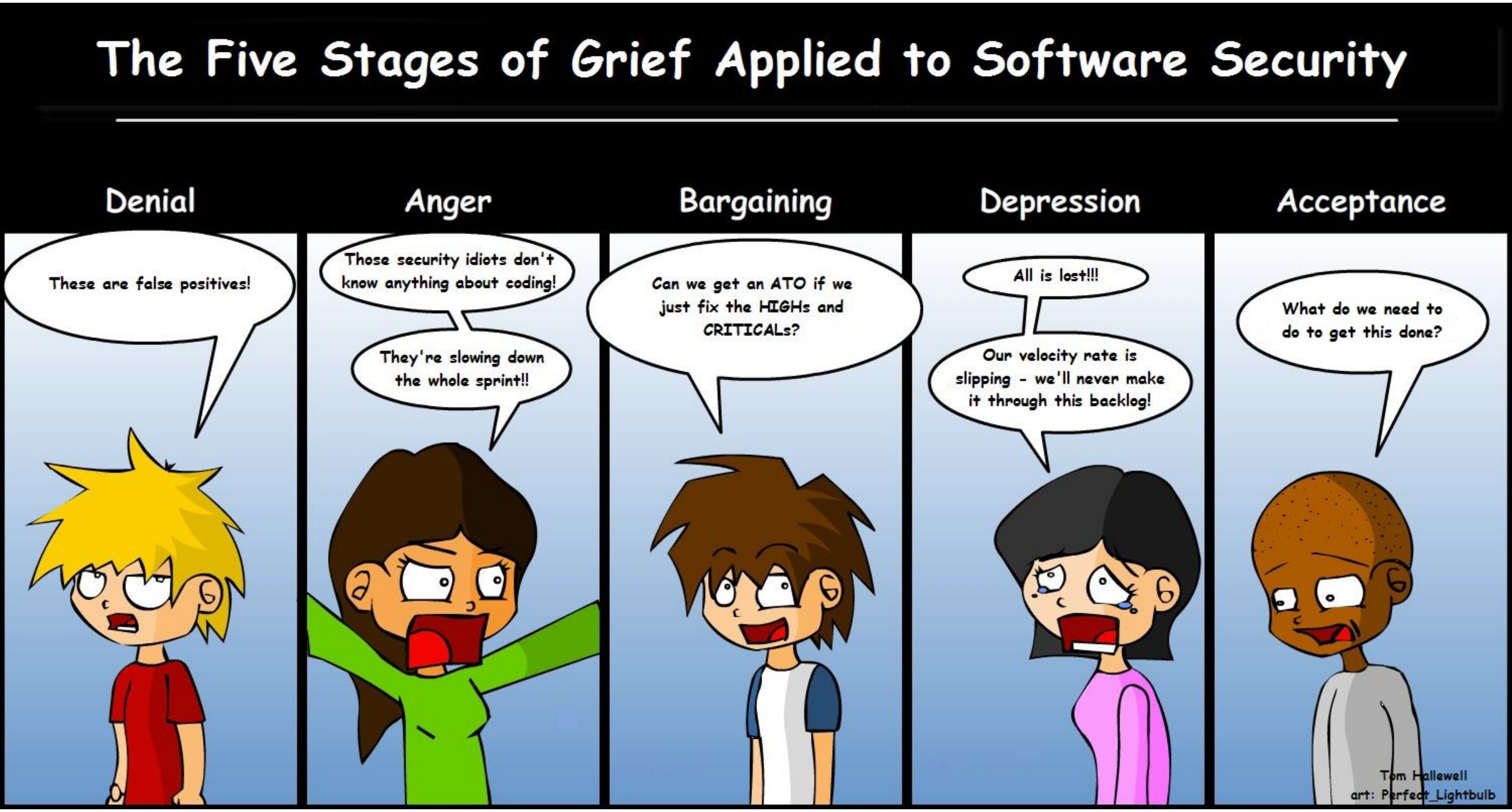

The Five Stages of Grief Applied to Software Security
Denial
Anger
Bargaining
Depression
Acceptance
These are false positives!
Those security idiots don't know anything about coding!
They're slowing down the whole sprint!!
Can we get an ATO if we just fix the HIGHs and CRITICALs?
All is lost!!!
Our velocity rate is slipping - we'll never make it through this backlog!
What do we need to do to get this done?
Tom Hallewell
art: Perfect_Lightbulb

Hello, World

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

Scott Lehman – Lifelong Geek
Tom Hallewell – Fed

The views expressed in this presentation do not reflect on those of our employers.
Any resemblance to real events is coincidental.

Why Software Assurance?

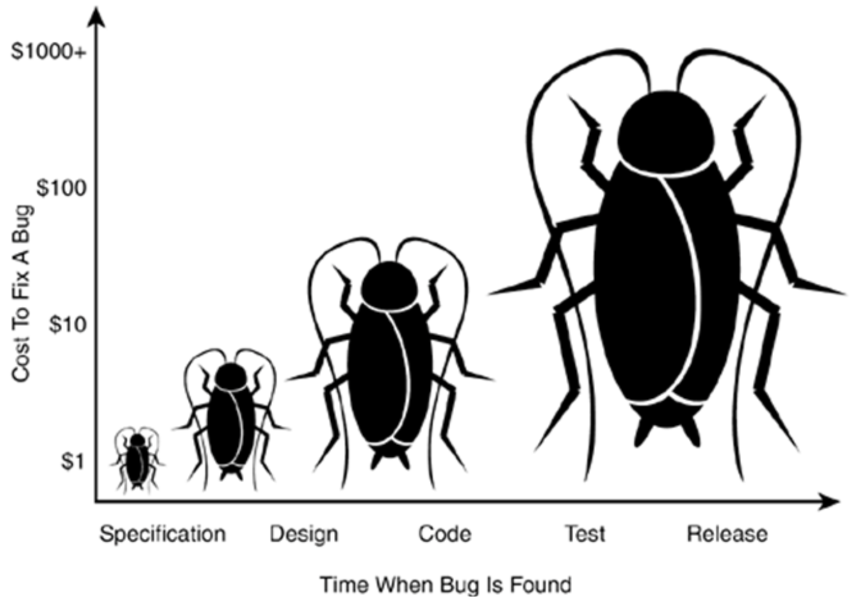
- Software vulnerabilities are one of the most common sources of compromise.
- Software flaws can directly impact Confidentiality, Integrity, and Availability.
- Software developers rarely receive any security-focused training.

Confidentiality
Integrity
Availability
Data

www.themailadmin.com

Why Static Analysis?

- Earliest possible detection of security-related flaws.
- Static Code Analysis can begin before functionality exists
- Manual code reviews are subject to human limits
- Reviews are quick, thorough and repeatable
- Humans are free to look the “big picture”



stevewedig.com

Alternatives to Static Analysis

- **Manual Code Review**
  - Requires your best developers to stop writing code
  - Nearly impossible to examine every line
  - The view is often myopic with very little consistency
- **Automated Penetration Testing**
  - High confidence in findings
  - Will find many deployment problems
  - Coverage is rarely 100%.
  - The best results require considerable “training” of the scanner
- **Manual Penetration Testing**
  - Requires an extreme skillset
  - As much an “art” as a “science”

What's your Angle?

Other groups have a stake in software quality

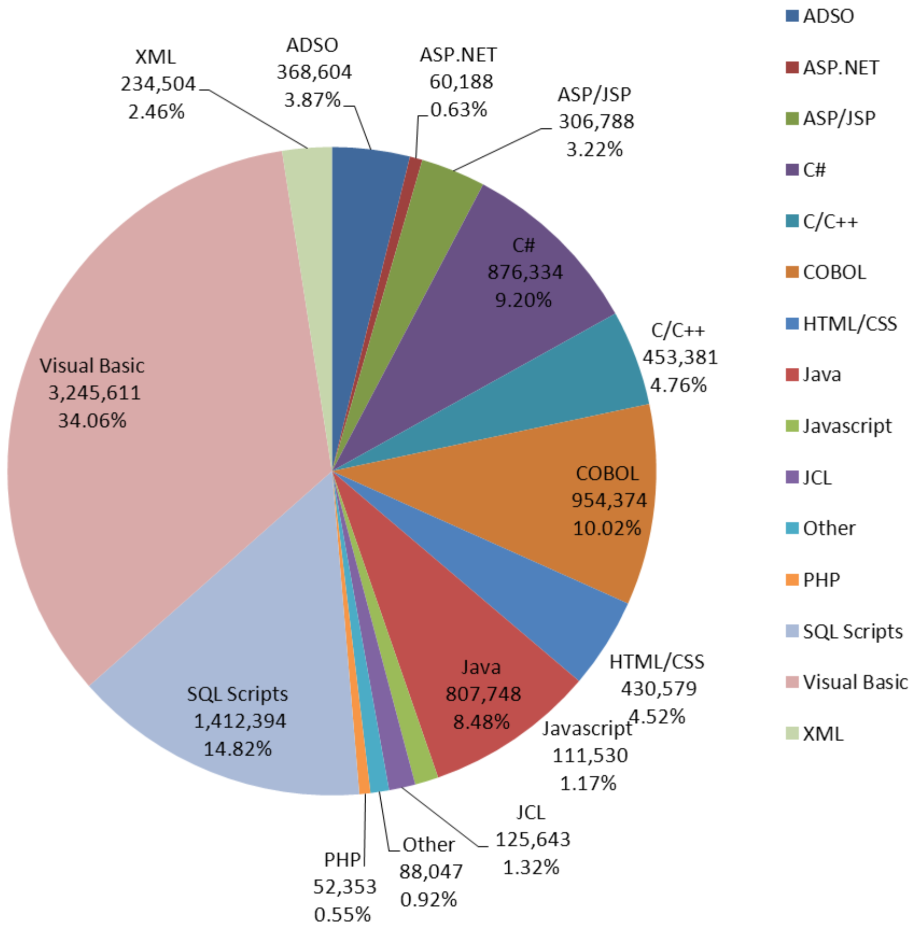
QA Teams look for

- Code reliability
- 508 compliance
- Adherence to organizational coding standards/best practices

Since we represent Information Security, we decided to focus on code that is:

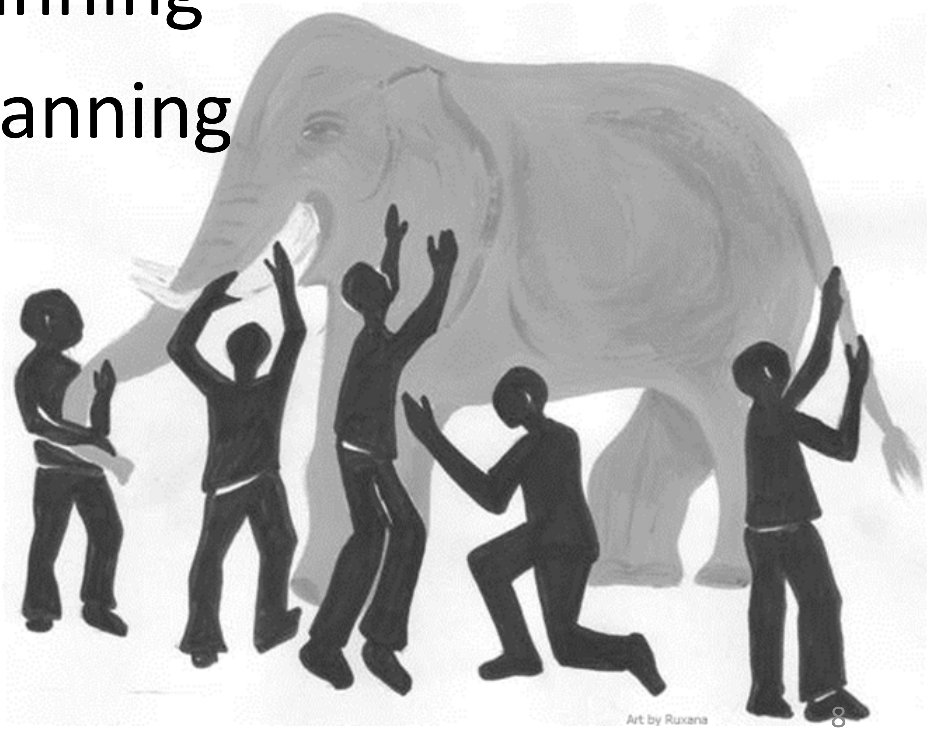
- Exploitable
- Could affect the Confidentiality, Integrity, or Availability of the system or data

What Languages do you need to support?



Static Analysis Delivery Models

- Cloud/Software as a Service
- Central, Manual scanning
- Developer desktop scanning
- Central, Automated scanning
- Hybrid model



Products: It’s a Jungle out There!

- Buguroo
- Cppcheck
- Grammatech
- LDRA Testbed
- Monoidics INFER
- C++test & Jtest
- CodeSecure
- Armorize
- Coverity
- SofCheck Inspector for Java
- Checkmarx
- Klocwork
- Fortify
- BugScout
- Codesonar
- Sparrow
- Goanna
- Veracode (service, not a tool)
- Aspect Security ASC (service, not a tool)

NASA.gov

And this was back in 2012!

Vendor Landscape

Product	Ada	Android	APEX	ASP	ASP.Net	C/C++	C#	COBOL	ColdFusion	Flash	Objective C	Java	Javascript	JSP	Mobile	.Net	Perl	PHP	PL/SQL	Python	Ruby	T-SQL	VB6	VB.Net	VBScript	Gartner Static[1]	Gartner Dynamic[1]	Method[2]	Delivery Platform
AdaCore Codepeer	X					X						X								X						N-A	N-A	S, D	Server app
Aspect Security ASC				X		X		X	X			X	X	X		X	X	X	X			X	X	X		N-A	N-A	S, D	SaaS, Service, Server app
Bugaroo BugScout & BugBlast				X			X					X						X								N-A	N-A	S, D	SaaS, Appliance
Checkmarx		X	X	X		X	X			X	X	X	X			X	X	X			X		X	X		V	N-A	S	Server app
Coverity						X	X					X														C	N-A	S	Server app
Cppcheck						X																				N-A	N-A	S	Stand-alone app (Open Source)
FindBugs												X														N-A	N-A	S	Stand-alone app (Open Source)
Fortify HP					X	X		X	X			X	X	X		X		X	X	X		X	X		X	L	L	S, D	Environment
GrammaTech						X						X														V	N-A	S	Environment
IBM Rational				X		X		X	X			X	X	X		X	X	X	X			X	X			L	L	S, D	Environment
Klocwork						X	X					X														V	N-A	S	Server app
LDRA Testbed	X					X																				N-A	N-A	S, D	Server app
Parasoft C++test & Jtest						X	X					X	X	X										X		C	N	S, D	Environment
Red Lizard Software Goanna						X																				N-A	N-A	S	Environment
Veracode (service, not a tool)					X	X			X			X		X				X			X			X		L	V	S, D	SaaS

Gartner Magic Quadrant rating

[1] N = Niche Players
C = Challengers
V = Visionaries
L = Leaders

[2] S = Static Analysis
D = Dynamic Analysis

Research performed 3/2012

If I Were King…

You don't need to **buy** a fancy tool to start your program

- Commercial tools are expensive, and may not fill your needs
- Do your proof-of-concept with a free tool
  - *Your organization may just not be ready for software assurance*
- Once you've built a process, start looking for the perfect tool
  - *According to some sources,* ***no*** *single static analysis tool will find all vulnerabilities*

Get Management Support

- Demonstrate to senior management that this is important
  - Statistics won’t do it by themselves
  - FUD only goes so far
- This isn’t an easy sell – software assurance is expensive, even if the tools are free
  - You need a solid business case – can you show ROI?
  - FUD only goes so far
- Line managers and project managers must clear time for developer training
  - Make sure the training is valuable

The Stakeholders

The Security Officer

http://bumpsandcurves.com

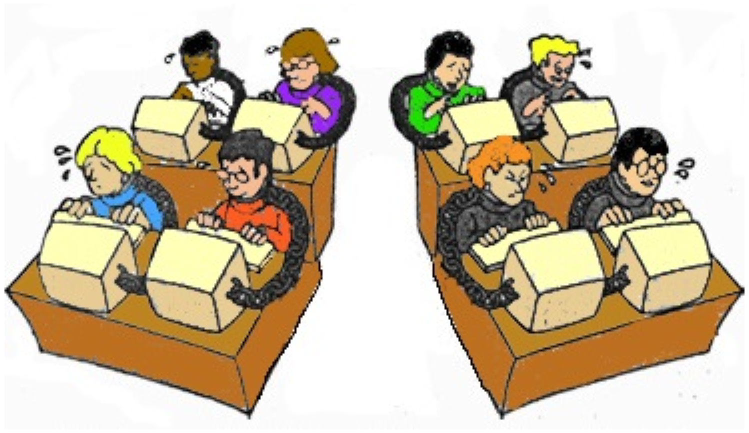
The Business Owner

Dreamstime.net

The Project Manager

www.cutevector.com

The Development Team



humanbeh-winter2010-topic11.wikispaces.com

The Security Officer

Focus: Security and Compliance
Usually runs the software assurance program (ie, your boss)

- "You don't comply unless you fix everything!"
- "Fix it all, regardless of cost!"
  - May not differentiate between severity/risk of findings
  - Most findings are valid
- "Why can't these people just fix it?"

http://bumpsandcurves.com

What you Need to Tell Him

- We are not going to get all findings resolved overnight.
- If we are too heavy-handed, the software assurance program will fail.
  - Developers will find ways to evade the scans
  - The Product Owner will get Senior Management to pull the plug

The Project Manager

www.cutevector.com

- "Will fixing this delay the project?"
- "What do the developers say?"
- May question the validity of findings
- May escalate to Senior Management

Focus: Product Delivery
Her career depends on getting a working product out on time

What you Need to Tell Her

- There will never be zero findings.
- We will focus on low-hanging, high-impact findings first, then raise the bar.
- We're always available to help understand and resolve findings

The Product Owner

Dreamstime.net

Focus: Product Usability
He needs a working product yesterday to support the mission

- "What is the business value of fixing this?"
- "This just a (....) application, why all the fuss about security?"
- May question the validity of findings
- **Is** Senior Management

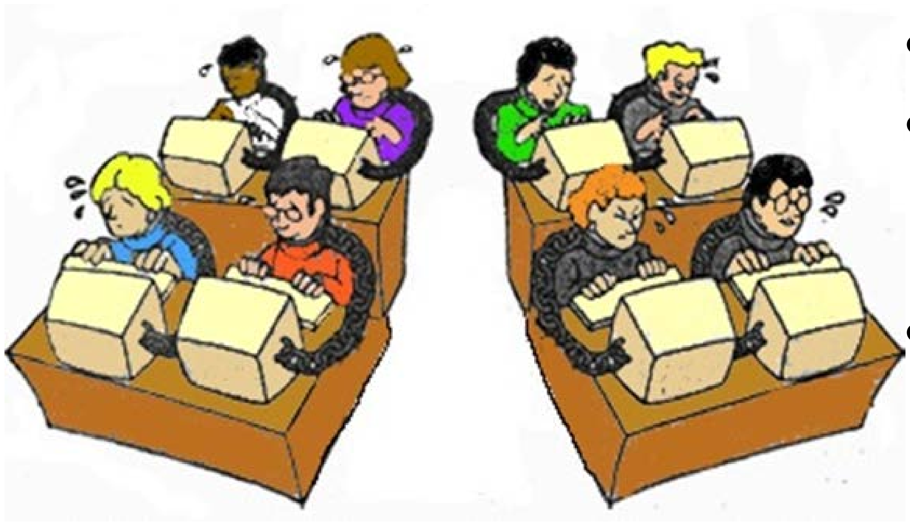
What you Need to Tell Him

- You can’t put a dollar value on a compromise that doesn't occur.
  - This means that traditional ROI models will not show value
- What does show value is improvement over time.
- We aren’t going to make the team fix findings just because they exist.
  - If a finding doesn’t prevent a risk, it doesn’t need to be fixed right away – or ever?

The Development Team

Focus: Get 'er Done
If we don't get this out on Tuesday, we're all fired!

- "Coding is my Art."
- "These findings are bulls***!"
- "This is test code!"
- "We don't have time/resources/skillset to fix these findings"
- May escalate to management
- Judged on code functionality, features and delivery time, not security
- Wants to deliver quality code, but feels time-constrained



humanbeh-winter2010-topic11.wikispaces.co

What you Need to Tell Them

Pledge not to:

- Score “points” by finding lots of issues
  - We’re in this together – there are enough real issues to focus on
- Point fingers or assign blame
  - Let’s get these findings resolved and move on
- Whitewash the results

Commit to:

- Only flag findings that affect the security stance of the application
- Being supportive, responsive and non-judgemental
- Provide meaningful feedback to development teams

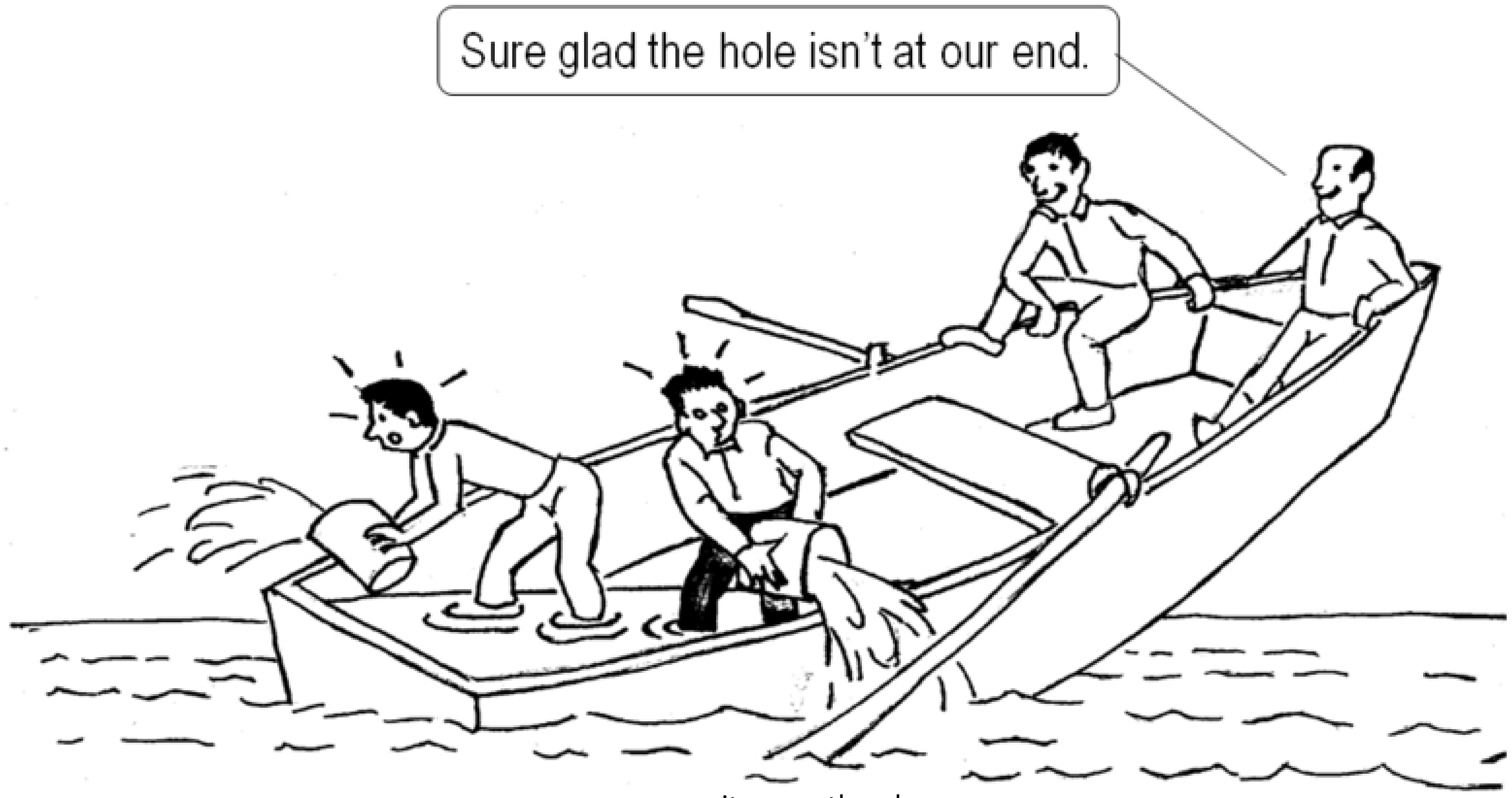
The Software Assurance Team

(That’s you)

www.wexfordgaa.ie

Focus: Get exploitable findings resolved

Remember, we’re all in this together!

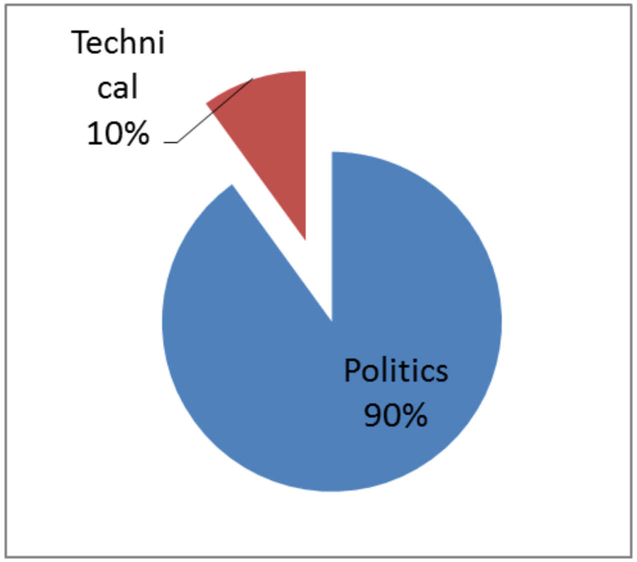


security-marathon.be

This is **not** a Technical Issue!

Technically, it is pretty trivial to start a software assurance program:

- Procure a maximum of two servers
- Install and configure platform and dependencies
- Install the scan tool
- Schedule Training
- Integrate it into the build process, if possible
- Scan code!
- Interpret and communicate results
- Get developers to fix findings



Politically, it is a nightmare to implement

- Get top-down support
- Schedule Training
  - Guarantee: at least 50% of scheduled students can't attend due to an emergency
- Integrate it into the build process, if possible
  - You'll be amazed at the reasons this is impossible
- Interpret and communicate results
- Get developers to fix findings

Process Depends on the Tool

You may point at a directory full of source code.

OR

You may have to scan in an actual build environment.

- You must scan ALL code that will be deployed.
- "Code Generation" must be taken into account.
- Only ignore “test code” when you are certain that it can't be accidentally promoted.

“This Scan Frequency is Just Right…”

Too often:

- Drains resources
- Generates results faster than they can be reviewed.

Too infrequently:

- Loses the advantage of fixing early in the cycle.

Suggestions:

- Daily: when code is changing rapidly
- Weekly: a good balance for many shops
- Timed with Sprints or Milestones
- Immediately before a Release

www.southernfriedscience.com

When Should you Fix Findings?

- **On check-in?**
  - Force developers to remediate findings before they are allowed to commit code to the repository
- **Daily?**
  - Creates a lot of overhead, but gives developers to fix findings early in the lifecycle
- **Weekly?**
  - Be sure to synchronize your scans with the development cadence
- **Timed with Sprints or Milestones?**
  - Findings may trigger re-work in a future Sprint
- **Immediately before a Release?**
  - Use this scan for your compliance go/no-go decision
  - **This is the most expensive time to fix findings!**

We Have Results!

http://funny-pictures.picphotos.net

What Now?

- Triage
- Communicate results to stakeholders
- Prepare to be underwhelmed

Triage

Divide your findings:

"False Positives"

- A good scanner is "pessimistic"
- There may be mitigations in place that the scanner doesn't see
- Even a valid finding might be irrelevant in your environment

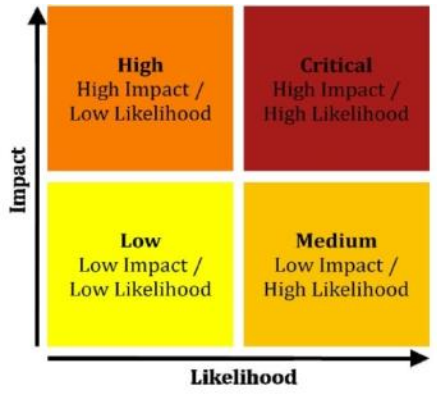
Prioritize Valid Security Issues

High Risk, Easy to Fix

to

Low Risk, Difficult to Fix

Take a Risk-based Approach


High
High Impact / Low Likelihood
Critical
High Impact / High Likelihood
Low
Low Impact / Low Likelihood
Medium
Low Impact / High Likelihood
Impact
Likelihood

Priority Level	Definition
Critical	Critical issues have a high impact and high likelihood. Critical issues are easy to discoverer and exploit and result in large asset damage. These issues represent the highest security risk to the program. As such, they should generally be remediated immediately. SQL Injection is an example of a critical issue.
High	High-priority issues have a high impact and low likelihood. High-priority issues are often difficult to discover and exploit, but can result in large asset damage. These issues represent a significant security risk to the program. High-priority issues should generally be remediated in the next scheduled patch release. Password Management: Hardcoded Password is an example of a high-priority issue.
Medium	Medium-priority issues have a low impact and high likelihood. Medium-priority issues are easy to discover or exploit, but often result in small asset damage. These issues represent a moderate security risk to the program. Medium-priority issues should be remediated in the next scheduled product update. ASP.NET Misconfiguration: Missing Error Handling is an example of a medium-priority issue.
Low	Low-priority issues have a low impact and low likelihood. Low-priority issues can be difficult to discover and exploit, and they typically result in small asset damage. These issues represent a minor security risk to the program. Low-priority issues should be remediated as time allows. Poor Error Handling: Empty Catch Block is an example of a low-priority issue.

Source: HP Fortify

Rule Number One

Don’t call them bugs, flaws, vulnerabilities, or errors.

They are **FINDINGS**.

Got it?

One More Time

Don’t call them bugs, flaws, vulnerabilities, or errors.

They are **FINDINGS**.

Regardless of risk, severity or potential impact.

Fanbox.com

Don’t make me tell you again, foo…

Triage

- Requires an understanding of programming
- Strive for a single point of responsibility
- It’s hard for a developer to judge her own code
  - But she’s still an invaluable resource.
    - Helps understand findings.
    - Helps determine the Level of Effort to resolve
- If developers perform the triage, **you must verify**!
- Developers typically underestimate risk and impact

Triage is a Process, not an Event

Triage is not decisive.

In an Agile environment, you still must convince the Product Owner, not only that the finding presents a risk, but that it is worth fixing.

Triage is not Final.

As long as code is in flux, scanning must continue.

Once code is stable, it's a good practice to re-scan whenever the scan tool/rule-packs are updated.

Communicating Results

Prepare for your findings to be challenged...

Communicating Results

Developers will tell you to show them an exploit before they will fix a finding.

- This is a trap - a problem that can be fixed in five minutes can require days to exploit!

Hackers:

- Work longer hours than we do
- Have less overhead than we do
- Often have stronger incentives than we do
- Might be smarter than we are

When in doubt, a finding should be addressed!

Remediation

- Establish and communicate clear priorities **before** the first scan
- A problem is "fixed" when it is no longer found in a scan
- Some findings are very hard to fix definitively
  - Consider mitigation strategies
- Don’t let developers game the system!

You’ve heard this before

TRUST

BUT

VERIFY

Train Your Teams

Secure coding training makes good coders better coders

businessinsider.com

Suggested Developer Curriculum

- **Intro to secure coding**
- **Use of the code-vetting tool**
  - **Interpreting scan results**
  - **Whitelisting false positives**
  - **Resolving coding errors**
  - **Reporting action taken**
- **Common coding errors and their impact**
- **How to resolve coding errors**
- **Resources and references**

Suggested Auditor Curriculum

- **Secure coding in-depth**
  - **Different languages**
  - **Understanding context**
- **How to tune/customize the analysis tool**
- **Whitelisting, Remediating findings**
- **Your remediation policy**

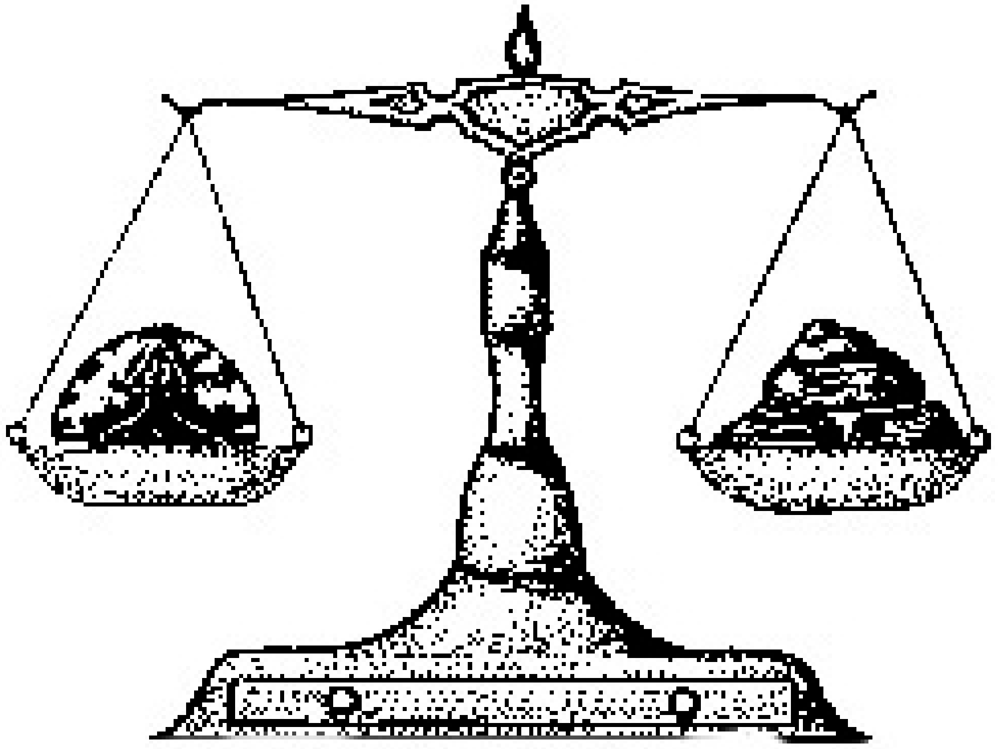
www.zazzle.com

Track your Team's Progress over Time

- Shows whether you are gaining traction
- Helps identify areas for future developer training
- Shows you when it's time to ingest another project
- Management loves graphics

Some scanning tools generate pretty reports and charts

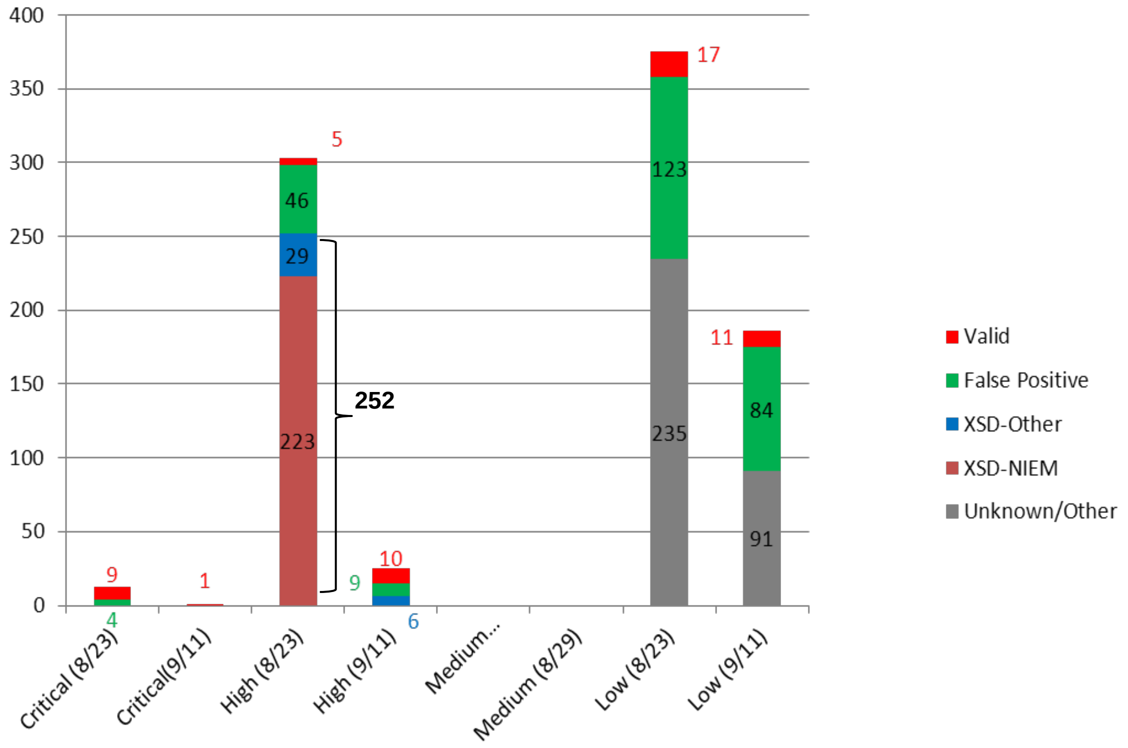
A Tale of Two Teams



sel.barc.usda.gov

We began a pilot with two teams working on two distinct development projects….

System A Code Scan Results



Statistics:

8/22

- Lines of Code: 37,191 *****
- Total Findings: 691
- Findings per line: 0.0186

9/11

- Lines of Code: 29,366 *****
- Total Findings: **212**
- Findings per line: 0.0072

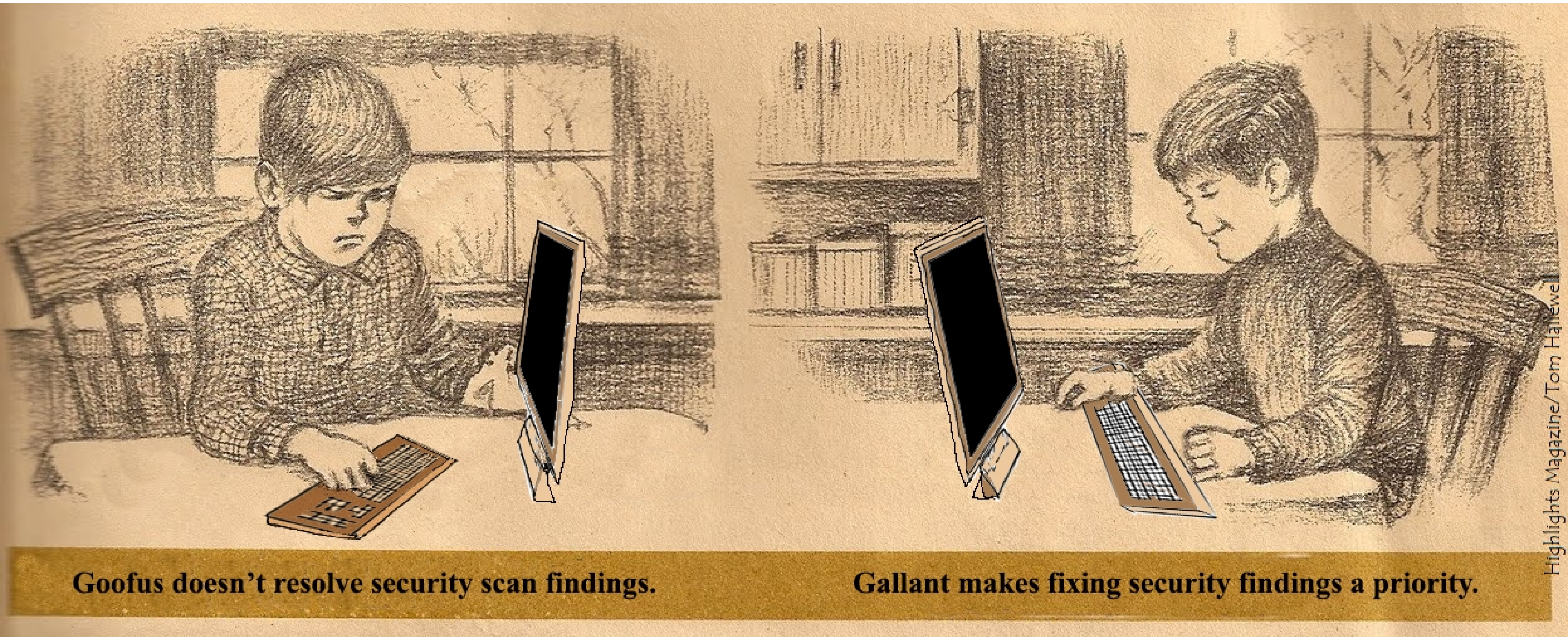
61 % reduction in FPL!

* For the 9/11 scan, we removed all directories named “test” from the scan base.

9/11 scan Results

Level	Total	Actionable	Reduction
Critical	1	1	92%
High	25	10	92%
Medium	0	0	N-A
Low	186	6	73%
Total	212	17	68%

Some Teams Get it…


Goofus doesn't resolve security scan findings.
Gallant makes fixing security findings a priority.
Highlights Magazine/Tom Hallewell

Highlights Magazine – manipulated by Tom Hallewell

Project B Code Scan Results

Statistics:

8/22

- Lines of Code: 141,224
- Total Findings: 1114
- Findings per line: 0.0079

9/11

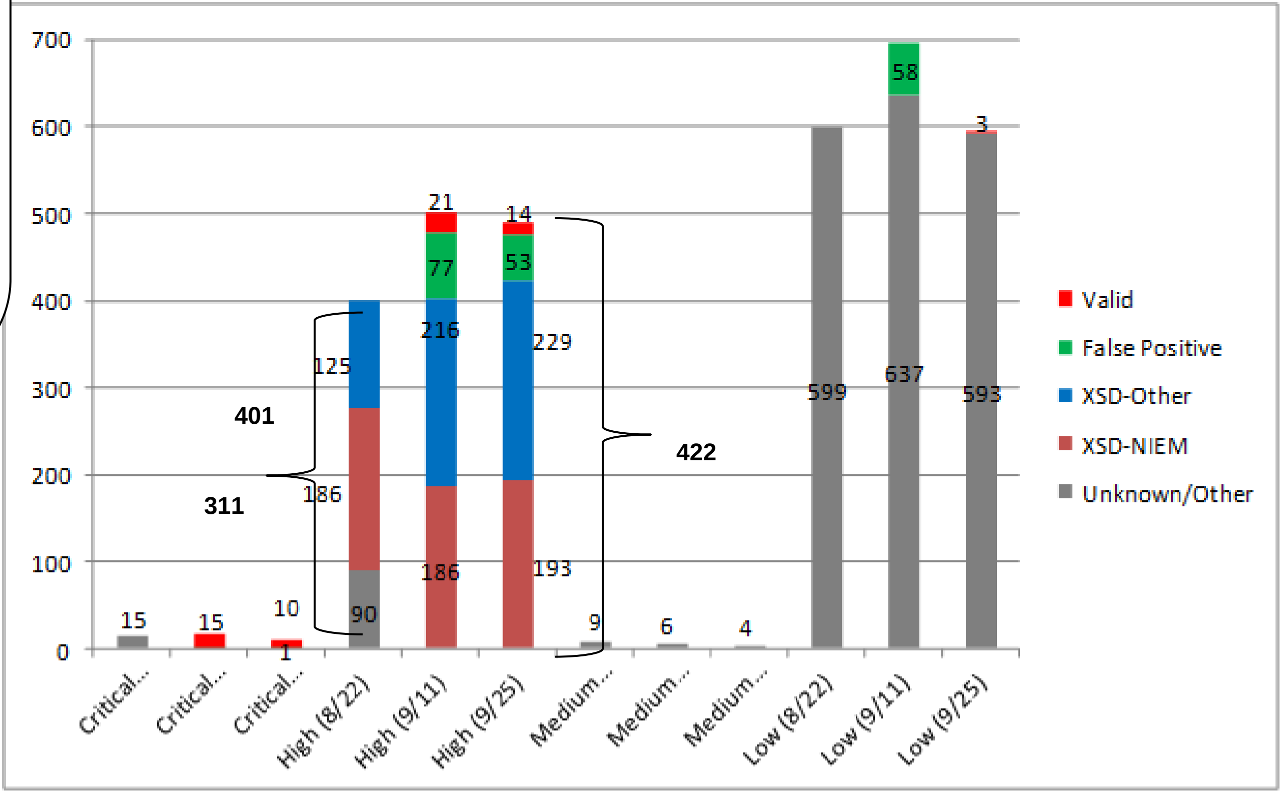
- Lines of Code: 110,812 *
- Total Findings: 1217
- Findings per line: 0.0110

9/25

- Lines of Code: 105,636 *
- Total Findings: 1100
- Findings per line: 0.0104

Observations:

- 38% of total findings due to issues with XML stylesheets
- Many are inherited from upstream systems and cannot be resolved by the Project B Team
- However, Project B continues to implement improperly configured stylesheets from other sources (5% increase from 9/11 scan)



* We continue to remove any directories identified as "test-related" from the scan base.

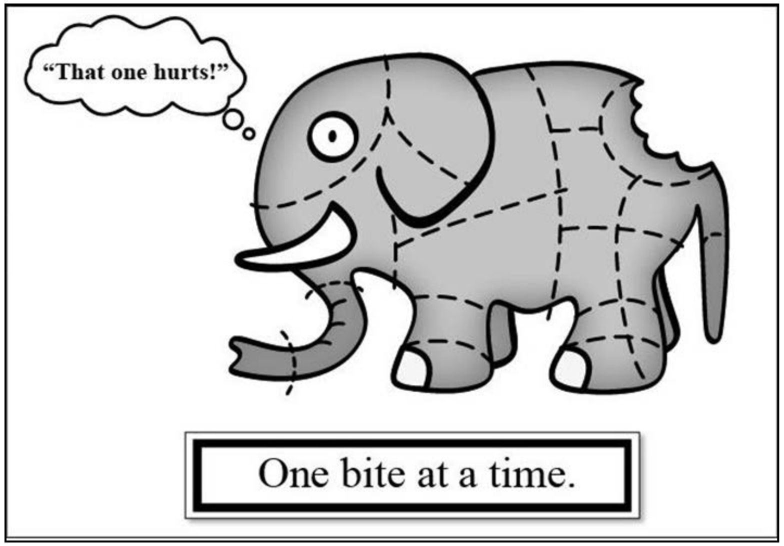
Limbo in Reverse

1. Keep your expectations low at first
2. Celebrate small successes
3. Gradually raise the bar

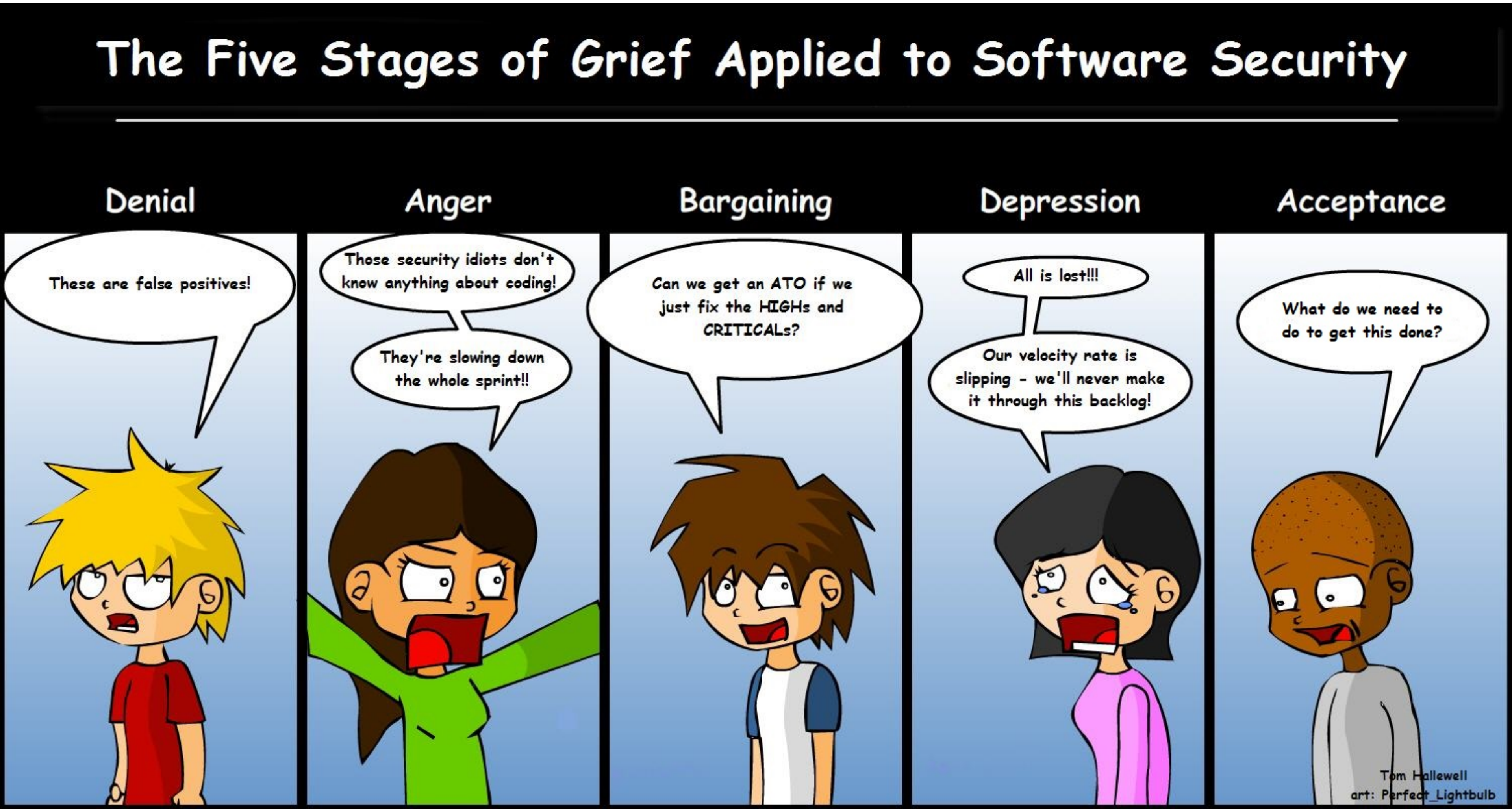
wipwapweb.com

Think big, start small

- Onboard projects in waves
- Make sure each project is a success before onboarding another one
- Remember, software grief is endless – be sure you have enough resources to continue to support existing projects before you take on another one



Gettingbusinessresults.wordpress.com


The Five Stages of Grief Applied to Software Security
Denial
Anger
Bargaining
Depression
Acceptance
These are false positives!
Those security idiots don't know anything about coding!
They're slowing down the whole sprint!!
Can we get an ATO if we just fix the HIGHs and CRITICALs?
All is lost!!!
Our velocity rate is slipping - we'll never make it through this backlog!
What do we need to do to get this done?
Tom Hallewell
art: Perfect_Lightbulb