

.
.

COS80013

Internet Security

Week 11

Presented by Dr Rory Coulter

19 May 2025



. . .
. . .

.
.



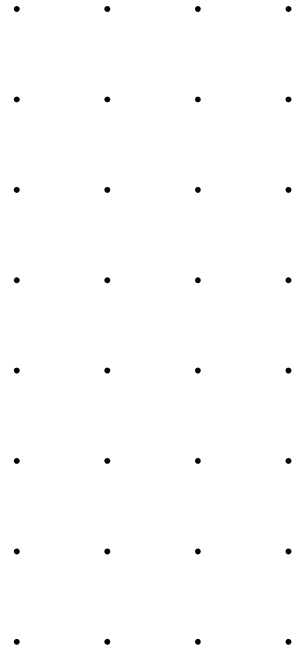
Week 11 Class

Assignment 2 Preparation

Logging, Hashing

Overview and basic usage of each element provided

- What
- Resources
- Demonstration



System Monitoring

How are systems events handled

Broadly between Unix-based (let's just say Linux) and Windows

- Events occur within a system
- Event logs capture:
 - Date, time
 - Device
 - Description
 - Level
 - Associated application/process
 - Specific event type
 - Characteristic
 - Networking information in relevant
- Typically, Operating system event logs relate to
- System events from the operating system itself
- E.g., Syslog/Auth (Linux), Sysmon (Windows)
- Applications
 - Security events
 - Application logging may include
 - Request type
 - Status
 - Message
 - Networking
 - Event type
- Log structures are standardised, structured
- Logs should be centralised for monitoring
- Not everything is logged from install

Logging Locations

Overview and Demonstration

Windows vs Linux

- Linux
 - Audit, kernel, events, scheduled tasks
 - `/var/log/`
- Windows
 - Application, Security, Setup, System, and Forwarded
 - `C:\Windows\System32\winevt\Logs`



Key Windows Events

Logon, Privilege Use, Defender

Key events

– Logon

- 4624: User successfully logged on to a computer
- 4625: Attempt made to logon with unknown user name or bad password and failed
- 4822: NTLM authentication failed because the account was a member of the Protected User group

– Privilege Use

- 4660: Object deleted
- 4698: A scheduled task was created
- 4699: A scheduled task was deleted

– Defender

- 1002: malware scan stopped before completing scan
- 1015: suspicious behaviour detected

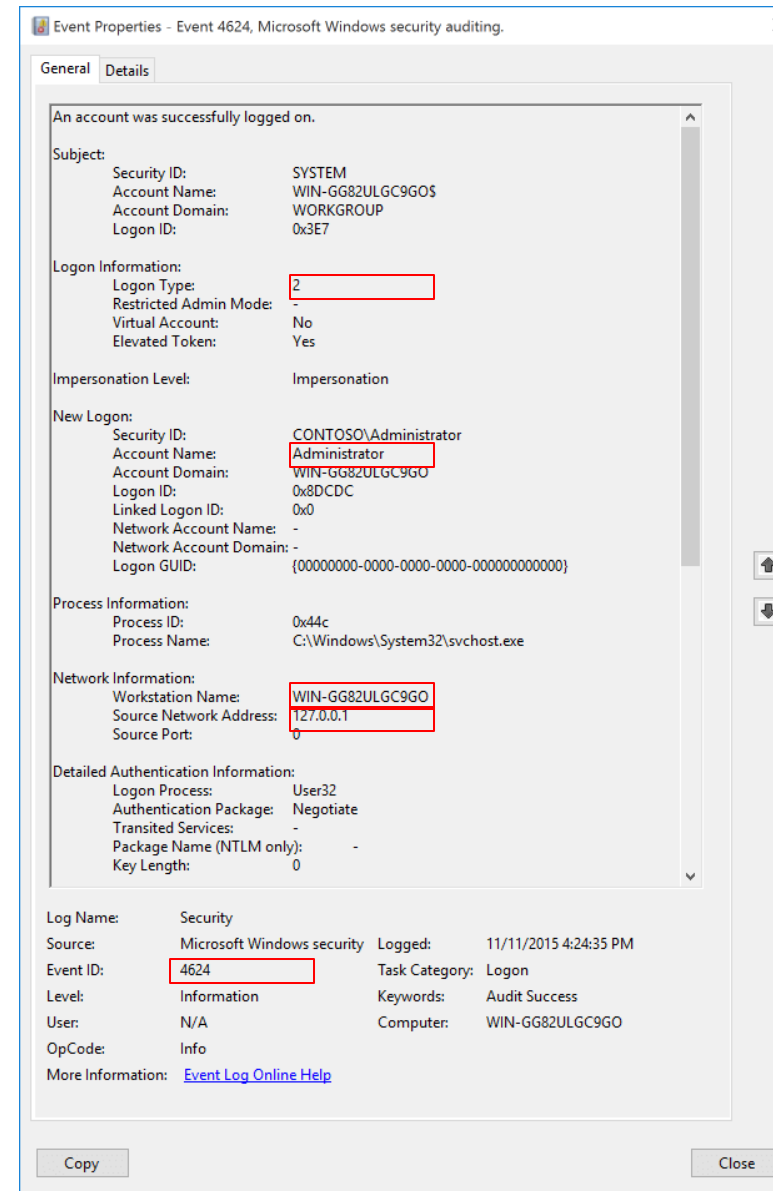


An Example

Windows System Monitor Log: Event Type 4624

4624(S): An account was successfully logged on

- Administrator account logged on
- Logon type is 2, interactive
- Workstation name: WINGG82ULGC9GO
- Source network address is 127.0.0.1



Auth.log

Linux authentication and authorisation events

Successful and failed attempts

- User logins
- Sudo usage
- Password changes
- Authorisation attempts

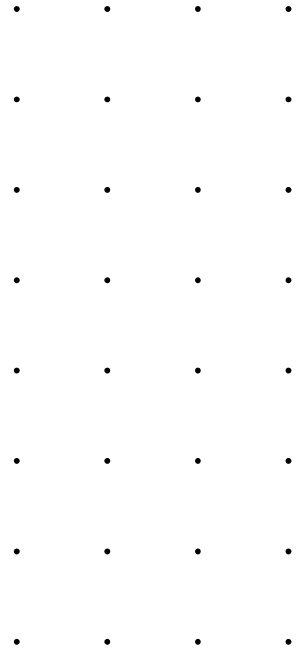


Demonstration

Simulate via terminal and logon

Event

- Sudo events
- Login events



Sysmon

More detailed logs

Demonstration

- Install
- Failed attempts
- Pause defender
- Drop malware

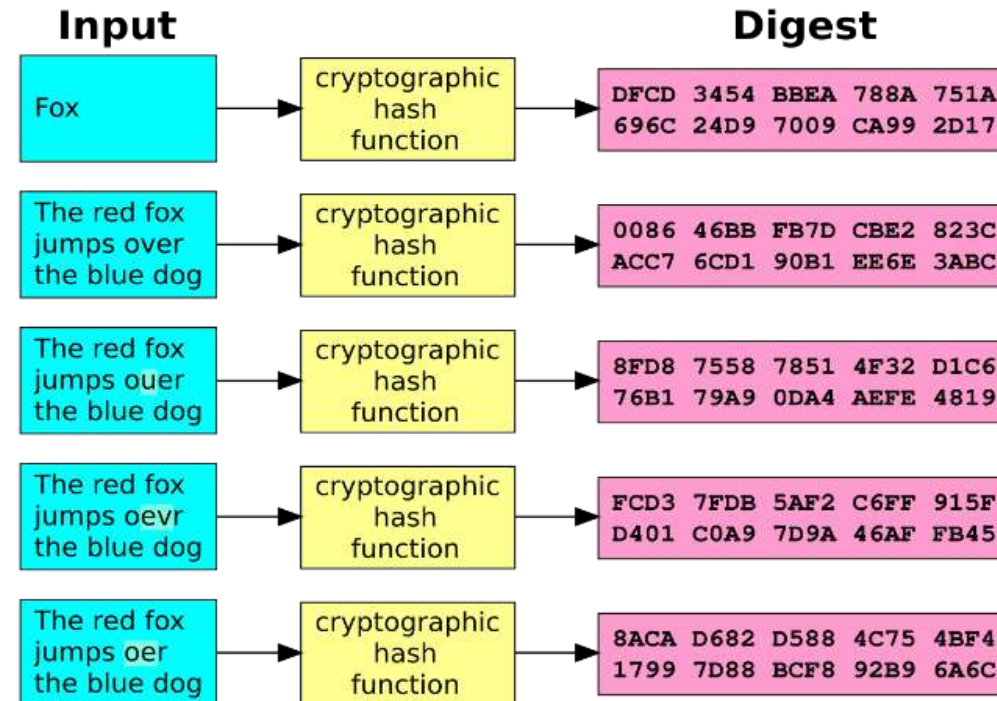


Hashing

Digital Signatures

has or digest

- Input of data (file, string) of an unfixed length and returning a fixed-length message digest/fixed-length string signature
- Aim is to ensure integrity
- Verification of files (e.g., transfer)
- Range of algorithms available
- In-built into OS, online tools

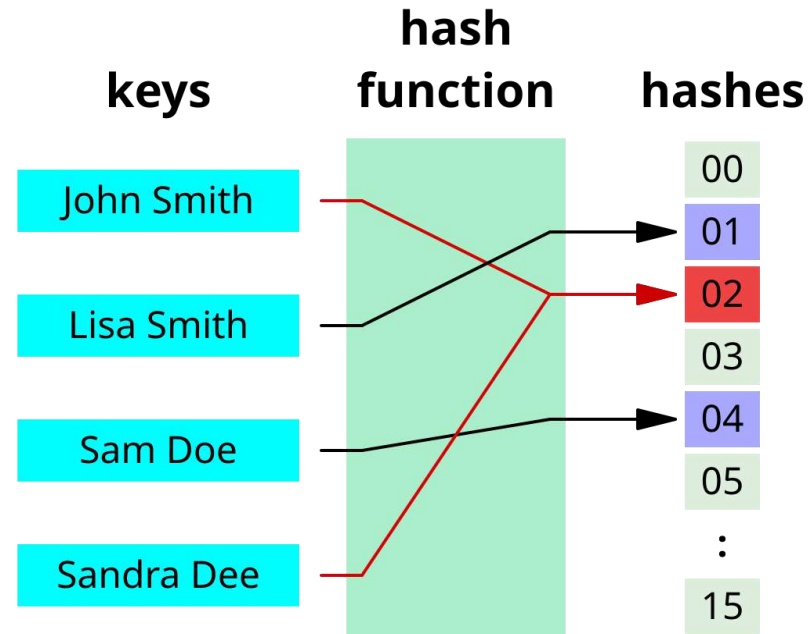


Hash Collisions

Different input, same output

MD5, SHA-1

- Collisions allow incorrect file to be accepted as correct
- Integrity cannot be guaranteed



. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

Available Tools

Inbuilt and Online

Demonstration

- Terminal
- Cyber Chef



Spam

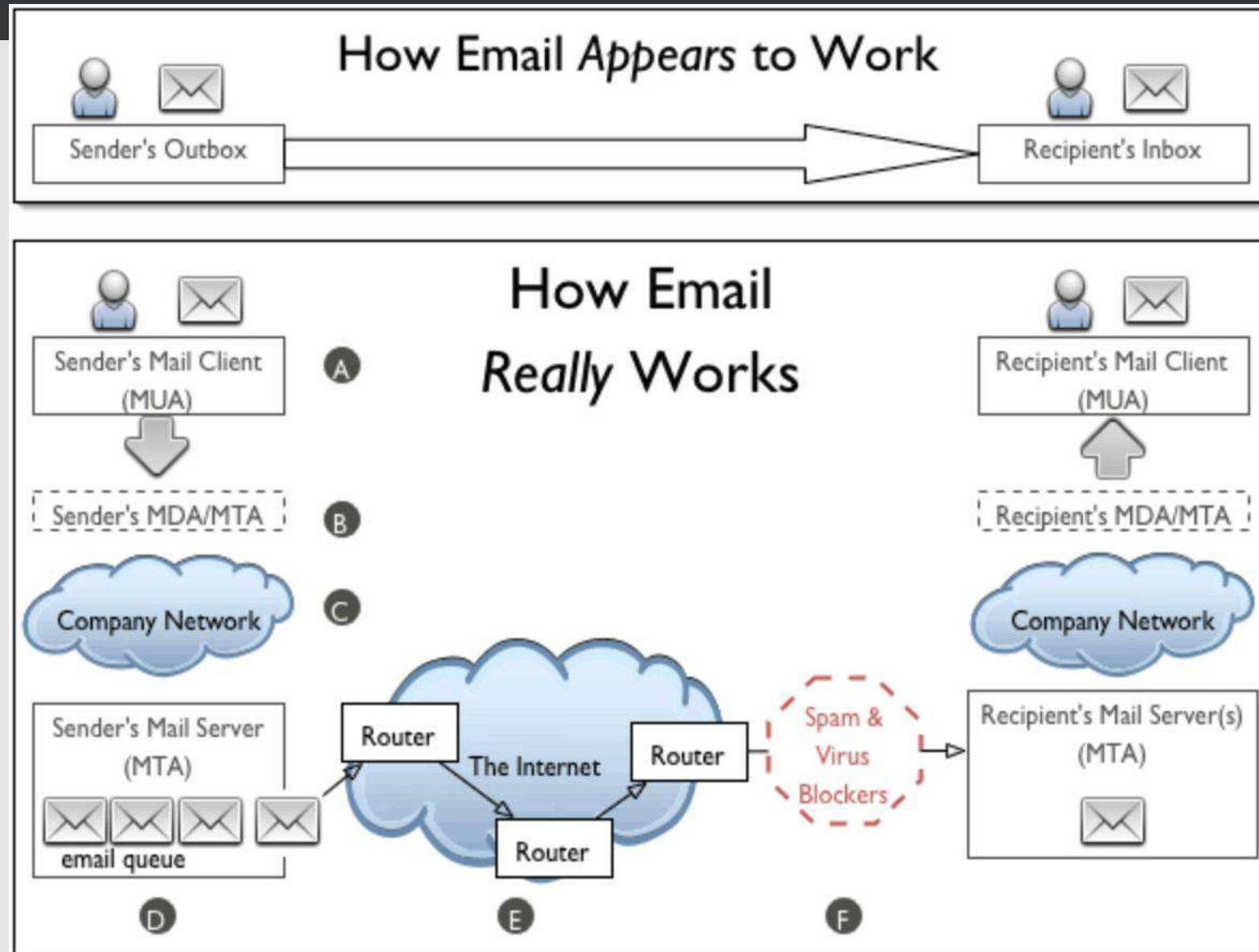
Unsolicited bulk email

Can be used to:

- Send advertising material
- Collect information
- Mount phishing attacks
- Mount pharming attacks
- Distribute malware
- Conduct social engineering attacks



Spam --- How Email works



Tracing Spam E-mails

- Spam e-mails comprise 70~90% of all e-mails.
- About 60% of all spam e-mails carry links to malware and pharming sites. About 30% are ads for drugs.
- Used extensively for scams, fraud, phishing attacks.
- Most spam is sent by **spam-bots** (automated spam generation and addressing).

Spam-Bot

- A type of malware which is used to send spam.
- Can create e-mail accounts.
- Search the web for e-mail addresses.
- Generate pseudo-random spam and send it.
- Some can
 - Crack passwords, solve CAPTCHA puzzles.
 - Install malware, host servers.

Tracing Spam E-mails

- Spam can be identified by the path it travels to get to you.
- Every email comes with **a header** listing the names and IP addresses of each mail server through which it has passed.
- The sender can be verified by performing a reverse DNS lookup on the sender's IP address.
- Known sources of spam can be looked up using a published black-list.

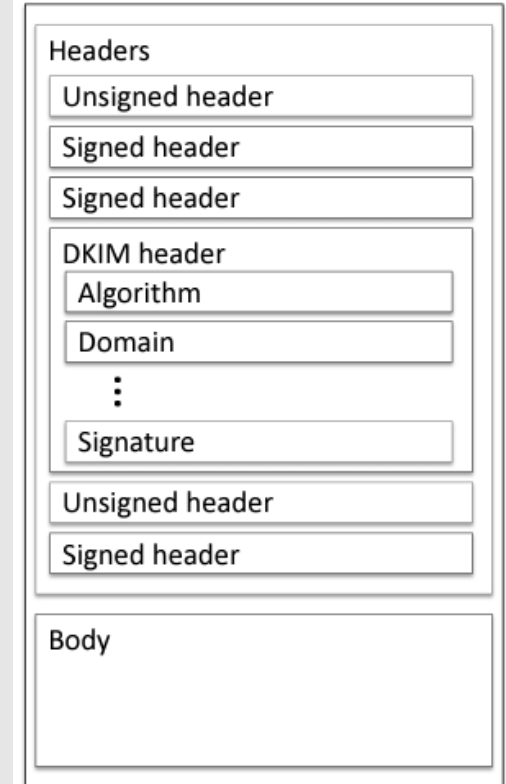
MIME --- Sending User Authentication

- MIME: Multipurpose Internet Mail Extension
- MIME message body:
 - Message itself including text and attachments
 - Signature



DKIM --- Sending MTA Authentication

- DKIM: DomainKeys Identifier Mail
- Some attributes of DKIM:
 - a: identifier of cryptographic algorithm
 - c: canonicalization algorithms for header and body
 - d: domain of the signing entity
 - s: selector of the signing key
 - bh: hash of the body of the message
 - b: signature



```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=brown.edu; s=cs;  
h=domainkey-signature:mime-version:received:in-reply-to:references  
:date:message-id:subject:from:to:cc:content-type;  
bh=L+J52L7uTfKTel/+2ywqQMH1eiGvl6tsXjDNAySew+8=;  
b=vE2bvcj8GVHGHeECJA4WJ/t1BRbLBvITQywbZl/HgFSMRfoIVUvH9lyVeMitOaNMeQ  
C29TNP5fJPphaFhHb9tf8EkJBlojRryWRAI5/r5RgT6z5DLWs8fgHe0wUbWEwBQ+sSTs  
A+vbfulObS1Gwdxu81HNOfiSLY0u2CM6R31s=
```

E-mail Headers

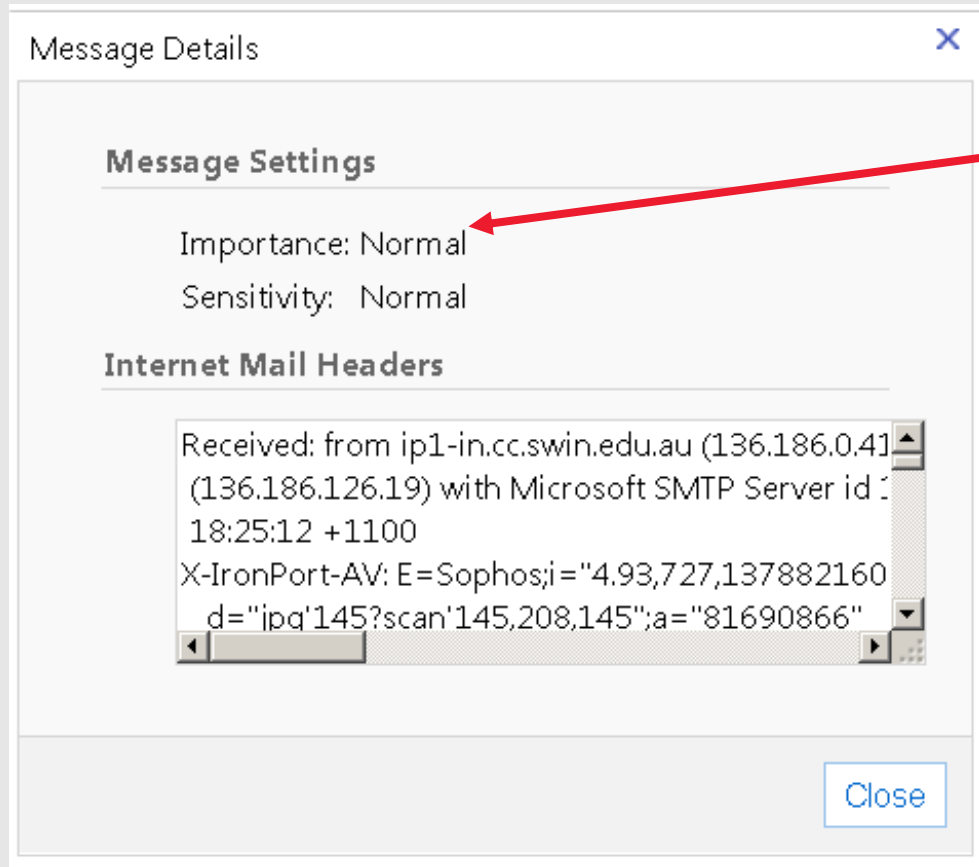


Fig 2. Internet Mail Headers

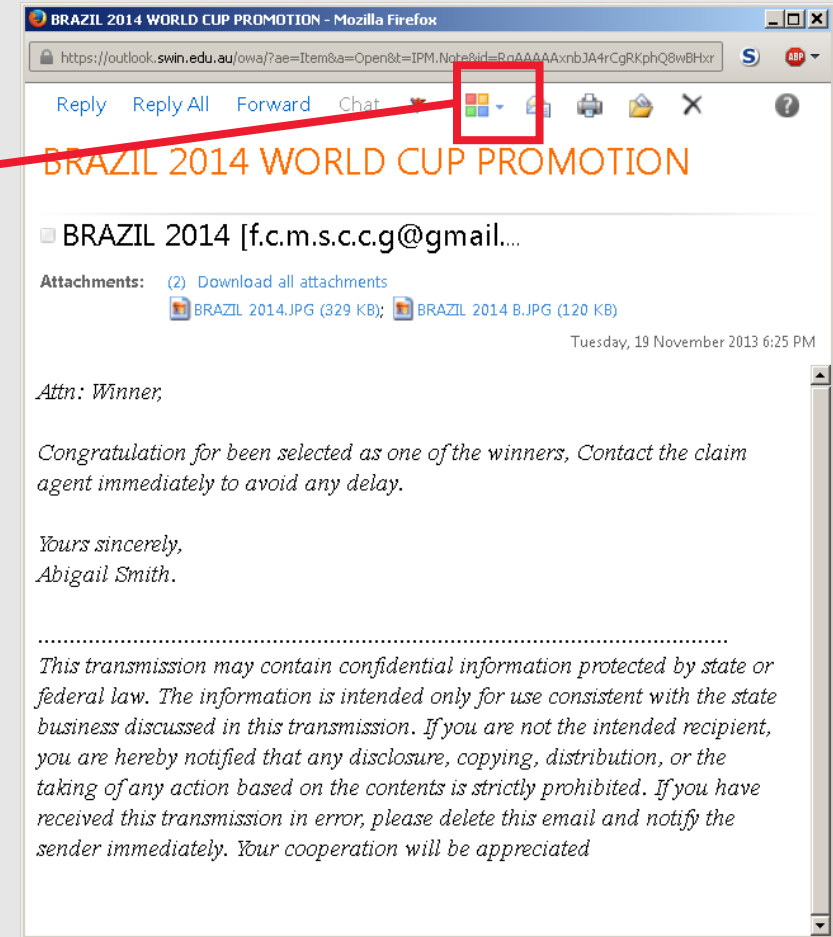



Fig 1. An example of an e-mail

E-mail Header

```
Content-Type: image/jpeg; name="145.jpg"; data:
  d="jpg'145?scan'145,208,145";a="81690866"
Received: from gpo4.cc.swin.edu.au ([136.186.1.33]) by ip1-in.cc.swin.edu.au
  with ESMTP; 19 Nov 2013 18:25:12 +1100
Received: from mail-oa0-f52.google.com (mail-oa0-f52.google.com
  [209.85.219.52]) by gpo4.cc.swin.edu.au (8.14.3/8.14.3) with ESMTP id
  rAJ7P2gH031114 (version=TLSv1/SSLv3 cipher=RC4-SHA bits=128 verify=FAIL)
  for
  <jhamlynharris@swin.edu.au>; Tue, 19 Nov 2013 18:25:06 +1100
Received: by mail-oa0-f52.google.com with SMTP id h16so2441195oag.11
  for <jhamlynharris@swin.edu.au>; Mon, 18 Nov 2013 23:25:02 -0800
(PST)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
  d=gmail.com; s=20120113;
  h=mime-version:reply-to:date:message-id:subject:from:to:content-type;
  bh=PnUts3Gl3c94wk5Da3k/D3nWn0cpih/ZY7pTWmhAPYw=;
  b=ivxuMhRYnDPAeH1R58QXjhFfOkfcOW7m/IouIT+R+YzBhemFVc+IGnqK6Jez3tVSXq
  DBQqdZHcr6qoImqHq3IjhX4zk+TexM4azjConDXDgxa4pruTnrhv3hFwFWQGMMyKFwFfX
  KtZqe9sXhPnSSWOf6mBzypzUnUTO7HMPb5FAdNFyIv9mHWHG6f9xB031S0XCBt2Mptir
  LmVvAcz3XcBwg4YvY7QwM3kOC5iVlFVahTzmeMDajTJE4JLwU24OcpxDH0t7sOS+lprl
  A+U/fXrdq3Vwajqqdo/vIW0CU4UAre69KU8u8bPCPCwOfv/wPbDYXM2+XBrNbk4Bkpno
  5w8A==
MIME-Version: 1.0
X-Received: by 10.182.66.164 with SMTP id g4mr887457obt.47.1384845901691;
  Mon,
  18 Nov 2013 23:25:01 -0800 (PST)
Received: by 10.182.59.70 with HTTP; Mon, 18 Nov 2013 23:25:01 -0800 (PST)
```



What does this
mean?
It's in Base64..

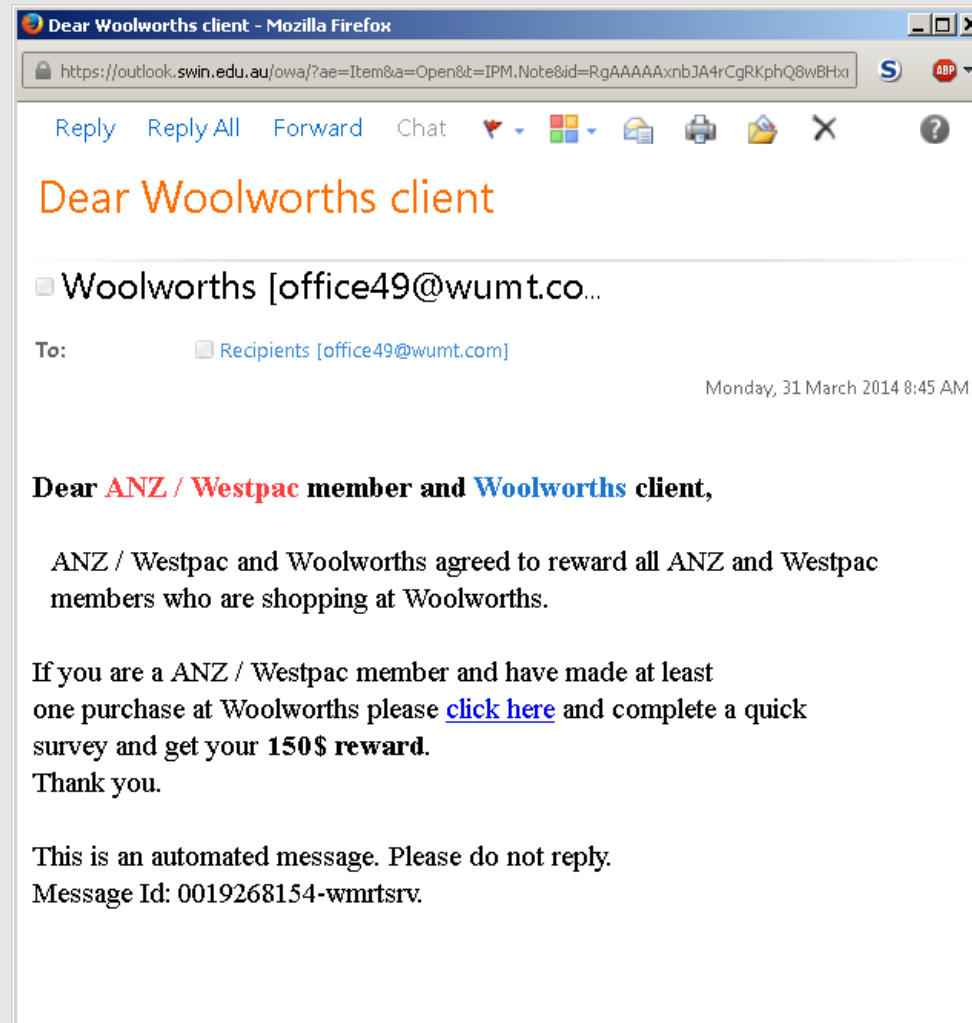
E-mail Spam Checker Report

Reply-To: <b2013.2014.bfwcoc@gmail.com>
Date: Tue, 19 Nov 2013 09:25:01 +0200
Message-ID: <CAG9WAb2v1ch7hburHTPft8hEfZgKHYhydml06mwFEQgkXaoT4A@mail.gmail.com>
Subject: BRAZIL 2014 WORLD CUP PROMOTION
From: BRAZIL 2014 <f.c.m.s.c.c.g@gmail.com>
To: undisclosed-recipients:;
Content-Type: multipart/mixed; boundary="089e0160c35e0a59d704eb8290ec"
X-Spam-Status: score=5.3
 tests=RCVD_IN_DNSWL_LOW,FREEMAIL_FROM,SUBJ_ALL_CAPS,HTML_MESSAGE,DKIM_VALID_AU,DKIM_SIGNED,,LOTTO_AGENT,
 FREEMAIL_REPLYTO
X-Spam-Level: *****
X-Spam-Report: * -0.7 RCVD_IN_DNSWL_LOW RBL: Sender listed at <http://www.dnswl.org/>, low
 * trust
 * [209.85.219.52 listed in list.dnswl.org]
 * 0.0 FREEMAIL_FROM Sender email is commonly abused enduser mail provider
 * (f.c.m.s.c.c.g[at]gmail.com)
 * 1.6 SUBJ_ALL_CAPS Subject is all capitals
 * 0.0 HTML_MESSAGE BODY: HTML included in message
 * -0.1 DKIM_VALID_AU Message has a valid DKIM or DK signature from author's
 * domain
 * 0.1 DKIM_SIGNED Message has a DKIM or DK signature, not necessarily valid
 * -0.1 DKIM_VALID Message has at least one valid DKIM or DK signature
 * 3.5 LOTTO_AGENT Claims Agent
 * 1.0 FREEMAIL_REPLYTO Reply-To/From or Reply-To/body contain different
 * freemails
 *

Remaining Headers

```
Return-Path: f.c.m.s.c.c.g@gmail.com
X-MS-Exchange-Organization-AuthSource: gsp-ex03.ds.swin.edu.au
X-MS-Exchange-Organization-AuthAs: Anonymous
X-MS-Exchange-Organization-PRD: gmail.com
X-MS-Exchange-Organization-SenderIdResult: SoftFail
Received-SPF: SoftFail (gsp-ex03.ds.swin.edu.au: domain of transitioning
  f.c.m.s.c.c.g@gmail.com discourages use of 136.186.1.33 as permitted sender)
X-MS-Exchange-Organization-AVStamp-Mailbox: MSFTFF;1;0;0 0 0
```


Another Example



E-mail Header

Received: from ENP-EX02.ds.swin.edu.au (136.186.126.148) by
gsp-ex03.ds.swin.edu.au (136.186.126.19) with Microsoft SMTP Server (TLS) id
14.3.158.1; Mon, 31 Mar 2014 08:53:23 +1100

Received: from ip1-in.cc.swin.edu.au (136.186.0.41) by outlook.swin.edu.au
(136.186.126.148) with Microsoft SMTP Server id 14.3.158.1; Mon, 31 Mar 2014
08:53:23 +1100

X-IronPort-Anti-Spam-Filtered: true

X-IronPort-Anti-Spam-Result:
AjBpAAGSOF0IugEgnGdsb2JhbABZgWwCAVN/SwEBqzECgSYBhUKIDYEmGYhEFg4BAQEBAQgUCTyCRIEAARw0Tog
LAQ2fVolrjRxRoQQXkTQPgXsEiRo2hgulc4EzhRqPJYFe

X-IronPort-AV: E=Sophos;i="4.97,761,1389704400";
d="scan'208,217";a="87288458"

Received: from gpo3.cc.swin.edu.au ([136.186.1.32]) by ip1-in.cc.swin.edu.au
with ESMTP; 31 Mar 2014 08:53:23 +1100

Received: from smtp42.singnet.com.sg (smtp42.singnet.com.sg [165.21.103.146])
by gpo3.cc.swin.edu.au (8.14.3/8.14.3) with ESMTP id s2ULr7Hm012561; Mon, 31
Mar 2014 08:53:21 +1100

Received: from [192.100.100.2] ([203.125.107.86]) by smtp42.singnet.com.sg //the source
(8.14.3/8.14.1) with ESMTP id s2ULpbv4021067; Mon, 31 Mar 2014 05:51:58 +0800

Message-ID: <201403302151.s2ULpbv4021067@smtp42.singnet.com.sg>

Content-Type: multipart/alternative; boundary="====1720182961=="

MIME-Version: 1.0

E-mail Header

office49@wumt.com //fake

From: Woolworths <office49@wumt.com>

Date: Mon, 31 Mar 2014 05:45:19 +0800

X-PMX-Version: 5.5.2.363555, Antispam-Engine: 2.6.1.350677, Antispam-Data: 2014.3.30.214218

X-PMX-AS: AS-Check

X-PMX-Score: Probability=10%

X-Spam-Status: score=2.5 tests=RCVD_IN_DNSWL_NONE,URIBL_BLACK,HTML_MESSAGE //blacklist

X-Spam-Level: **

X-Spam-Report: * -0.0 RCVD_IN_DNSWL_NONE RBL: Sender listed at <http://www.dnswl.org/>, no

- * trust
- * [165.21.103.146 listed in list.dnswl.org]
- * 2.5 URIBL_BLACK Contains an URL listed in the URIBL blacklist
- * [URIs: sungazette.com]
- * 0.0 HTML_MESSAGE BODY: HTML included in message
- *

Return-Path: office49@wumt.com

X-MS-Exchange-Organization-PRD: wumt.com

X-MS-Exchange-Organization-SenderIdResult: None

Received-SPF: None (ENP-EX02.ds.swin.edu.au: office49@wumt.com does not designate permitted sender hosts)

X-MS-Exchange-Organization-AVStamp-Mailbox: MSFTFF;1;0;0 0 0

X-MS-Exchange-Organization-AuthSource: ENP-EX02.ds.swin.edu.au

X-MS-Exchange-Organization-AuthAs: Anonymous

Tracing Spam E-mails

- The link inside take us to:
 - <http://extras.sungazette.com/wool.html>
(Now broken)
- It's the Sun Gazette -- a local Williamstown newspaper
- It's US hosting

12.169.112.230

Lookup IP Address

General IP Information

IP: 12.169.112.230

Decimal: 212431078

Hostname: sungazette.com

ISP: AT&T Services

Organization: Ogden Newspapers

Services: None detected

Type: [Corporate](#)

Assignment: [Static IP](#)

Blacklist: [Blacklist Check](#)

Geolocation Information

Country: United States 

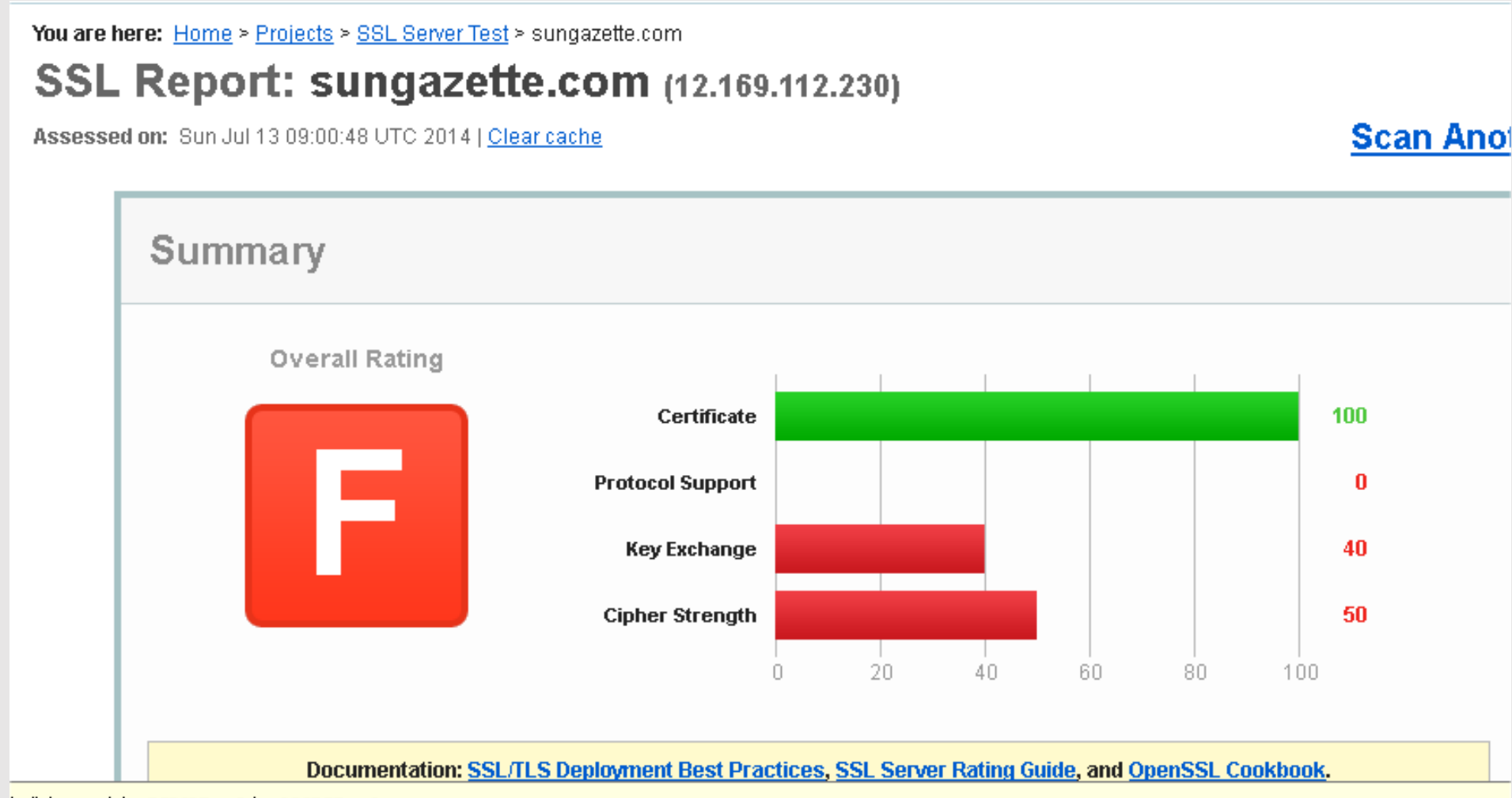
State/Region: West Virginia

City: Wheeling

Latitude: 40.0582 (40° 3' 29.52" N)

Tracing Spam E-mails

- Vulnerable to being hacked?



Tracing Spam E-mails

- Vulnerable to being hacked?

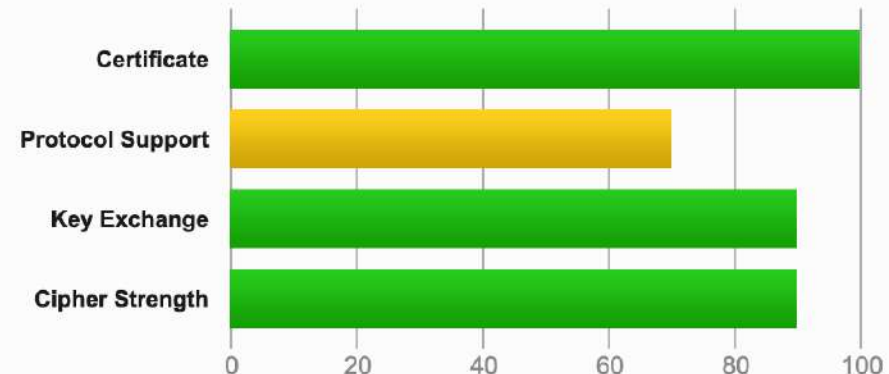
You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > [sungazette.com](#) > 35.170.139.128

SSL Report: [sungazette.com](#) (35.170.139.128)

Assessed on: Wed, 07 Oct 2020 05:39:51 UTC | [Hide](#) | [Clear cache](#)

Summary

Overall Rating



Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

Blacklists

- System Administrators have to option of subscribing to various **blacklists**.
 - Lists of domain names and hosts identified as sources of spam e-mail.
 - <http://www.rahul.net/falk/#blacklists>
 - <http://www.spamhaus.org/lookup.lasso>
 - Some sites are blacklisted by mistake.
 - There are whitelist services available as well for default-deny e-mail servers.

Open Relays

- There are a few **open relays** – public and anonymous e-mail servers which allow anyone to send an e-mail.
 - They are a leftover from the days when the internet was used for good and not evil
 - Highly sought-after by spammers
 - A compromised or owned PC can act as an open relay.
 - <http://tools.rosinstrument.com/proxy/>
 - http://multiproxy.org/all_proxy.htm

Free E-mail Services

- Rather than using **open relays**, spammers tend to set up an e-mail server on an owned PC. (never use their own e-mail account)
- Spammers also use public web-based e-mail sites to send spam. Bots can be used to create new e-mail accounts with random or dictionary-based names for the purposes of sending spam.
- Such services are increasingly trying to prevent this by adding puzzles that bots can't solve:
 - CAPTCHA puzzles
 - Phone call-backs
 - Audio CAPTCHAs

CAPTCHA Puzzles

- **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
 - Require the person opening the account to interpret a scrambled image or sound. The theory is that a bot is not smart enough to solve the puzzle, but the algorithms for this already exist
 - <http://www.cs.sfu.ca/~mori/research/gimpy/>
- Can be avoided if the bot tricks a real user on another site to decode the puzzle on his behalf (Security Now 101):
 - Some CAPTCHA puzzles have been implemented on the client-side using Javascript (epic fail!)

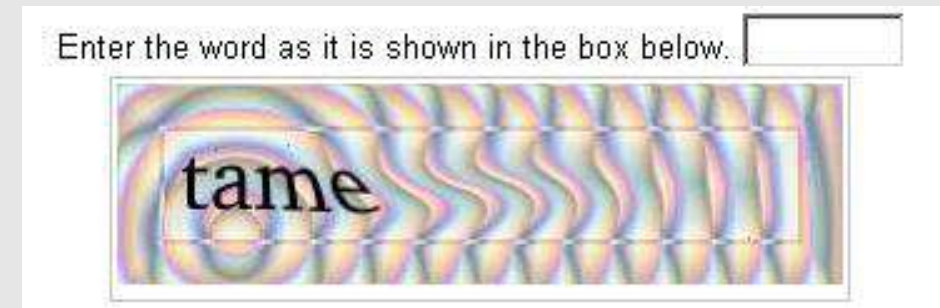


Fig 1. Picture of a CAPTCHA in use at Yahoo

E-mail Harvesting

- Collecting e-mail addresses is part of the enumeration process.
 - E-mail addresses reveal usernames and domain names.
- Names can be lifted by automatic tools (spiders) which sift through web sites on the web.
 - Companies doing this represent themselves as legitimate companies providing a service or a "web directory" product
 - Include directories of coffee shops, health care and education providers
- E-mails are sent to the harvested e-mail addresses inviting the recipients to visit the web site and confirm the details. Confirmed e-mail addresses are worth more on the hacker/spam market.

E-mail Harvesting

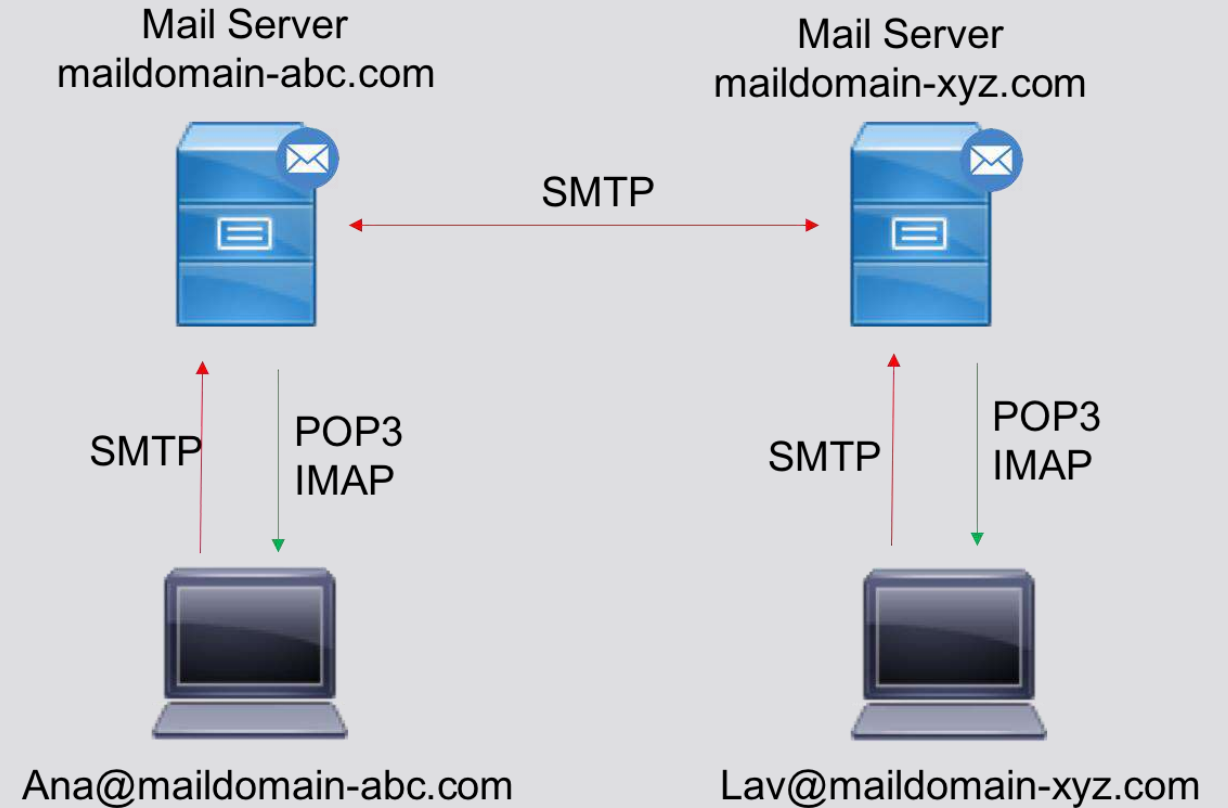
- Spam-bots also generate random e-mail addresses and send spam to them.
 - The messages include an **unsubscribe link** which if followed logs the e-mail address of the victim, confirming it as real and therefore saleable.
- The messages often say something like "**don't reply to this e-mail address**".
 - The "**from**" e-mail address does not exist – it was spoofed or has been shut down by the ISP managing the bot's domain

E-mail Security

E-mail Security

Three main protocols, clients and mail servers:

- SMTP
- POP3
- IMAP



Spam Filtering

- Keyword matching.
 - Check through blacklist of words.
 - Easily bypasses by spammer adding spaces, punctuation, substitute letters
- Bayesian Filtering.
 - Uses machine learning to distinguish between Spam and normal e-mail
 - Needs to be “trained”
- ALPACAS: A Large-scale, Privacy-Aware Collaborative Antispam System.
 - Identifies “fingerprints” of spam e-mail based on style, layout
 - Changes on content, obfuscation don’t trick it

Pretty Good Privacy

- E-mail messages are:
 - Digitally signed
 - Encrypted
 - Hashed
 - Uses Web of Trust (instead of CA) to verify public keys
 - Based on reputation of public keys
 - Open source version is GPG (Gnu Privacy Guard)

E-mail Authentication

- Authentication of sending user (client) relies on public key crypto:
 - Everyone must have a certificate
 - Not used much
- Authentication of the organization:
 - Uses certificate embedded in gateway (e.g. Astaro appliance)
 - Easier to use, so more common

Sender Policy Framework

- SPF field in DNS record used to authenticate e-mail server.
- Easy to spoof.
- Does not check message integrity.
- No privacy (encryption).
- Does not support mail forwarding.
- Some adoption, but not commonplace

Structured Query Language (SQL)

- A non-standard (RDBMS-specific) set of commands for creating database tables and accessing them.
- DDL: Data Definition Language.
- DML: Data Manipulation Language.

CREATE TABLE
SELECT
SQL
INSERT INTO
UPDATE

Database Table

- The **top row** are the column names.
- Each **subsequent row** contains data relating to one thing (in this case, a person).
- Each row should be different, although the same value can appear in different rows of the same column.
- Each row can be uniquely identified by a **Primary key** (a column where no value is repeated).

Num	Name	Inaugural Age	Age at Death
1	George Washington	57.2	67.8
2	John Adams	61.3	90.7
3	Thomas Jefferson	57.9	83.2
4	James Madison	58.0	85.3
5	James Monroe	58.8	73.2
6	John Quincy Adams	57.6	80.6
7	Andrew Jackson	62.0	78.2
⋮	⋮	⋮	⋮

Fig 1. A relational database table,
Presidents

DDL

- The DDL to create this table in a database looks like this:

```
CREATE TABLE Presidents (  
  Num Int PRIMARY KEY,  
  Name VARCHAR(10) ,  
  Inaugural_Age DOUBLE ,  
  Age_at_Death DOUBLE  
);
```

More DDL

- Delete the table (completely):
 - `DROP TABLE Presidents`
- Change the columns or other details of the table:
 - `ALTER TABLE Presidents ADD Email VARCHAR(255) ;`
`ALTER TABLE Presidents DROP COLUMN Email;`

DML

- These commands add data to tables, delete it, modify it or get data from the table :
 - `SELECT Name FROM Presidents`
- Gets the contents of the name column.
 - `SELECT * FROM Presidents;`
- Gets all of the columns.

More DML

```
INSERT INTO Presidents (Num, Name)  
VALUE (8, Jimmy Carter);
```

- Adds a new row to the table, with an 8 in the **Num** column and *Jimmy Carter* in the **Name** column.
- Note that we can leave some cells (intersection of row and column) empty.

DML --- WHERE

- The **WHERE** command limits the action of the rest of the command to specific rows.
- Does a logical test on each row and if the logic returns TRUE, performs the action.
 - `SELECT * FROM Presidents
WHERE Name = 'Jimmy' ;`
- Delete all the rows where Age_at_Death is larger than 90.
 - `SELECT * FROM Presidents
WHERE Age_at_Death > 90 ;`

DML --- WHERE Logic

- We can use boolean logic to combine **WHERE** conditions.
- To remove all of the rows containing the age at death around 80 to 90.
 - `SELECT * FROM Presidents
WHERE Age_at_Death > 80 AND Age_at_Death < 90;`
- To remove only the row containing the age at death larger than 90 with the inaugural age smaller than 60.
 - `SELECT * FROM Presidents
WHERE Age_at_Death > 90 AND Inaugural_Age < 60;`

SQL Injection and Prevention

- SQL injection covers a range of database attacks from the injection of exploit code through a buffer overflow in a DBMS to execution of arbitrary SQL script through a web page form.
 - Example: <http://www.unixwiz.net/techtips/sql-injection.html>
 - Documentation: <http://msdn.microsoft.com/en-us/library/ms161953.aspx>

SQL Injection and Prevention

- Problem occurs because lazy programmers pass un-sanitized user-input directly into SQL command strings.
- Can be prevented by:
 - Sanitizing:
 - using **Trim()** and **Replace()** (asp) to remove escaping and long strings
 - sanitising with regex (php: **preg-replace()**, asp: **rewrite**)
 - remove or escape these characters: ' ; - " () = / #
 - Passing parameters to DBMS
 - Using stored procedures on the server
 - **mysql_real_escape_string()**, **addslashes()**, **urlencode()**

SQL Parameters

- **Wrong:**

```
Dim SQL As String = "SELECT Count(*) FROM Users  
                    WHERE UserName = '" & username.text & "'  
                    AND Password = '" & password.text & "'"
```

```
Dim thisCommand As SqlCommand = New SqlCommand(SQL, Connection)
```

```
Dim thisCount As Integer = thisCommand.ExecuteScalar()
```

- **Right:**

```
Dim thisCommand As SqlCommand = New SqlCommand("SELECT Count(*) " &  
        "FROM Users WHERE UserName = @username AND Password = @password",  
        Connection)
```

```
thisCommand.Parameters.Add ("@username", SqlDbType.VarChar).Value =  
username
```

```
thisCommand.Parameters.Add ("@password", SqlDbType.VarChar).Value =  
password
```

```
Dim thisCount As Integer = thisCommand.ExecuteScalar()
```

SQL Stored Procedures

- **Code running on the DBMS**

```
Dim thisCommand As SqlCommand = New SqlCommand  
("proc_CheckLogon", Connection)
```

```
thisCommand.CommandType = CommandType.StoredProcedure  
thisCommand.Parameters.Add ("@username", SqlDbType.VarChar).Value =  
username  
thisCommand.Parameters.Add ("@password", SqlDbType.VarChar).Value =  
password  
thisCommand.Parameters.Add ("@return", SqlDbType.Int).Direction =  
ParameterDirection.ReturnValue  
Dim thisCount As Integer = thisCommand.ExecuteScalar()
```

Ref: SQL Injection Prevention Cheat Sheet

https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

SQL Logic Attack

- Inject <something> OR <TRUE>
- x' or $'x' = 'x$
- $1 \text{ OR } 1=1 \setminus^*$ //MySQL (comments: out the rest of the SQL/php)
- $A' \text{ OR } 2=2 ; --$ //Other DBMSs
- Solution: Sanitize, filter, restrict privileges

SQL Logic Attack

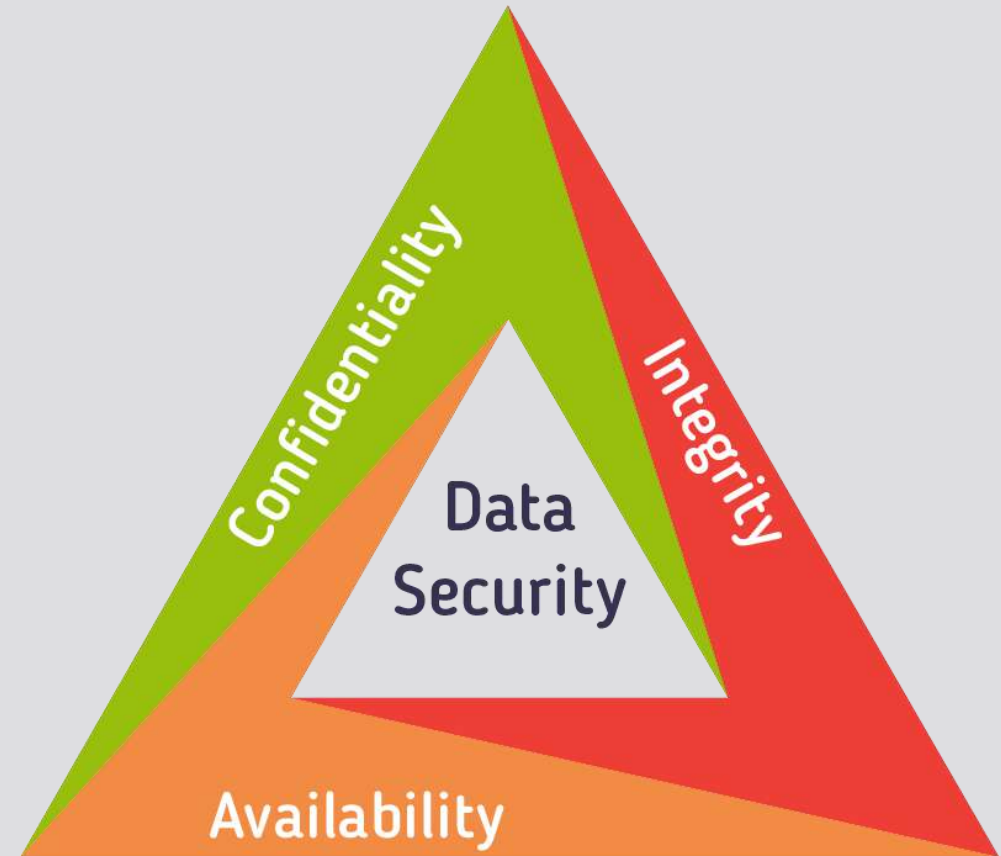
- UNION
- Concatenates two DML queries
- As long as the number of columns returned is the same.
- MySQL only
- Solution: Sanitize, filter, restrict privileges

Database CIA:

- Confidentiality
- Integrity
- Availability

Databases have several systems in place to:

- maintain privacy
- prevent data corruption and
- ensure availability



Two-Phrase Commit

- Allows simultaneous write access to a database without risk of data corruption.
- Request phase
 - Upload proposed changes to database
 - DBMS locks needed records. If it can't lock them, it changes nothing and aborts the transaction.
- Commit phase
 - Changes all the records it has locked and returns a success code. If anything goes wrong, it reverses and changes it has made (returning the database to the state it had before commit was called)

DB Access Control

- DBMSs use ACLs or permission attributes to control who reads from and writes to the database, tables, columns.
- Implements DAC, MAC (users may not be given permission to change permissions for other users).
- Should be set up according to the principle of least privilege.

Granting Permission

- GRANT SELECT ON `users` TO `EvilHacker`
- Can also grant DELETE, INSERT, UPDATE
- Can grant permissions to ALL, PUBLIC

DAC Permission

- DAC can be implemented by creating a user-specific view and granting GRANT rights to it's "owner"
- CREATE VIEW `user_alice`
AS SELECT * from `users` where `name='Alice'`;
GRANT SELECT ON `user_alice` to `Alice` with GRANT OPTION;

Removing Permission

REVOKE SELECT ON `users` FROM `Alice`;

- If a user gets demoted or leaves the organisation, permissions to the database must be removed before they can retaliate:



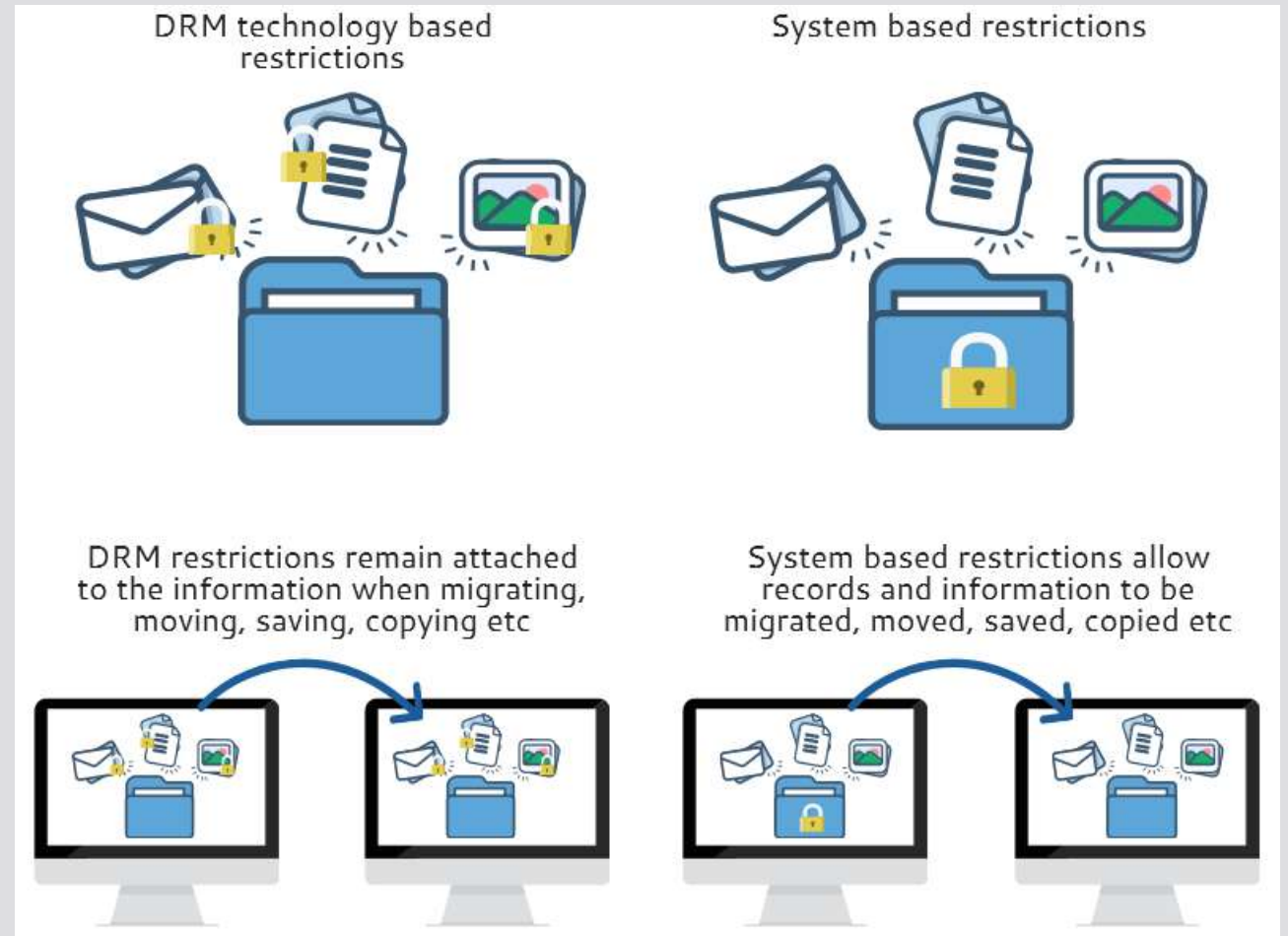
- This has a cascade effect of removing permissions from everyone who was approved by Alice

Database Encryption

- Sensitive data should be encrypted before storing it in the database.
- Some databases store data in plain text or human-readable form, so an attacker with hard-drive access can read data without using the DBMS
- DBMSs offer a range of encryption and decryption functions ranging from symmetric keys to public-private crypto.

Digital Rights Management (DRM)

- Schemes for protecting the rights of copyright holders, recording companies and media companies.
- Restricts rights to sell, copy, give, backup, transfer and broadcast.
- Applies to software, audio, video, print (e-books), transmissions (pay-TV).
- Restricts “fair use” (study, review).



DRM Schemes

- CDs

- 2002 – Copy protection attempted – quickly defeated
- 2005 – Sony put root-kit on CDs to prevent copying. Replaced audio with noise, disabled CD drive if removed. Vuln. in root kit exploited.

- DVDs

- Use Content Scrambling System
- Encrypts channel between player and display
- Reverse engineered (key kept on disk)

DRM Schemes

- HD-DVDs
 - CSS-like protection cracked soon after the format came into use.
- Blue-Ray
 - AACS
 - AES block cipher
 - Many ways to crack this
- Physical media protection schemes becoming irrelevant

Fuzzing



Fuzzing

Software testing technique

- implementation bugs
- automated

Invalid or random data called FUZZ

University of Wisconsin in 1989

Uncover security flaws

Cost effective

Used by both good and bad

Fuzzing Cont.

Different types for different use cases

- Mutation: Alter data to create new test data
- Generation: Define input parameters to generate input
- Protocol: Define protocol and modify requests

Fuzzer Types

Application

- UI: button sequence, inputs
- Command line
- Import/Export

File

- Parser
- Application layer

Protocol

- Sequence

Attack Vectors

Attacks will centre around

- Numbers
- Chars
- Metadata
- Binary sequence

Integers: zeros, negative numbers, large numbers

Chars: escaped, various formats

Binary: random sequence

Considerations for Fuzzing

Less effective for threats which might not result in a crash

Simple, not as in depth

Overhead to set up

Is it deterministic, how do you determine boundaries?

Cost/time to run, concurrent testing?

Serious Fuzzing

The industry workhorse is AFL – the American Fuzzy Lop

- <https://github.com/google/AFL>
- “...a brute-force fuzzer coupled with an exceedingly simple *but rock-solid* instrumentation-guided genetic algorithm. It uses a modified form of edge coverage to effortlessly pick up subtle, local-scale changes to program control flow.”

AFL process

1. Load user-supplied initial test cases into the queue,
2. Take next input file from the queue,
3. Attempt to trim the test case to the smallest size that doesn't alter the measured behavior of the program,
4. Repeatedly *mutate* the file using a balanced and well-researched variety of traditional fuzzing strategies,
5. If any of the generated mutations resulted in a new state transition recorded by the instrumentation, add mutated output as a new entry in the queue.
6. Go to 2.

Fuzzing is Brute Force

Coverage

The range of and combination of inputs fed into a process. Should test all code paths in the minimum time. Each combination of inputs makes up a **test case**.

- Much research on optimizing this:

<https://researchbank.swinburne.edu.au/items/665ae947-cd8c-4261-b108-7b0a97b8ed94/1/>

Instrumentation

Fuzzers have to record the inputs and the resultant outputs and detect if a crash or anomaly has occurred.

Fuzzing advantages

1. Provides results with little effort - once a fuzzer is running, it can be left for hours, days, or months to look for bugs with no interaction.
2. Can reveal bugs missed in a manual audit.
3. Provides an overall picture of the robustness of the target software.
4. “Fuzz while you sleep”

adapted from <https://www.f-secure.com/en/consulting/our-thinking/15-minute-guide-to-fuzzing>

Fuzzing disadvantages

1. Won't find all bugs - may miss bugs that don't trigger a program crash, and may miss bugs that require sequences of state changes.
2. Interesting test cases may be difficult to analyze. Fuzzing doesn't expose how the software operates internally.
3. Programs with complex inputs can require massive numbers of test cases.

adapted from <https://www.f-secure.com/en/consulting/our-thinking/15-minute-guide-to-fuzzing>