UNCLE SAM'S GUIDE TO GRAILS SECURITY

JOE RINEHART

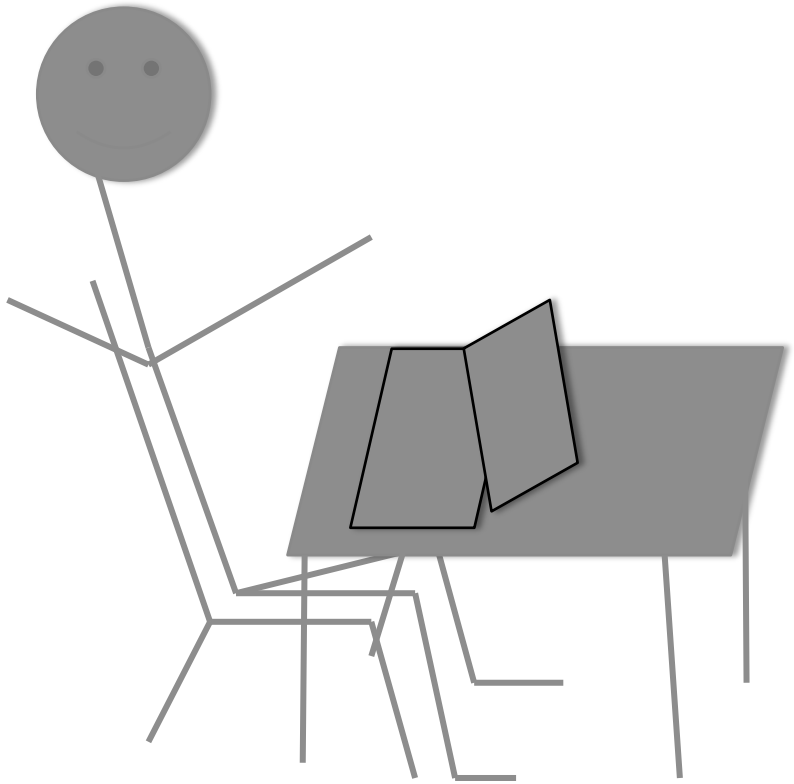
BOOZ | ALLEN | HAMILTON

Photo: AJ Cann

WHAT ARE WE TALKING ABOUT?

Photo: milos milosevic

WHO AM I?



WHO AM I?

Photo: eli duke

STIGS.

STIGS.

IASE Home | IA News | What's New | Consent Notice

STIGs

Search

– STIGs Home
- Home
- STIGs News
- FSO Release Schedule
- TIM / DSAWG Release Schedule
- IAVM to CVE

– STIGs Technologies
- STIGs Master List (A to Z)
- STIG Viewing Guidance
- Operating Systems
- Application Security
- Network / Perimeter / Wireless
- Cross Domain Solutions
- Sunset Products
- HBSS

IASE Home > STIGs Home

STIGs RSS Feeds

Updates!

IAVM to CVE Mapping Spreadsheet - Update August 23, 2013

Draft Office 2013 STIG Version 1 - Update August 21, 2013

Draft SQL Server 2012 - Version 1, Release 0.2 - Update August 19, 2013

IAVM to CVE Mapping Spreadsheet - Update August 16, 2013

Draft Windows Server 2012 STIG - Version 1 - Update August 12, 2013

The Security Technical Implementation Guides (STIGs) and the NSA Guides are the configuration standards for DOD IA and IA-enabled devices/systems. Since 1998, DISA Field Security Operations (FSO) has played a critical role enhancing the security posture of DoD's security systems by providing the Security Technical Implementation Guides (STIGs). The STIGs contain technical guidance to "lock down" information systems/software that might otherwise be vulnerable to a malicious computer attack. DISA FSO is in the process of moving the STIGs towards the use of the NIST Security Content Automation Protocol (S-CAP) in order to be able to “automate” compliance reporting of the STIGs.

A STIG Security Checklist, typically a companion of a STIG, is essentially a document that contains instructions or procedures to manually verify compliance to a STIG. STIGs have been under optimization efforts since 2008 to begin to combine the STIG and STIG Security Checklist into one document.

IN A NUTSHELL

A STIG is a list of security/assurance controls for a given technology.

There are many available: operating system, database, and application development.

We'll concern ourselves with the Application Development STIG.

PROPER PLANNING AND PREPARATION

PREVENTS PISS POOR PERFORMACE

MEA CULPA

These are things I've rarely done, or seen done, in the commercial space.

They're not fun.

They're good ideas.

Left to my own, I'd take a common sense, "good-enough" documentation approach.

SYSTEM SECURITY PLAN

Who can get to what.

What you think can go wrong.

What to do about it.

CONFIGURATION GUIDE

Hosting requirements

Deployment instructions

CLASSIFICATION GUIDE

How critical is data you're collecting?

Is it PII?

Are there legal ramifications if data is leaked?

3RD PARTY USE

What are you using that you didn't invent?

Where did it come from?

What support does it have?

KNOW THYSELF

Who's responsible for management? Design? Testing?

What are their strengths?

What are their weaknesses?

WHEN THINGS GO WRONG

What do you do when a vulnerability is identified?

Exploited?

When a natural disaster occurs?

SYSTEM DESIGN

AN OUNCE OF PREVENTION

I'M NOT TALKING "RATIONAL"

Please don't confuse encouraging some degree of system design with a push for anti-agile bits like RUP

KNOWING WHAT YOU DON'T KNOW YOU DON'T KNOW

Develop a threat model

- Assumptions
- Interactions
- Entries
- Risk
- Mitigations

BROAD GUIDANCE

How should an exception be handled, and what should end users see?

What kinds of encryption should be used? Avoided?

What should never be stored?

Basically, *document choices developers don't like to have to make.*

SIGH.

THIS COULD BE A LOT OF WORK.

DEVELOPMENT == GREAT TRAIL.

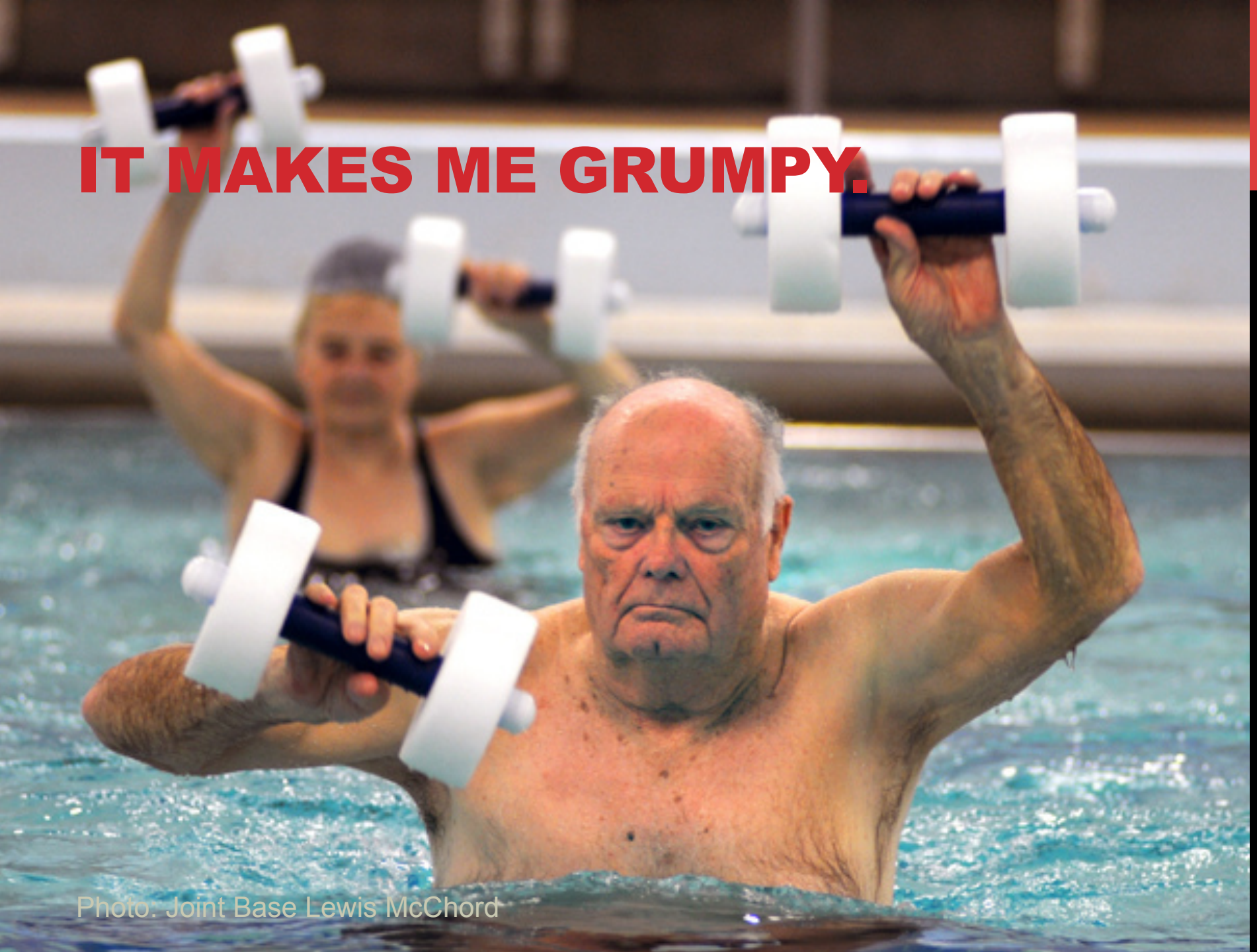
SECURITY == RICHARD SIMMONS

THIS

STUFF

IS

BORING


IT MAKES ME GRUMPY.
Photo: Joint Base Lewis McChord

SECURITY, AGILE, AND YOU.

IT'S NOT THE END OF THE WORLD.

INEFFECTIVE SECURITY

Most of the time I've seen STIGing, it's a tacked-on waterfall

…a sidecar.

…a box to check to get on with life.

I don't deal well with getting my ticket punched.

A JOB WORTH DOING

What's worse than security work?

Doing it poorly.

PROBLEMS?

1. Copied / pasted paperwork, designs, and plans
2. Documentation for documentation's sake
3. Ineffective process

BE ORIGINAL

Write from scratch. You need to think about these things.

If nothing else, it makes auditors do their job.

SERVE YOURSELF

Too often, people do these things just to get accredited.

Instead, cover your …

Make it known that these things are important and should take place.

Word processors are evil: use a wiki-style solution!

PART OF EVERYDAY LIFE

Don’t STIG and walk off.

It’s a task like any other: prioritize, backlog, and attack.

Ingrain security into your normal workflow.

GRAILS, FINALLY.

HOW WE'VE STIG'D IN RECORD TIME

TWO WEEKS.

Largely thanks to Grails and its community, we took a very basic JEE application and reached a secured state in two weeks.

MANY BIRDS, ONE STONE.

HUGE THANKS TO BURT BECKWITH

SPRING-SECURITY

Half of you have probably used it.

The other half will tell you why they haven't.

We have, and we like it, and here's why.

WHY SPRING-SECURITY?

1. Open source and well examined
2. Out-of-the-box good practices
3. Native x509 integration
4. LDAP, AD, and many other authentication providers are supported
5. Simple API!

SPRING-SECURITY: CONTROLS SATISFIED

- Role-based security
- Passwords must be stored in an encrypted format
- Passwords cannot be displayed in clear text
- Many PKI controls
- Session logout
- Token rotation
- Logon limitations

MANY BIRDS, MANY PEBBLES.

A BRIEF TOUR OF SOLVING SOME CONTROLS WITH GRAILS

CSRF VULNERABILITIES

```
<g:form useToken="true" action="save" />
```

CODE COVERAGE

We've used the code-coverage plugin. Big thanks to Mike Hugo and Jeff Beck.

```
> grails test-app -coverage
```

TRANSACTIONS

Does data change? Does a controller perform two data access operations?

Service time.

```
> grails create-service com.acme.Book
```

DATABASE CREDENTIALS

Please don’t let anyone find this. Externalize!

```
environments {
    production{
        dataSource {
            username = "sa"
            password = "password"
        }
    }
}
```

LAST TIME/DATE OF CHANGE

```
package com.acme


class Book {
      Date dateCreated
      Date lastUpdated
}
```

LAST TIME/DATE OF CHANGE – EXTRA CREDIT

```
class DataAccessAuditingListener implements
ApplicationListener {


void onApplicationEvent( ApplicationEvent e ) {
  if ( e instanceof AbstractPersistenceEvent ) {
    // do stuff
  }
}


}
```

XSS

1. Filter posts to check referers!
2. grails.views.default.codec = ‘html’

SQL INJECTION

It's hard to really screw this up with Grails/GORM, but if you're using straight SQL, please:

1. **Use bound parameters instead of GStrings**
2. **Understand the ramifications of toString()'ing a GString and then using it as a statement!**

ACCOUNT LOCKOUT

```
class UserLoginAuditingListener implements
ApplicationListener {


void onApplicationEvent( ApplicationEvent e ) {
  if ( e instanceof AbstractAuthenticationEvent ) {
    // do stuff
  }
}


}
```

VALIDATION

This one's hard to enforce programatically, but we try to cover it by:

1. **Using commands**
2. **Testing domain validation**

CONCLUSIONS?

THE STIG IS YOUR FRIEND.