PKI:
Lemon Markets and Lemonade

Peter Gutmann
University of Auckland

Session ID: STAR-304
Session Classification: Intermediate

RSACONFERENCE2011

Agenda

- **Lemon Markets/PKI Markets**
- **What's the Problem?**
- **Consequences**
- **Solutions**

Lemon Markets / PKI Markets

RSACONFERENCE2011

Lemon Markets

- Lemon Market: One in which buyers can't distinguish between good-quality and poor-quality goods
  - Won its author George Akerlof the joint Nobel Prize in Economics
- An analogy used to analyze the problem of information asymmetry
  - One side knows more about the product than the other
- Tend to collapse unless a correcting force is applied

RSA 2011 CONFERENCE

Lemon Markets

- Buyers can't distinguish between good-quality and poor-quality used cars ("lemons"), but sellers can
- Sellers of good-quality used cars can't get a fair price for them
  - Better-quality used cars are withdrawn from the market
  - Buyers revise their expectations downwards
- Sellers of medium-quality used cars can't get a fair price for them
  - Medium-quality used cars are withdrawn …
- Eventually only lemons are left
  - Correcting force: third-party vehicle checks, after-sales warranties, …

Lemon Markets / PKI Markets

- What happens when *neither* side has accurate information about the quality of the product?
  - This leads to a market for silver bullets
  - Insert joke about "a used car salesman knows when he's lying"
- In a lemon market, a failure is obvious
  - If the car you bought breaks down, it's a lemon
- In a silver-bullet market, failures are silent
  - The security is ineffective, but no-one ever notices
    - Any security technology whose effectiveness can't be empirically determined is indistinguishable from blind luck — Geer's Law
  - The security is silently bypassed by attackers, and again no-one notices

What's the Problem?

RSA CONFERENCE 2011

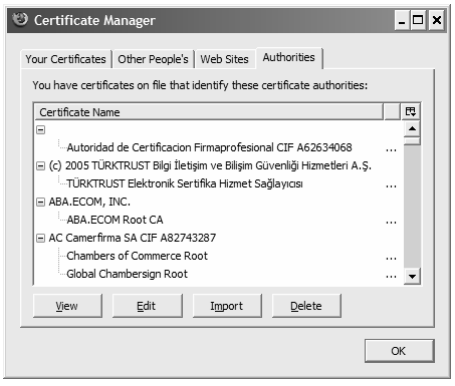
What's the Problem?

- With PKI software, users do have a means of evaluating the product
  - The more capable the software is of accepting any certificate, the "better" it appears to be
- Software that correctly rejects invalid and broken certificates gets dropped in favour of software that blindly accepts anything thrown at it
  - With the amazing invalid certificate, the complaint was that an application was actually rejecting it!
- Acceptance of invalid certificates is a silent failure
  - Rejection of invalid certificates is a very obvious failure of functionality

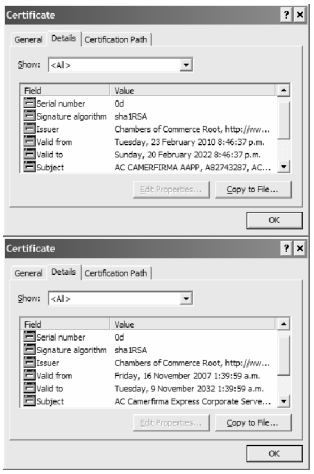
What's the Problem? (ctd)

- In economic terms users are relying not on metrics but on signals
  - A signal is a proxy for information in the absence of a metric that encompasses actually useful information
  - Branding of badge-engineered products is an example of a signaling market
- For PKI software, the deciding metric should be the quality of the implementation, the accuracy with which it rejects invalid certificates
  - (On a more abstract level it's the effectiveness with which it secures transactions/messages, but this is hard to quantify)
- In the absence of this information, users rely on signaling, the ability to accept and process the widest possible range of certificates

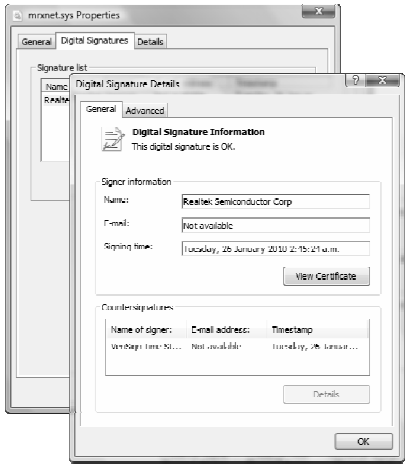
What's the Problem? (ctd)



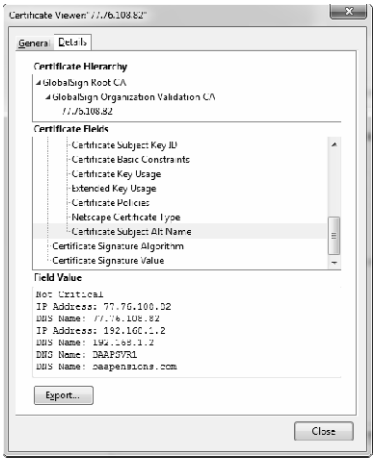
What's the Problem? (ctd)



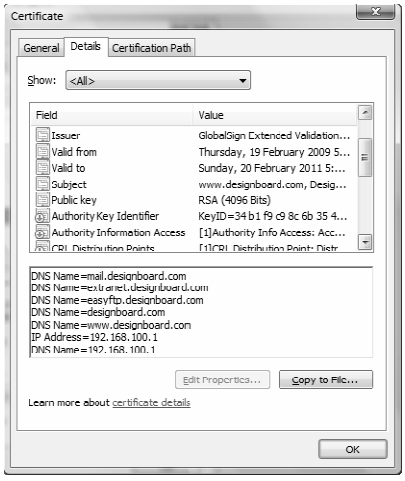
What's the Problem? (ctd)



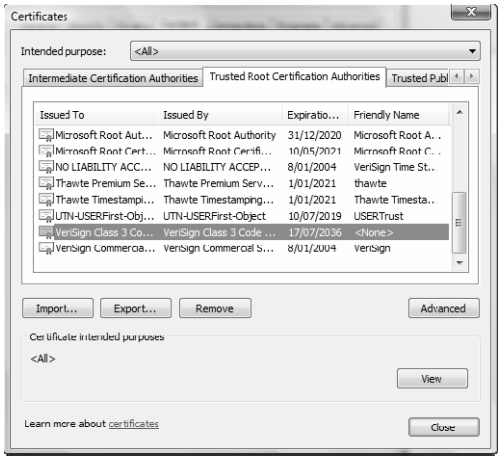
What's the Problem? (ctd)



What's the Problem? (ctd)



What's the Problem? (ctd)

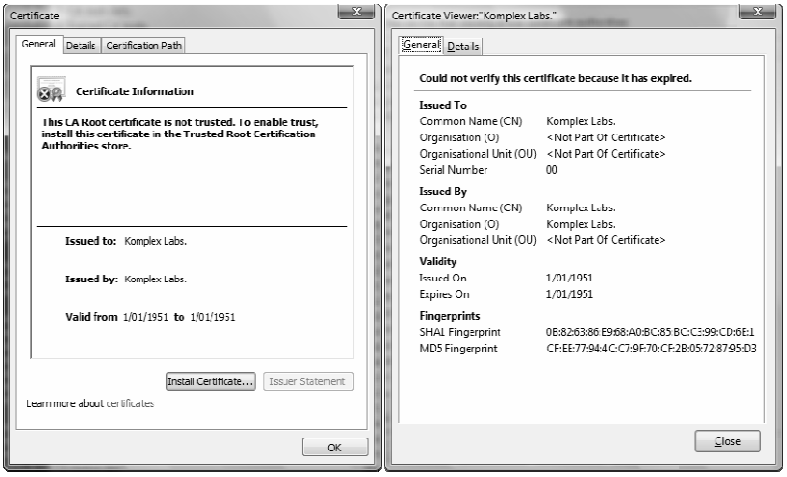


What's the Problem? (ctd)

What's the Problem? (ctd)

```
-----BEGIN CERTIFICATE-----
MIIQojCCCIoCAQAwDQYJKoZIhvcNAQEEBQAwGDEWMBQGA1UEAxMNS29tcGxleCBM
YWJzLjAeFw01MTAxMDEwMDAwMDBaFw01MDEyMzEyMzU5NTlaMBgxFjAUBgNVBAMT
DUtvbXBsZXggTGFicy4wggggMA0GCSqGSIb3DQEBAQUAA4IIDQAwgggIAoIIAQCA
A++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+////////////////////////////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///++++HELLO+THERE++++////////////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///And/welcome/to/the/base64/coded/x509/pem/certificate/of///+
+////////////////////////////////////////////////////////////////+
+///KOMPLEX/MEDIA/LABS//////////////////////////////////////////+
+///www/dot/komplex/dot/org/////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///created/by/Markku+Juhani/Saarinen//////////////////////////+
+///22/June/2000///dw3z/at/komplex/dot/org/////////////////////+
+////////////////////////////////////////////////////////////////+
+///You/are/currently/reading/the/public/RSA/modulus/////////////+
+///of/our/root/certification/authority/certificate/////////////+
+////////////////////////////////////////////////////////////////+
+///Which/happens/to/be/16386/bits/long/////////////////////////+
+////////////////////////////////////////////////////////////////+
+///And/fully/working/and/shit//////////////////////////////////+
+////////////////////////////////////////////////////////////////+
+///And/totally/insecure///////////////////////////////////////+
+////////////////////////////////////////////////////////////////+
```

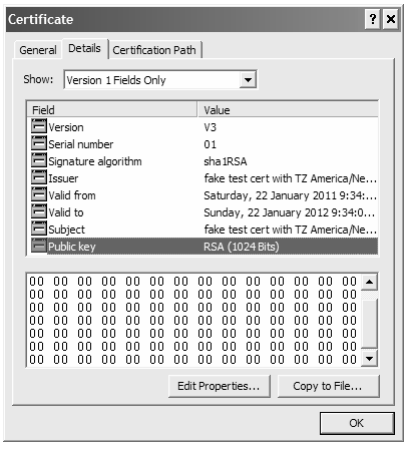

What's the Problem? (ctd)



What's the Problem? (ctd)

- This certificate…
  - Looks a bit suspicious
  - Dates from the 1950s
  - Has a negative validity period
  - Is unsigned (!!)
- Apart from trust-related bookkeeping issues, neither Windows nor Firefox see a problem with this

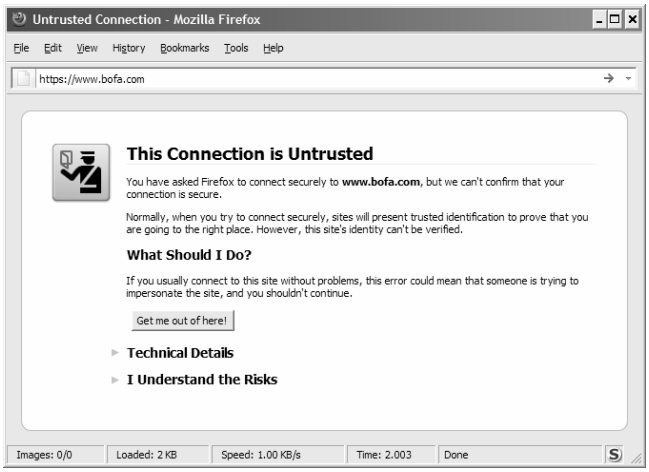
What's the Problem? (ctd)



What's the Problem? (ctd)

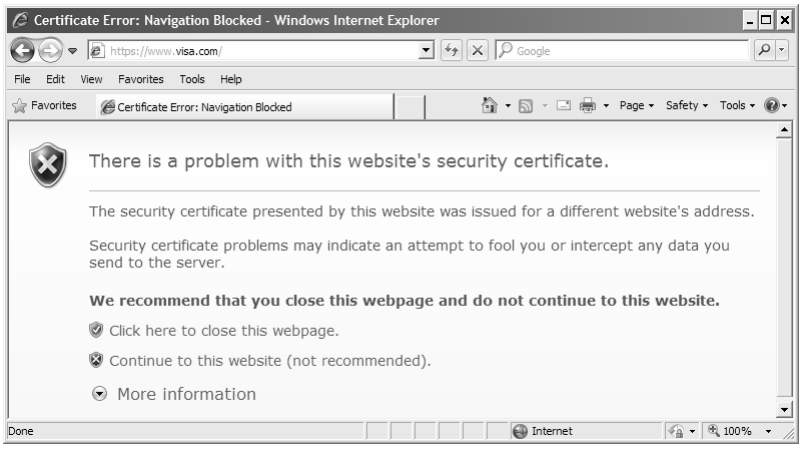
- This certificate has…
  - An invalid issuer name
  - An invalid subject name
  - An invalid start date
  - An invalid end date
  - An invalid public key
  - An invalid signature
- It's actually hard to find anything in this certificate that's valid
  - Well there's the serial number…
- It's OK though, Windows and OpenSSL accept it

What's the Problem? (ctd)

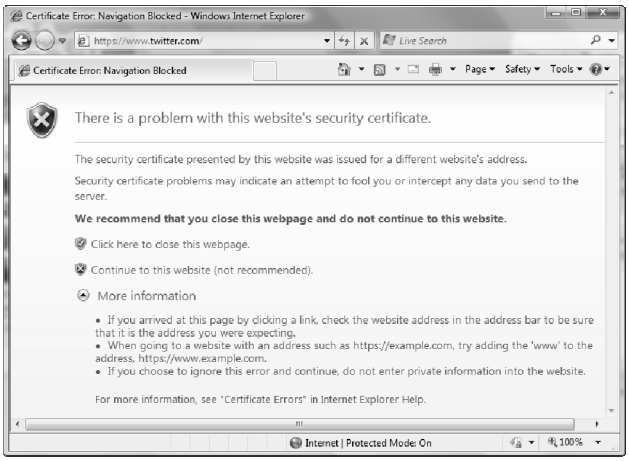


What's the Problem? (ctd)

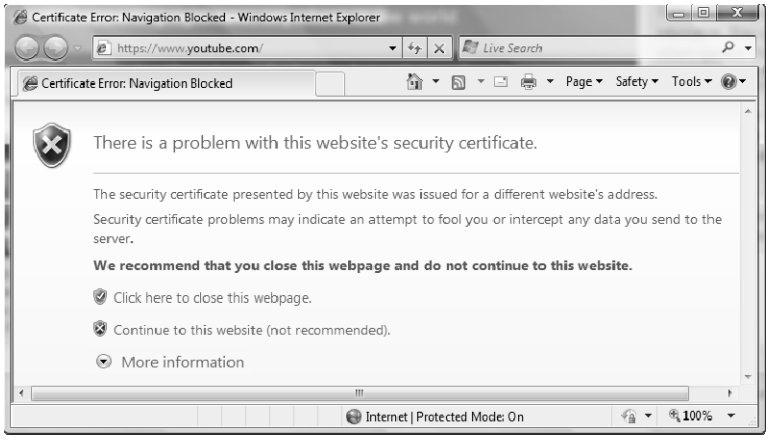
What's the Problem? (ctd)



What's the Problem? (ctd)



What's the Problem? (ctd)



What's the Problem? (ctd)

- Admittedly some of the problems illustrated are more due to-lax-to-nonexistent CA checking
  - Getting screenshots of bugs in software is difficult
- Still, we have a (serious) problem
- There is no economic term for such a situation
  - This is something that can't occur in conventional economics, since it leads to market failure
- Since there isn't a term defined for this, I propose "PKI Market" to match the existing concept of a "Lemon Market"

Consequences

Consequences

- A PKI market produces a toxic codependency of broken certificates and broken code
  - Certificates can be broken because the code doesn't reject them
  - As a result, code can't reject broken certificates because there are too many of them out there, and users would switch to code that doesn't reject them
- Why is this stuff so hard to get right?
  - ACLs/Firewall rules: Allow/disallow based on a pattern-match
  - Certificates: Vast amounts of custom business logic

Consequences (ctd)

- Disabling validity checks in order to make PKI "work" is fairly widespread
  - Two widely-used security toolkits allow user-defined verification callbacks to supplement or replace standard checks
  - Many applications implement this as 'return 1'
  - Practice is institutionalised in manuals and user guides
  - Financial transaction processor "by way of some awful documentation and sample code" tells vendors how to make an SSL connection insecurely
  - stunnel does this by default
  - German national ID card software didn't bother performing any checking, so any certificate was regarded as valid
  - Many, many more examples of PKI apps doing similar things

Discussion Overview

- Problems
  - BasicConstraints/KeyUsage
  - Key Identifiers
  - DNs
  - CRLs
  - PKI Services
- Solutions

Basic Constraints

- basicConstraints.cA flag
  - The single most basic value in a certificate
  - Boolean flag, “is a CA” / “is not a CA”
- Many major platforms simply ignored this until 2002 when bad publicity involving a fake Amazon site “certified” by Verisign forced a fix
  - For the first ~10 years in which some of these technology platforms were deployed, they couldn’t get a basic boolean flag right

Key Usage

- Conformance is more or less arbitrary
  - One often-seen practice is to ignore the flag and use the first certificate you find for any purpose you feel like
- Windows happily uses encryption-only (AT_KEYEXCHANGE) keys for signing…
  - … and signing-only keys for encryption:
  - "the certificates [has the digitalSignature flag set] so the public key can only be used to verify a signature, but in the logon procedure the key is also used to [decrypt]. This is NOT allowed because the [keyEncipherment flag is not set]"
- This was particularly distressing in this case because it voided guarantees provided under European digital signature laws

Key Usage (ctd)

- European PKI vendor ran an interop server for other PKI vendors to test against
  - A who's-who of vendors successfully did
  - After two years someone pointed out that the keyUsage in the server's certificate didn't actually allow this
- Global software vendor ran an interop site for its flagship server product
  - Server authentication key was marked as unusable for server authentication
  - After several years' operation, no-one had noticed

Key Usage (ctd)

- Microsoft NDES SCEP server used to provision Apple iPhones
  - iPhone happily encrypts to a signature-only certificate, ignoring the keyUsage constraint
  - Works OK though because the Microsoft server at the other end ignores it as well
- European CA marked its signature key as not being valid for signatures
  - CA marked a certificate used to encrypt data for a national tax authority as usable only for digital signatures
  - Another CA reversed the order of the flags in keyUsage due to confusion over endianness, effectively setting random flags

Key Usage (ctd)

- keyUsage flags seem to be set arbitrarily by some public CAs
  - Specify keyUsage.keyEncipherment or keyUsage.keyAgreement when the algorithm in the cert isn't capable of doing this
- One CA set DH keyUsage.keyAgreement (for an RSA key)
  - Set keyUsage.encipherOnly
  - Just to be fair, set keyUsage.decipherOnly as well

Key Usage (ctd)

- European PKI project approached this from another angle
  - Marked encryption-only certificates with "ENC" in the DN and signature-only certificates with "SIG"
  - Tested the certificates with PKI software
  - "ENC" certificates worked fine for encryption, "SIG" certificates worked fine for signatures
  - Product was shipped and widely used
  - Quite some time later, a technically-minded user noticed that the software would select and use "ENC" and "SIG" keys more or less at random
    - "ENC" keys had supposedly been kept in escrow
    - Destroyed the validity of the signing process since keys held by a third party had been used for signing

Key Identifiers

- Certificates contain two binary identifier fields, subjectKeyIdentifier (SKID) and authorityKeyIdentifier (AKID)
  - These have very different encodings
- Some CAs memcpy() the SKID to the AKID, creating an invalid encoding
  - When tested against a wide range of PKI software, nothing noticed this
  - Not only were they not paying any attention to the keyIdentifier values, they weren't even trying to decode the extension that held it

Key Identifiers (ctd)

- Variations on this abound…
- European CA encoded the AKID as an empty value
  - Implying the certificate was issued by nobody?
- CAs create circular references
  - AKID points back to itself
  - Presumably an implementation would need to go into an endless loop to process this
- CAs use duplicate SKIDs
  - In one case probably due to it being derived from a time-based value, because batches of certificates issued in close proximity had identical SKIDs
- Adobe's cert handling for signed PDFs does pretty much the exact reverse of what it's supposed to with the KIDs

Key Identifiers (ctd)

- Copy-and-paste PKI
  - Find something that works elsewhere and copy and paste it into your PKI
  - A good idea for regex's, SQL expressions, Perl scripts, …
  - Less good for PKI
- AKIDs point to random unrelated CAs
- SKIDs for all certificates are identical
- authorityInfoAccess points to unrelated CAs
  - Blacklist-based operation in CRLs and OCSP means that such certificates can never be revoked
  - The use of blacklist- rather than whitelist-based checking also means that the failure isn't noticed during normal use

DNs

- If two implementations disagree over what goes where in a DN, they will/won't check different portions of the DN and related fields
- How to abuse DNs/altNames as hiding places
  - Request a certificate with different identifiers placed in locations regarded as being equivalent
    - See earlier screenshots of RFC 1918 certificates
  - CA verifies the identifier in one location
  - PKI software uses a supposedly-equivalent but unverified identifier from another location

DNs

- Examples include Moxie Marlinspike's '\0' DN strings
  - Request a certificate for 'www.microsoft.com\0mydomain.com'
  - CA verifies 'mydomain.com', PKI software uses 'www.microsoft.com\0'
  - At Defcon 2009, a selection of geeks bought certificates for Adobe, Apple, Microsoft, Verisign, Yahoo, and others, until they ran out of money
- As with many other PKI failures, this wasn't fixed until it got media attention due to the creation of a bogus Paypal certificate

DNs (ctd)

- LDAP format represents DNs in reverse order to how they're present a certificates and cert-using protocols
  - Some Java implementations do this too
  - As a result, DNs in certificates can be encoded forwards or backwards
- .NET GetIssuerName and GetSerialNumber return the information in reverse order to the MMC certificate snap-in
  - Different versions of software, e.g. IIS 4 and IIS 5, processed the bytes in opposite order
- This interferes destructively with X.509's blacklist-based checking

DNs (ctd)

- Not only can DNs end up encoded forwards or backwards, they can even be forwards and backwards in the same certificate
  - One European national CA encodes DNs forwards and backwards apparently at random
  - Others are more consistent and get the DN backwards in all certificates
  - Others get the issuer name, via memcpy(), forwards, but the subject name backwards
  - Some certificates contain DN components in more or less arbitrary order
    - This includes duplicate AVA instances in different parts of the DN

Make it a Feature!

- Some European CAs use this to their advantage when the CA is also the PKI vendor
  - CA will only process certificates produced by its own buggy software
  - Software will only accept buggy certificates issued by the CA
- Use of a particular European CA was mandated by government decree
  - Would only issue certificates to users using the CA's broken PKI toolkit
  - The term for this in the country in question is apparently "appointing a goat as gardener"

Make it a Feature! (ctd)

- Another CA quietly dropped requests created by anything other than its own software
  - Use of the CA's services was government-mandated
  - Ensured that only products sold by the CA's consulting arm could be used
- Another CA added incompatible modifications to a standard PKI protocol "for security reasons"
  - The financial security of the CA, that is
  - Had to buy the CA's software to get your request processed

PKI Services

- No better than the basic certificate handling…
- TSA had a soft-failure that caused it to reject any request for a timestamp
  - No-one using the service, which delivers tens of thousands of timestamps a month, noticed that their data wasn't being timestamped any more
- TSA client submitted not a hash of the document to be timestamped but the entire document
  - Server took the first 20 bytes and timestamped that
  - Used with European high-assurance (qualified) signatures

Summary

- After twenty years, we've almost got to the point where we can rely on the most basic extension in a certificate, basicConstraints.cA
  - Even the next most basic one, keyUsage, is handled more or less arbitrarily
  - Beyond that, it's a crapshoot
- "There's not a single X.509v3 extension defined in PKIX a PKI designer can really rely on. For each and every extension somebody planning/deploying a PKI has to check each and every implementation if and how this implementation interprets this extension. This is WEIRD!"
  — PKI developer Michael Ströder

Summary (ctd)

- There is a complete absence of any kind of quality control in PKI software
  - One large PKI vendor for many years had no documentation whatsoever for their code's functionality
  - Developers were handed the code and told that the software's functionality was defined to be whatever you got when you fed it a certificate
  - One new developer's first task was to reverse-engineering what the code did based on observed behaviour with various certificates
- You can't build something so broken that it can't claim to be X.509…
  - … and vendors frequently do

Solutions

Solutions

- Good news: We have near-infinite scope for improvement!
- There are four ways to deal with this problem…
  - 1. The Ostrich algorithm
  - 2. PKI overlay networks
  - 3. Field-qualify your PKI applications
  - 4. Work defensively

1. The Ostrich Algorithm

- Everything's working fine, nothing to see here, move along
- Popular with PKI created for its own sake
  - Target for the consultants was "You asked for PKI, here is PKI, you didn't specify that it had to work"
- This isn't as bad as it sounds: Attackers seem to be using the Ostrich Algorithm as well
  - Baffled by its complexity?
  - Easier targets elsewhere?
  - Not protecting anything worth attacking?
- (Probably the last one on the list)

2. PKI Overlay Networks

- Layer your own custom security controls on top of the general-purpose PKI
  - Requires at least some control over the PKI software
- Leverages existing investment in PKI software while providing add-on functionality that provides the services/functionality that you need
  - A bit like an overlay network built on top of the Internet

2. PKI Overlay Networks (ctd)

- Done by Microsoft for its code-signing certificates
  - Code-signing certificates need a special code-signing extendedKeyUsage
  - Must be present in CA root certificates to prevent a downstream CA from manufacturing their own code-signing certificates
  - Signatures can be verified after the certificate expires via a countersignature mechanism
  - Assorted other special-case handling, e.g. for boot code that can't rely on a CRL being available
  - Verification code is created and controlled by Microsoft to do what it wants
- Disadvantage: Not everyone is Microsoft

3. Field-qualify Your PKI Apps

- Try to field-qualify every version of every application on every platform that you plan to use
  - This is impossible in general
- It may be effectively impossible even for specific cases…
  - One survey of SSL/TLS server certificates found 219 different combinations of keyUsage and basicConstraints.cA flags, including many that were totally illogical

3. Field-qualify Your PKI Apps

- Tests are extremely tricky and tedious
  - Need to verify that things that should happen, do happen
  - Need to verify that things that shouldn't happen, don't happen
  - This comes close to trying to prove a negative
  - Need to re-run the tests every time an application update occurs
- CRLs and OCSP make this especially tricky
  - A successful verification against a blacklist is indistinguishable from a failed check

3a. Field-qualify Your PKI Apps (ctd)

- Variant: Require the use of one specific piece of software everywhere
- Possible in closed environments
  - Inside corporates
  - Closed B2B
- Still need to qualify the PKI software, but now the scope of the operation is limited

3b. Field-qualify Your PKI Apps (ctd)

- Variant: Only use the restricted subset of PKI functionality that you can verify works
  - Don't discard it all, since it least some of it works some of the time
- Possible in controlled environments
  - All parties agree in advance on which subset to use
- As before, still need to qualify the PKI software, but again the scope of the operation is limited

4. Work Defensively

- Assume that nothing will quite work as expected and build your system appropriately
  - When building a system from unreliable components, the less of the unreliable components that you have to depend on, the smaller the chances of an unpleasant surprise later on
- There's nothing to say that you have to use certificates as anything more than a complex bit-bagging mechanism

4. Work Defensively (ctd)

- Example: Use standard presence checks to replace unreliable PKI mechanisms
  - Certificate present in a database $\Rightarrow$ access allowed / certificate is OK
  - To revoke access, remove the certificate from the database
- Avoids the need for CAs, CRLs, OCSP, bridge PKI, certificate path building, chain verification, …
  - All the dysfunctional portions of PKI are eliminated
- Use of Active Directory to manage certificates for account login is an example of this

4. Work Defensively (ctd)

- Variant: Drop PKI entirely
- Why exactly are you using it?
  - "It seems to be the expected thing to do" isn't a reason
- Seriously, why do you actually need a PKI for what you're doing?
  - Name five alternative options that solve the same problem, and provide reasons why PKI is the better choice

4. Work Defensively (ctd)

- Example: Network authentication
  - TLS-SRP and TSL-PSK solves this problem far better than PKI ever can
  - True failsafe mutual authentication
  - PKI can only provide unilateral authentication in both directions …
  - … and then you have to deploy, manage, and run a PKI to get there
- Example: Secure email between corporate offices
  - STARTTLS solves this problem far better than S/MIME can
  - S/MIME: … and then you have to deploy a PKI …
- Do you really want to hold your business hostage to PKI?

Apply Slide

- Assume that a certificate may be little more than a complex bit-bagging scheme
- Treat certificates as a simple signed access token
- No need for external CAs, PKI heirarchies, OCSP servers, or other complex and expensive PKI folderol
- Presence in a database ⇒ certificate is OK
- Access control is handled by removing the certificate from the database, not hoping that a CRL or other check works
- Do you really need a PKI for what you're trying to do?

Further Reading

- Detailed writeup, background material, and references to sources can be found in
- http://www.cs.auckland.ac.nz/~pgut001/pubs/book.pdf
- Chapters "Problems" and "PKI"