Security in knowledge

Writing Applications that are Easier to Defend than Attack

Alan H. Karp

Hewlett-Packard Laboratories

Marc Stiegler

Hewlett-Packard Laboratories

RSA CONFERENCE 2013

Session ID: ASEC-T17

Session Classification: Advanced

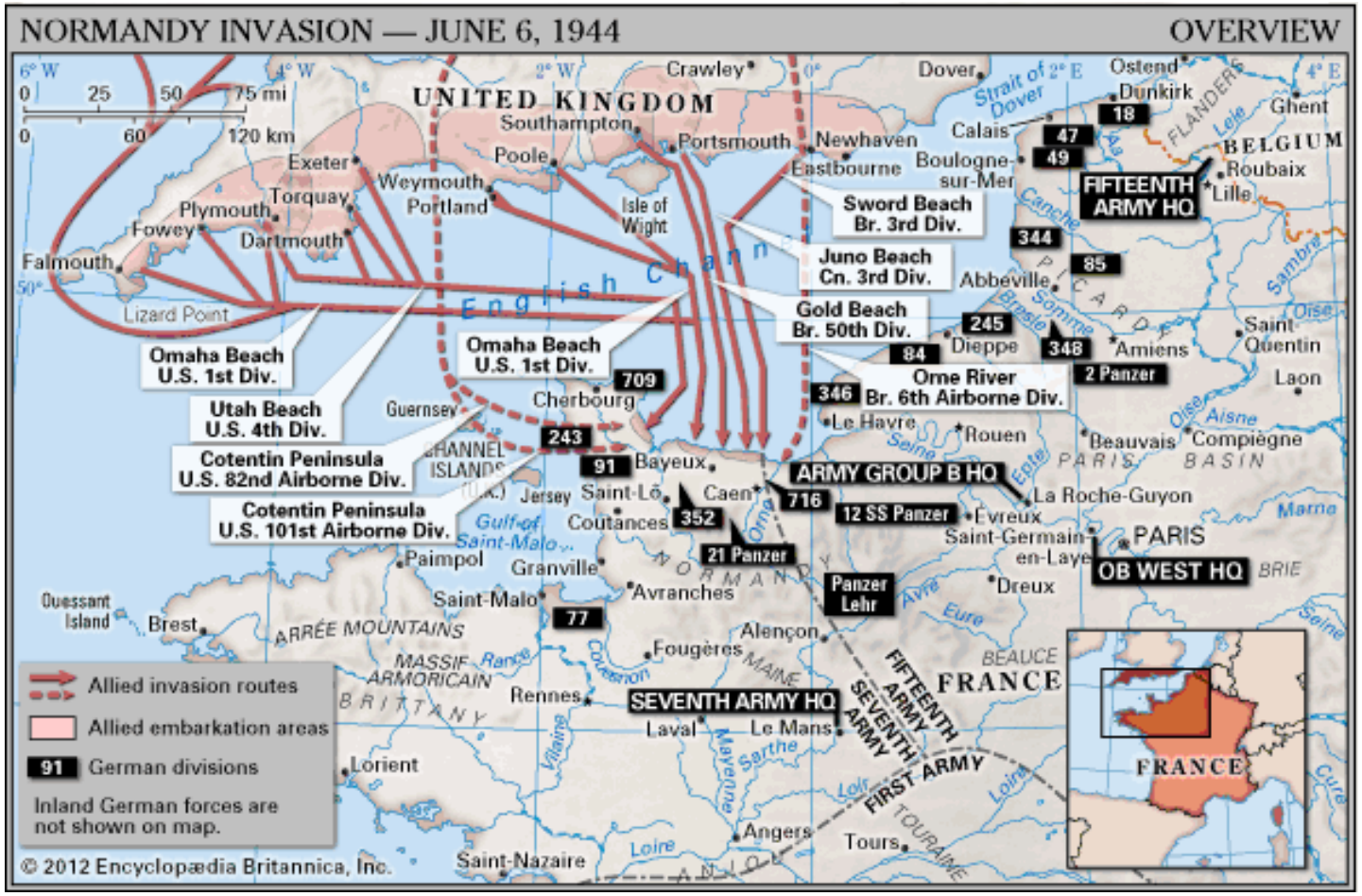
Most Common Security Patches

- 75% of Microsoft security bulletins are for applications
  - Year, after year, after year

"If what we were doing was effective, wouldn't you expect things to be getting better?"

-- Marcus Ranum

We Know Why


NORMANDY INVASION — JUNE 6, 1944
OVERVIEW
UNITED KINGDOM
English Channel
BELGIUM
FRANCE
Omaha Beach U.S. 1st Div.
Utah Beach U.S. 4th Div.
Cotentin Peninsula U.S. 82nd Airborne Div.
Cotentin Peninsula U.S. 101st Airborne Div.
Omaha Beach U.S. 1st Div.
Sword Beach Br. 3rd Div.
Juno Beach Cn. 3rd Div.
Gold Beach Br. 50th Div.
Orne River Br. 6th Airborne Div.
FIFTEENTH ARMY HQ
ARMY GROUP B HQ
OB WEST HQ
SEVENTH ARMY HQ
Panzer Lehr
12 SS Panzer
21 Panzer
2 Panzer
709
243
91
352
716
77
47
49
18
344
85
245
84
348
346
Allied invasion routes
Allied embarkation areas
91 German divisions
Inland German forces are not shown on map.
© 2012 Encyclopædia Britannica, Inc.

Anderson's Economic Analysis

- Defender's cost
  - 1,000,000 line program
  - 1 exploitable bug/10,000 lines
  - 100 hour/bug
  - 10,000 hours
- Attacker's cost
  - 1.000 hours/bug
  - Need to exploit 1 bug
  - 1000 hours
- Defender's cost/Attacker's cost >> 1

What does the attacker win?

- A clue for finding an answer

"Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights."

-- Microsoft Security Bulletins

Principle of Least Privilege

"Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job."

-- Jerome H. Saltzer
Michael. D. Schroeder

"Protection and the control of information sharing in multics". *Communications of the ACM* **17** (7): 389, (1974).

Root Cause of the Problem

Every program you run can use all your permissions.

Don’t do that!

A Short Detour

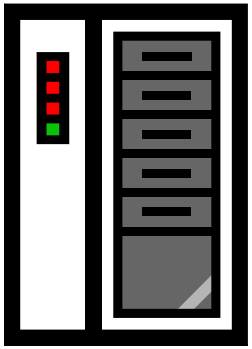
Security in knowledge

What is a Privilege?

- Saltzer and Schroeder didn't say precisely
- Principle of Least Authority
  - Easier to say (POLA vs POLP)
  - Precise meaning

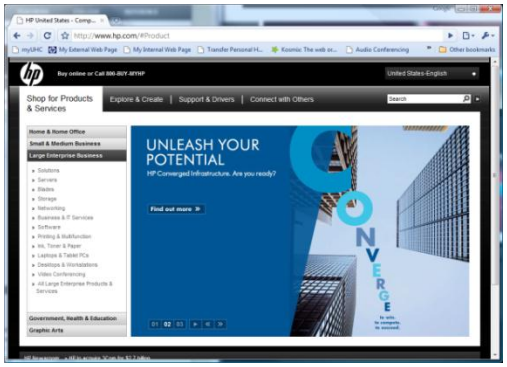
May versus Can

- Permission analysis tells you what may happen.
- Authority analysis tells you what can happen.



index.html
(Apache,R; admin,RW)

Apache



Random User

- Permission analysis: Put secrets on home page
- Authority analysis tells why you shouldn't

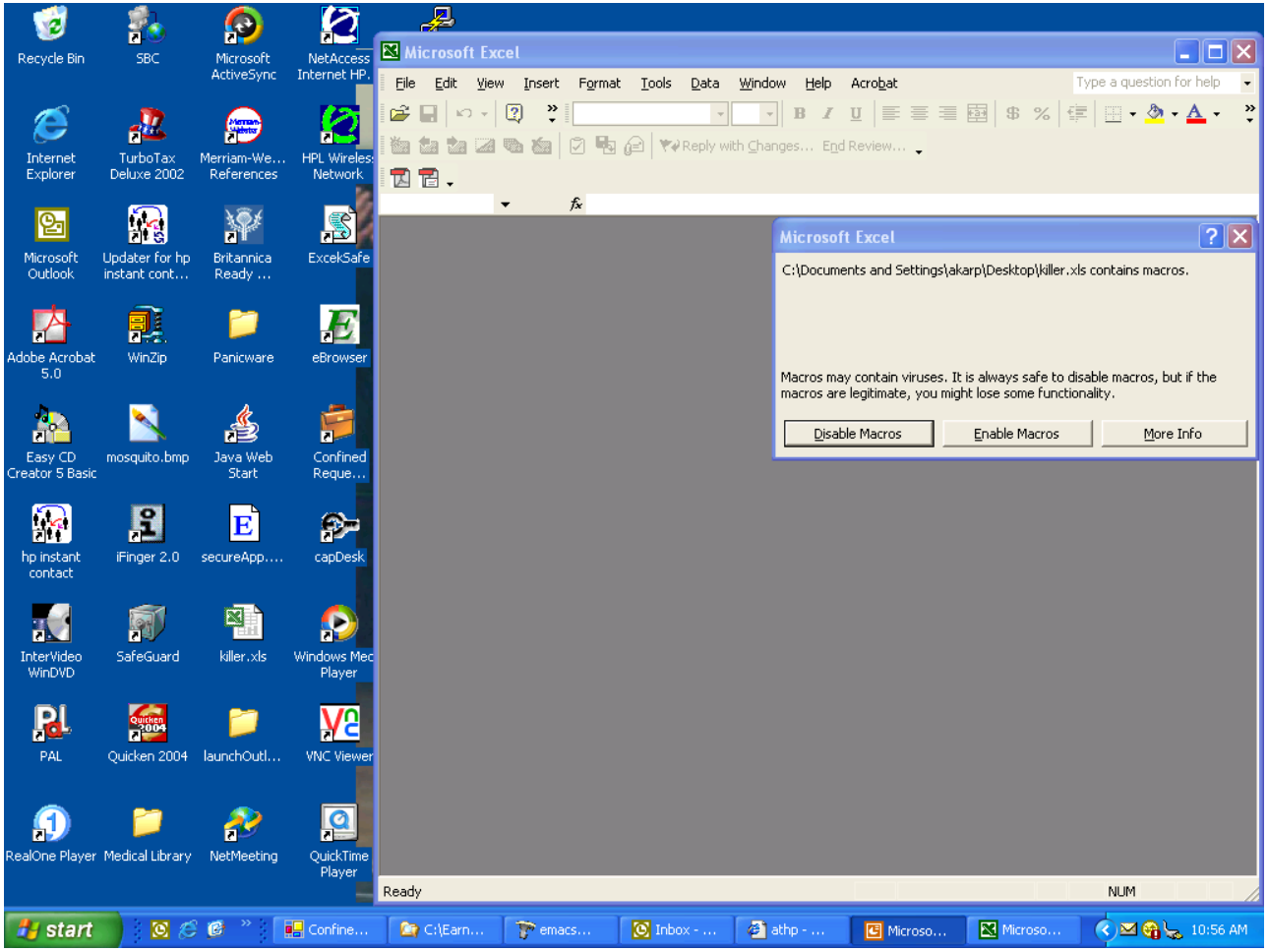
Back on the Main Road

Security in knowledge

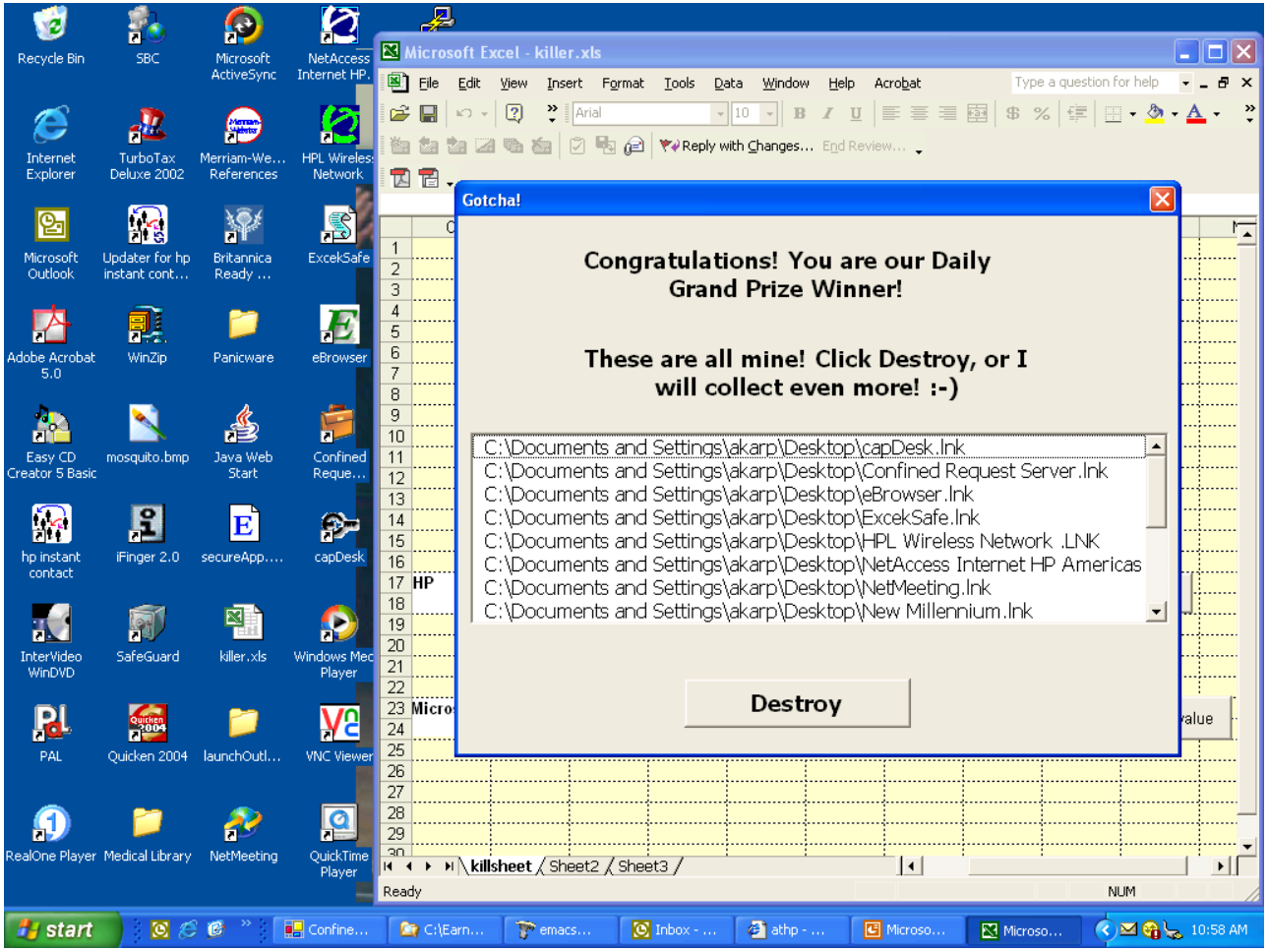
Killer App

- Wonderful spreadsheet
  - Important calculation
  - May have a virus
- Choice today
  - Turn off macros – useless
  - Turn on macros – risk my machine
- POLA approach
  - Leave macros on
  - Virus can do no harm I care about

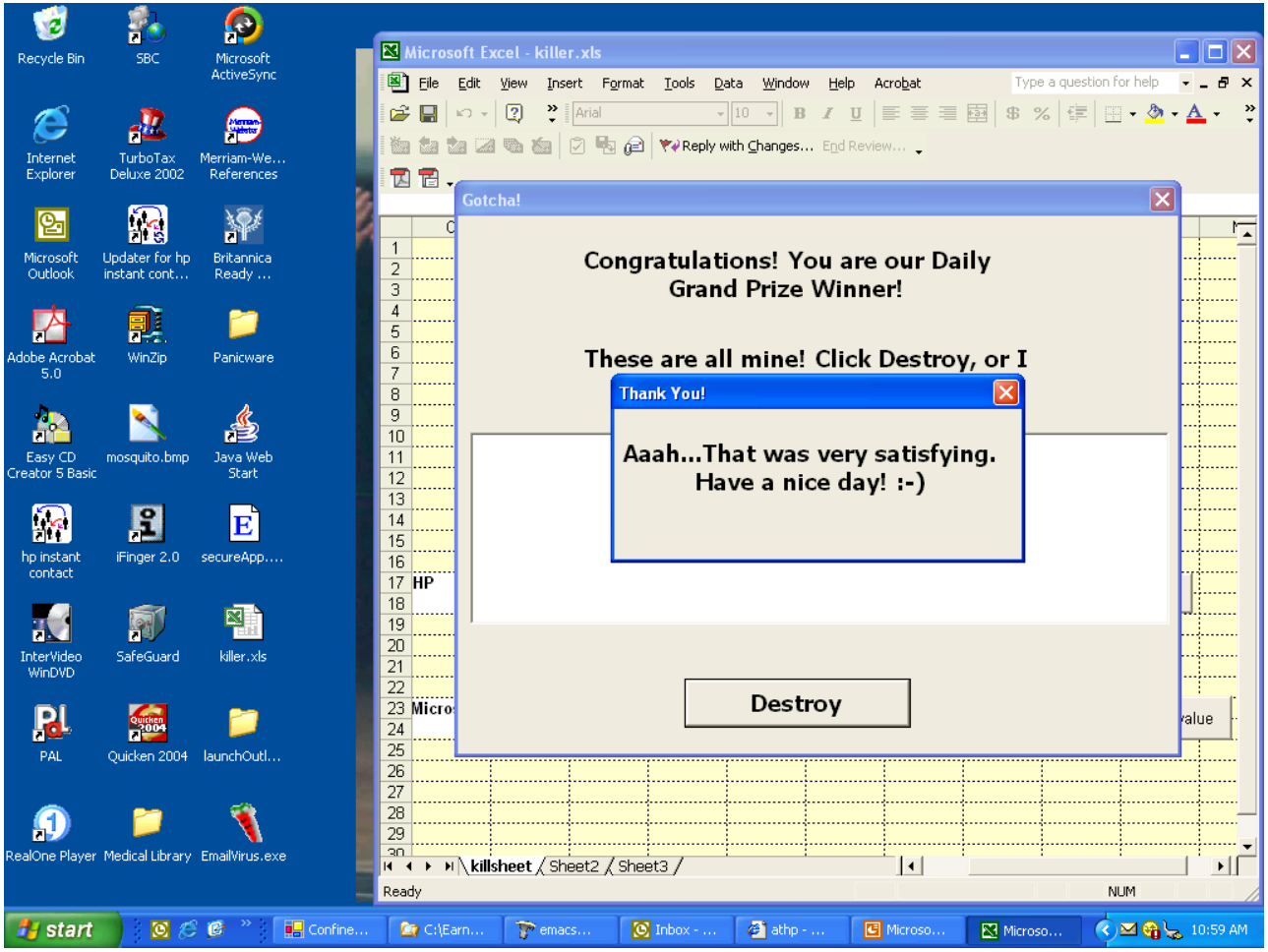
Current Approach



Where's my paddle?



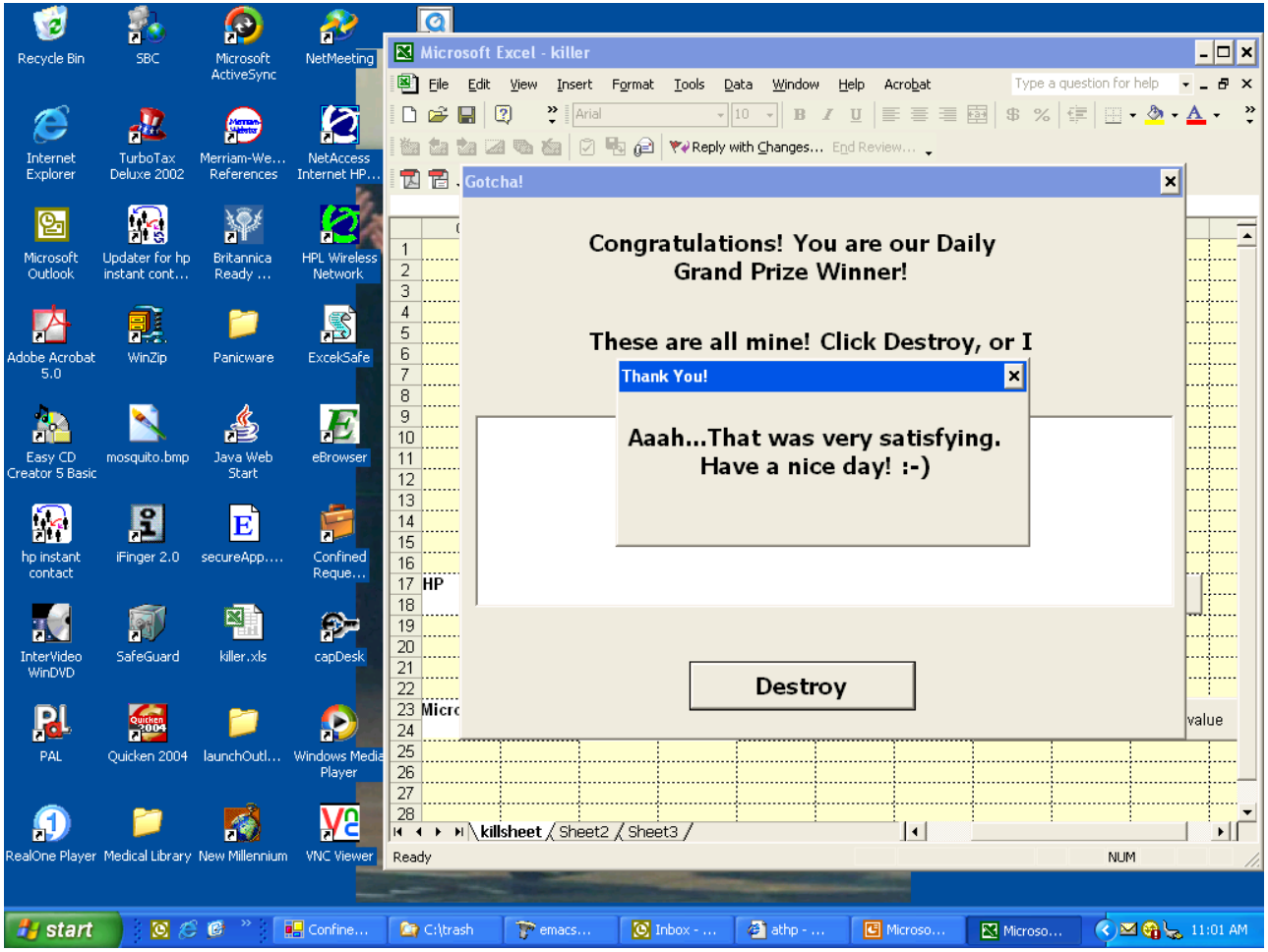
It Ate My Desktop



RSACONFERENCE2013

POLA at Application Granularity

Can’t Hurt Anything I Care About



Polaris Seems Magical

- No change to operating system
- No change to application
- Sandboxed only with standard Windows API
- No need to run in a VM
- No need to intercept system calls

- Use runAs to launch app in a restricted user account
- Write some code to enable SaveAs, etc.

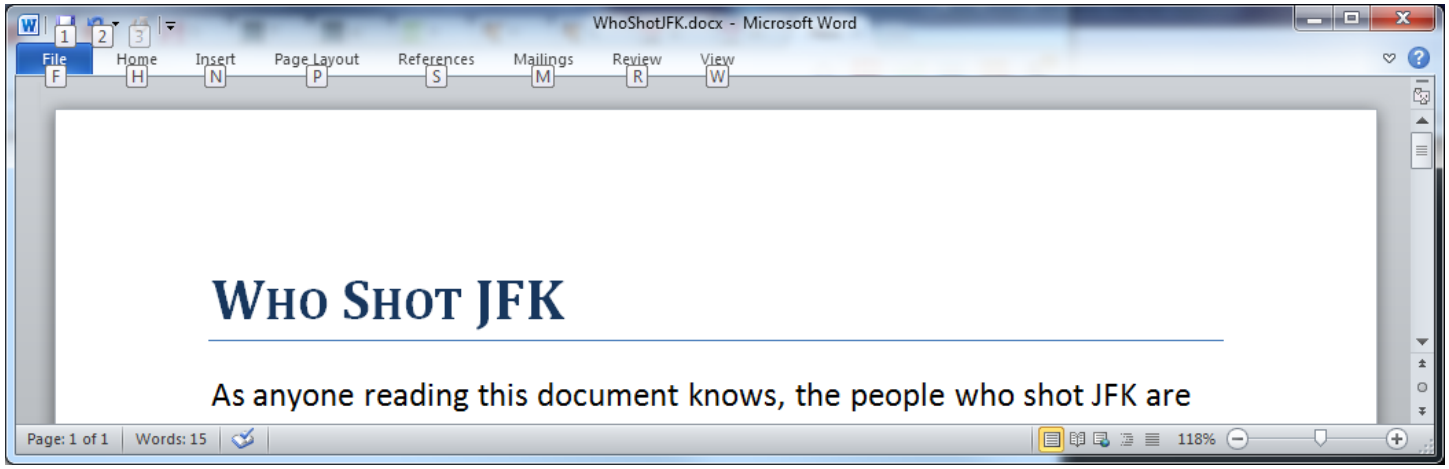
Caveat

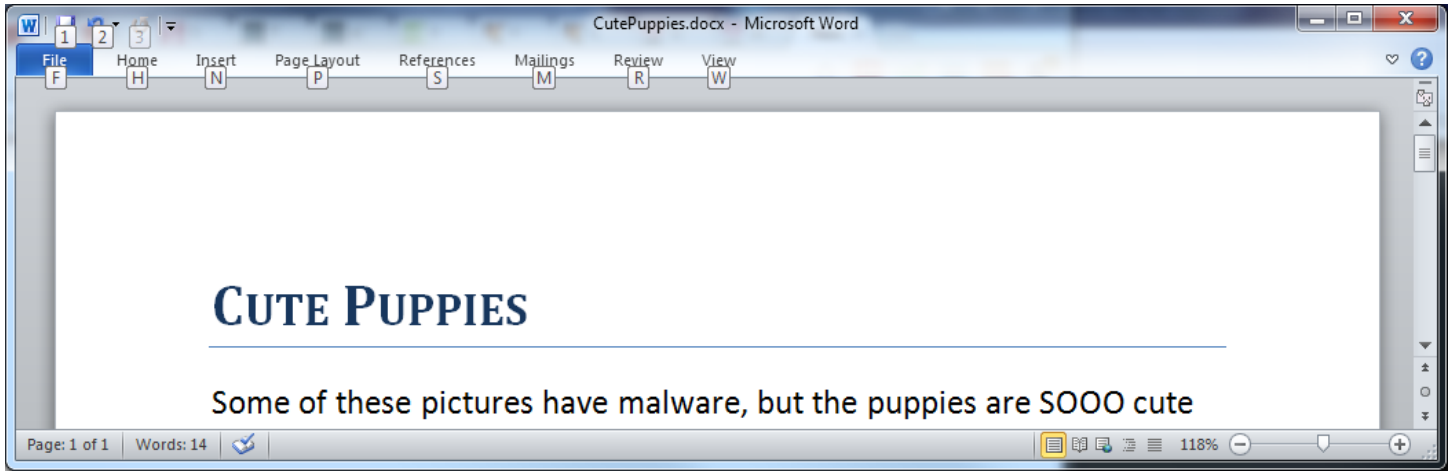
- COM communications hole required special handling

A Lot of Protection

- All of the 75% of Microsoft patches
  - Zero day attacks against Office
  - Drive-by downloads in IE
- Other vulnerabilities
  - Adobe Reader
  - RealPlayer
  - QuickTime
  - Malicious email, including malware attachments

The Problem with Polaris





POLA for Application Instances

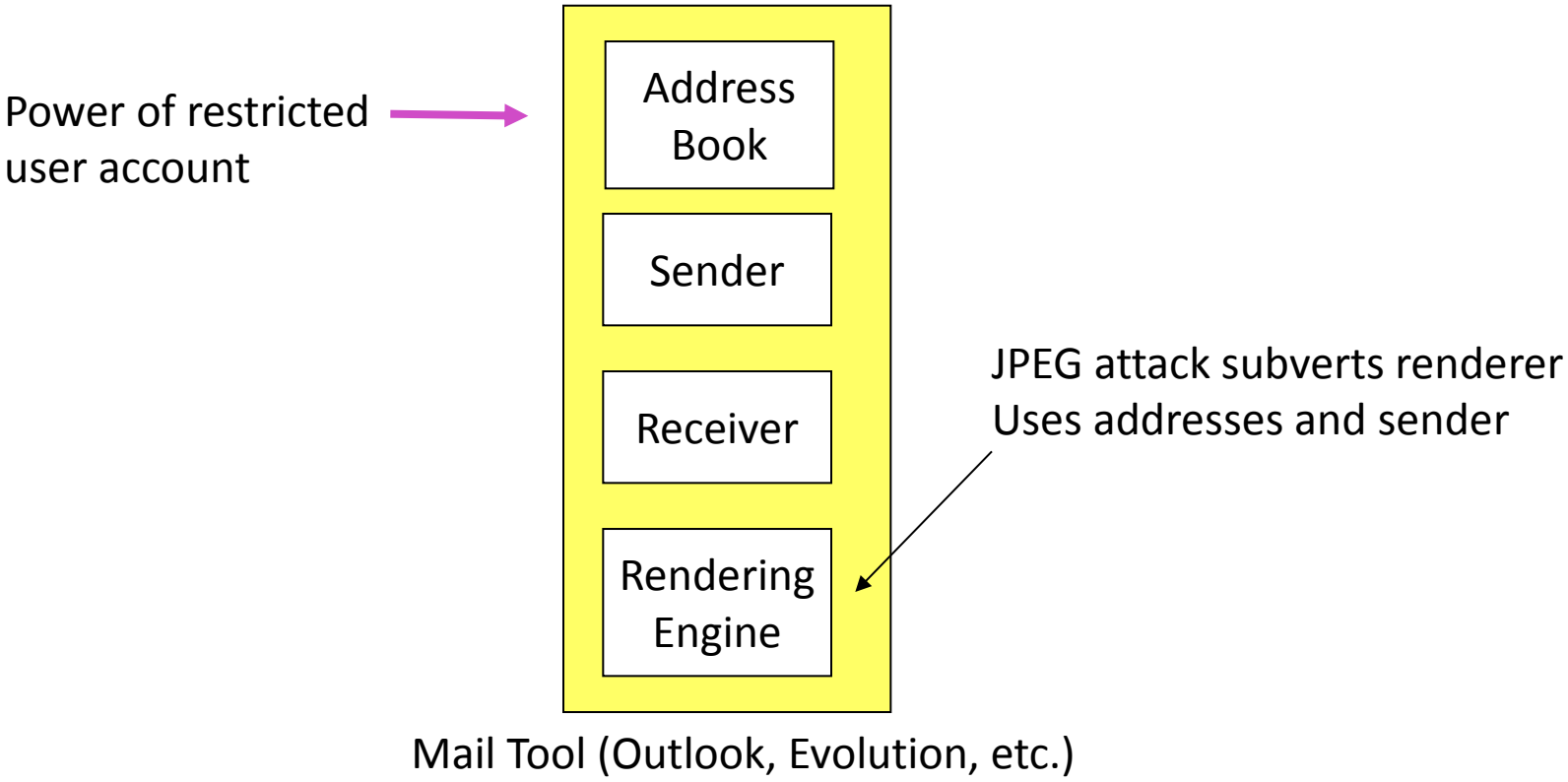
Security in knowledge

The Solution

- Run each instance in a different account
- Surprisingly hard
  - Creating accounts is slow
  - Common operations fail
  - Clipboard is a security problem
  - Apps don’t all obey account boundaries (e.g., Firefox)
- Probably need help from software vendors

The Problem with App Instances

Any Breach == Full Breach

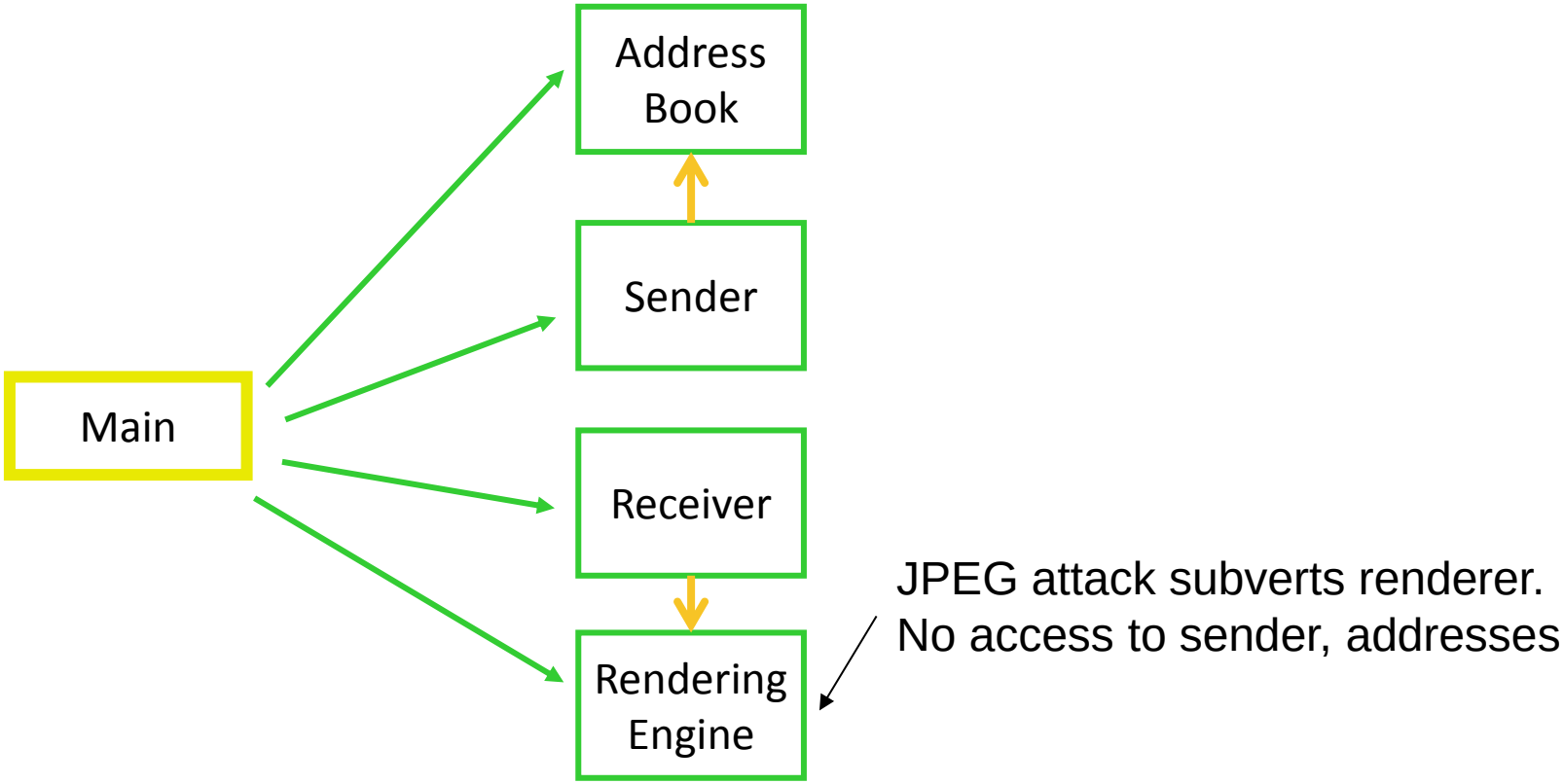


POLA for Modules

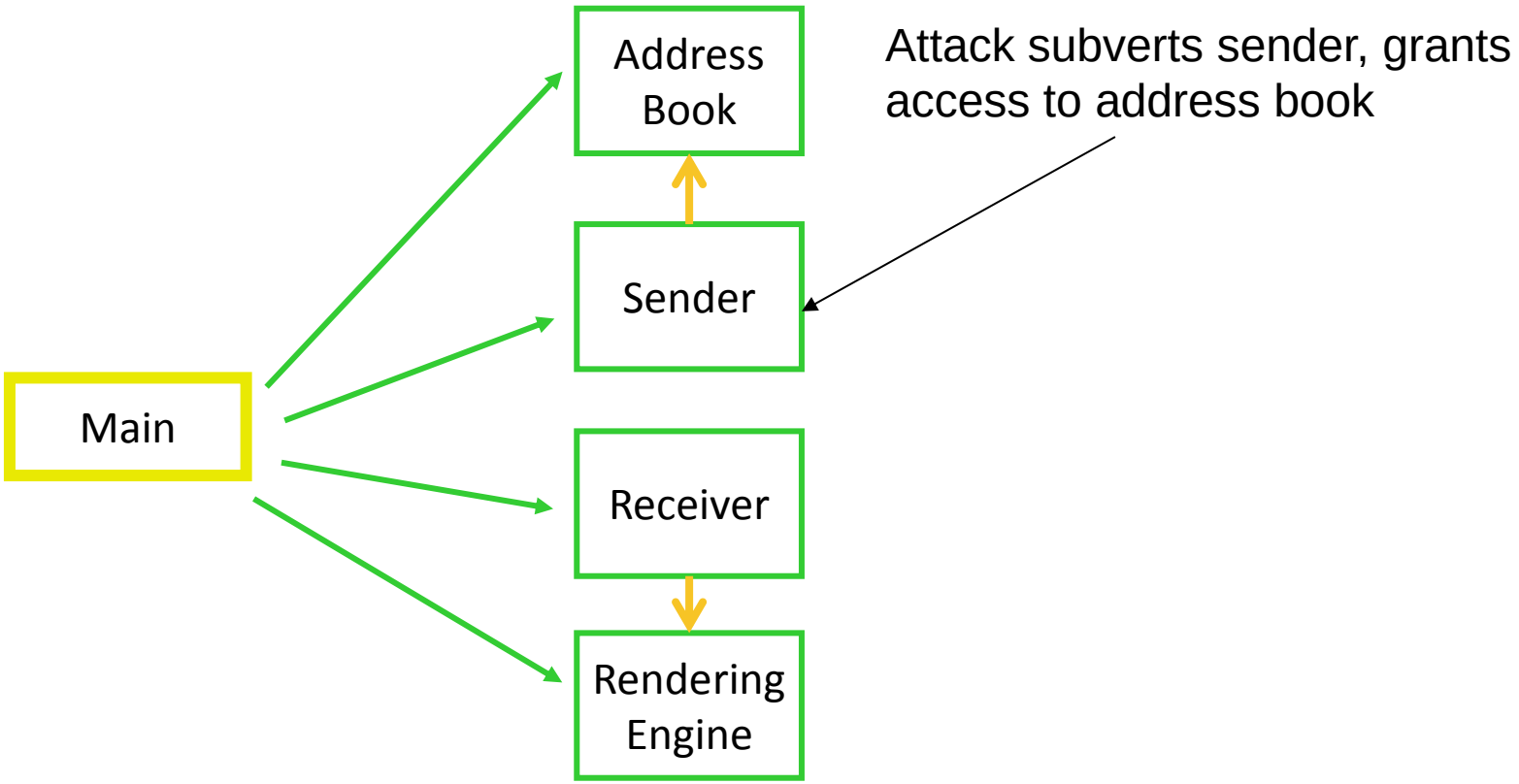
Security in knowledge

Virus versus POLA Client

Modularize *Authority*, not just *Code*



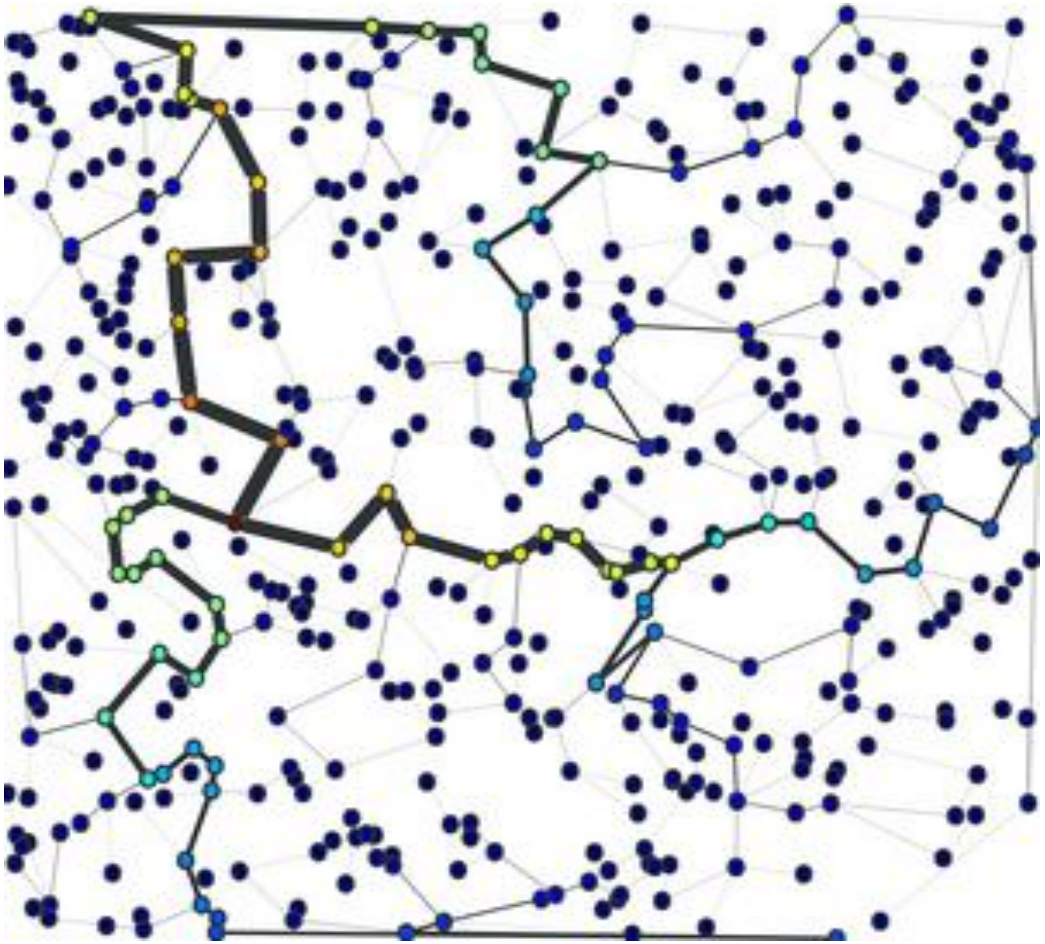
The Problem with Modules



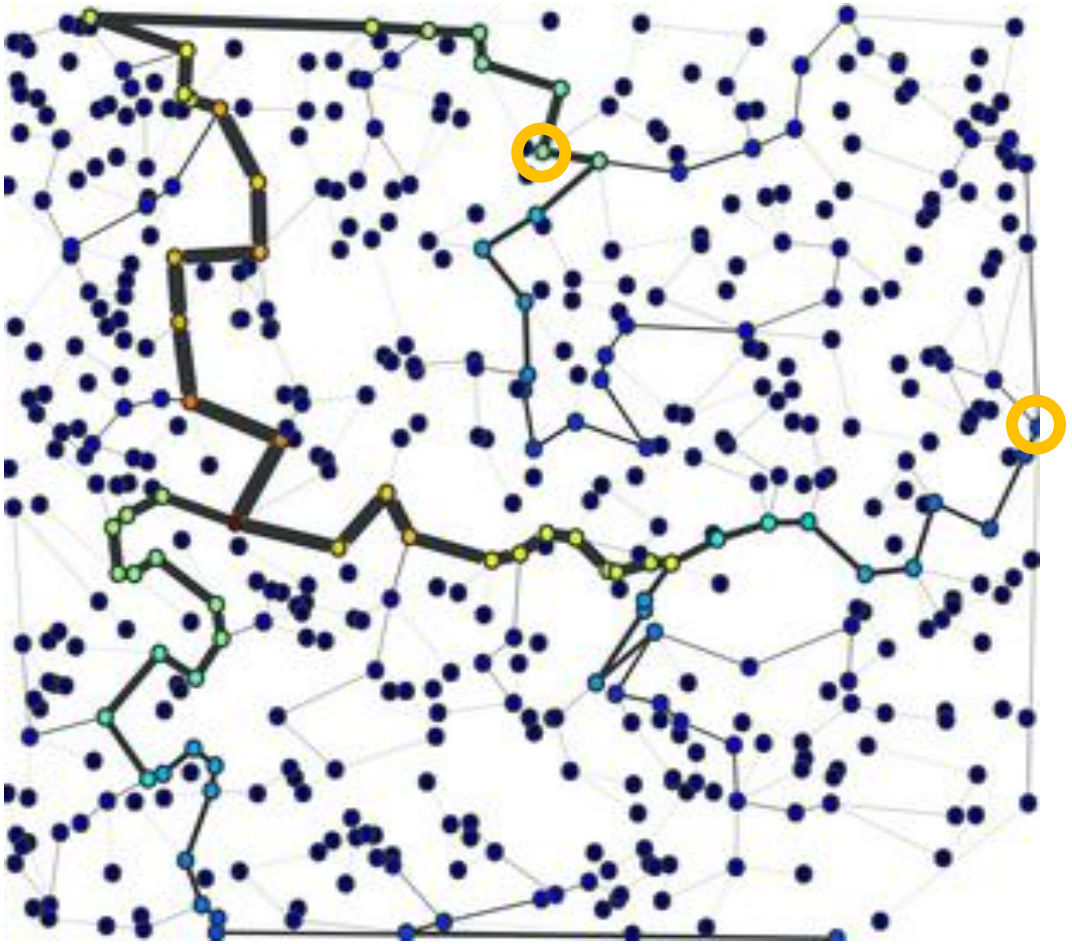
POLA for Objects

Security in knowledge

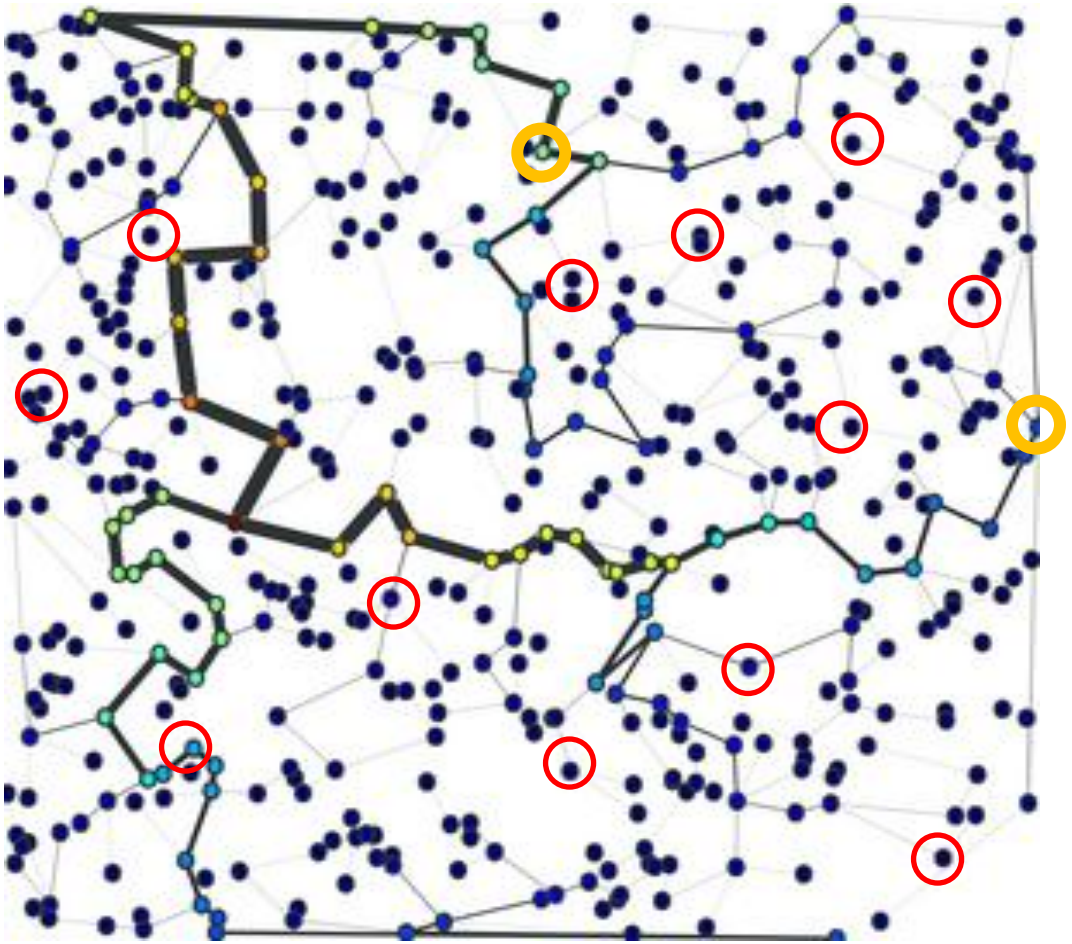
Object Graph of Sender



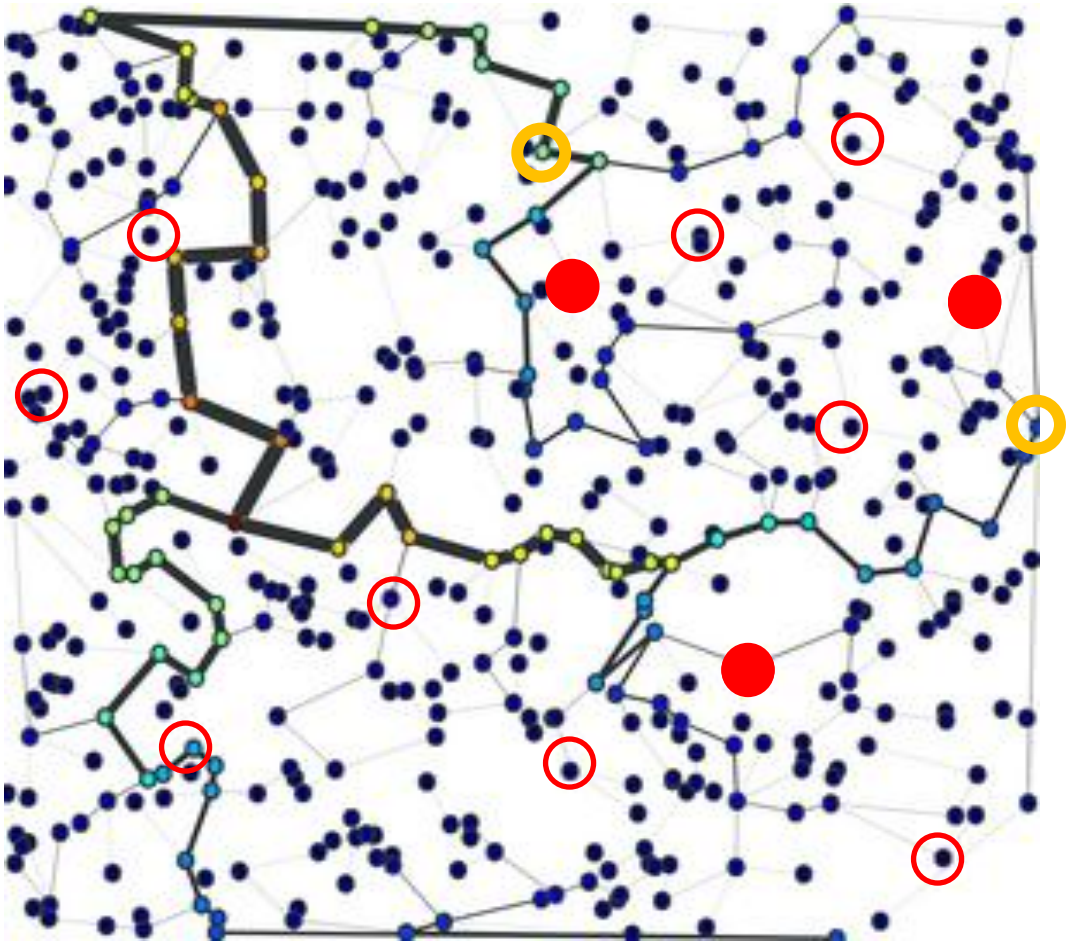
Critical Objects



Vulnerable Objects



Exploited Objects



Revised Economic Analysis

- Defender's cost
  - 1,000,000 line program
  - 1 exploitable bug/10,000 lines
  - 100 hour/bug
  - 10,000 hours
- Attacker's cost
  - 1,000 hours/bug
  - Need to exploit k bugs
  - Not an arbitrary k, cost $\alpha \binom{n}{k} \alpha (1{,}000^{k})$
- Defender's cost/Attacker's cost << 1

(Don't take math too seriously. It says you are safer with more bugs, so only applies a small percentage of objects.)

Code Examples

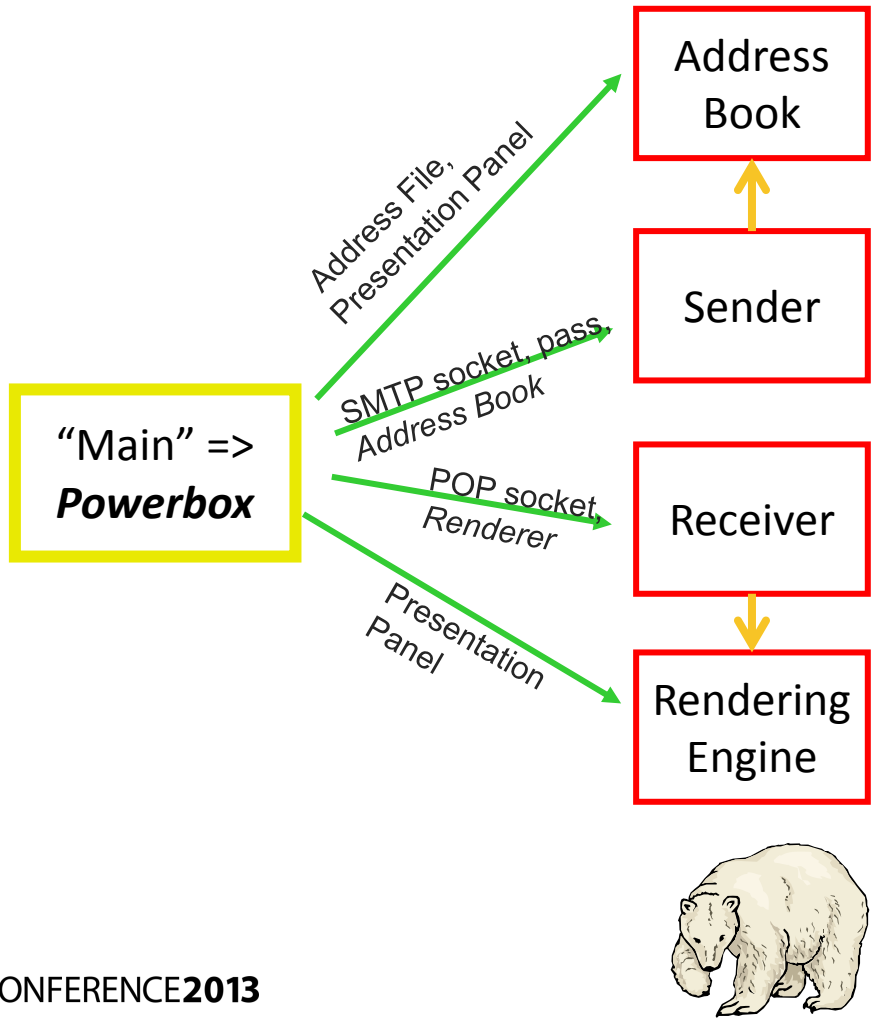
Security in knowledge

Through the (Cost) Looking Glass

- Authority Modularization: How to quantify cost/rewards?
- Security Review: lower cost, equal quality
- Currently, every line of code needs review
  - java.io.File passFile = new java.io.File(“password”);
- Basic Principle: Objects/Modules without strong powers do not need review (Defend Calais, not Brittany)
- If only 2 in 10 modules have risky powers, reduce review cost by 80%

POLArized Modules

Strict Isolation + Explicit Delegation of Least Powers == Authority Modularization



3 Threats:

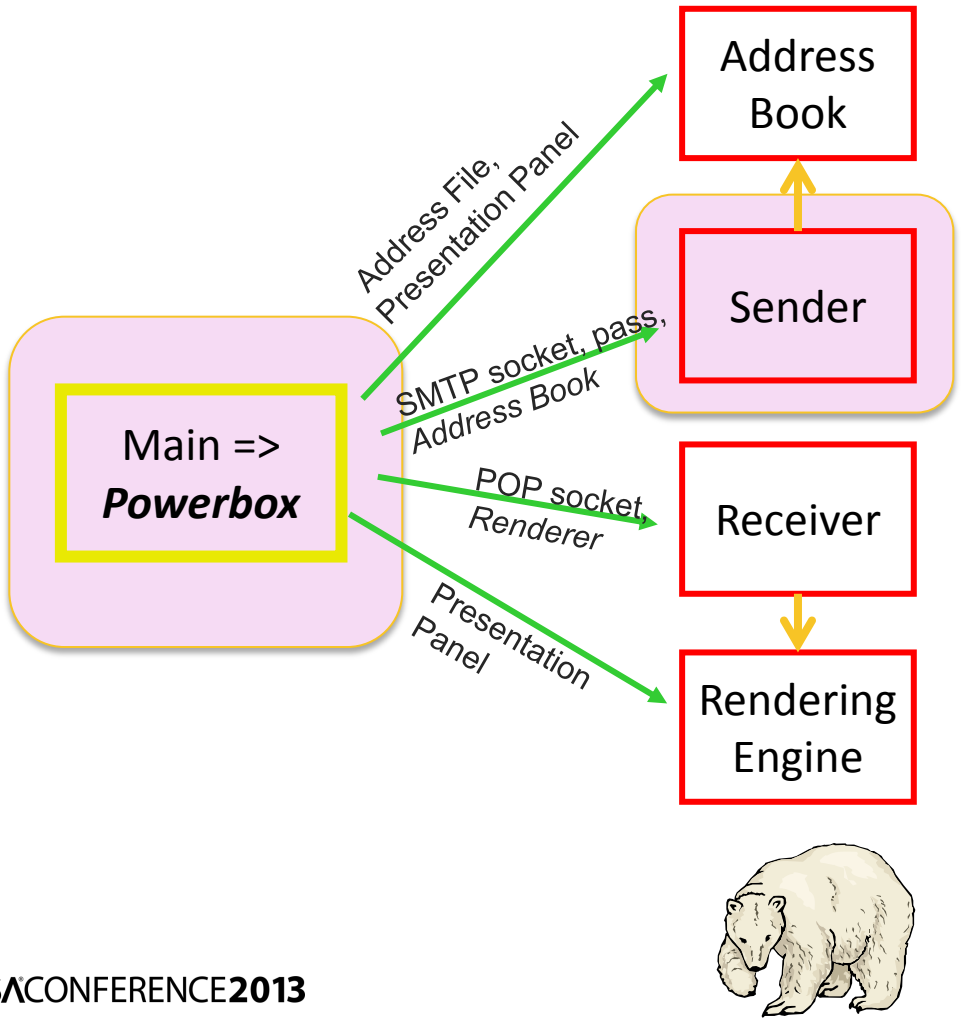
- SpamBot
  - Address Book
  - Sender
- Private Data Theft
  - Receiver
  - Sender
- SMTP Password Theft

1 Special Vulnerability:
Rendering Engine

What Modules Need Review?

POLArized Modules

Strict Isolation + Explicit Delegation of Least Powers



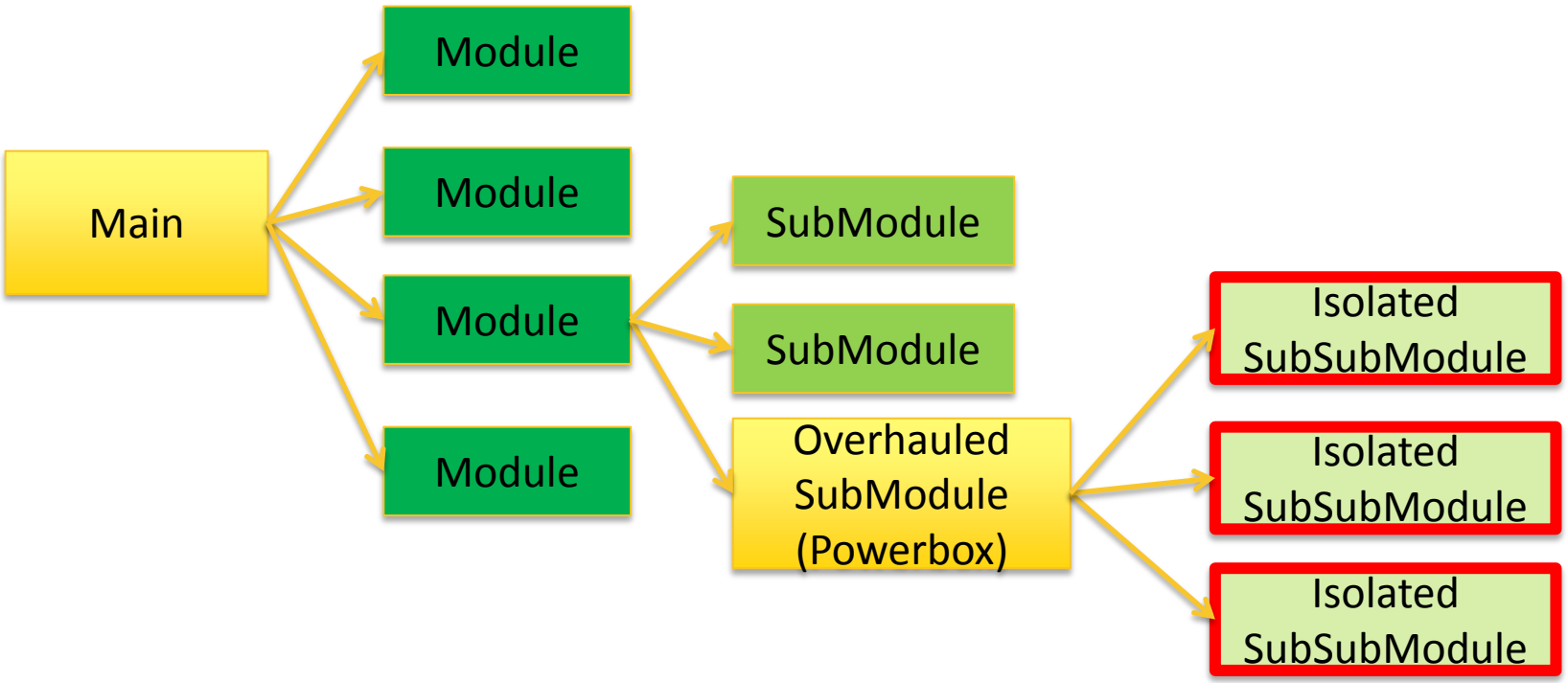
3 Threats:

- SpamBot
  - Address Book
  - Sender
- Private Data Theft
  - Receiver
  - Sender
- SMTP Password Theft

1 Special Vulnerability:
Rendering Engine

Powerbox

A reusable pattern at many coding levels
Enables incremental retrofit of legacy apps, submodule by submodule

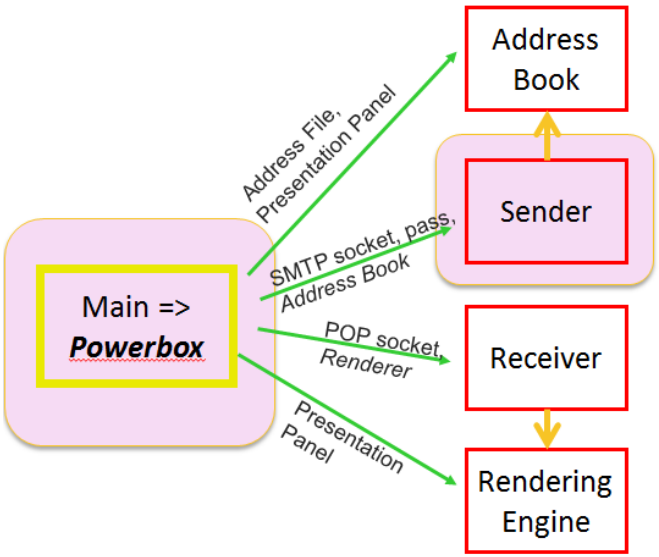


POLArized Modules Part 2

Security in knowledge

Must We Review the Sender?

- Small code change, big review payoff?
- Must have the send authority: this is its purpose!
- Does it need the smtp password? If so, must review in detail
- Does it need full access to the address book? If so, must review in detail
- If we can eliminate password and limit address book, no review needed



Encapsulate Password in SMTPLogin

```
package com.hp.rsa2013.email;
import java.net.Socket;
public class SMTPLogin {
    private String password;
    private Socket socket;
    public SMTPLogin(String password, Socket socket) {
        this.password = password;
        this.socket = socket;
    }
    public Socket login() {
        ... open socket,
        ...use password,
        return socket;
    }
}
```

Sender no longer has access to password

```
package com.hp.rsa2013.email;
public class Sender {
    private SMTPLogin login;
    public Sender(AddressBook book, SMTPLogin login) { ... }
    public void send(Friend addressee, String subject, String body) { ... }
}
```

Protect Address Book Behind *Facet*

```
package com.hp.rsa2013.email;
import java.io.File;
import java.awt.Panel;
public class AddressBook {
    public AddressBook(File addressFile, Panel panel) { ... }
    public Friend[] addressees() { ... return addressees;}
    public void addFriend(Friend newPerson) { ...}
    public void removeFriend(Friend oldPerson) { ... }
    public Friend selectRecipient() {
        ... use panel to ask user for friend ...
        return selectedFriend;
    }
}
```

Grant Violates Threat Model

Grant Required for Operation

Least Privilege Grant

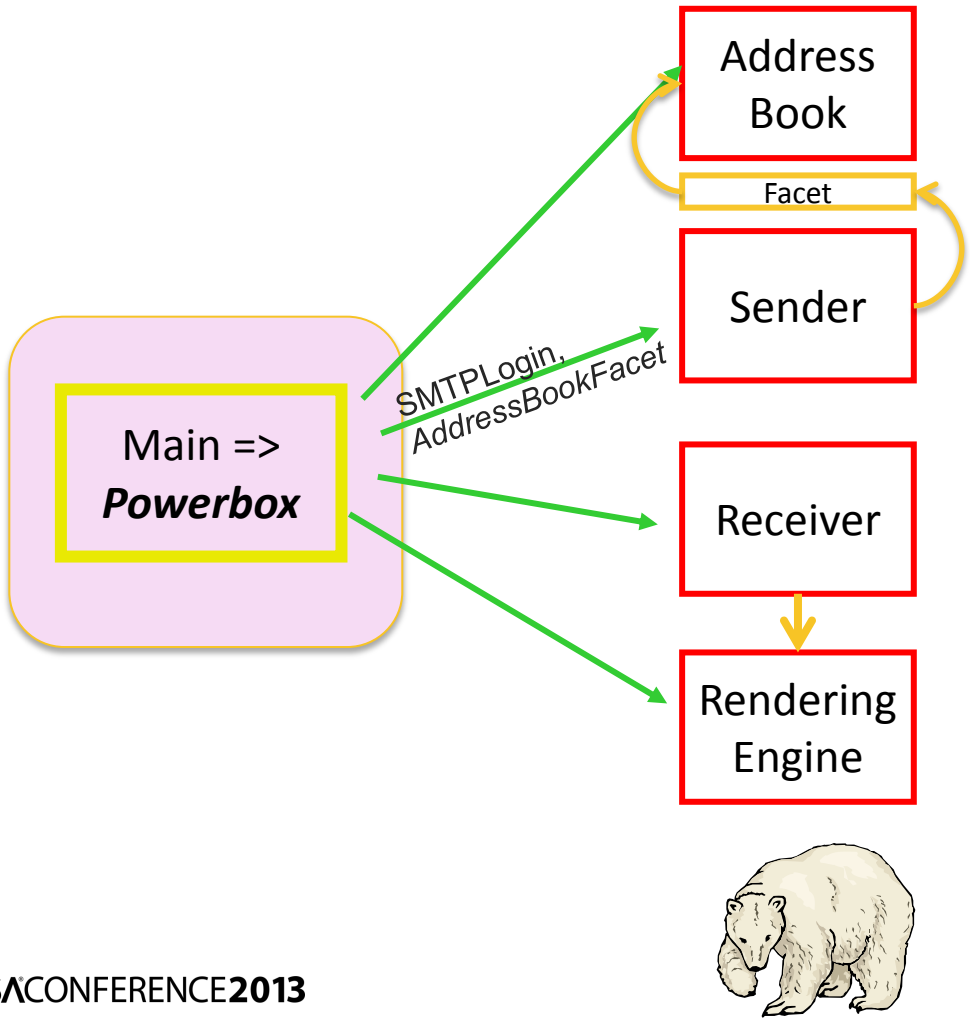
```
package com.hp.rsa2013.email;
class AddressBookFacet {
    private AddressBook book;
    public AddressBookFacet(AddressBook book) {this.book = book;}
    public Friend selectRecipient() {return book.selectRecipient();}
}
```

POLA-rized Sender

```
package com.hp.rsa2013.email;
public class Sender {
    private SMTPLogin login;
    private AddressBookFacet bookFacet;
    public Sender(AddressBookFacet bookFacet, SMTPLogin login) { ... }
    public void send(Friend addressee, String subject, String body) { ... }
}
```

Object-POLArized Modules

Provide Authority-Limited Arguments to Sender
Achieve Closer Approximation to Perfect Least Privilege



Result: Simple Architecture Analysis Demonstrates Only "Main" Module Has Risk

Sounds Good. Strict Isolation?!

- Java Protection Domains useless
  - No delegation: new File creation indistinguishable from explicitly granted File authority, disallows both or neither
  - Yet another complicated, confusing mechanism outside the flow of program operation.
  - Tenuous relationship to POLA: No Control on Address Book
- 2 Solutions to verify object isolation:
  - Joe-E Verifier
    - Adrian Mettler/David Wagner at UCB
  - Coding Standards to support Visual Inspection
  - Understand basic rules by looking at simple violations

Breaking All the Rules

Code that makes Visual Verification Too Hard: Bad Class Sender

Unneeded Powerful Import: Why does Sender need this?

```
package com.hp.rsa2013.email;
import java.io.File;
import java.net.*;
public class Sender {
    public static Sender publicSender;
    String smtpPassword;
    public Sender(AddressBook book, Receiver rcvr) { ... }
    public AddressBook book() {return book;}
    public void send(String addressee, String subject, String body) { ... }
    public void init() {File passwordFile = new java.io.File("password.txt"); ...
}
```

Excess Privilege Package Import, must inspect for Socket, URL, etc.

Mutable Ambient Authority, Powerful Ambient Authority, isolation broken for rest of system

Authority String, Hard to track on way to accidental exposure

Unneeded Privilege Required, easily avoided with mere laziness

Excess Power Grant To Sender Clients

Backdoor Access to Powerful Authority, breaks isolation

Inline powerful authority creation, requires line-by-line scrutiny to detect isolation break

Basic Java Coding Standards

- Rules
  - Explicitly list each imported class in each source header
  - Only powerboxes create new java.io.File, java.net.URL, java.net.Socket, etc.
  - Only powerboxes use java.lang.Runtime.exec, etc.
  - Files, sockets, etc., explicitly granted as object references
  - No powerful or mutable statics
  - No strings carrying authority (encapsulate immediately)
  - Powerbox architecture
- Reviews:
  - Checkin: Quick checkin scan confirm isolation, coding standards
  - Security review only of threat-model-risk classes

Revised Economic Analysis II

- Defender's cost
  - 1,000,000 line program
  - 1 exploitable bug/10,000 line *module*
  - 2 powerful modules requiring review per 10 modules
  - 100 hour/bug
  - 2,000 hours (not 10,000)
- Attacker's cost
  - 1,000 hours/bug
  - Need to exploit k bugs
  - Not an arbitrary k, cost $\alpha \binom{n}{k} \alpha (1000^{k})$
- Defender's cost/Attacker's cost << 1

(Don't take math too seriously!)

The Secret Sauce

- OO design taken seriously
  - Which is better?
    - public void setFile(String path) {this.file = new File(path);}
    - public void setFile(File file) {this.file = file;}
  - The *preferred* OO choice is the *crucially required* securely isolated, authority-modularizing choice
  - Authority Modularization == OO modularization *…on steroids*
- Strong security *properties:* inexpensive lunch (TANSTAAFL)
- Strong security *policy* is *still hard*. But it should not be impossible.

Examples of Where It Works

We Can't Find Any

- Few widely-used applications follow the rules
  - C/C++ so not memory safe
  - Java but use mutable global state
  - One or two hops between any pair of objects
- One possibility – Cajoled apps
  - Rewritten by Google's Caja to a "safe" javascript
  - Widgets on a page isolated by virtualizing global "this"
  - Rules in the Secret Sauce enforced
- Caja vulnerability list
  - Examined ~200 entries
  - All were against the runtime platform (TCB)
  - None were against cajoled apps

Conclusions

Security in knowledge

Finer-grained POLA is Safer

Granularity of POLA	Example
Machine	DOS, Windows XP
User	Windows Vista UAC, MacOS, Linux
Application	Polaris, Android, MacOS Lion
Application Instance	Bromium
Module	Chrome Browser, Mashups with ES 5
Object	Waterken, CapDesk

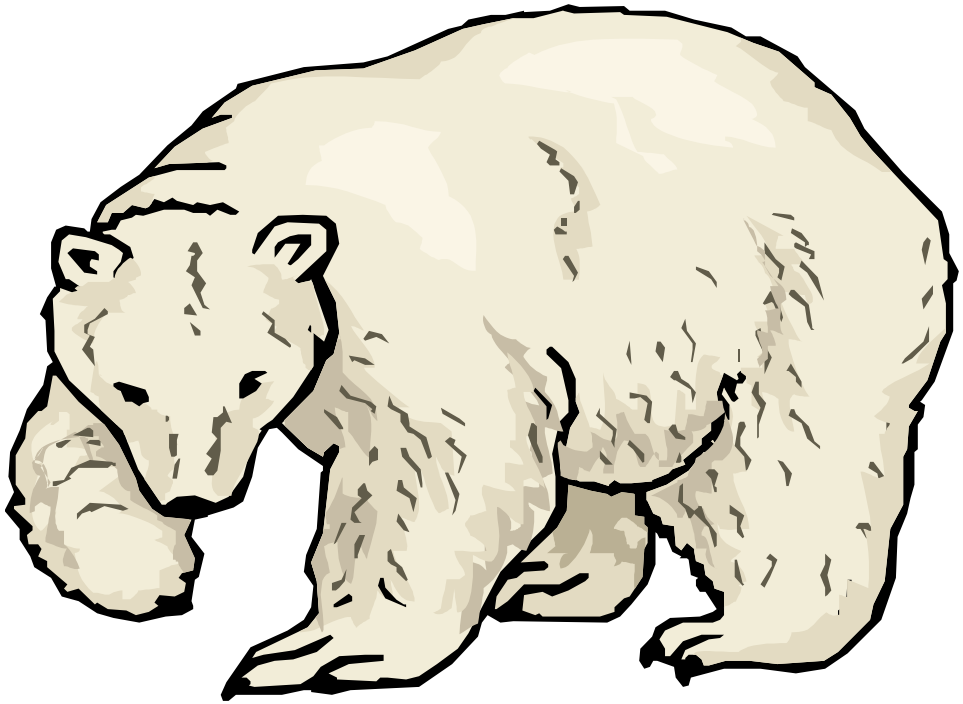
Take Homes

- Immediately, for Java Applications
  - Coding Standard Upgrade as described earlier
  - Checkin Review procedure as described earlier
  - New and overhauled subsystems, powerbox architecture:
    - no whole system rewrite required to start benefitting
  - Investigate Joe-E automated isolation verifier
- Immediately for JavaScript Applications
  - Use Caja to ensure isolation for new code at checkin
  - Or use EcmaScript 5 and “use strict” and visual verification
- Both Java and JavaScript
  - For isolation-verified subsystems
    - Security review only threat-model-risk components

What's With the Bear?

Security in knowledge

The POLA Bear



Questions

http://www.hpl.hp.com/personal/Alan_Karp/