Carnegie Mellon
Software Engineering Institute

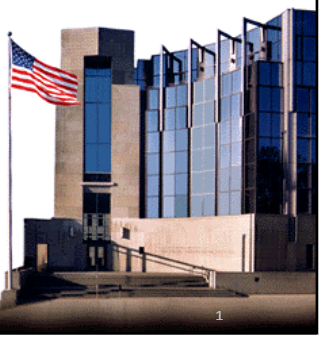
CERT
Coordination
Center

CERT/CC Overview

Presented by Kevin J. Houle
FIRST TC, October 16 Karlshure, Germany

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

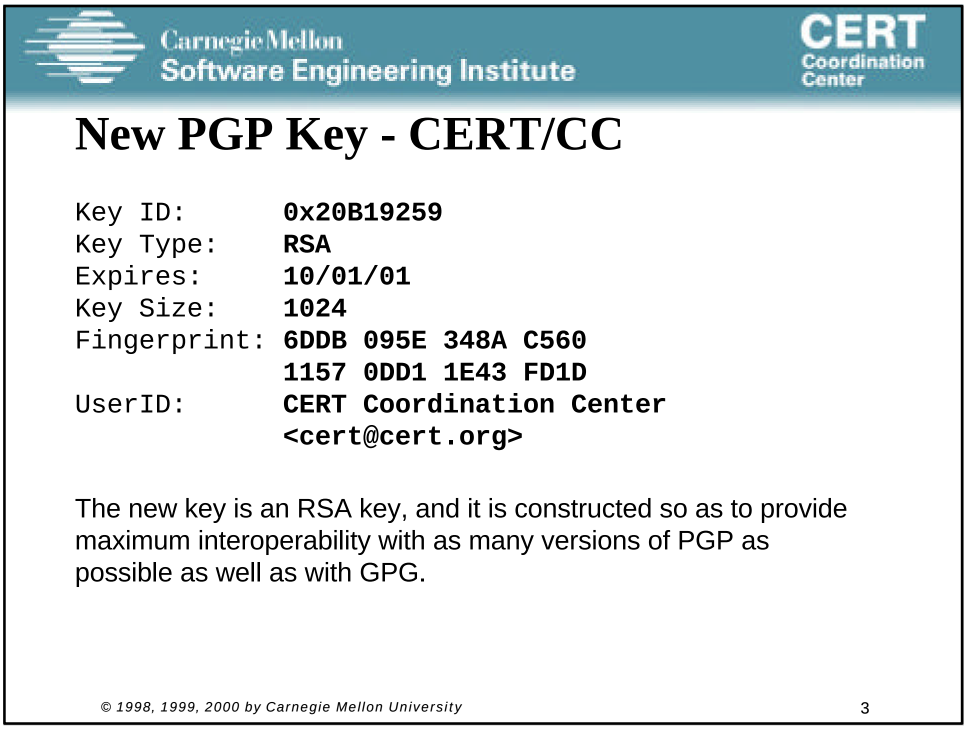
The CERT Coordination Center is part of the Software Engineering Institute. The Software Engineering Institute is sponsored by the U.S. Department of Defense.



Agenda

- New PGP Key
- Incident Statistics and Trends
- Vulnerability Disclosure Policy
- Automating Site Notification
- AirCERT

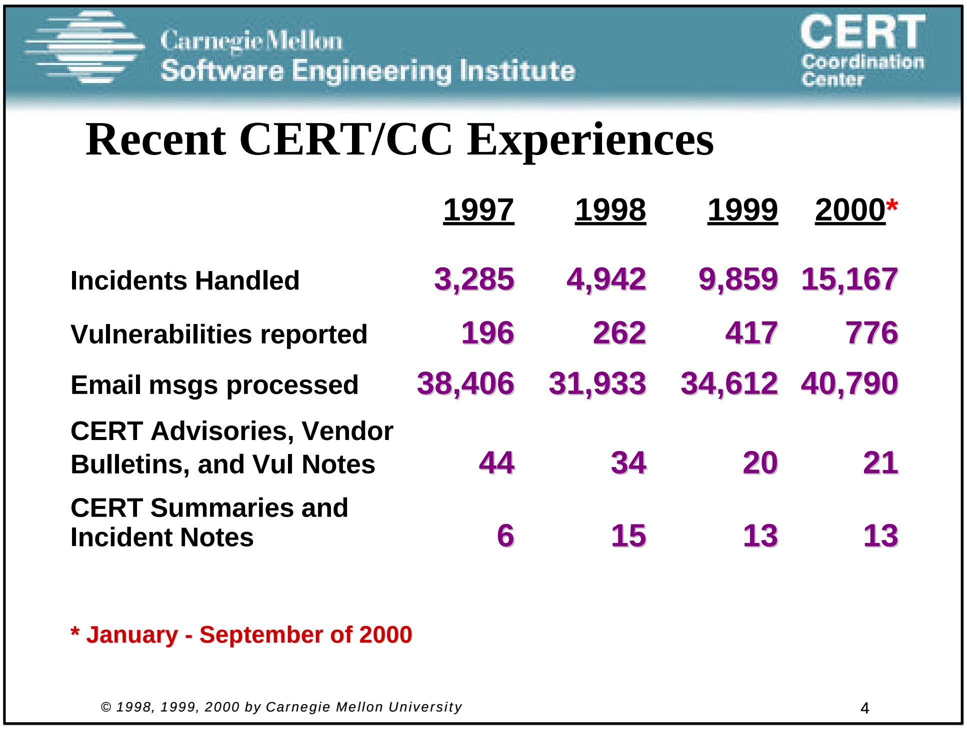
Questions and discussion are welcomed.



New PGP Key - CERT/CC

```
Key ID:      0x20B19259
Key Type:    RSA
Expires:     10/01/01
Key Size:    1024
Fingerprint: 6DDB 095E 348A C560
             1157 0DD1 1E43 FD1D
UserID:      CERT Coordination Center
             <cert@cert.org>
```

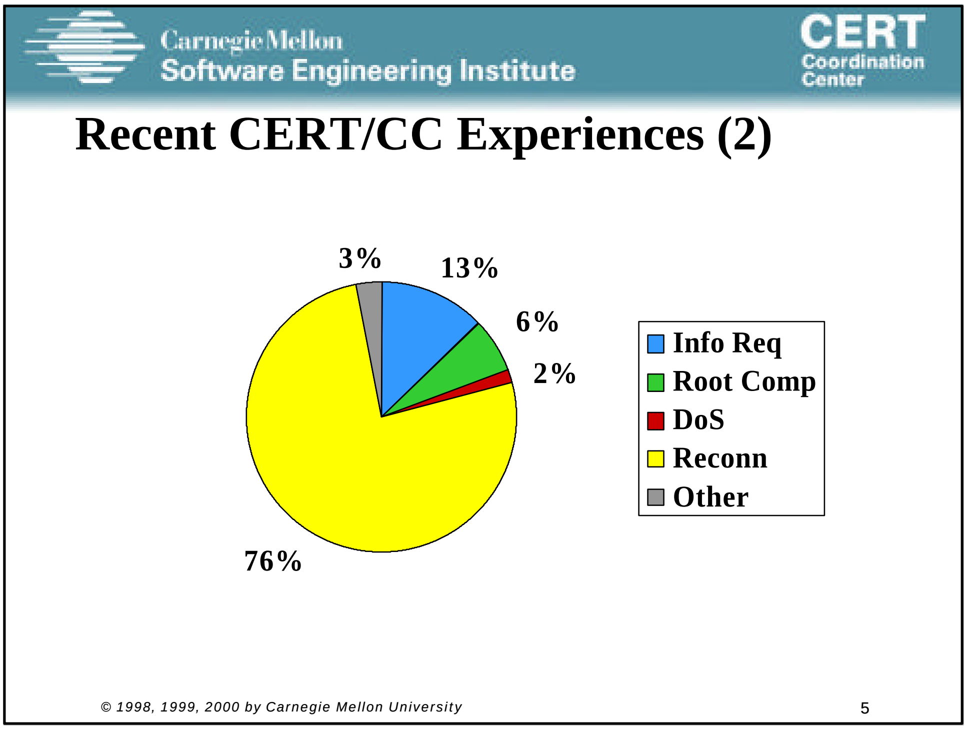
The new key is an RSA key, and it is constructed so as to provide maximum interoperability with as many versions of PGP as possible as well as with GPG.



Recent CERT/CC Experiences

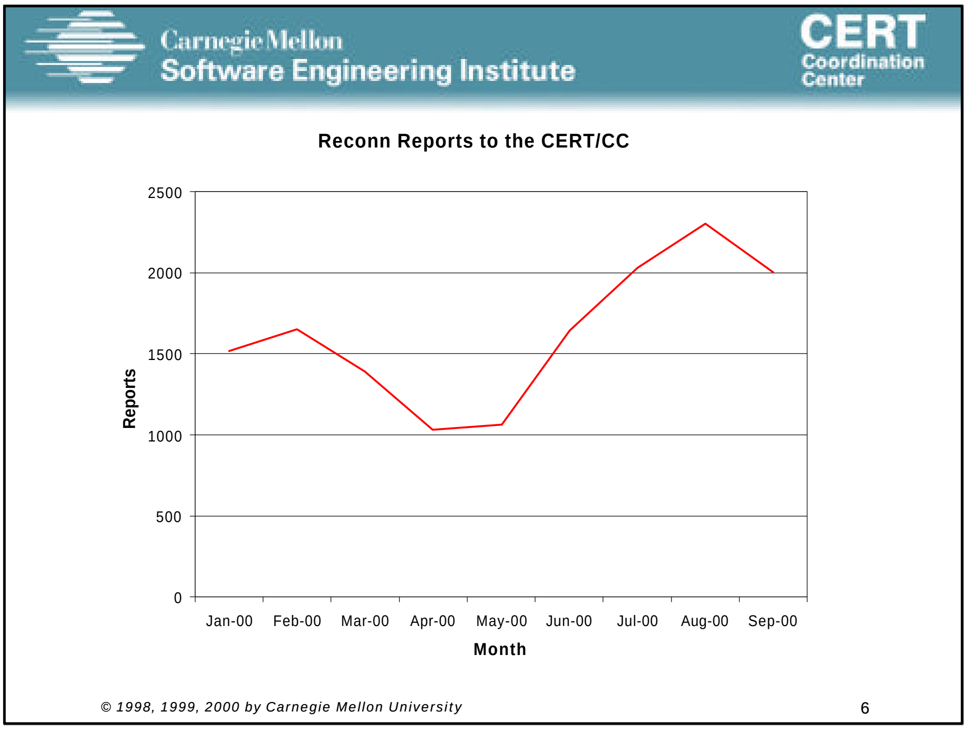
	1997	1998	1999	2000*
Incidents Handled	3,285	4,942	9,859	15,167
Vulnerabilities reported	196	262	417	776
Email msgs processed	38,406	31,933	34,612	40,790
CERT Advisories, Vendor Bulletins, and Vul Notes	44	34	20	21
CERT Summaries and Incident Notes	6	15	13	13

* January - September of 2000

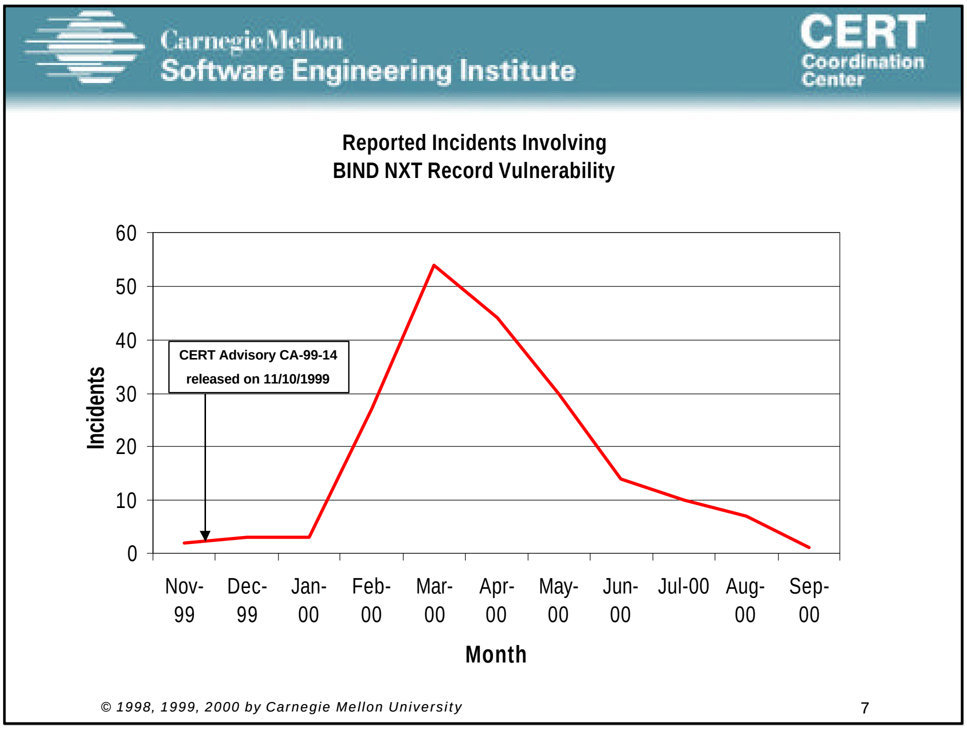


Recent CERT/CC Experiences (2)

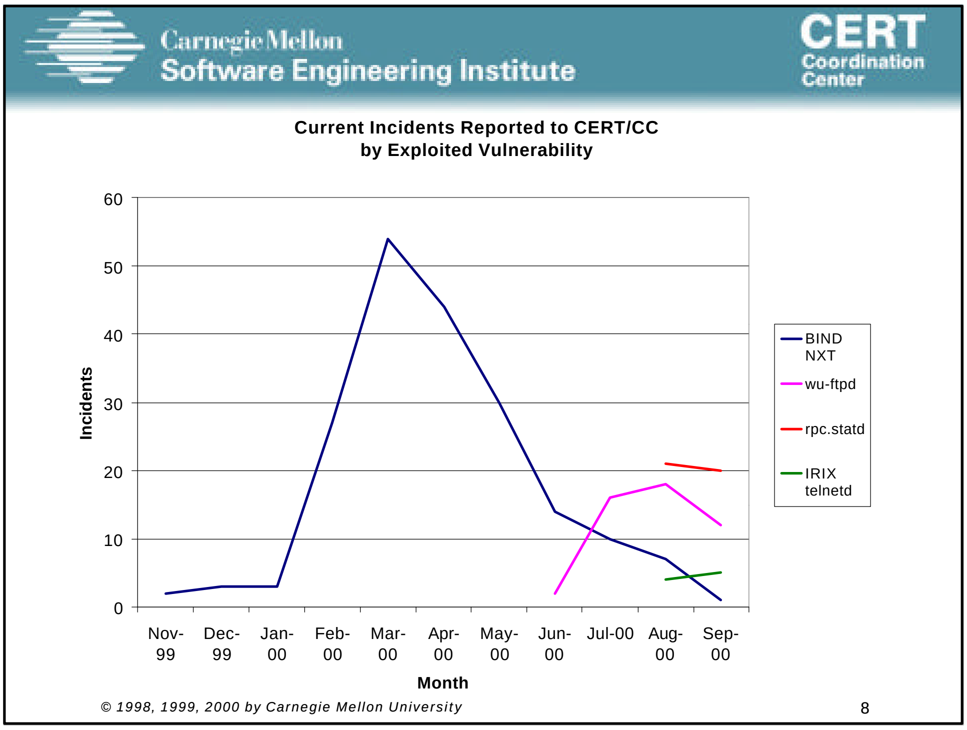
🟦 Info Req
🟩 Root Comp
🟥 DoS
🟨 Reconn
◼ Other



Reconn Reports to the CERT/CC



Reported Incidents Involving BIND NXT Record Vulnerability



Current Incidents Reported to CERT/CC by Exploited Vulnerability

Vulnerability Disclosure Policy

Effective October 9, 2000

http://www.cert.org/faq/vuldisclosurepolicy.html

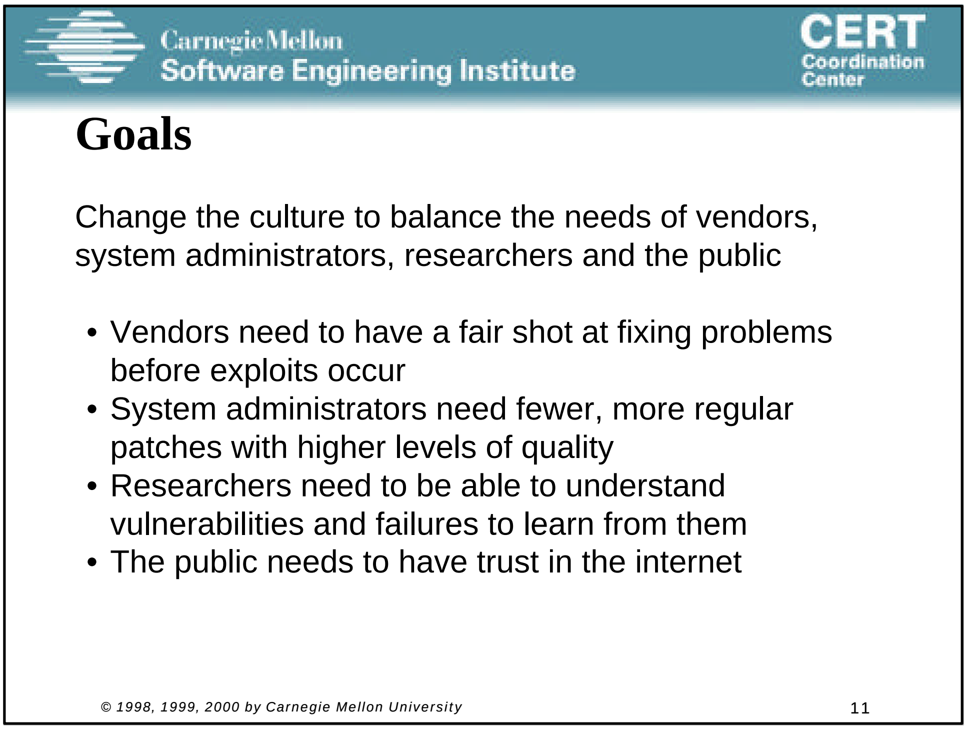
Motivation: the Problems

Problem 1: *the bad old days*

- vulnerability information was available only to a small number of people
- vendors were unresponsive to security concerns
- poor quality, incomplete fixes

Problem 2: *the bad new days*

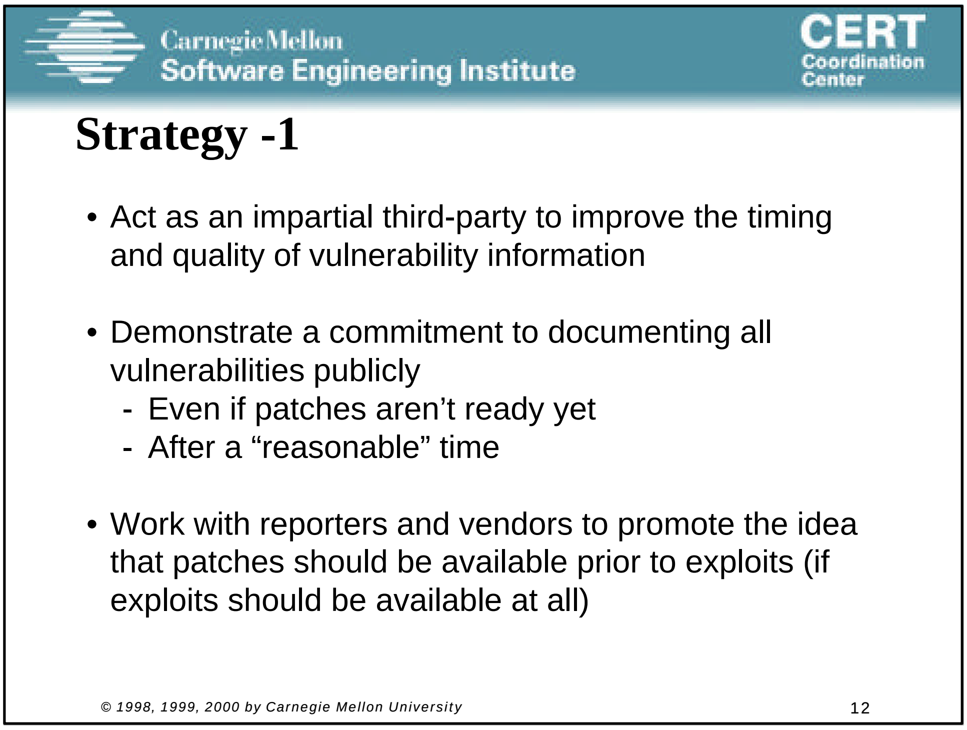
- Exploit information made available before fixes available
- Vendors forced to react immediately (with attendant quality problems)
- Lots of hyperbole and exaggeration
- System administrators unable to manage the information flow, let alone the sheer number of patches



Goals

Change the culture to balance the needs of vendors, system administrators, researchers and the public

- Vendors need to have a fair shot at fixing problems before exploits occur
- System administrators need fewer, more regular patches with higher levels of quality
- Researchers need to be able to understand vulnerabilities and failures to learn from them
- The public needs to have trust in the internet



Strategy -1

- Act as an impartial third-party to improve the timing and quality of vulnerability information

- Demonstrate a commitment to documenting all vulnerabilities publicly
  - Even if patches aren't ready yet
  - After a "reasonable" time

- Work with reporters and vendors to promote the idea that patches should be available prior to exploits (if exploits should be available at all)

Strategy -2

- Low-overhead publishing mechanism
- Document dates of notification to vendor, patch availability, and significant public events
- Provide well-scoped and accurate first information without hype
- Get administrators off the “patch treadmill” -- support aggregation of patches
- Support self-prioritized categorization of vulnerabilities
- Support public discussion of vulnerability information
- Support private collaboration prior to public disclosure

Details

- 45-day nominal disclosure
  - Some earlier (e.g. exploitation, serious threats)
  - Some later (e.g. “hard” problems requiring complex fixes)
- Availability of patches or workarounds from vendors is not necessarily a concern
- No exploits
- Vendors given opportunity to comment upon or rebut our assessments
- Information shared prior to public disclosure with experts, vendors, sponsors, and others who can contribute and with whom we have a trusted relationship

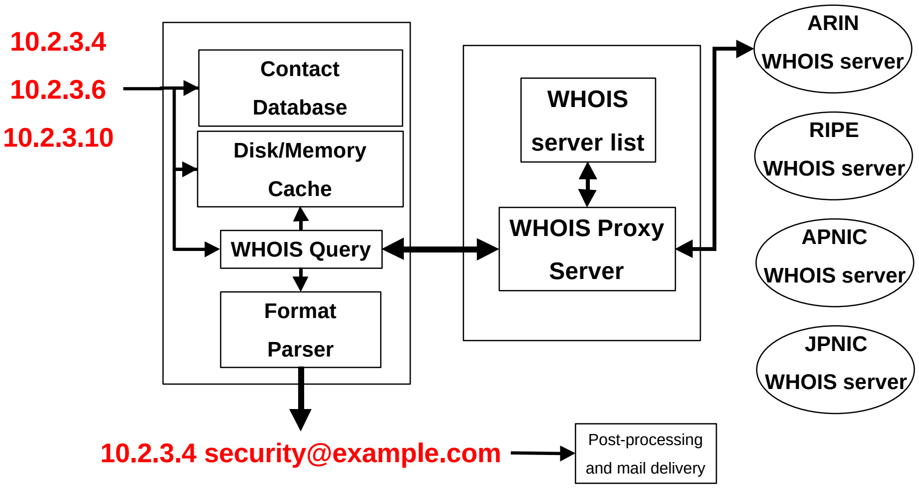
Automating Site Notification

Issues:

- Large-scale incidents involving numerous hosts/sites
- Diverse collection of public contact information
- Special handling for specific constituencies
- Lack of automation

Goal: Automate the process of gathering "accurate" contact information for hosts involved in incidents.

Contact Information Architecture



10.2.3.4
10.2.3.6
10.2.3.10

Contact Database

Disk/Memory Cache

WHOIS Query

Format Parser

WHOIS server list

WHOIS Proxy Server

ARIN WHOIS server

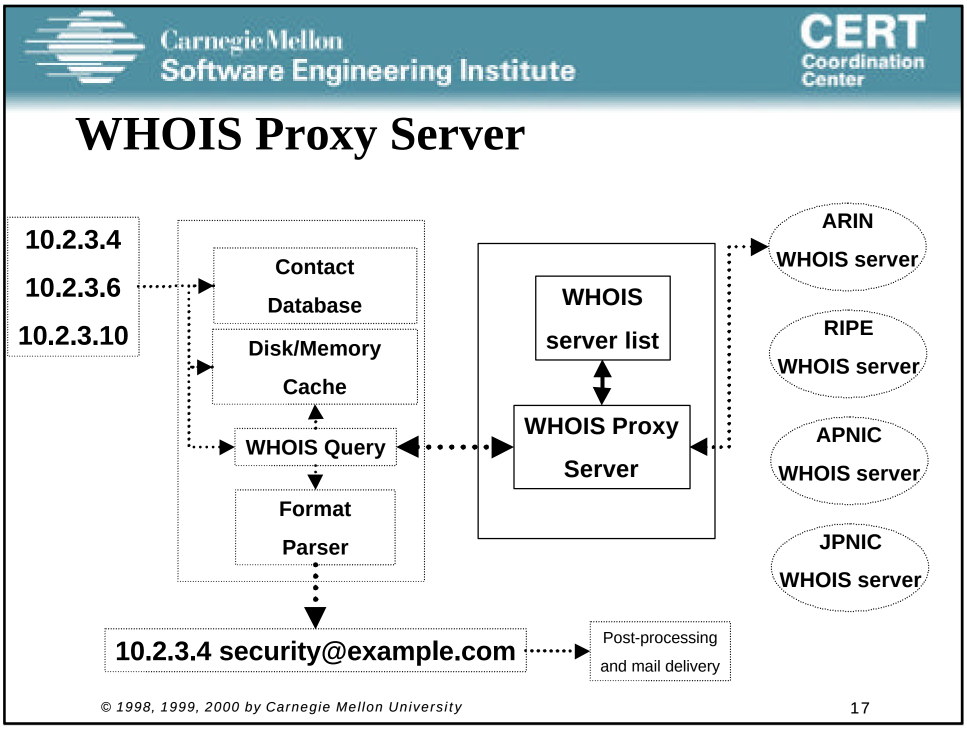
RIPE WHOIS server

APNIC WHOIS server

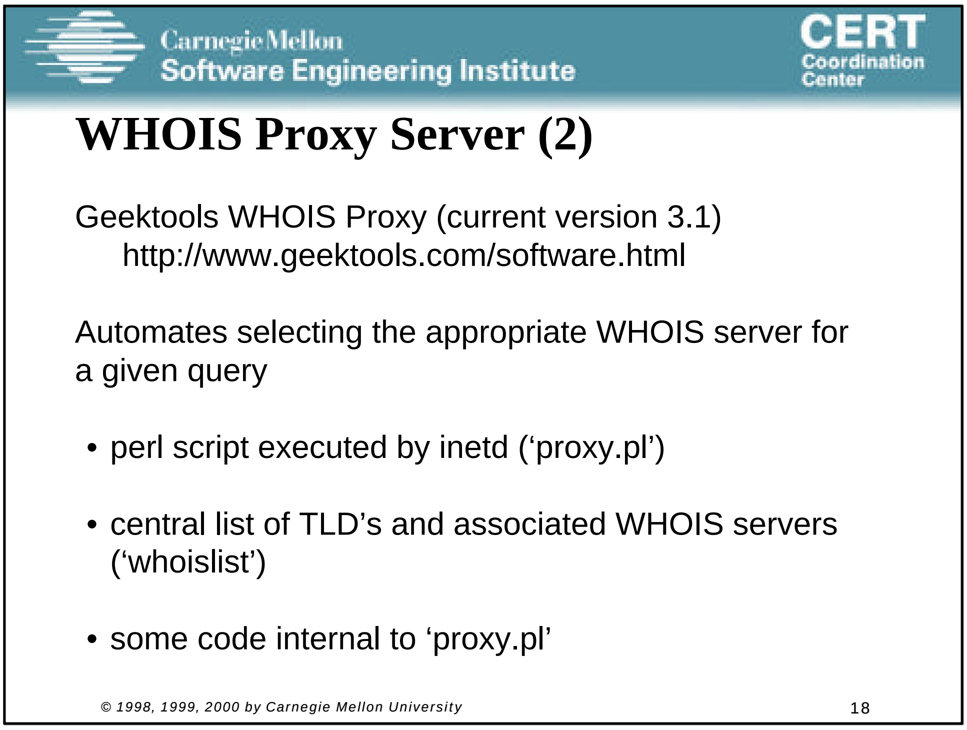
JPNIC WHOIS server

10.2.3.4 security@example.com

Post-processing and mail delivery



WHOIS Proxy Server



WHOIS Proxy Server (2)

Geektools WHOIS Proxy (current version 3.1)
http://www.geektools.com/software.html

Automates selecting the appropriate WHOIS server for a given query

- perl script executed by inetd (‘proxy.pl’)
- central list of TLD’s and associated WHOIS servers (‘whoislist’)
- some code internal to ‘proxy.pl’

WHOIS Proxy Server (3)

Implementation of Geektools WHOIS Proxy

- Added a map of IP address blocks for major IP registries that are not ARIN
  - whois -h whois.arin.net ripe.
  - whois -h whois.arin.net apnic.
  - whois -h whois.arin.net jnic

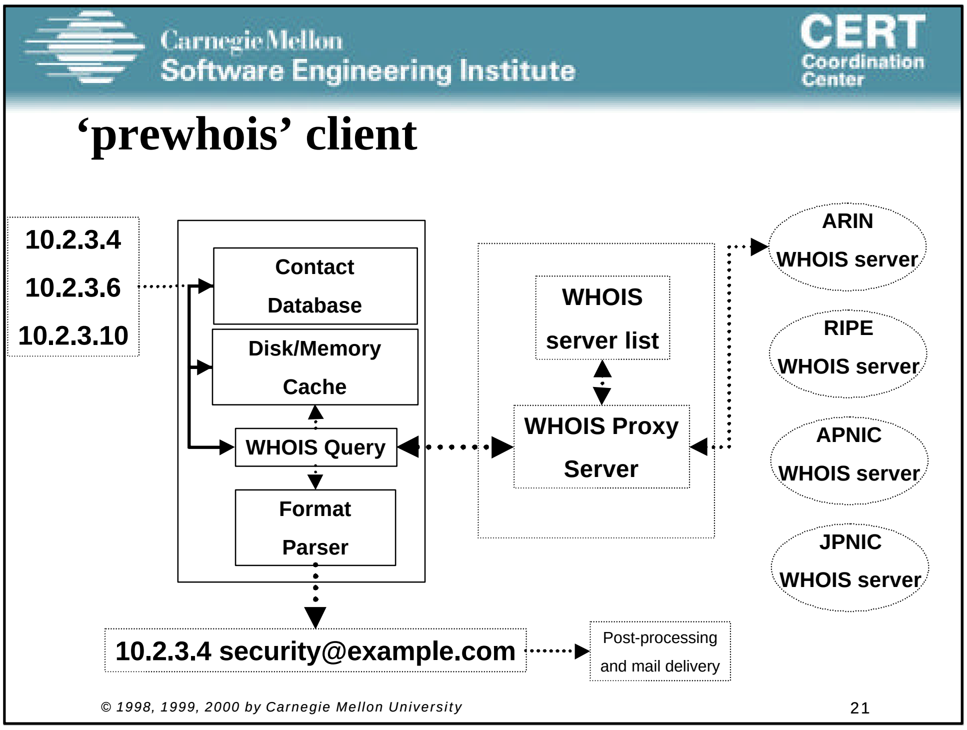
  Improves efficiency over default method of querying ARIN first for every query

- Multiple proxy servers with round-robin DNS

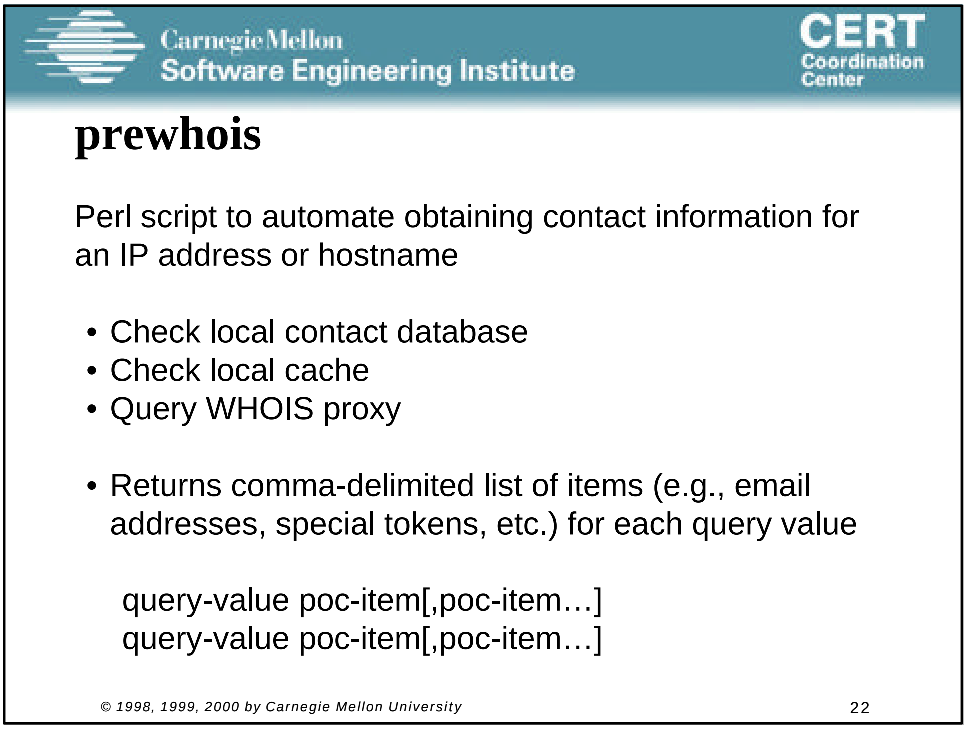
WHOIS Proxy Server (4)

Geektools WHOIS Proxy output:

```
$ whois -h whois-proxy.cert.org cert-dom
Query:     cert-dom
Registry:  whois.networksolutions.com
Results:

Registrant:
CERT Coordination Center (CERT-DOM)
   Software Engineering Inst. Carnegie
   Mellon University
   Pittsburgh, PA 15213

   Domain Name: CERT.ORG

   Administrative Contact, Technical Contact, Zone Contact,
   Billing Contact:
      CERT Coordination Center  (CERT)  cert@CERT.ORG
```



‘prewhois’ client



prewhois

Perl script to automate obtaining contact information for an IP address or hostname

- Check local contact database
- Check local cache
- Query WHOIS proxy

- Returns comma-delimited list of items (e.g., email addresses, special tokens, etc.) for each query value

```
query-value poc-item[,poc-item…]
query-value poc-item[,poc-item…]
```

prewhois - Contact Database

Special handling for specific constituencies

- by {sub}domain
- by machine name
- by IP address block

.{domain}	poc-item[,poc-item...]
.{sub}.{domain}	poc-item[,poc-item...]
{host}.{domain}	poc-item[,poc-item...]
{host}.{sub}.{domain}	poc-item[,poc-item...]
10.0.0.0/8	poc-item[,poc-item...]
10.1.0.0/16	poc-item[,poc-item...]
10.1.2.0/24	poc-item[,poc-item...]

prewhois - Contact Database (2)

.{domain}   poc-item[,poc-item]

- Format for poc-item is arbitrary (with one exception)
- Local convention can be used with post-processing

.gov   premail:fedcirc,cert@cert.org

prewhois - Contact Database (3)

There is one reserved token for ‘poc-item’

```
 .gov          cc:fedcirc@fedcirc.gov
```

Intent is to enable carbon-copy policies for defined constituencies.

- can be used to carbon-copy reports to CSIRT’s when contacting sites directly
- match most specific case only

prewhois - Contact Database (4)

Uses and advantages:

- Automated use of non-public contact information
  - for those times when WHOIS is just plain wrong
  - abuse@your.favorite.isp.here
  - IANA reserved IP address and domain names
- Improved detection of constituent hosts involved in incidents (e.g., large IP address lists)
- Encode preferences of FIRST teams (e.g., send us all reports vs. CC us on all reports)

prewhois - Cache

Store WHOIS proxy answers in a client-side cache

{zone} {timestamp} {poc-list}

‘zone’ is a ‘near-TLD’ or an IP address

cert.org 970503967 cert@cert.org

192.88.210.0/24 970503968 cert@cert.org

prewhois - Cache (2)

IP address blocks are stored using CIDR notation

- use /24, /16, and /8 only (multiples if needed)
- for networks shorter than /24, just cache the IP address without CIDR

Advantages:

- reduces WHOIS queries for sets with numerous hosts in a single domain or IP address range
- aborted processes can be efficiently restarted

prewhois - WHOIS Queries / Parsing

prewhois queries a WHOIS proxy, parses result for email addresses based on the source of the answer

```
%WHOISFormatMap = (
  'whois.networksolutions.com'    => "generic_format",
  'whois.nic.mil'                 => "generic_format",
  'whois.nic.gov'                 => "generic_format",
  'whois.cdnnet.ca'               => "generic_format",
  'whois.awregistry.net'          => "generic_format",
  'whois.internetnamesww.com'     => "generic_format",
  'whois.opensrs.net'             => "generic_format",
  'whois.arin.net'                => "arin_format",
  'whois.ripe.net'                => "ripe_format",
  'whois.apnic.net'               => "ripe_format",
  'whois.nic.br'                  => "ripe_format",
  'whois.denic.de'                => "ripe_format”);
```

prewhois Example

```
$ echo www.example.net | prewhois -d
# debug: load_contact_file(prewhois.contact.txt) loaded 29 contacts
# using contact database 'prewhois.contact.txt'
# debug: load_tld_map(tld-site) loaded info for 256 TLD's
# debug: using cachefile 'prewhois.cache.db'
# debug: check_contactdb(www.example.net) returns ''
# debug: get_domain(www.example.net) returns 'example.net'
# debug: check_cache(example.net) returns ''
# querying server 'whois-proxy' for 'example.net' ...
# debug: whois_lookup(example.net, whois-proxy)
# debug: tcp_connect(whois-proxy, whois)
# debug: find_registry(...) returns 'whois.networksolutions.com'
# response received from 'whois.networksolutions.com'
# debug: generic_format(...) returns ('iana@IANA.ORG','')
# debug: 'example.net 970511743 iana@iana.org' added to cache
# debug: unique(iana@iana.org) returns 'iana@iana.org'
# POC list for 'www.example.net' is 'iana@iana.org'
www.example.net iana@iana.org
```

prewhois Example (2)

```
$ echo test.example.net | prewhois -d
# debug: load_contact_file(prewhois.contact.txt) loaded 29 contacts
# using contact database 'prewhois.contact.txt'
# debug: load_tld_map(tld-site) loaded info for 256 TLD's
# debug: using cachefile 'prewhois.cache.db'
# debug: check_contactdb(test.example.net) returns ''
# debug: get_domain(test.example.net) returns 'example.net'
# debug: check_cache(example.net) returns 'iana@iana.org'
# debug: unique(iana@iana.org) returns 'iana@iana.org'
# POC list for 'test.example.net' is 'iana@iana.org'
test.example.net iana@iana.org
```

prewhois Example (3)

```
$ echo 192.88.209.1 | prewhois -d
# debug: load_contact_file(prewhois.contact.txt) loaded 29 contacts
# using contact database 'prewhois.contact.txt'
# debug: load_tld_map(tld-site) loaded info for 256 TLD's
# debug: using cachefile 'prewhois.cache.db'
# debug: check_contactdb(192.88.209.1)
# debug: netblock(192.88.209.1,8) returns 192.0.0.0/8
# debug: netblock(192.88.209.1,16) returns 192.88.0.0/16
# debug: netblock(192.88.209.1,24) returns 192.88.209.0/24
# debug: check_cache(192.88.209.1)
# debug: netblock(192.88.209.1,8) returns 192.0.0.0/8
# debug: netblock(192.88.209.1,16) returns 192.88.0.0/16
# debug: netblock(192.88.209.1,24) returns 192.88.209.0/24
# querying server ’whois-proxy’ for '192.88.209.1' ...
# debug: whois_lookup(192.88.209.1, whois-proxy)
# debug: tcp_connect(whois-proxy, whois)
# debug: find_registry(...) returns 'whois.arin.net'
# response received from 'whois.arin.net'
# debug: arin_format(...) returns ('cert@CERT.ORG','192.88.209.0:192.88.209.0')
# debug: find_netblock(192.88.209.0:192.88.209.0)
# debug: '192.88.209.0/24 970512535 cert@cert.org' added to cache
# debug: unique(cert@cert.org) returns 'cert@cert.org'
# POC list for '192.88.209.1' is 'cert@cert.org'
192.88.209.1 cert@cert.org
```

prewhois Example (4)

```
$ cat hostlist | prewhois
# querying server ’whois-proxy' for 'ripe.net' ...
# response received from 'whois.networksolutions.com'
# POC list for 'www.ripe.net' is 'dfk@ripe.net,ops@ripe.net'
www.ripe.net dfk@ripe.net,ops@ripe.net
# querying server ’whois-proxy' for 'cert.org' ...
# found NSI hostname and domain name 'cert.org'
# querying server ’whois-proxy' for 'CERT-DOM' ...
# response received from 'whois.networksolutions.com'
# POC list for 'www.cert.org' is 'cert@cert.org'
www.cert.org cert@cert.org
# POC list for 'ns1.example.com' is 'iana@iana.org'
ns1.example.com iana@iana.org
# POC list for 'ns2.example.com' is 'iana@iana.org'
ns2.example.com iana@iana.org
```

prewhois - Status

Usable, but still being developed…

- Input from FIRST teams for contact database to automate use point of contact policies for teams
- Parsing support for more registry output formats
- Needs more comprehensive testing and continued logic improvements
- Post-processing tools needed (srmail)

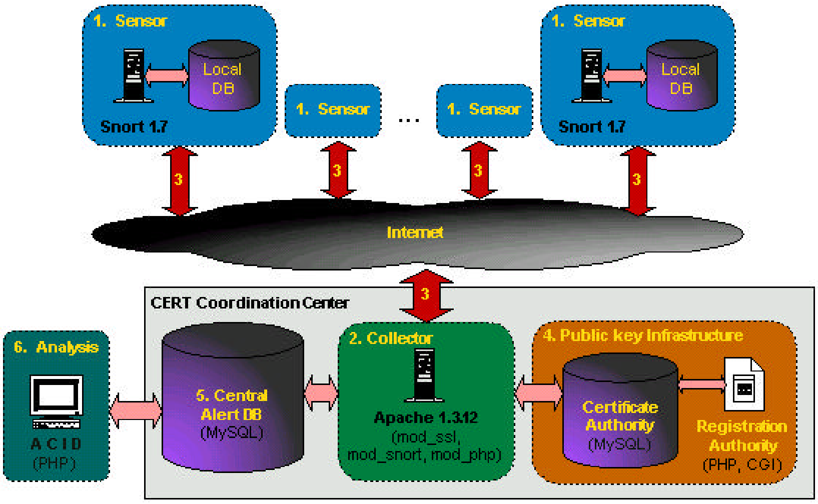
AirCERT - Overview

Automated Incident Reporting (AIR)

- Collect a data-stream of “events” from remote (participating) Internet sensors
  - Local events of concern, different from AirCERT events of concern

- Reduce manual processing of known events
  - start with scans/probes (>75% of reports)

AirCERT - Prototype Architecture



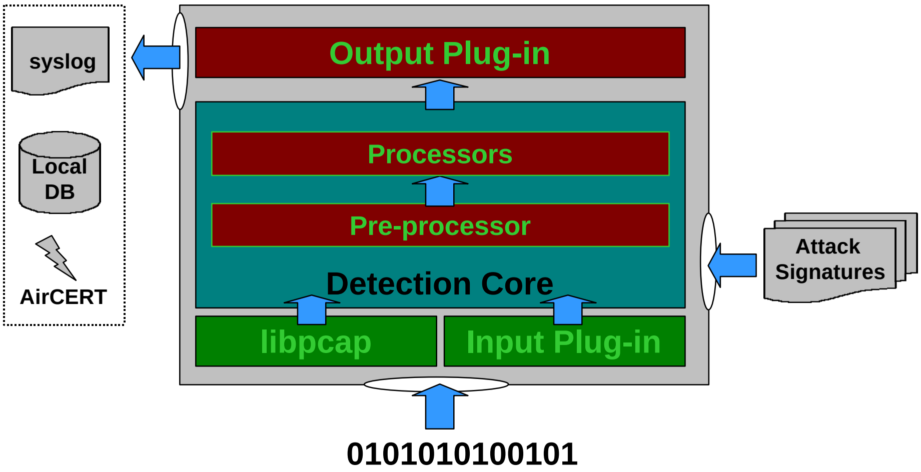
AirCERT Sensor - ‘snort’

Function: gather intrusion data

Snort - (http://www.snort.org/)
- Open-source NIDS
- Signature-based: triggered on single packet
- Single-threaded
- Logging facilities: TCPDump, database, or syslog

AirCERT awareness
- Alert encoding: XML
- Network infrastructure code
- Local and AirCERT logging (Andrew Baker)
- Sanitization

AirCERT Sensor - ‘snort’ (2)



Data: network traffic, stored data

AirCERT Collector - ‘mod_snort’

Function: aggregate sensor data

Apache
- Open-source Web server

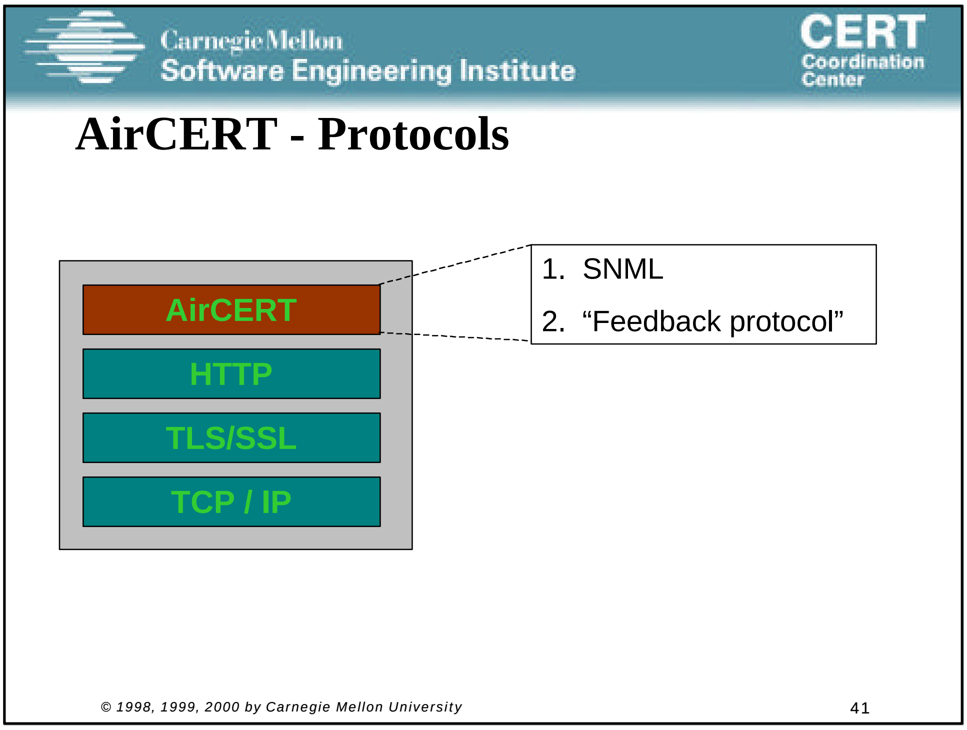
AirCERT awareness
- Alert “processing” = mod_snort
    - specialized POST handler

AirCERT Collector - Alert Processing

Sensor=>Collector: Client sends alert(s) via POST

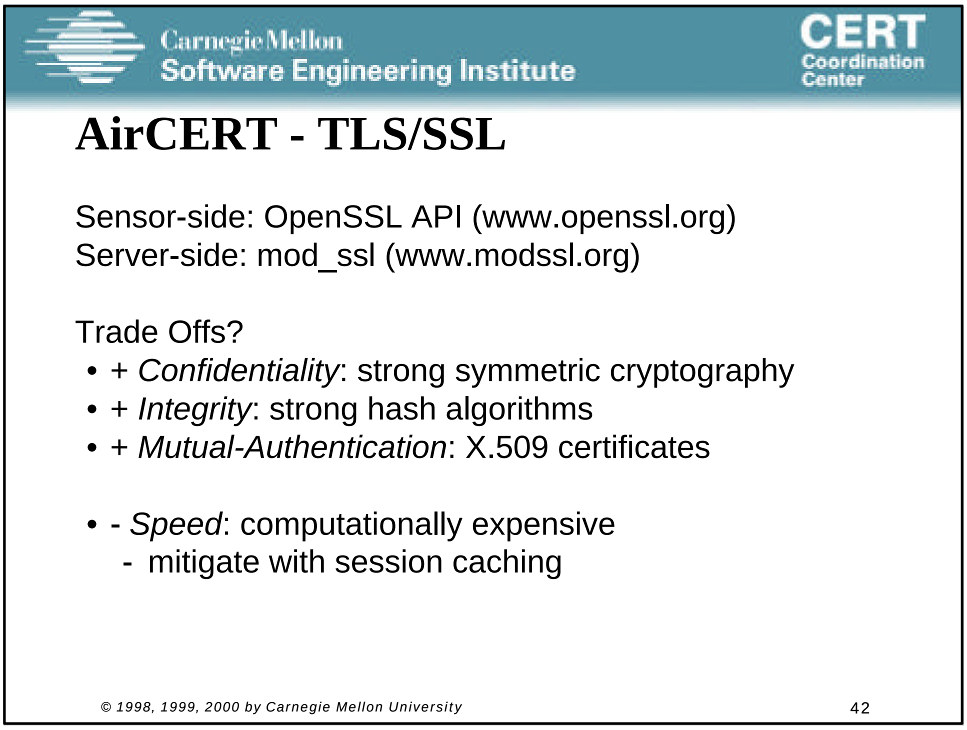
Apache core: gets request
- mod_ssl: Verify credentials
- mod_snort: Authentication sensor
- mod_snort: Throttling
- mod_snort: Alert Parsing (well-formed)
- mod_snort: Log Alert

Collector=>Sensor: returns Alert processing status



AirCERT - Protocols

1. SNML
2. “Feedback protocol”



AirCERT - TLS/SSL

Sensor-side: OpenSSL API (www.openssl.org)
Server-side: mod_ssl (www.modssl.org)

Trade Offs?

- + *Confidentiality*: strong symmetric cryptography
- + *Integrity*: strong hash algorithms
- + *Mutual-Authentication*: X.509 certificates

- - *Speed*: computationally expensive
  - mitigate with session caching

AirCERT - SNML

Snort Network Markup Language (SNML)
Function: standardized alert description language

XML-based

- Meta detection information
  - date and time
  - triggering signature
  - sensor: name, interface, and filters
- Contents of packet (header and data)

Related Work:

- IETF IDWG: Intrusion Detection Message Exchange Format (IDMEF)

AirCERT - SNML Example

```
<?xml version="1.0" encoding="UTF-8">
<!DOCTYPE snort-message-version-0.1 PUBLIC>

<report>
  <event version="1.0">
    <sensor encoding="hex" detail="full">
      <interface>eth0</interface>
      <ipaddr version="4">128.2.66.93</ipaddr>
      <hostname>box</hostname>
    </sensor>
    <signature>IDS279 - BACKDOOR SIGNATURE - SubSeven Login</signature>
    <timestamp>2000-08-25 13:17:27-05</timestamp>
    <packet>
      <iphdr saddr="128.2.66.93" daddr="128.2.237.74" proto="6" ver="4" hlen="5"
             len="60" id="6681" ttl="64" csum="61686">
        <tcphdr sport="1213" dport="23" flags="2" seq="3864041455" ack="0" off="2560"
                 win="32120" csum="56664">
          <option code="2" len="4">05B40402</option>
          <option code="4"/>
          <option code="8" len="10">00173A5E000000000103</option>
          <option code="1"/>
          <option code="3" len="3">000000</option>
        </tcphdr>
      </iphdr>
    </packet>
  </event>
</report>
```

AirCERT - Feedback Protocol

Function: return feedback from collector to sensor

Rudimentary “Command-and-Control” (C2)
Text-based protocol
3-classes of messages

- Authentication (3xx)
- Input validation (4xx)
- Throttle (5xx)

Related Work:

- IETF IDWG: Intrusion Alert Protocol (IAP)

AirCERT - PKI

Function: infrastructure for disseminating and validating public-keys

Required to support mutual authentication in TLS/SSL
Components:

- Certificate Authority (CA): validate and store certificate info
  - certificate database: MySQL
- Registration Authority (RA): sign public-keys
  - certificate signing code: OpenSSL, PHP

AirCERT - PKI (2)

Creating certificate

- Create a public-key
- Create a certificate signing request (CSR)
- Submit CSR to the RA
- RA returns a valid signed CSR = certificate

Validating a certificate

- Verify integrity of certificate’s signature
- Verify valid date
- Verify against a Certificate Revocation List (CRL)

AirCERT - ACID

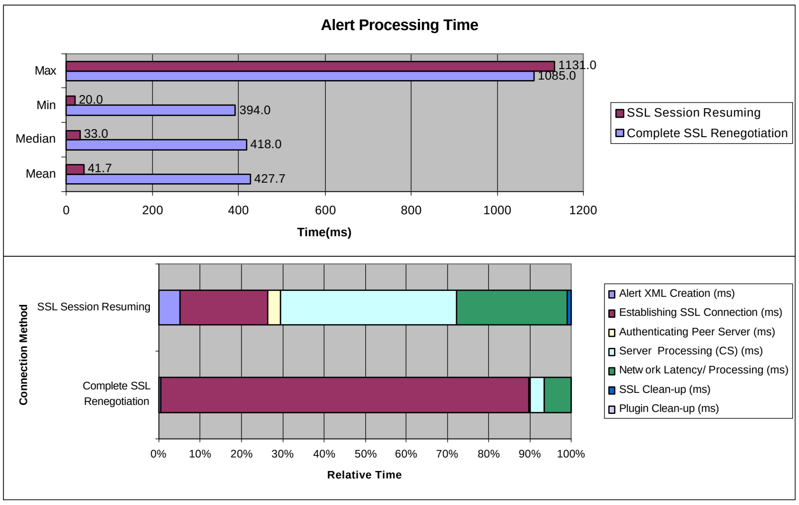
Analysis Console for Intrusion Databases
<u>Function</u>: analyze collected incident data

PHP-based scripts

Current Features:

- Search interface
- Statistics
- Alert groups
- Alert purging

AirCERT Performance Analysis



AirCERT Status

Prototype completed

- Sensor, Collector, PKI, and Analysis Engine implementations

Current State: Testing

- *Internal* (ongoing): CERT
  - large-volume data testing
  - usability
- *External* (late October): Snort-users, CERT clients
  - validate signature-set
  - stress scalability
  - usability
- *Public* (??): Internet community
  - validate usefulness of AirCERT data collection

AirCERT Information

AirCERT Documentation

(http://www.cert.org/kb/aircert)
- XML (http://www.cert.org/kb/snortxml)
- ACID (http://www.cert.org/kb/acid)

CERT® Contact Information

CERT Coordination Center
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh PA 15213-3890
USA

Hotline:	**+1 412 268 7090**	**CERT personnel answer 8:00 a.m. — 8:00 p.m. EST(GMT-5) / EDT(GMT-4), and are on call for emergencies during other hours.**
Fax:	**+1 412 268 6989**	
Web:	**http://www.cert.org/**	
Email:	**cert@cert.org**	