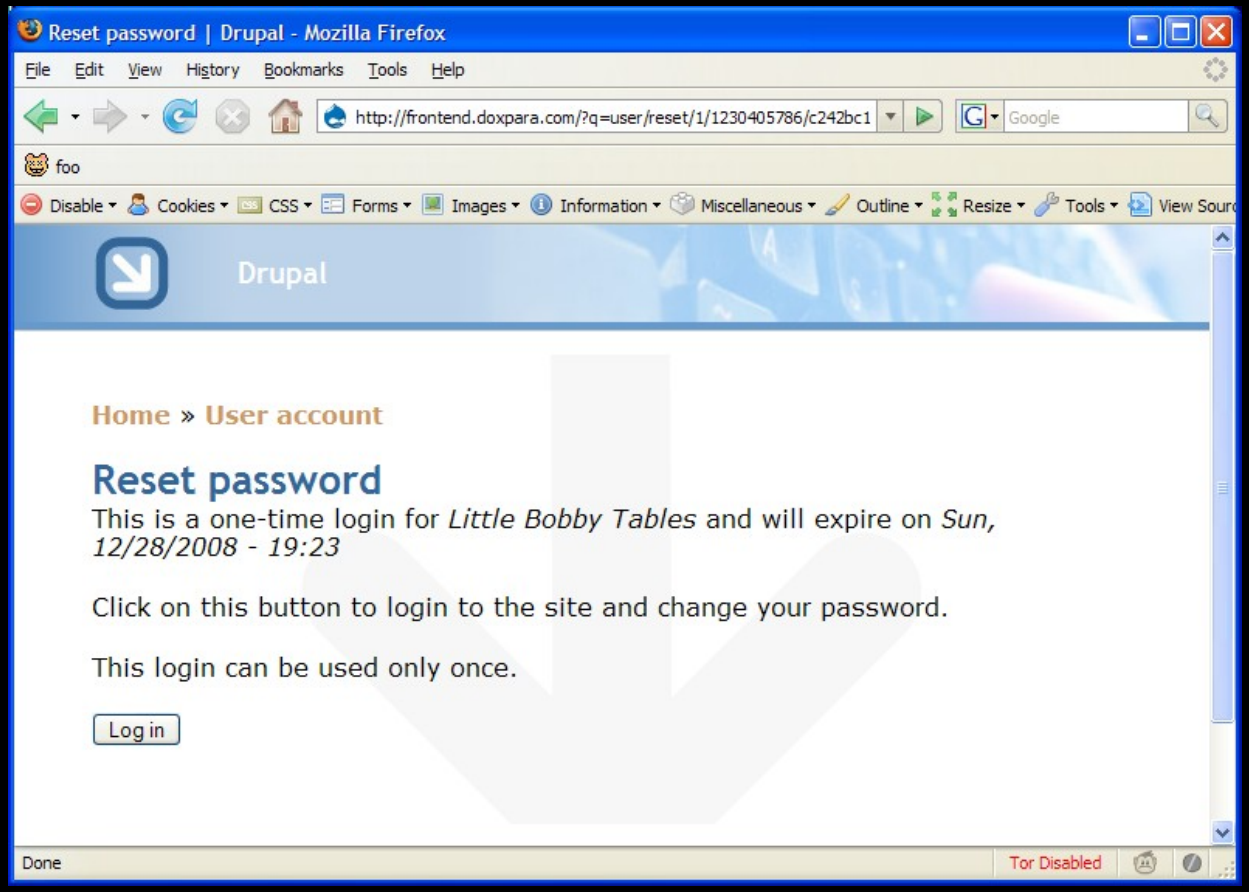
7) Forget Admin's Password



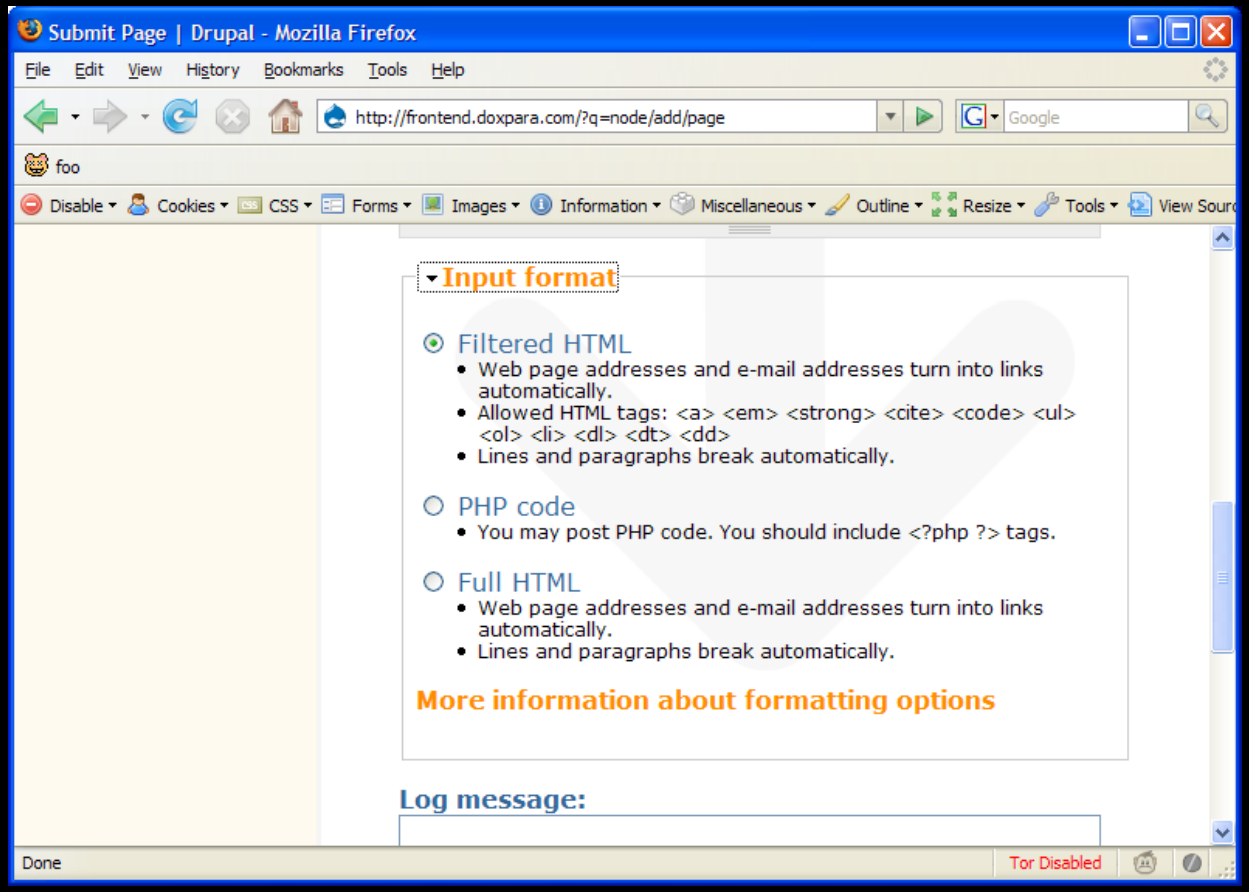
8) Click recovery link (wrote a small mail server)



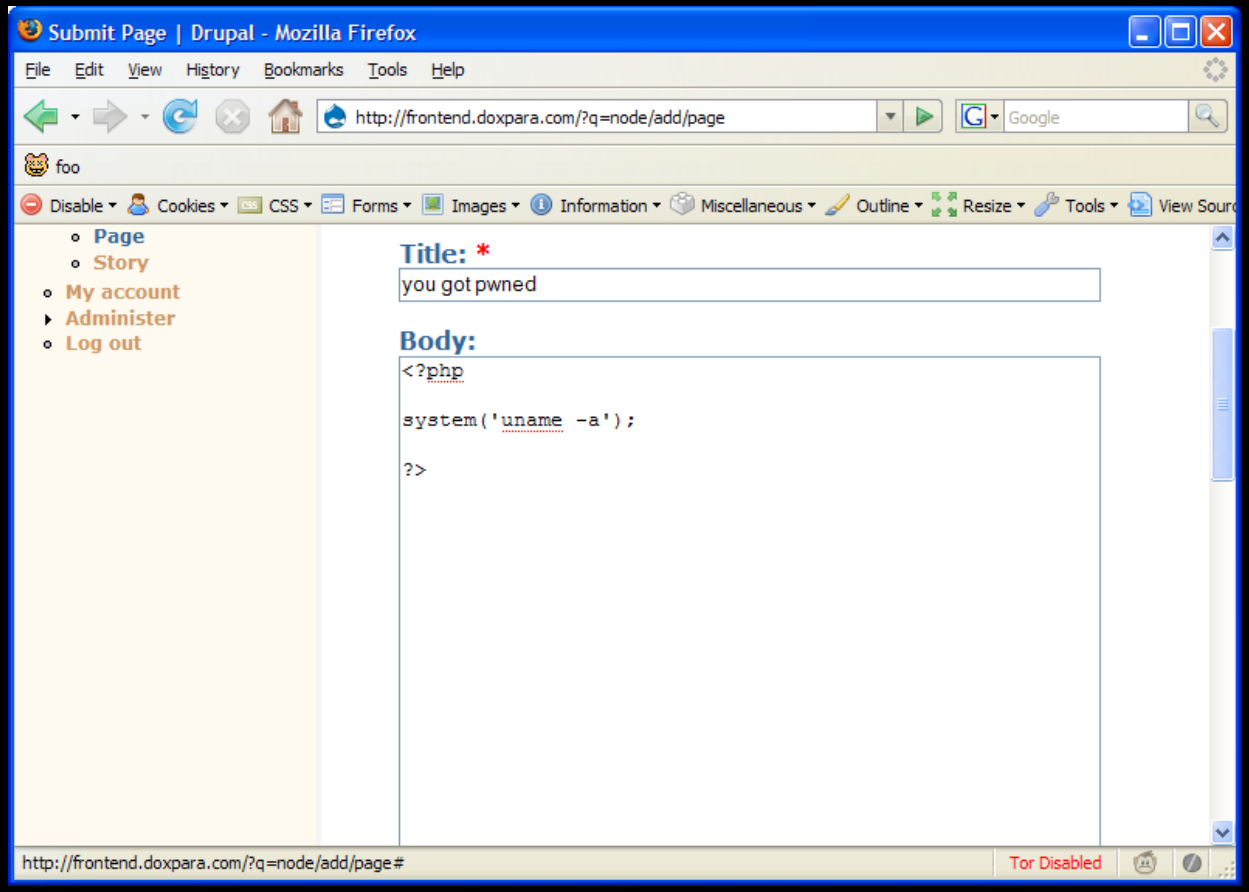
9) Enter Administrative Interface



10) Post content. Be sure to select "PHP Code"

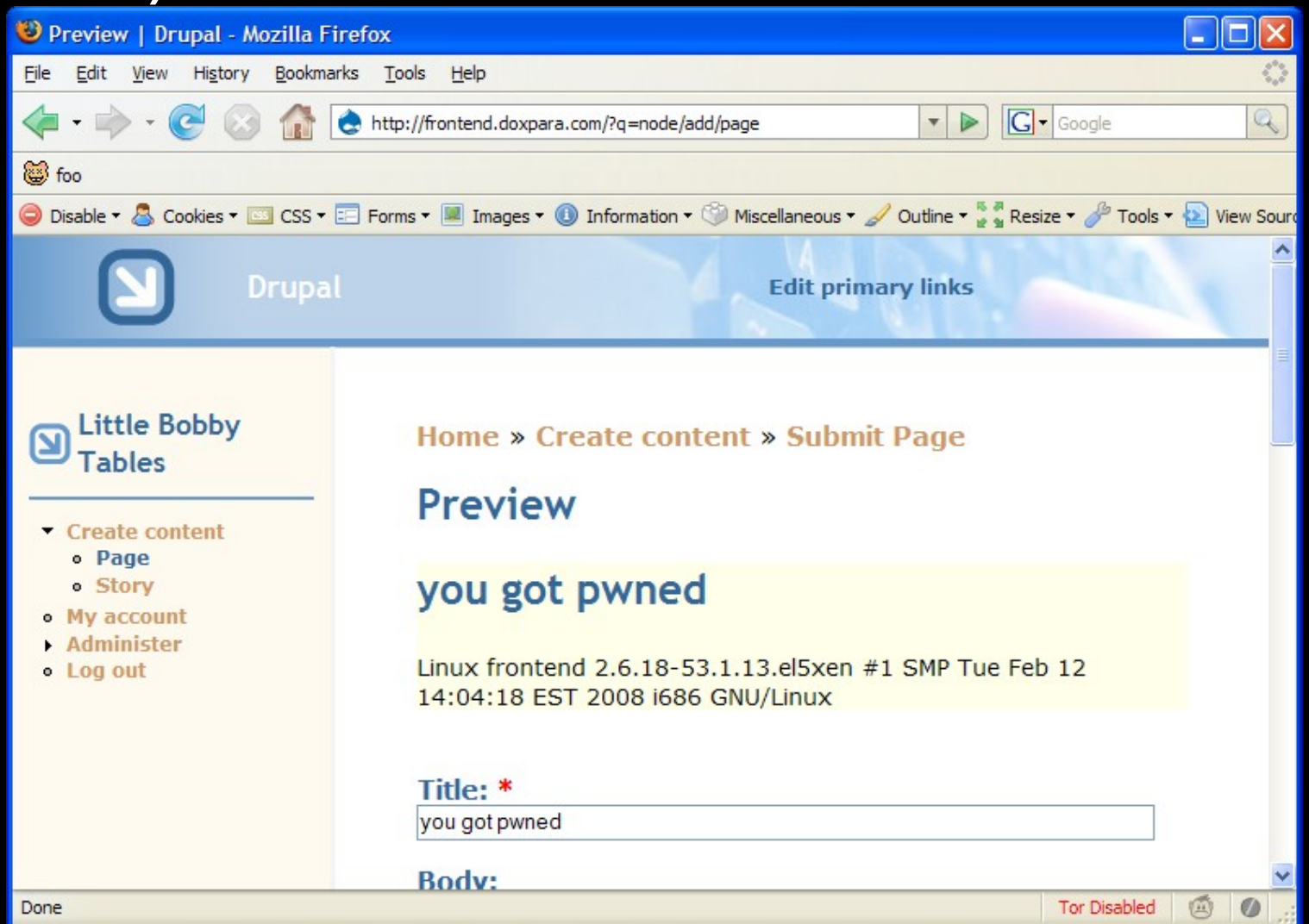


11) Post PHP


Submit Page | Drupal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://frontend.doxpara.com/?q=node/add/page
Google
foo
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Sourc
Page
Story
My account
Administer
Log out
Title: *
you got pwned
Body:
<?php
system('uname -a');
?>
http://frontend.doxpara.com/?q=node/add/page#
Tor Disabled

IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

12) Uh oh



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
- What, no rush to sign up? ☺

If You Don't Understand The Problem, You Won't Find The Solution

- We have some bad habits in the security industry:
  - Appeal to Morality: "If you were a *good* programmer, you'd use SSL Client Certificates instead of passwords."
  - Appeal to Intelligence: "If you were a *smart* programmer, you'd use SSL Client Certificates instead of passwords."
  - This is nice because it makes us feel good and smart
    - Reality: What can take a team of programmers two years to build, can take us two weeks to break.
      - We have the easy job.
  - Neither argument is very compelling.
- SSL client certificates are awkward and expensive to manage, and fail many critical use case scenarios
  - Put another way: **PASSWORDS SCALE**
- So does DNS – like nothing else does.

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
- 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
- DNS had first mover advantage, being built in 1983
  - Every IT shop has someone whose job it is to update the DNS
  - Interactions with the global DNS are limited after initial registration

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
      - CN=DNSName.com

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. Heh, lets see how they’re doing…

2008 Has Been A Rough Year For Authentication

- Mike Zusman: SSL VPN's, which have no other purpose but to prevent bad guys from intercepting traffic, don't make sure a bad guy isn't intercepting traffic
  - They aren't validating SSL certs
- Mike Perry: HTTP Cookies received over SSL, which are used to authenticate access to https://www.bank.com, are by default also sent to http://www.bank.com. Most sites do not change the default.
- Francisco Amato's Evilgrade: You'd think anyone who went out onto the Internet and said, "Dear Internet, please give me arbitrary code to execute", would actually authenticate the code retrieved. But in general, they don't.
  - Java plugin - Winzip - Winamp - MacOS - OpenOffices - iTunes - Linkedin Toolbar - DAP [Download Accelerator] - notepad++ - speedbit

An Interesting Theory

- Historical games: “What if?”
  - What if DNS had been secure from the start?
  - Would all of these design bugs have actually happened?
    - SNMPv3 and NRNG definitely would have
    - But what of SSL-VPNs, Cookie Monster, and EvilGrade?

Why are SSL-VPN's insecure?

- *My* original reaction to hearing that SSL-VPN's didn't authenticate who they were encrypting to was shock and indignation
  - Appeal to Morality and Intelligence
  - Blamed "commercial realities"
- But then I started listening
  - "We have to ship test gear to customers. It has to still work after they buy it. How are customers supposed to get their certificates?"
- The present system for securing SSL, for devices, requires interactions between *three* companies – a device manufacturer, a customer, and a third party root CA
  - Inter-company interactions are *expensive*
  - This is completely inelegant
  - **And yet, every one of those devices is showing up in DNS.**
    - DNS scales. The SSL infrastructure, not so much here.
- A question: If DNS had been a secure place to put certificate hashes, might SSL VPN's have just told people "heh, throw this in your DNS w/ the IP"?

Why are HTTP Cookies via SSL insecure?

- There is no good way to know that a site is HTTPS-only.
  - So it's common that we *try* to access things insecurely, and then upgrade
    - 302 redirect to HTTP
      - Which can of course be hijacked
  - Also common that we need to be logged into both the secure and insecure side of a site
- What if we had a secure, scalable way to declare that a site was HTTPS only?

Why are automatic upgrades insecure?

- An automatic upgrade over SSL would have almost every security property of Windows Update
  - Missing property: Signing box offline in a vault
  - It would also be much slower, since SSL session negotiation is much slower than TCP session negotiation
    - It Has To Work
- People distribute updates over HTTP because it scales
  - It’s very difficult to correctly generate signatures across packages and validate them
    - Wrong version
    - Wrong package
    - Wrong signing key
    - Etc
- What if there was a secure, scalable way to distribute hashes for new versions of packages?

Just Sayin...

- What if we had a secure, scalable place to put PGP keys?
  - Maybe we'd finally have 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.

The "Shouldn't Be In DNS" Problem

- There are some who think that people "shouldn't put strange things in DNS"
  - "It wasn't designed for that"
- Well, the jig is up: A whole mess of things have been seen in DNS
  - SSH Fingerprints (SSHFP – how is that secure?)
  - Anti-SPAM TXT records
  - VoIP directories
  - IPSec keys (again, how is that secure?)
- The reality of federation: Nobody gets to tell you what you host in your own DNS
  - Whether you cache or not is your choice, but that's it
- We're using it for auth. We might as well do it securely.

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.

Automation

- Apparently, the big thing in the DNS world right now is “automation” – how do we make key signing just happen?
  - DNS scales. DNSSEC in its “normal operating modes” requires a fairly extensive amount of manual manipulation.
  - Yeah, no. **DNSSEC needs to be as close to a deployment story as the SPR patch as technically feasible**.

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 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 getting worse.

DNSCurve?

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

DNSCurve [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

DNSCurve [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.

DNSCurve 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
- DNSCurve 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 heirarchy), or abandon caching (stub resolver does talk to the heirarchy).
    - The DNS cannot absorb a 100x increase in load, even without added CPU hit from the crypto.

Nonetheless

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

Conclusions

- 1) Federation is hard, and everyone has been using DNS to do it. That DNS isn't secure, and that people needed DNS to be secure, wasn't actually enough to stop people from using the only workable federation scheme out there.
- 2) Authentication in general is a mess.
- 3) If DNS is to be at the heart of authentication, then perhaps it should be more secure.
- 4) DNSCurve offers some cool possibilities for making DNSSEC better.

One More Thing...

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

IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

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