# ColdFusion Security and Risk Management

Justin Mclean
Email: justin@classsoftware.com
Twitter: @justinmclean
Blog: http://blog.classsoftware.com

# Who am I?

- Director of Class Software for 10 years
- Developing and creating web applications for 15 years
- Programming for 25 years
- Adobe Community Professional
- Adobe certified developer and trainer in ColdFusion and Flex
- Based in Sydney Australia

# Security

- No system is 100% secure
- Security takes time and effort and can be costly
- How much security is actually needed?
- Is a security feature actually effective?

# Risk Assessment

Risk assessment is a tool used to balance business objectives and security requirements in order to achieve cost effective security measures.

# Process

1. Identify assets
2. Identify and quantify the possible threats
3. Determine the consequence of each threat
4. Evaluate the current risk
5. Decide acceptable level of risk
6. Treat each risk

# Frequency

- This is not a once off process
- Repeat when:
  - System is in place
  - Changes are made to the system or assets
  - Made aware of new threats

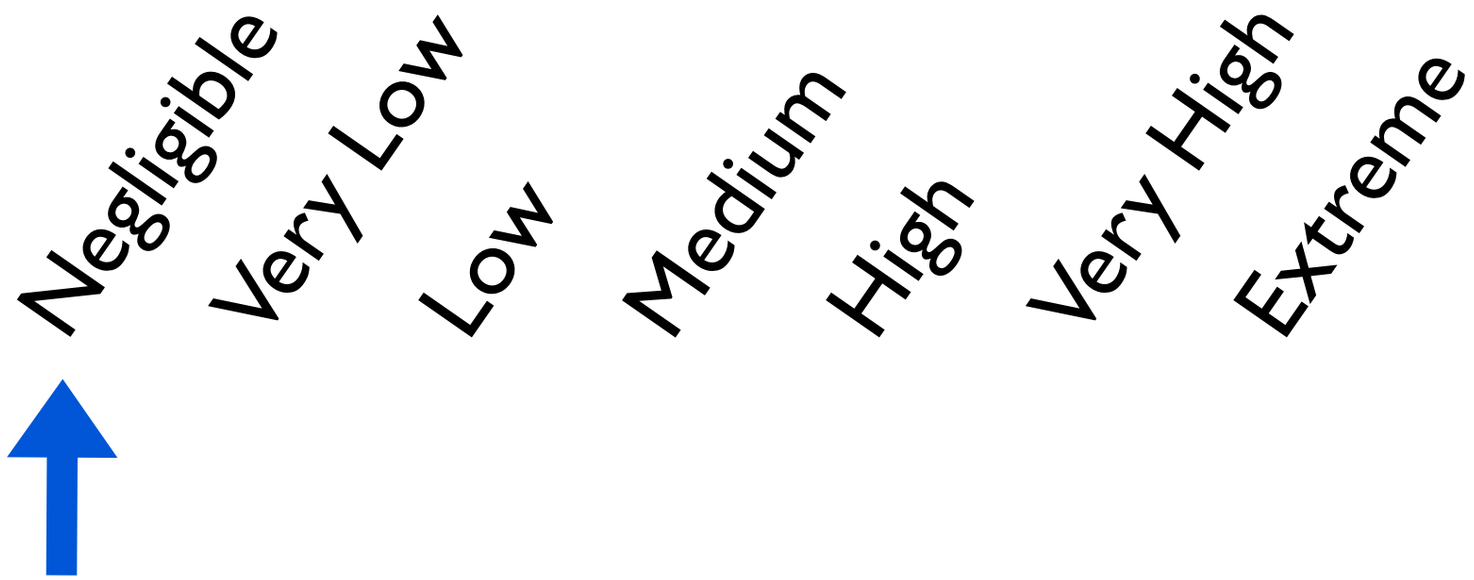
# Identify Assets

- Create list of assets
  - Hardware
  - Availability of service
  - Integrity of information
  - Reputation of system and/or organisation
  - Staff

# Identify Threats

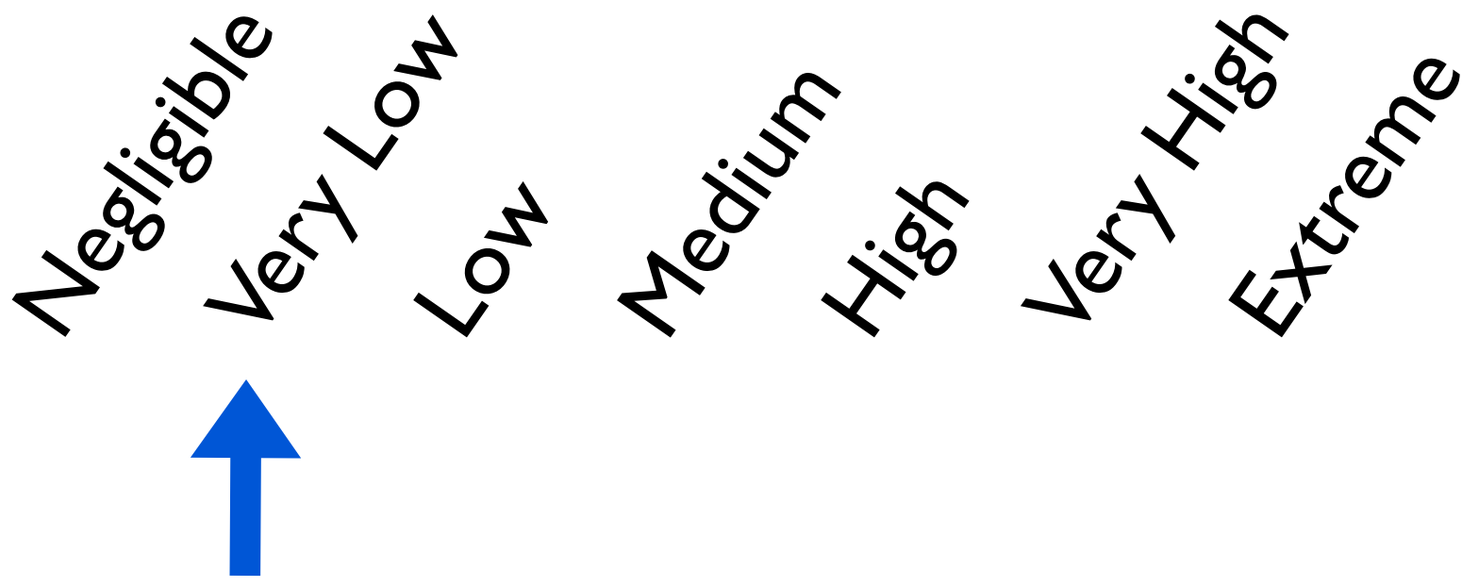
- Create a list of threats
- Be creative include unlikely threats
- Tendency to ignore obvious threats
- Careful of preconceived attitudes

# How likely is each threat?



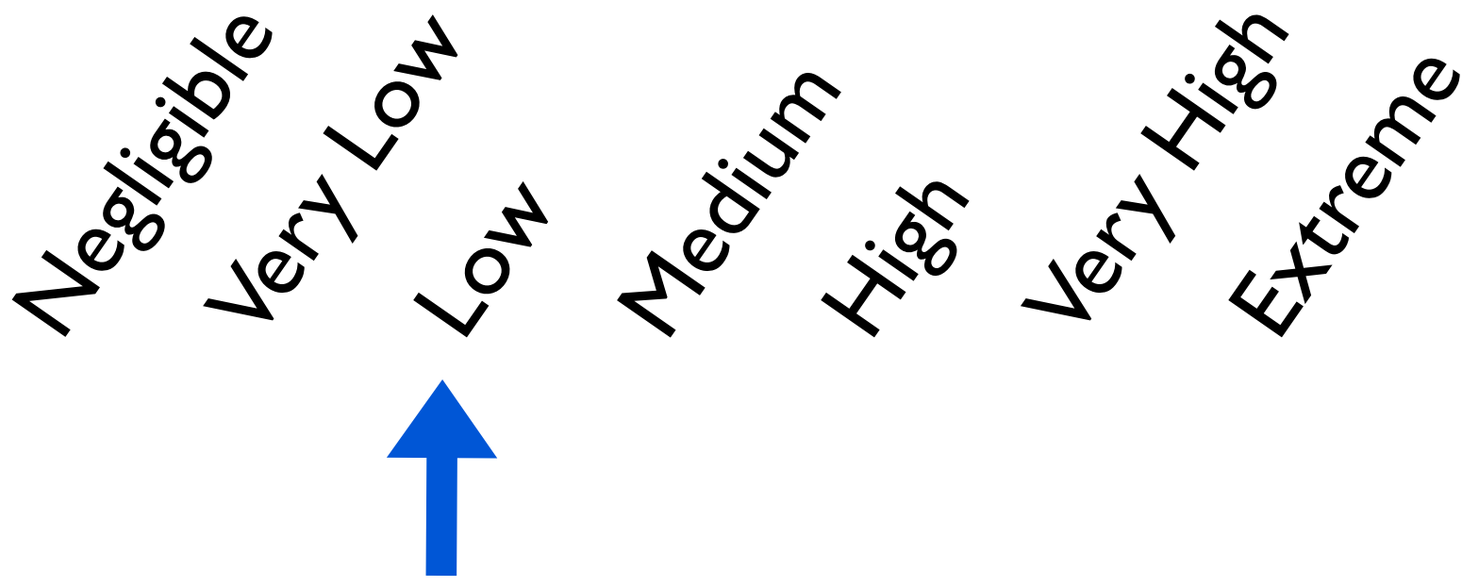

Unlikely to occur

# How likely is each threat?



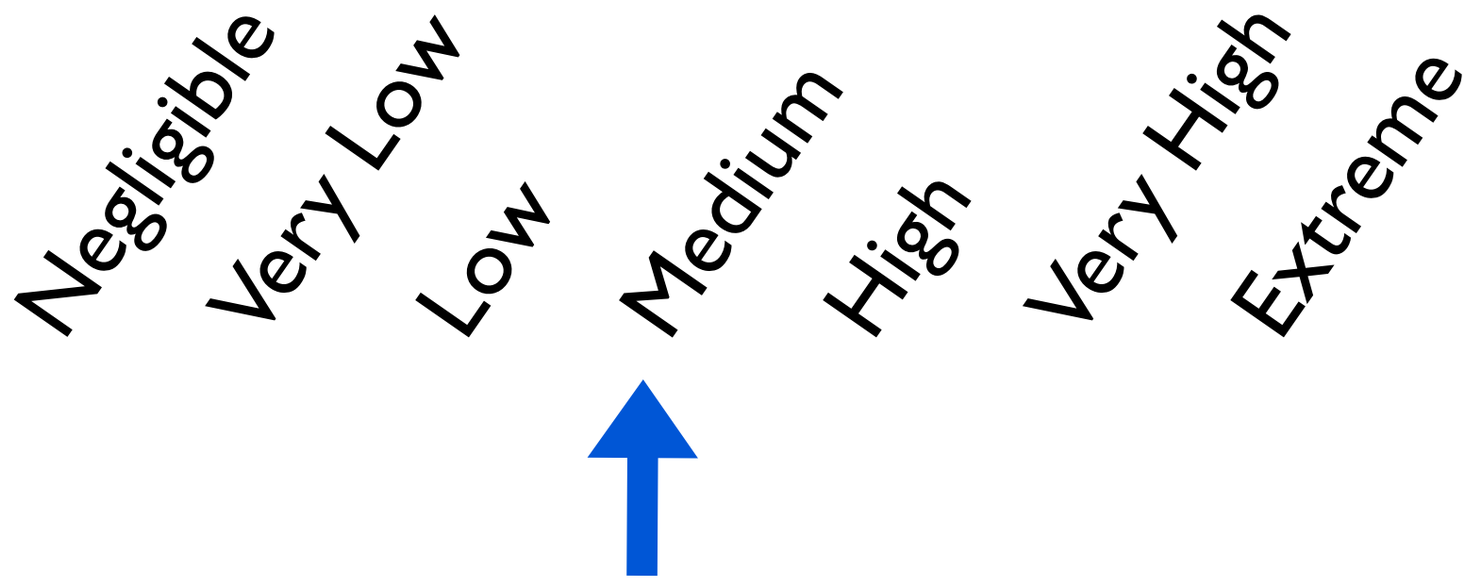

Occurs a couple of times in 5 years

# How likely is each threat?



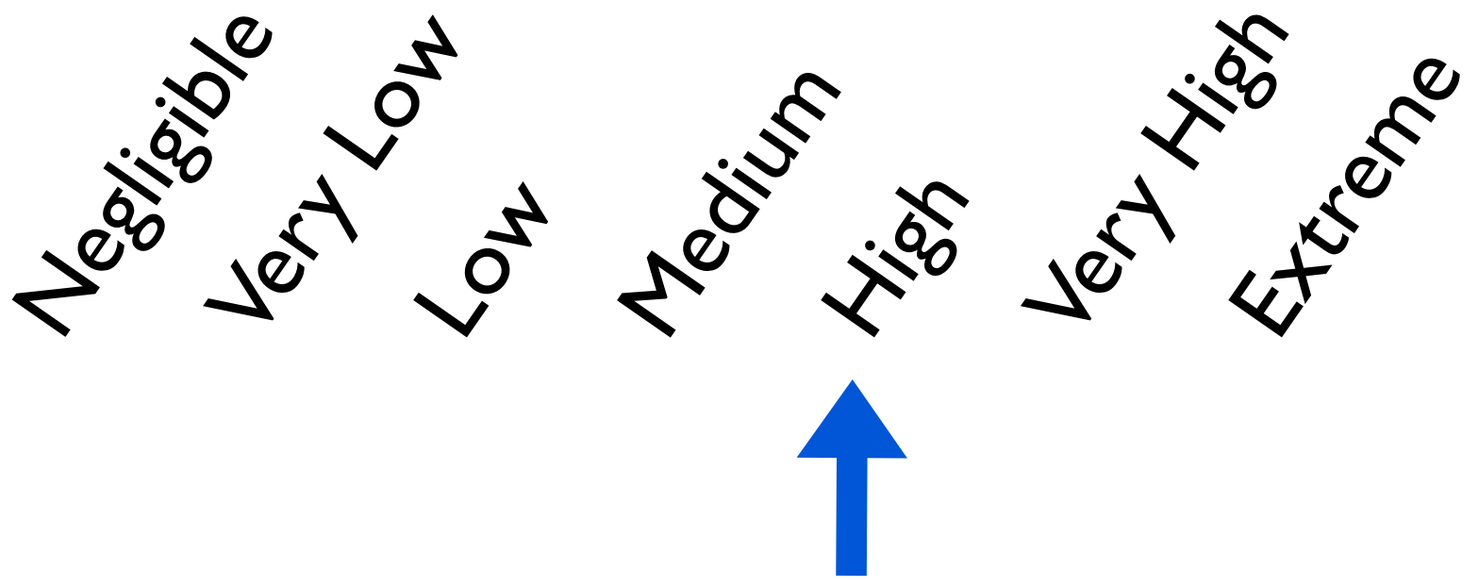

Occurs once every year

# How likely is each threat?




Occurs every six months

# How likely is each threat?




Occurs every month

# How likely is each threat?

Negligible
Very Low
Low
Medium
High
Very High
Extreme

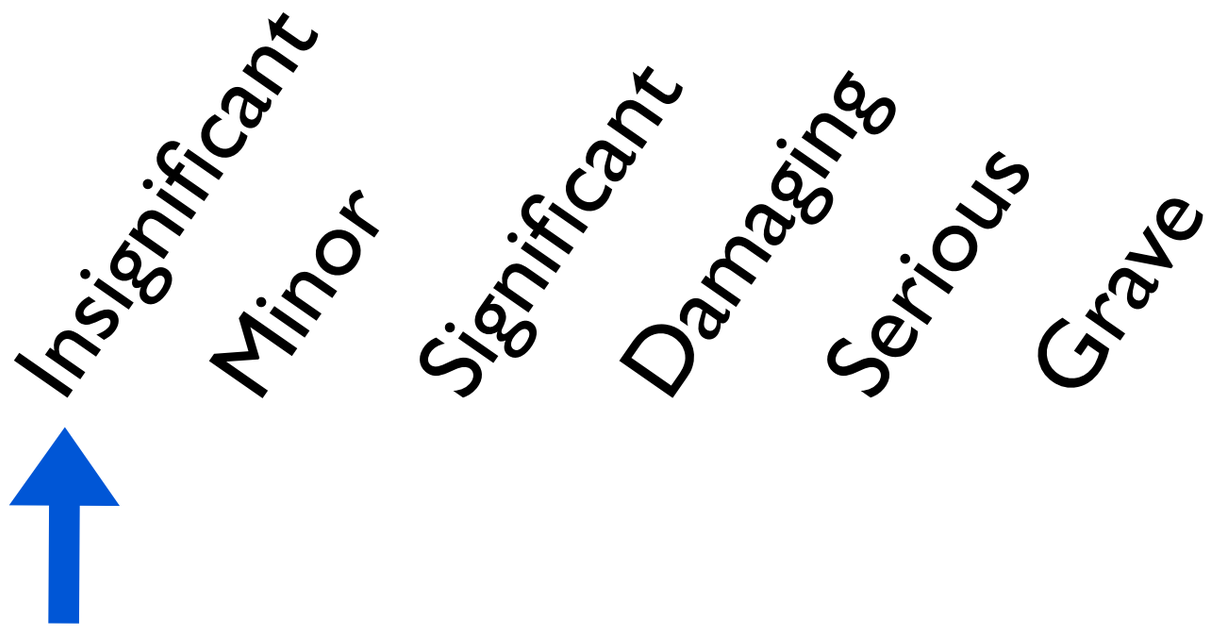
Occurs multiple times a month

# How likely is each threat?

Negligible
Very Low
Low
Medium
High
Very High
Extreme

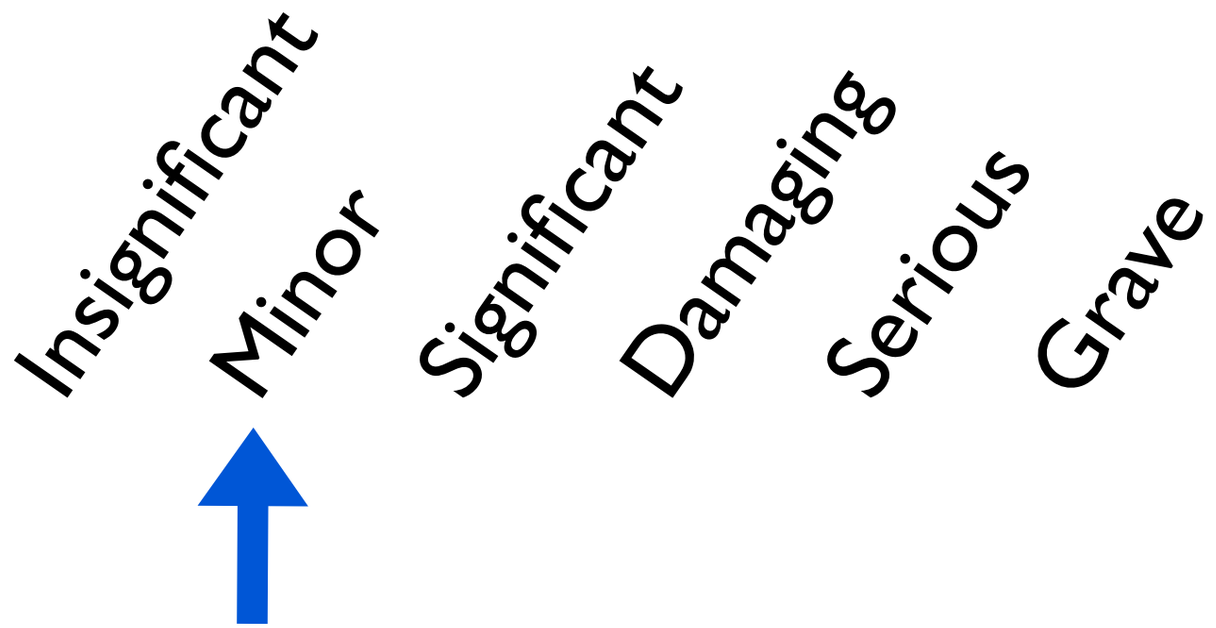
Occurs multiple times a day

# Consequence of each threat?



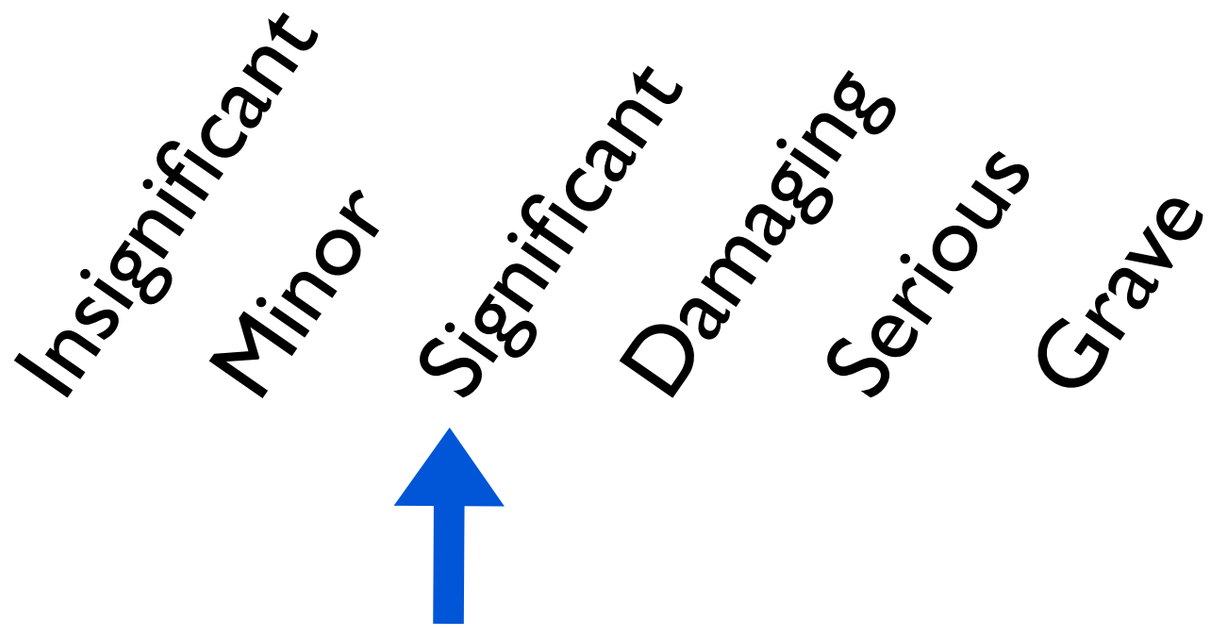

No extra effort to repair

# Consequence of each threat?



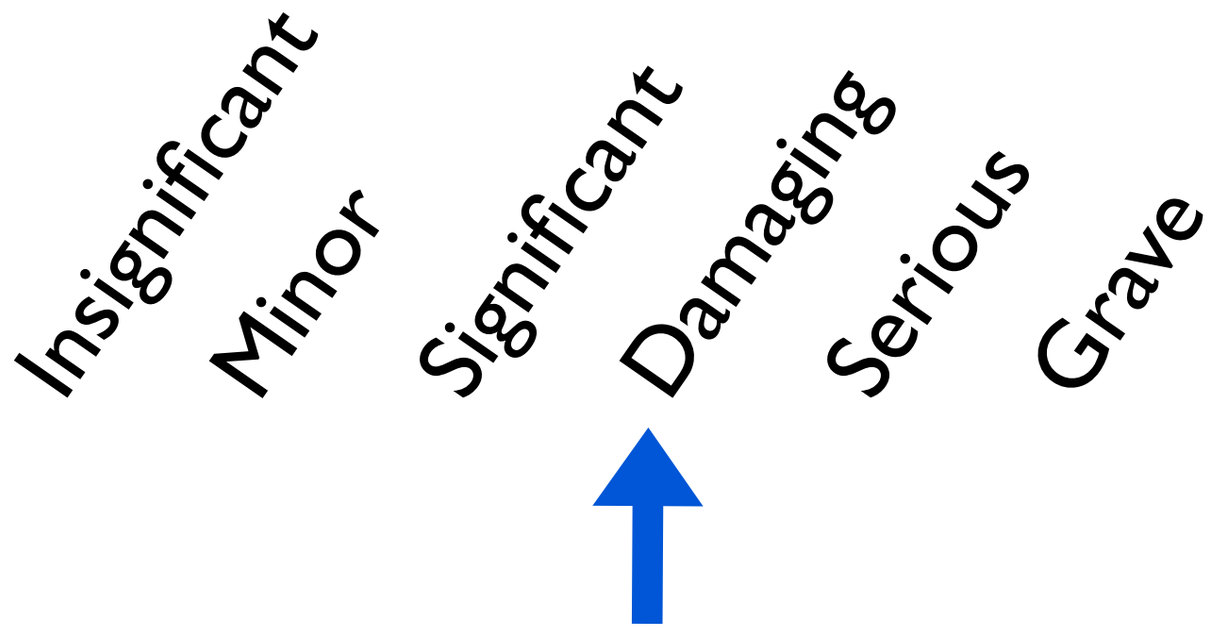

Few people notice. Small effort to repair

# Consequence of each threat?



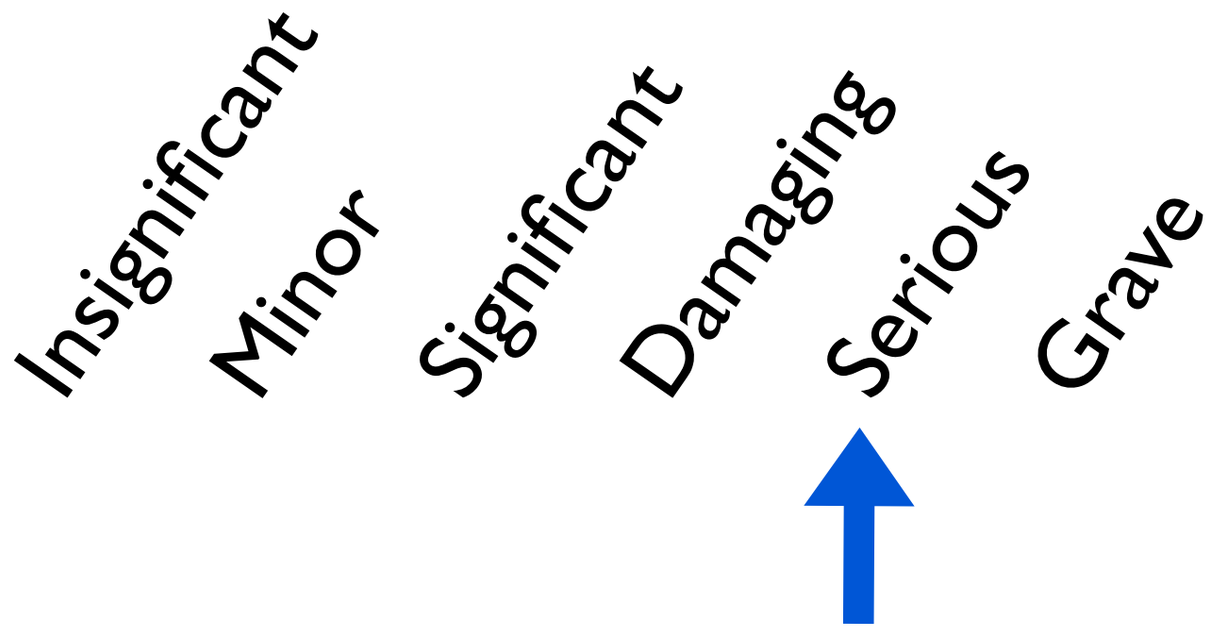

Some harm. Some effort to repair.

# Consequence of each threat?



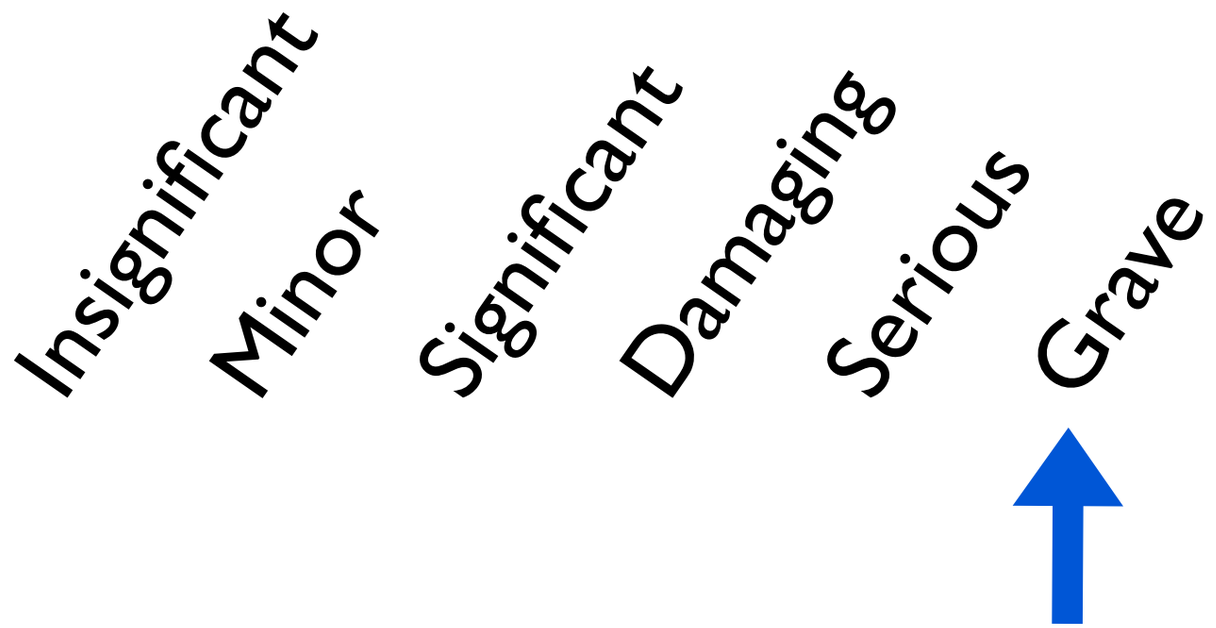

Serious. Extend system outage or loss of customers.

# Consequence of each threat?




Loss of confidence/reputation. Major effort to repair.

# Consequence of each threat?




Completely compromised. System permanently closed or offline.

# Risk

- Risk is a combination of frequency and consequence
- The more likely a threat increases risk
- The more serious a threat increases risk

# Risk

## Consequence

| Threat | Insignificant | Minor | Significant | Damaging | Serious | Grave |
|---|---|---|---|---|---|---|
| **Negligible** | Nil | Nil | Nil | Nil | Nil | Nil |
| **Very Low** | Nil | Low | Low | Low | Medium | Medium |
| **Low** | Nil | Low | Medium | Medium | High | High |
| **Medium** | Nil | Low | Medium | High | High | Critical |
| **High** | Nil | Medium | High | High | Critical | Extreme |
| **Very High** | Nil | Medium | High | Critical | Extreme | Extreme |
| **Extreme** | Nil | Medium | High | Critical | Extreme | Extreme |

# Acceptable Risk

- Set level of acceptable risk
- Assign priorities for each threat based on acceptable risk and risk of threat

# Priorities

- A risk greater than acceptable risk + 1 level
- B risk is acceptable risk + 1 level
- C risk same as acceptable risk
- D risk less than acceptable risk
- Aim to do all of A,B and C priorities
- Do D priorities if you have time and budget

# Treatments

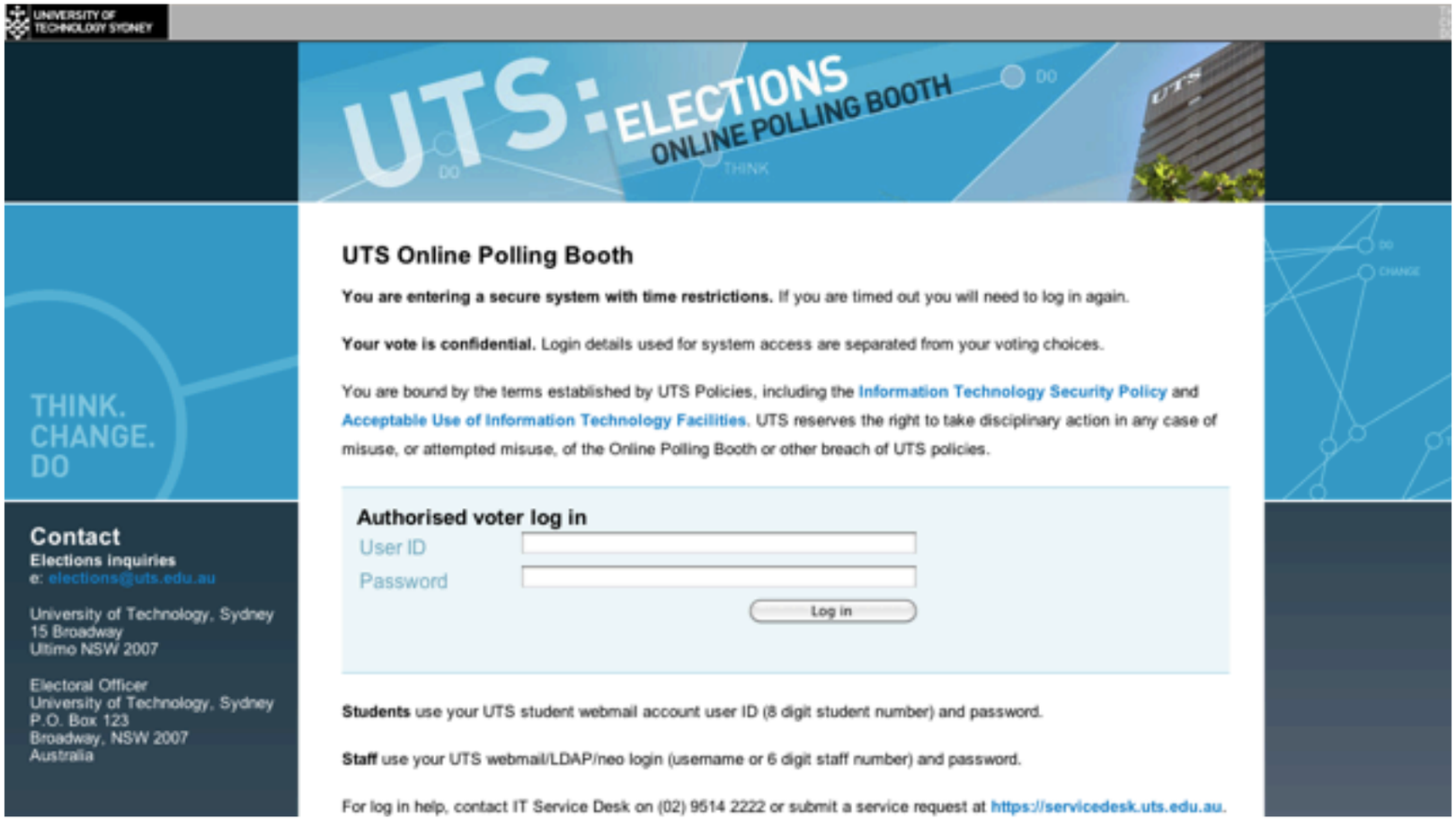
- Addition of security measures
- Reduction of security measures
- Minimisation of harm
- Change of service or system specifications
- Transference of risk
- Acceptance of risk

# Effort and Cost

- May be many ways to treat a single threat
- Amount of effort or cost may decide which treatment chosen

# Election System

Student election system for the
University of Technology Sydney




# Student Election System

OBJECTIVE (ANZ)

# Server Configuration

- Run with minimal down time
- Perform well under load
- Limited external access to server

# Server Treatment

- Staging/production system
- Not a shared server
- Standalone separate machines for database and CF server
- No access to production server
- Code reviewed by external agency

# Network Issues

- Occasional network outages
- Occasional slow access from outside

# Network Treatment

- Ability to change the end date after an election has started
- Date could only be extended not reduced

# SQL Security Issues

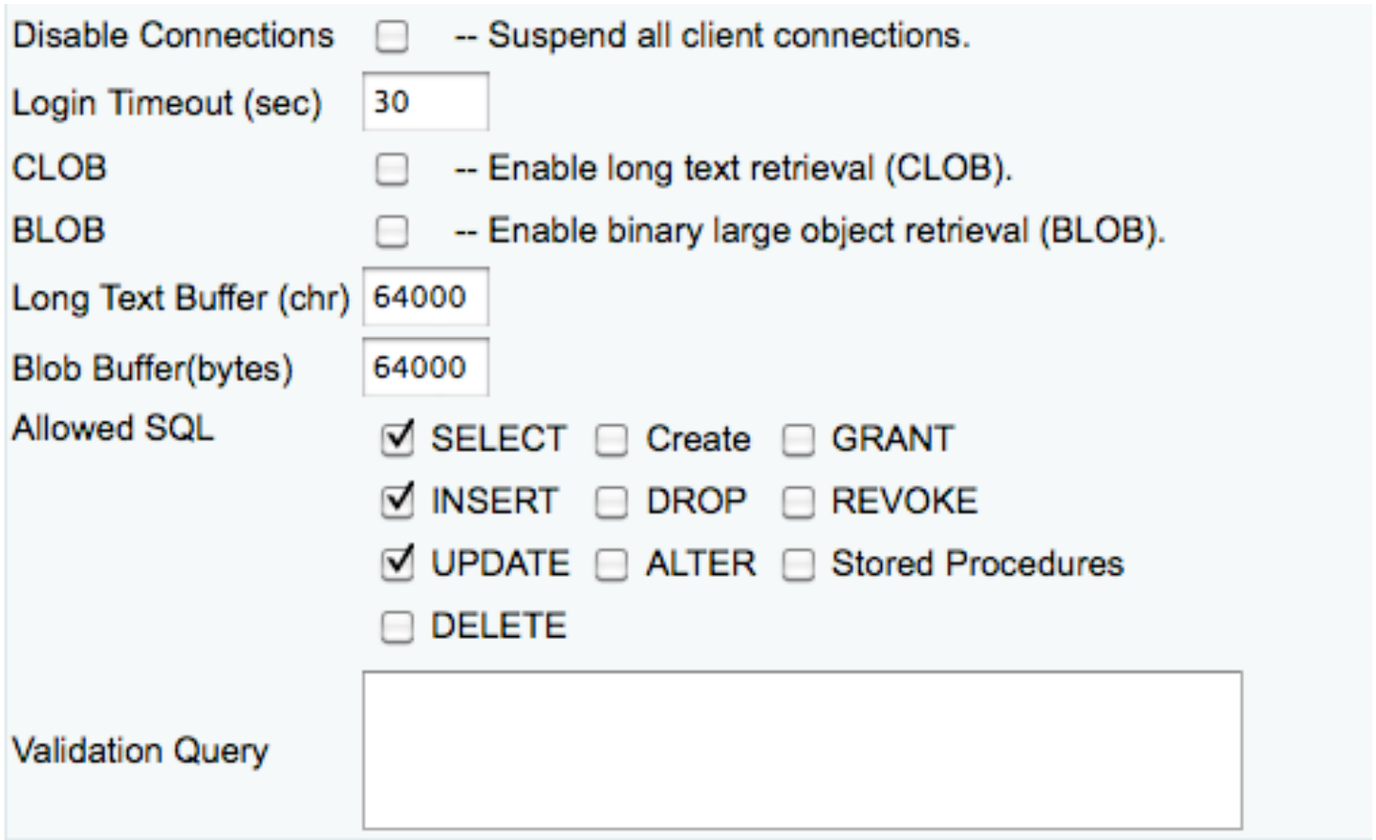
- SQL injection attacks
- Sensitivity of data
- Trust and integrity of election results

# SQL Injection

- Most common form of attack
- Malformed form or URL parameters to run evil SQL statements
- Wrap SQL in methods with type safe arguments
- Cfqueryparam is your friend!

# SQL Security Treatment

- Multiple data sources
- Multiple database users
- Restrict SQL actions. No deletes and almost no updates few inserts and mainly selects.
- Table level permissions




# Datasource Options

# Multiple Database Users

- Table level permissions
- SQL operation permissions

# SQL Permissions

- Deny all to all users to all tables
- Add permissions for each SQL operation as needed
- Don’t be tempted to give admin user all permissions

# Deny All

deny all on elections to electionvoter, electionadmin, electionlogin

deny all on candidates to electionvoter, electionadmin, electionlogin;

deny all on rolls to electionvoter, electionadmin, electionlogin;

deny all on ballots to electionvoter, electionadmin, electionlogin;

# Grant Access

```
grant select on roll to electionvoter,
electionadmin;

grant update on roll to electionvoter;

grant insert on roll to electionadmin;
```

# Login Issues

- Dictionary attacks
- Timing attacks
- Storing passwords

# Login Treatment

- Account lock out if password wrong x times
- Random time delay

# Java Sleep

```
<!--- delay is to hinder timing style attacks --->

<cfset thread=createObject("java","java.lang.Thread")>
<cfset thread.sleep(300 + int(rand()*21)*10)>
```

# Code Modification

- Pages code not modified
- Only run trusted pages

# Code Modification Treatment

- Finger print each page via MD5
- Check finger print when page is run via Application onRequest method

# onRequest

```
        <!--- read the cfm file --->
      <cftry>
          <cffile action="read" variable="pagecontents"
file="#CGI.PATH_TRANSLATED#">
          ........
       </cftry>

       <!--- get page from database --->
       <cfquery name="dbpage" datasource="#request.datasource#">
          select page, hash from pages
          where page = <cfqueryparam value="#hash(listlast(arguments.page,'/'))
#" cfsqltype="cf_sql_varchar">
       </cfquery>

       <!--- check if page exists and page hash is correct --->
       <cfif dbpage.recordcount is 1 and hash(pagecontents) is dbpage.hash>
           <cfinclude template="#arguments.page#">
       <cfelse>
          <cfinclude template="./elections/security.cfm">
       </cfif>
```

# Limiting Information

- Assume someone will break into the system
- What information can they obtain?
- What could they modify?
- Limit what they can see/use
- Minimise damage they can do
- Log everything

# Why do this?

- Know that you're spent you budget efficiently
- Confidence that your system is secure as it needs to be
- An understanding of the risks in your system
- Minimal damage occurs if the worse does happen

# Questions?

Ask now, see me after the session,
follow me on twitter @justinmclean
or email me at justin@classsoftware.com