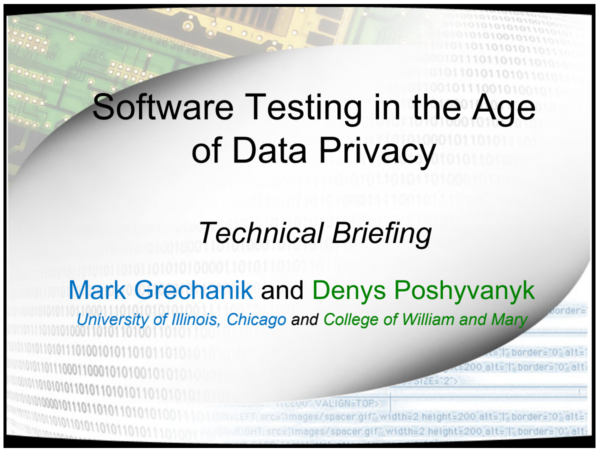
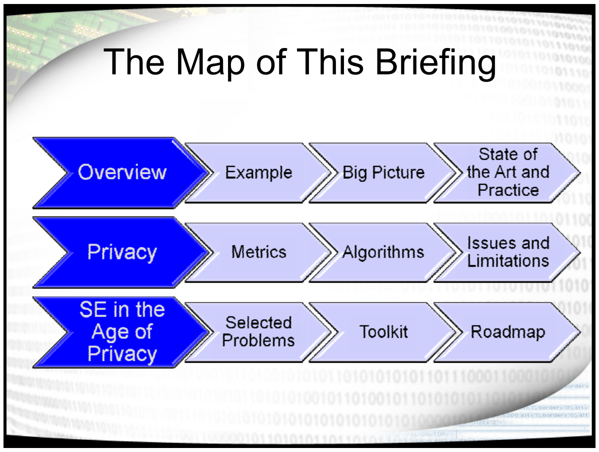


Software Testing in the Age of Data Privacy

Technical Briefing

Mark Grechanik and Denys Poshyvanyk
University of Illinois, Chicago and *College of William and Mary*

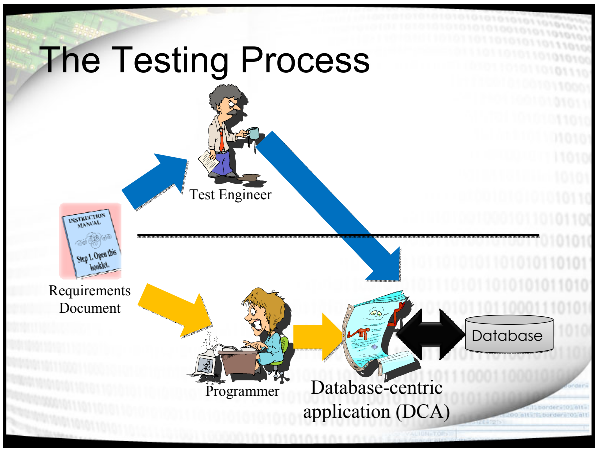


The Map of This Briefing

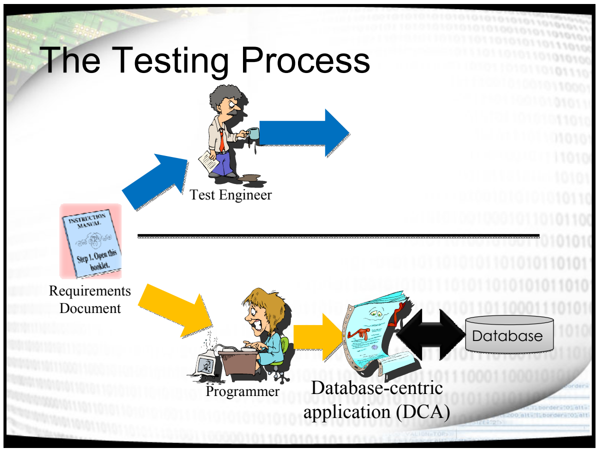
Overview → Example → Big Picture → State of the Art and Practice

Privacy → Metrics → Algorithms → Issues and Limitations

SE in the Age of Privacy → Selected Problems → Toolkit → Roadmap



The Testing Process



The Testing Process

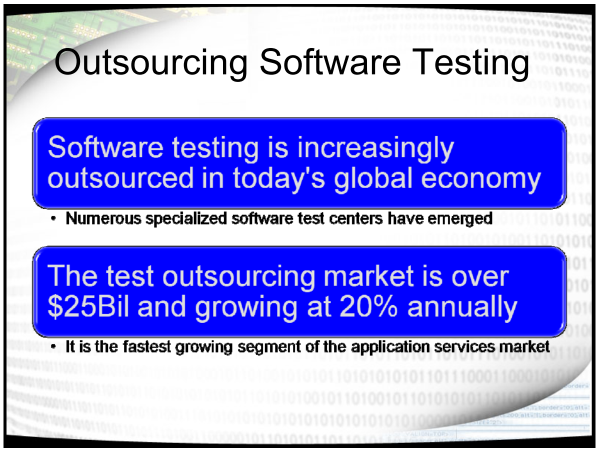
Test Engineer

Requirements Document

Programmer

Database-centric application (DCA)

Database



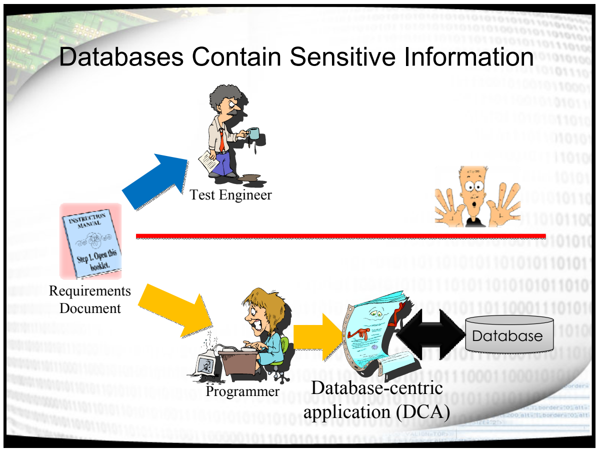
Outsourcing Software Testing

Software testing is increasingly outsourced in today's global economy

- Numerous specialized software test centers have emerged

The test outsourcing market is over $25Bil and growing at 20% annually

- It is the fastest growing segment of the application services market



Databases Contain Sensitive Information

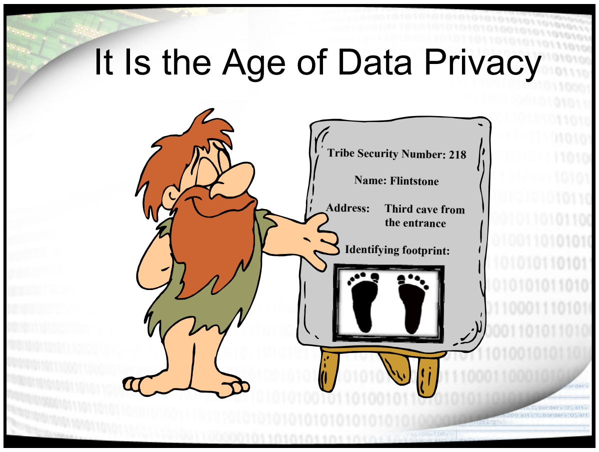
Test Engineer

Requirements Document

Programmer

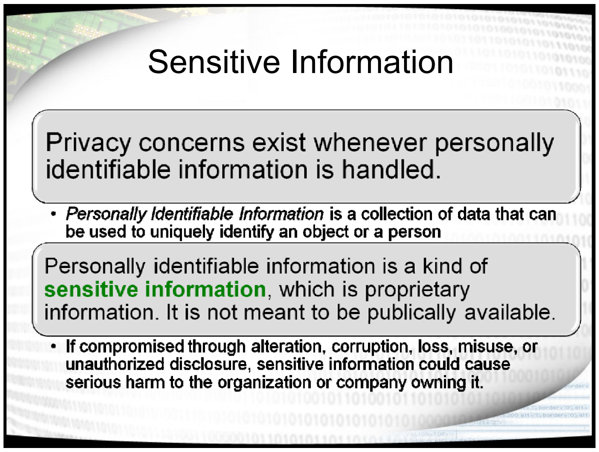
Database-centric application (DCA)

Database



It Is the Age of Data Privacy

Tribe Security Number: 218
Name: Flintstone
Address: Third cave from the entrance
Identifying footprint:



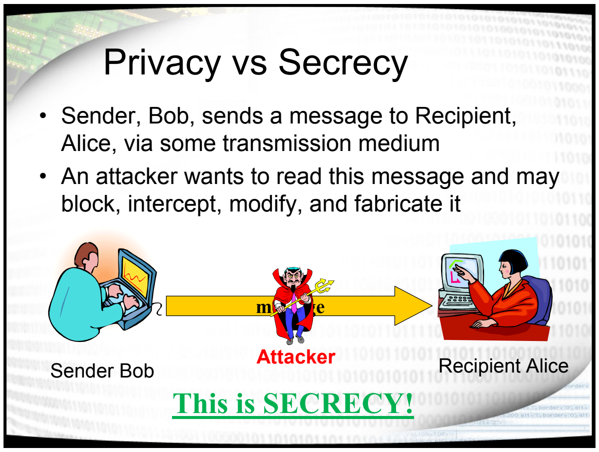
Sensitive Information

Privacy concerns exist whenever personally identifiable information is handled.

- *Personally Identifiable Information* is a collection of data that can be used to uniquely identify an object or a person

Personally identifiable information is a kind of **sensitive information**, which is proprietary information. It is not meant to be publically available.

- If compromised through alteration, corruption, loss, misuse, or unauthorized disclosure, sensitive information could cause serious harm to the organization or company owning it.



Privacy vs Secrecy

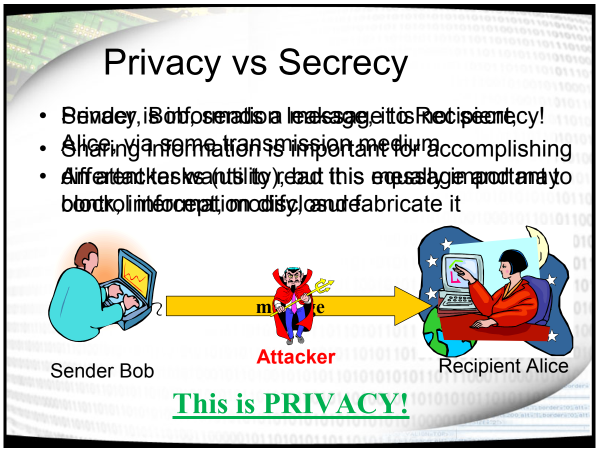
- Sender, Bob, sends a message to Recipient, Alice, via some transmission medium
- An attacker wants to read this message and may block, intercept, modify, and fabricate it

Sender Bob

Attacker

Recipient Alice

This is SECRECY!

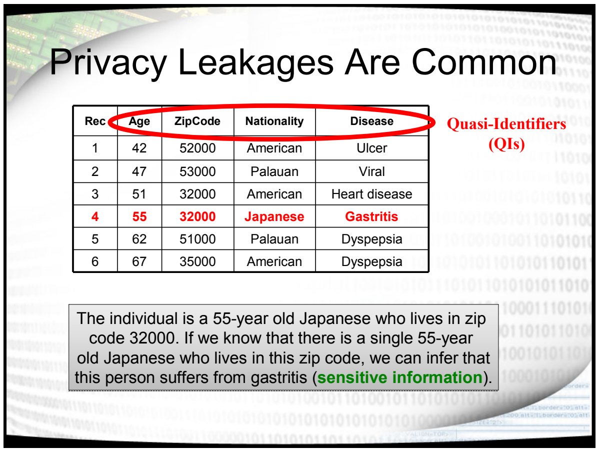


Privacy vs Secrecy

- Sender, Bob, sends a message to Recipient, Alice, via some transmission medium
- An attacker wants to read this message and may block, intercept, modify, and fabricate it

- Privacy is information leakage, it is not secrecy!
- Sharing information is important for accomplishing different tasks (utility), but it is equally important to control information disclosure

This is PRIVACY!

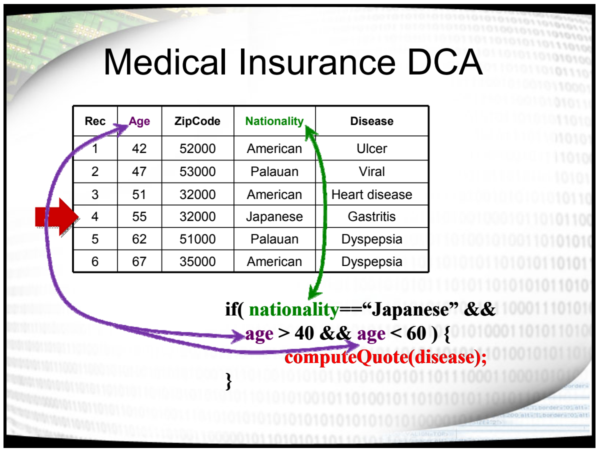


Privacy Leakages Are Common

Rec	Age	ZipCode	Nationality	Disease
1	42	52000	American	Ulcer
2	47	53000	Palauan	Viral
3	51	32000	American	Heart disease
4	55	32000	Japanese	Gastritis
5	62	51000	Palauan	Dyspepsia
6	67	35000	American	Dyspepsia

Quasi-Identifiers (QIs)

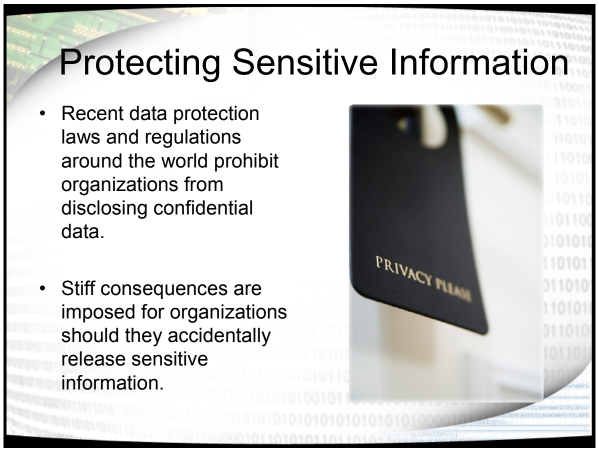
The individual is a 55-year old Japanese who lives in zip code 32000. If we know that there is a single 55-year old Japanese who lives in this zip code, we can infer that this person suffers from gastritis (**sensitive information**).



Medical Insurance DCA

Rec	Age	ZipCode	Nationality	Disease
1	42	52000	American	Ulcer
2	47	53000	Palauan	Viral
3	51	32000	American	Heart disease
4	55	32000	Japanese	Gastritis
5	62	51000	Palauan	Dyspepsia
6	67	35000	American	Dyspepsia

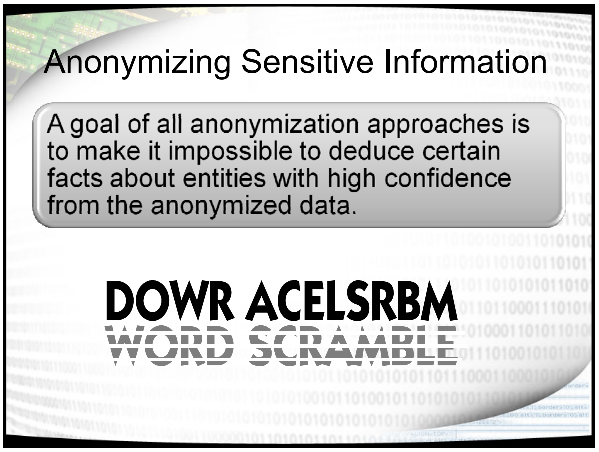
```
if( nationality=="Japanese" &&
    age > 40 && age < 60 ) {
        computeQuote(disease);
}
```



Protecting Sensitive Information

- Recent data protection laws and regulations around the world prohibit organizations from disclosing confidential data.
- Stiff consequences are imposed for organizations should they accidentally release sensitive information.

PRIVACY PLEASE

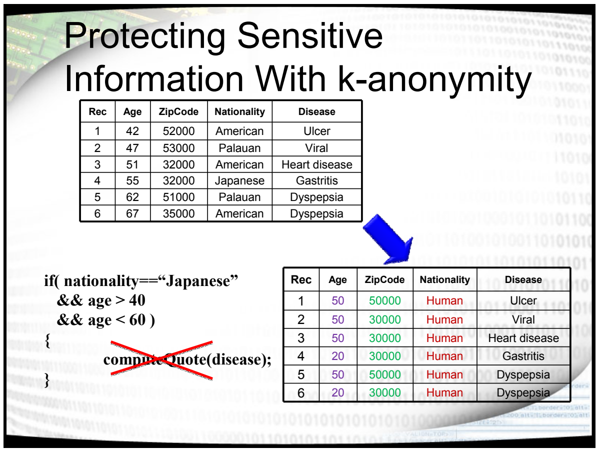


Anonymizing Sensitive Information

A goal of all anonymization approaches is to make it impossible to deduce certain facts about entities with high confidence from the anonymized data.

DOWR ACELSRBM

WORD SCRAMBLE

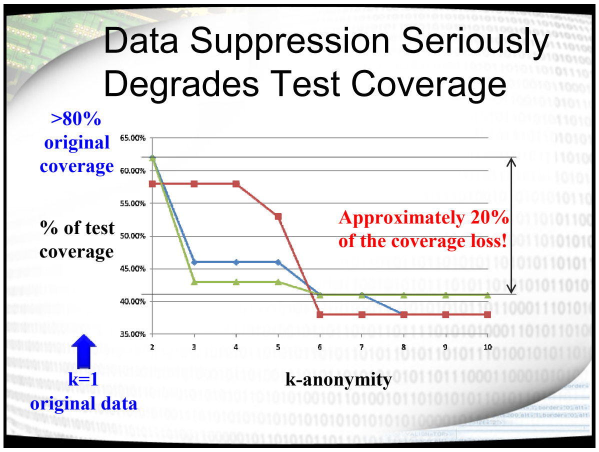


Protecting Sensitive Information With k-anonymity

Rec	Age	ZipCode	Nationality	Disease
1	42	52000	American	Ulcer
2	47	53000	Palauan	Viral
3	51	32000	American	Heart disease
4	55	32000	Japanese	Gastritis
5	62	51000	Palauan	Dyspepsia
6	67	35000	American	Dyspepsia

if(nationality==“Japanese”
&& age > 40
&& age < 60)
{
~~computeQuote(disease);~~
}

Rec	Age	ZipCode	Nationality	Disease
1	50	50000	Human	Ulcer
2	50	30000	Human	Viral
3	50	30000	Human	Heart disease
4	20	30000	Human	Gastritis
5	50	50000	Human	Dyspepsia
6	20	30000	Human	Dyspepsia



Data Suppression Seriously Degrades Test Coverage

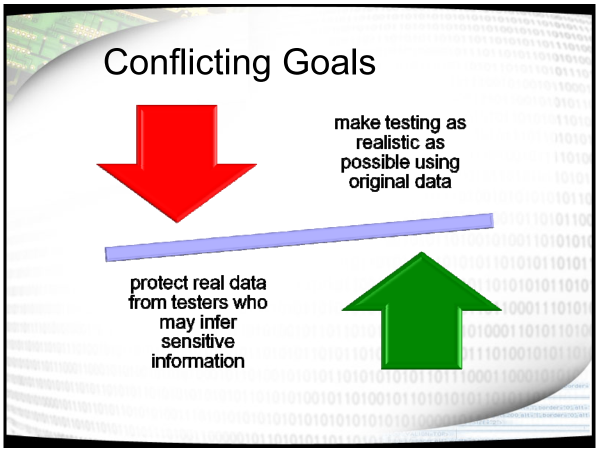
>80% original coverage

% of test coverage

Approximately 20% of the coverage loss!

k=1 original data

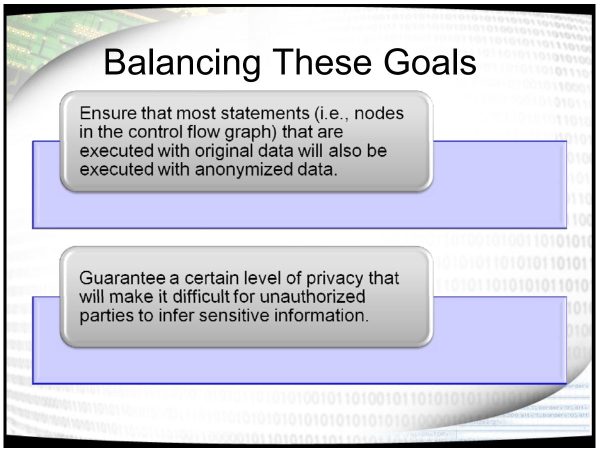
k-anonymity



Conflicting Goals

make testing as realistic as possible using original data

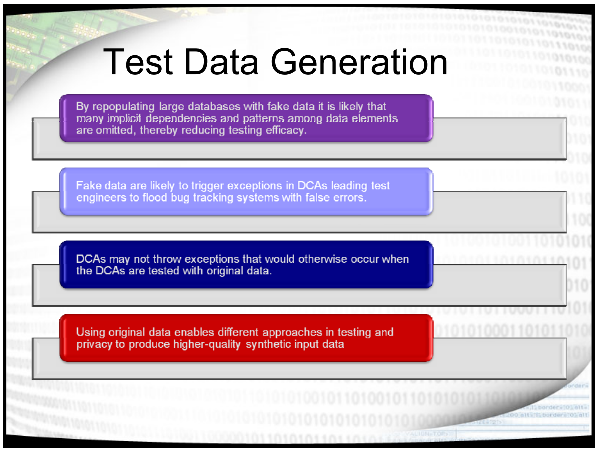
protect real data from testers who may infer sensitive information



Balancing These Goals

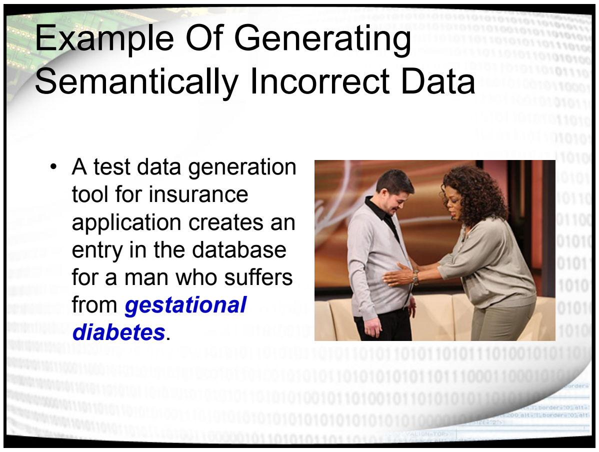
Ensure that most statements (i.e., nodes in the control flow graph) that are executed with original data will also be executed with anonymized data.

Guarantee a certain level of privacy that will make it difficult for unauthorized parties to infer sensitive information.



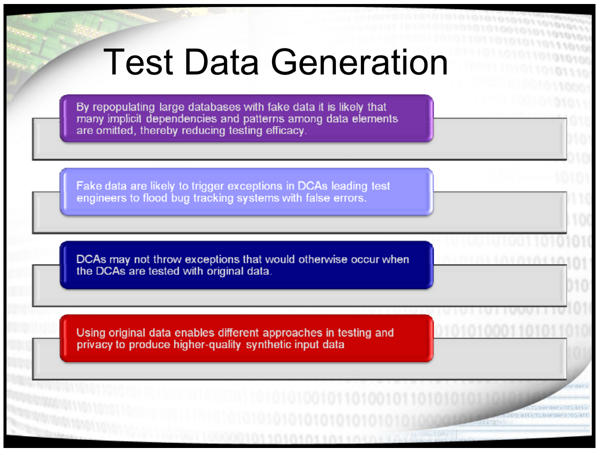
Test Data Generation

- By repopulating large databases with fake data it is likely that many implicit dependencies and patterns among data elements are omitted, thereby reducing testing efficacy.
- Fake data are likely to trigger exceptions in DCAs leading test engineers to flood bug tracking systems with false errors.
- DCAs may not throw exceptions that would otherwise occur when the DCAs are tested with original data.
- Using original data enables different approaches in testing and privacy to produce higher-quality synthetic input data



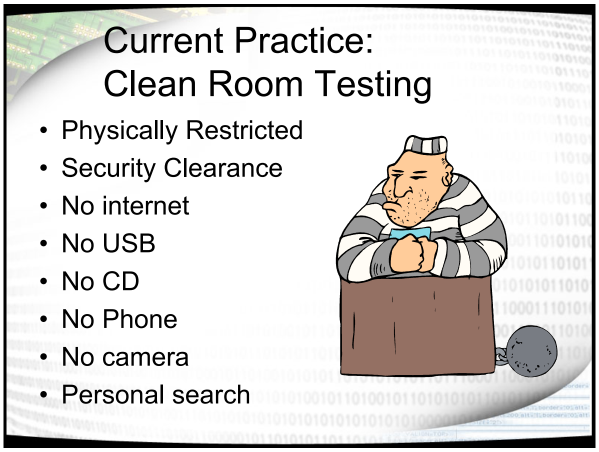
Example Of Generating Semantically Incorrect Data

- A test data generation tool for insurance application creates an entry in the database for a man who suffers from ***gestational diabetes***.



Test Data Generation

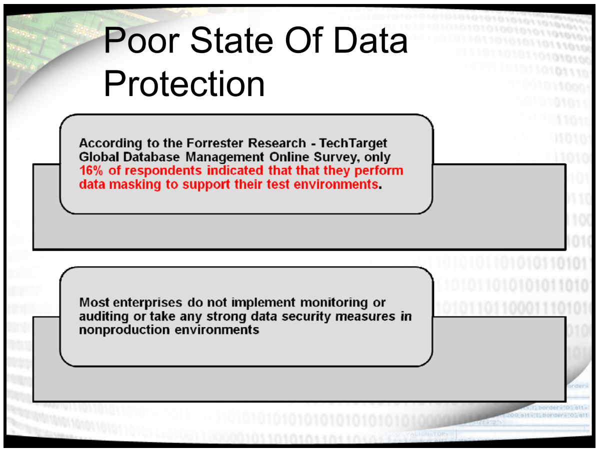
- By repopulating large databases with fake data it is likely that many implicit dependencies and patterns among data elements are omitted, thereby reducing testing efficacy.
- Fake data are likely to trigger exceptions in DCAs leading test engineers to flood bug tracking systems with false errors.
- DCAs may not throw exceptions that would otherwise occur when the DCAs are tested with original data.
- Using original data enables different approaches in testing and privacy to produce higher-quality synthetic input data



Current Practice: Clean Room Testing

- Physically Restricted
- Security Clearance
- No internet
- No USB
- No CD
- No Phone
- No camera
- Personal search

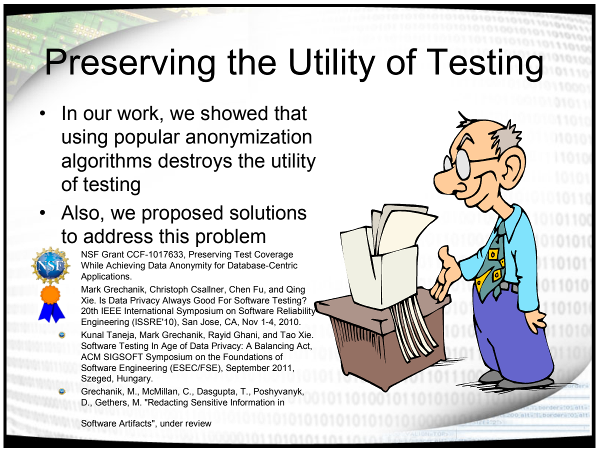
Walking the Tightrope



Poor State Of Data Protection

According to the Forrester Research - TechTarget Global Database Management Online Survey, only **16% of respondents indicated that that they perform data masking to support their test environments**.

Most enterprises do not implement monitoring or auditing or take any strong data security measures in nonproduction environments



Preserving the Utility of Testing

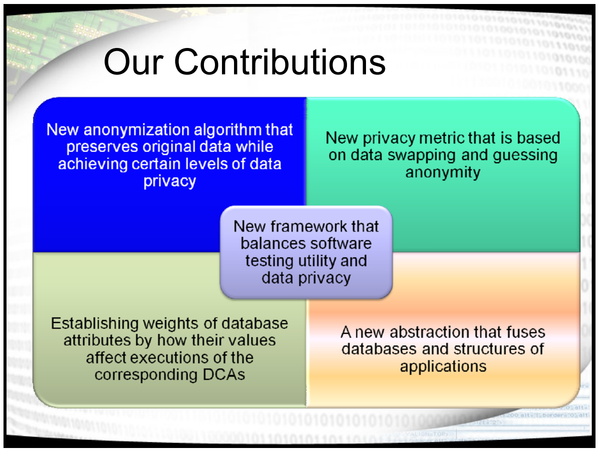
- In our work, we showed that using popular anonymization algorithms destroys the utility of testing
- Also, we proposed solutions to address this problem

NSF Grant CCF-1017633, Preserving Test Coverage While Achieving Data Anonymity for Database-Centric Applications.

Mark Grechanik, Christoph Csallner, Chen Fu, and Qing Xie. Is Data Privacy Always Good For Software Testing? 20th IEEE International Symposium on Software Reliability Engineering (ISSRE'10), San Jose, CA, Nov 1-4, 2010.

- Kunal Taneja, Mark Grechanik, Rayid Ghani, and Tao Xie. Software Testing In Age of Data Privacy: A Balancing Act, ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), September 2011, Szeged, Hungary.
- Grechanik, M., McMillan, C., Dasgupta, T., Poshyvanyk, D., Gethers, M. "Redacting Sensitive Information in

  Software Artifacts", under review



Our Contributions

New anonymization algorithm that preserves original data while achieving certain levels of data privacy

New privacy metric that is based on data swapping and guessing anonymity

New framework that balances software testing utility and data privacy

Establishing weights of database attributes by how their values affect executions of the corresponding DCAs

A new abstraction that fuses databases and structures of applications

Goals of Our Solution

Enable Organizations to Balance Testing Utility and Privacy

by preserving test coverage while releasing DCAs to external test centers with a controlled disclosure of sensitive information.

Support Software Evolution

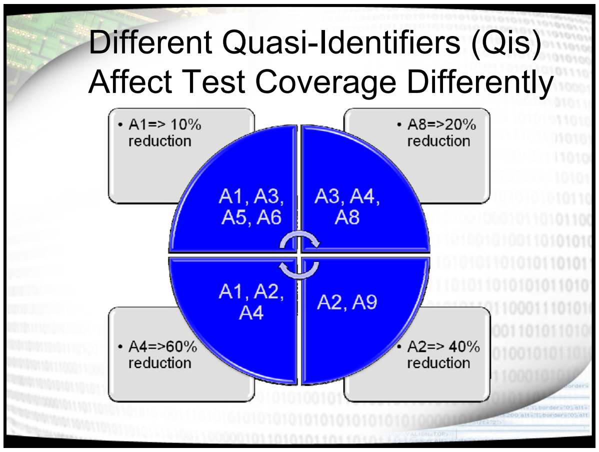
by re-anonymizing the original data multiple times without enabling statistical data inference.

Keep original values

in sanitized databases thus enabling testers to achieve higher test coverage.

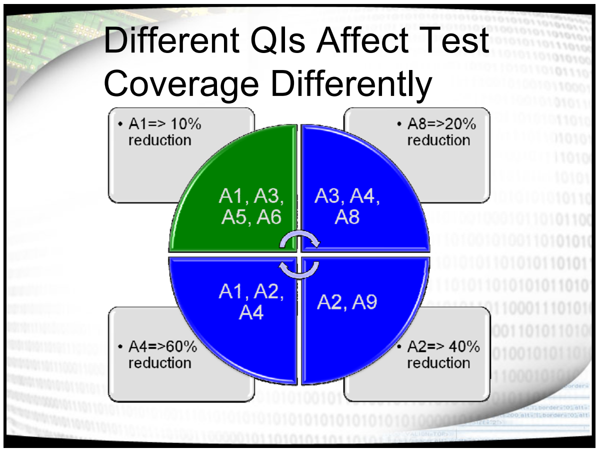
Ensure that Privacy Metric

measures the difficulty of attackers.



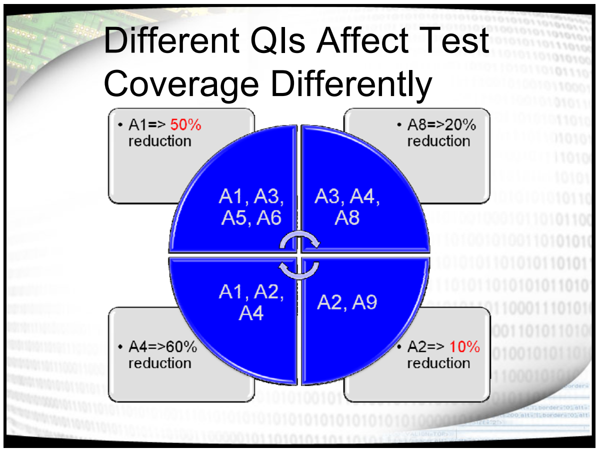
Different Quasi-Identifiers (Qis) Affect Test Coverage Differently

- A1=> 10% reduction
- A8=>20% reduction
- A4=>60% reduction
- A2=> 40% reduction



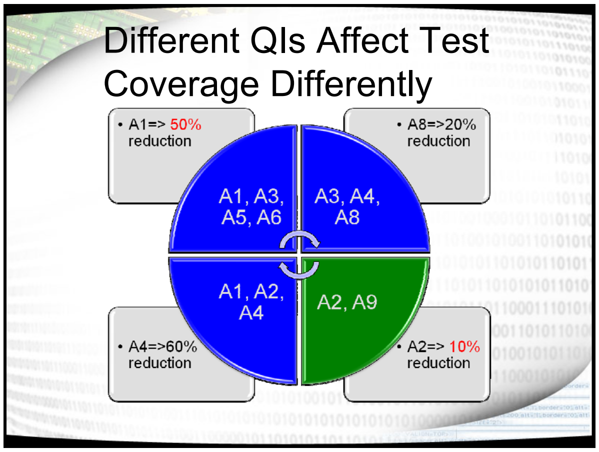
Different QIs Affect Test Coverage Differently

- A1=> 10% reduction
- A8=>20% reduction
- A4=>60% reduction
- A2=> 40% reduction



Different QIs Affect Test Coverage Differently

- A1=> 50% reduction
- A8=>20% reduction
- A4=>60% reduction
- A2=> 10% reduction



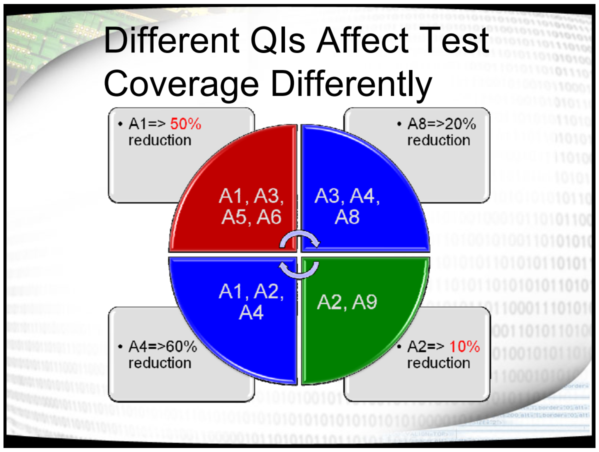
Different QIs Affect Test Coverage Differently

- A1=> 50% reduction

- A8=>20% reduction

- A4=>60% reduction

- A2=> 10% reduction



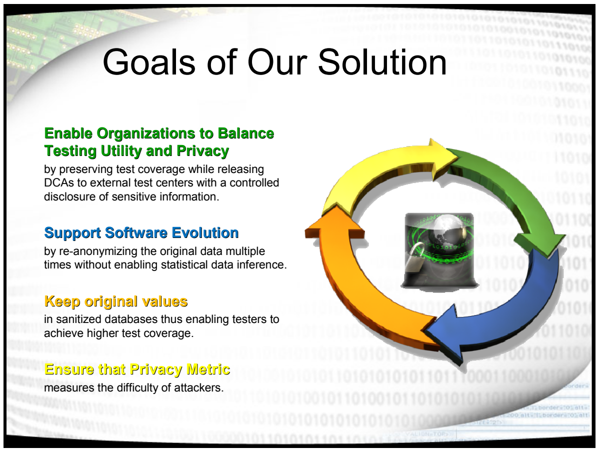
Different QIs Affect Test Coverage Differently

- A1=> 50% reduction

- A8=>20% reduction

- A4=>60% reduction

- A2=> 10% reduction



Goals of Our Solution

Enable Organizations to Balance Testing Utility and Privacy

by preserving test coverage while releasing DCAs to external test centers with a controlled disclosure of sensitive information.

Support Software Evolution

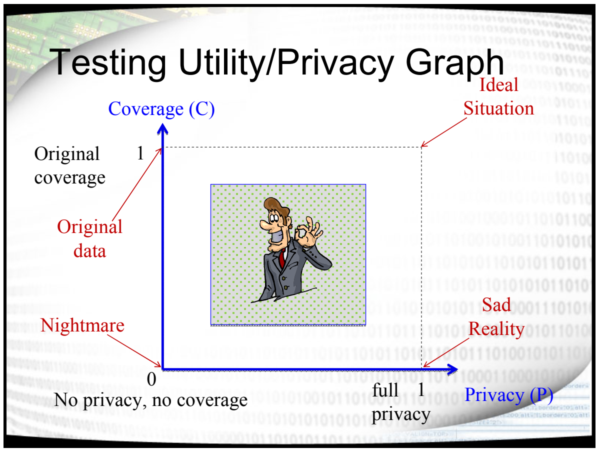
by re-anonymizing the original data multiple times without enabling statistical data inference.

Keep original values

in sanitized databases thus enabling testers to achieve higher test coverage.

Ensure that Privacy Metric

measures the difficulty of attackers.



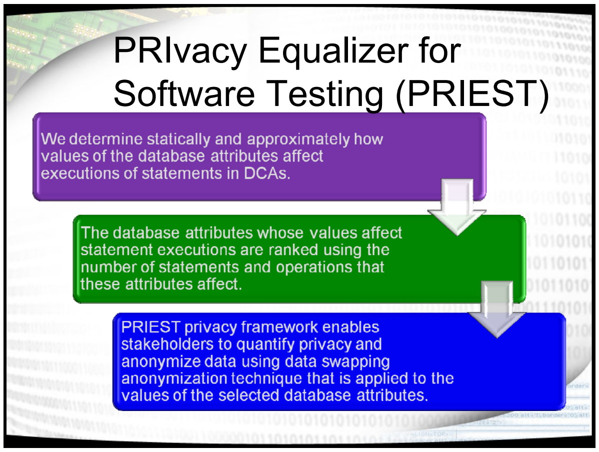
Testing Utility/Privacy Graph

A Main Goal

Enable stakeholders to balance privacy and utility of testing

Desired Privacy

Acceptable test coverage

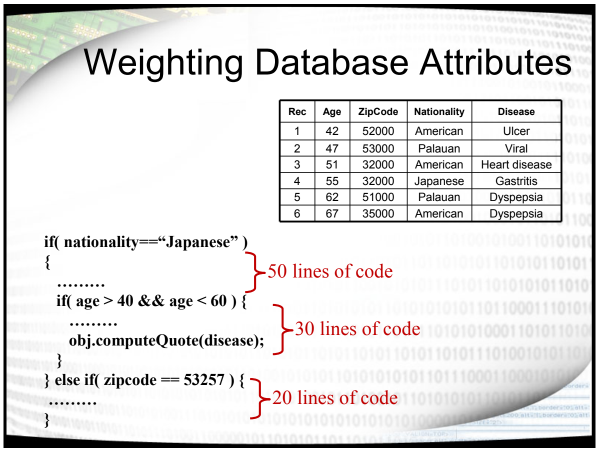


PRIvacy Equalizer for Software Testing (PRIEST)

We determine statically and approximately how values of the database attributes affect executions of statements in DCAs.

The database attributes whose values affect statement executions are ranked using the number of statements and operations that these attributes affect.

PRIEST privacy framework enables stakeholders to quantify privacy and anonymize data using data swapping anonymization technique that is applied to the values of the selected database attributes.



Weighting Database Attributes

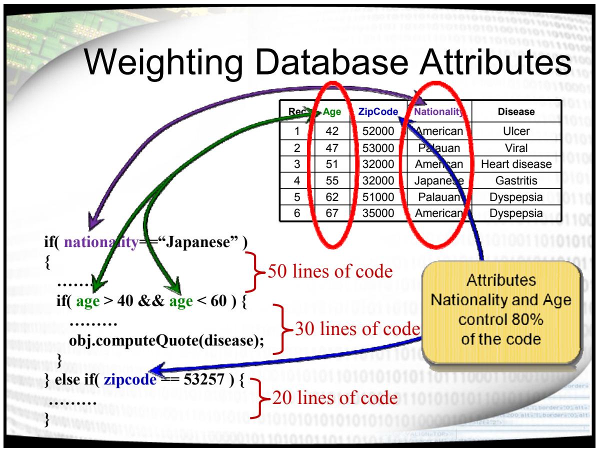
Rec	Age	ZipCode	Nationality	Disease
1	42	52000	American	Ulcer
2	47	53000	Palauan	Viral
3	51	32000	American	Heart disease
4	55	32000	Japanese	Gastritis
5	62	51000	Palauan	Dyspepsia
6	67	35000	American	Dyspepsia

```
if( nationality=="Japanese" )
{
  .........
  if( age > 40 && age < 60 ) {
    .........
    obj.computeQuote(disease);
  }
} else if( zipcode == 53257 ) {
  .........
}
```

50 lines of code

30 lines of code

20 lines of code



Weighting Database Attributes

Rec	Age	ZipCode	Nationality	Disease
1	42	52000	American	Ulcer
2	47	53000	Palauan	Viral
3	51	32000	American	Heart disease
4	55	32000	Japanese	Gastritis
5	62	51000	Palauan	Dyspepsia
6	67	35000	American	Dyspepsia

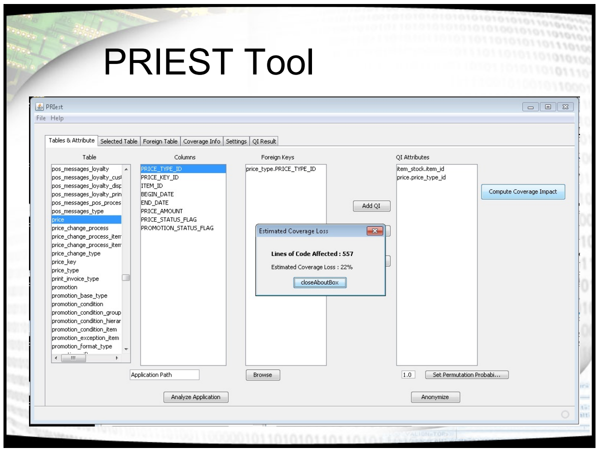
```
if( nationality=="Japanese" )
{
  .........
  if( age > 40 && age < 60 ) {
    .........
    obj.computeQuote(disease);
  }
} else if( zipcode == 53257 ) {
  .........
}
```

50 lines of code

30 lines of code

20 lines of code

Attributes Nationality and Age control 80% of the code



PRIEST Tool

Table
pos_messages_loyalty
pos_messages_loyalty_cust
pos_messages_loyalty_disp
pos_messages_loyalty_prin
pos_messages_pos_proces
pos_messages_type
price
price_change_process
price_change_process_item
price_change_process_item
price_change_type
price_key
price_type
print_invoice_type
promotion
promotion_base_type
promotion_condition
promotion_condition_group
promotion_condition_hierar
promotion_condition_item
promotion_exception_item
promotion_format_type

Columns
PRICE_TYPE_ID
PRICE_KEY_ID
ITEM_ID
BEGIN_DATE
END_DATE
PRICE_AMOUNT
PRICE_STATUS_FLAG
PROMOTION_STATUS_FLAG

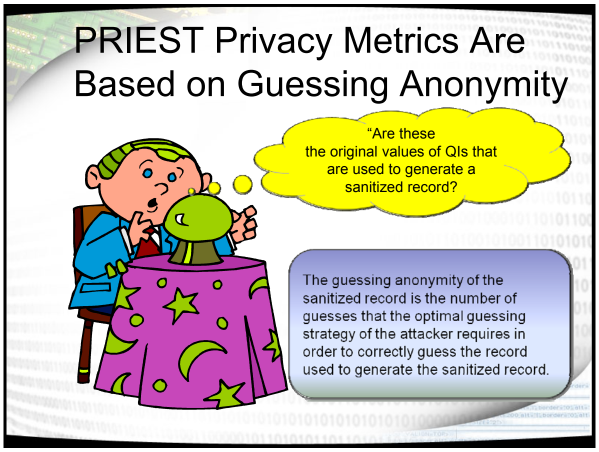
Foreign Keys
price_type.PRICE_TYPE_ID

QI Attributes
item_stock.item_id
price.price_type_id

Estimated Coverage Loss

Lines of Code Affected : 557

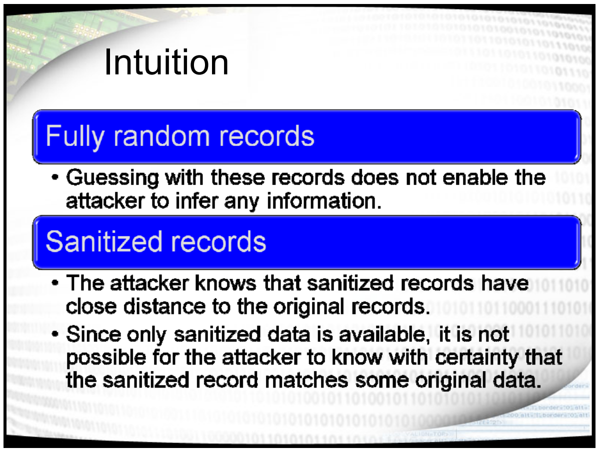
Estimated Coverage Loss : 22%



PRIEST Privacy Metrics Are Based on Guessing Anonymity

"Are these the original values of QIs that are used to generate a sanitized record?

The guessing anonymity of the sanitized record is the number of guesses that the optimal guessing strategy of the attacker requires in order to correctly guess the record used to generate the sanitized record.



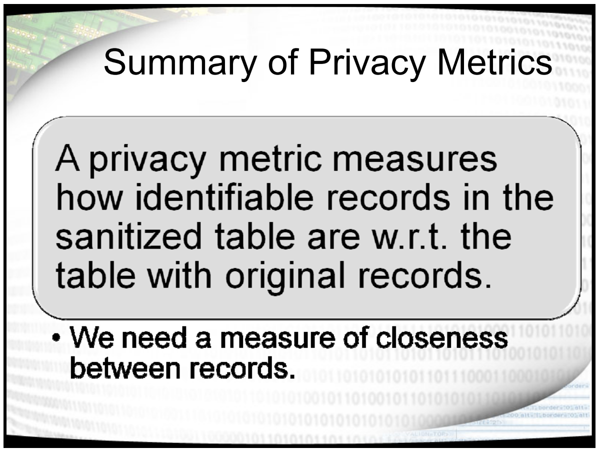
Intuition

Fully random records

- Guessing with these records does not enable the attacker to infer any information.

Sanitized records

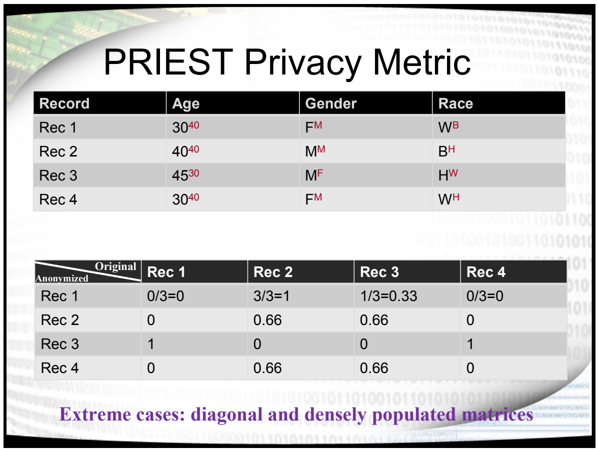
- The attacker knows that sanitized records have close distance to the original records.
- Since only sanitized data is available, it is not possible for the attacker to know with certainty that the sanitized record matches some original data.



Summary of Privacy Metrics

A privacy metric measures how identifiable records in the sanitized table are w.r.t. the table with original records.

- We need a measure of closeness between records.

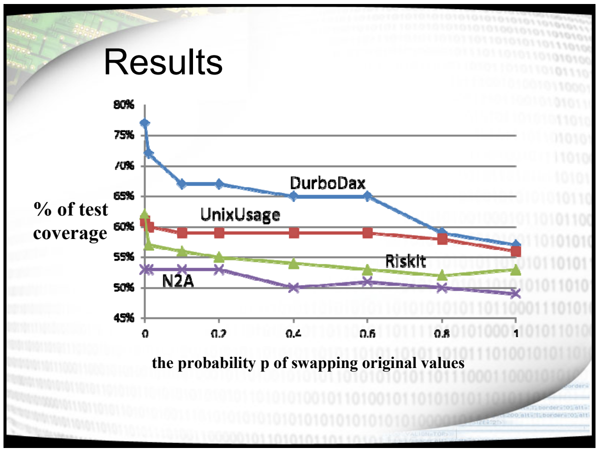


PRIEST Privacy Metric

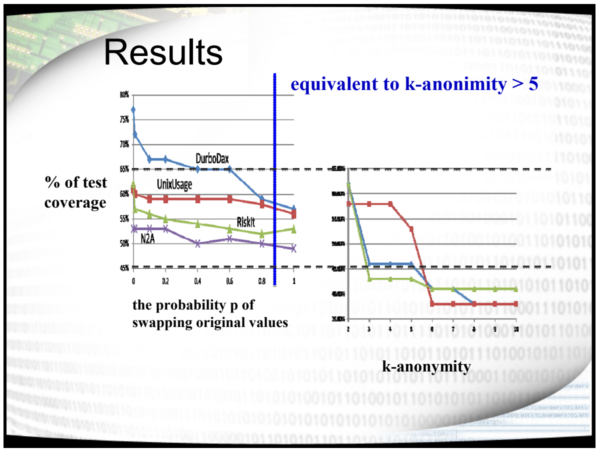
Record	Age	Gender	Race
Rec 1	30^{40}	F^{M}	W^{B}
Rec 2	40^{40}	M^{M}	B^{H}
Rec 3	45^{30}	M^{F}	H^{W}
Rec 4	30^{40}	F^{M}	W^{H}

Original Anonymized	Rec 1	Rec 2	Rec 3	Rec 4
Rec 1	0/3=0	3/3=1	1/3=0.33	0/3=0
Rec 2	0	0.66	0.66	0
Rec 3	1	0	0	1
Rec 4	0	0.66	0.66	0

Extreme cases: diagonal and densely populated matrices



Results

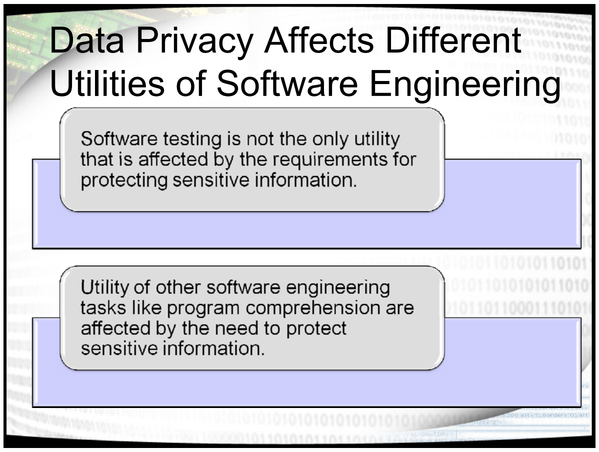


Results

equivalent to k-anonimity > 5

What's Next?

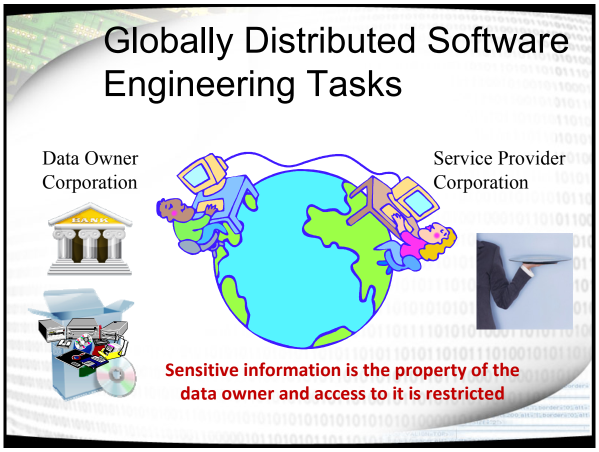
Privacy and Utilities of different software engineering tasks



Data Privacy Affects Different Utilities of Software Engineering

Software testing is not the only utility that is affected by the requirements for protecting sensitive information.

Utility of other software engineering tasks like program comprehension are affected by the need to protect sensitive information.

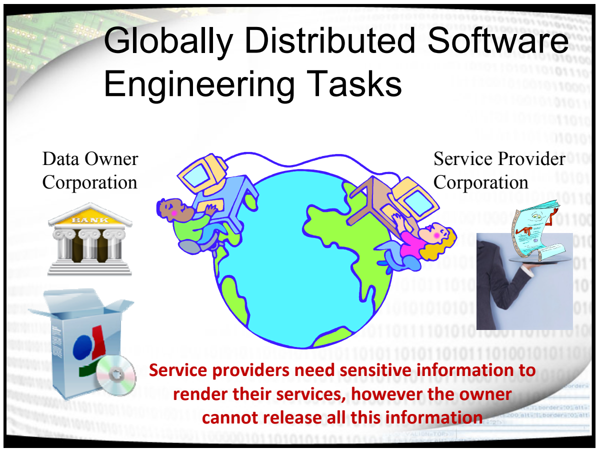


Globally Distributed Software Engineering Tasks

Data Owner Corporation

Service Provider Corporation

Sensitive information is the property of the data owner and access to it is restricted



Globally Distributed Software Engineering Tasks

Data Owner Corporation

Service Provider Corporation

Service providers need sensitive information to render their services, however the owner cannot release all this information

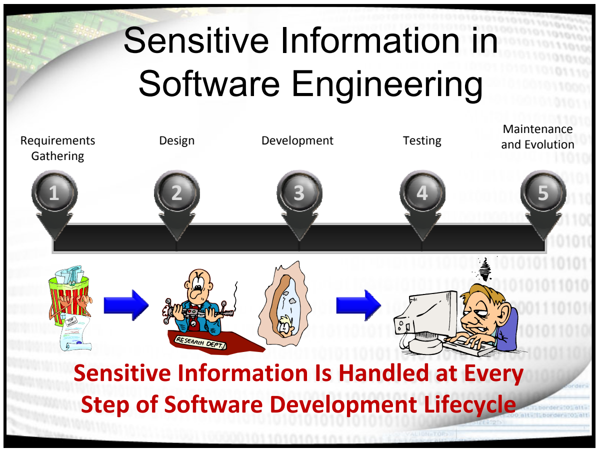
Attackers and Victims in the Globally Distributed Context

Data Owner Corporation

Service Provider Corporation

Victim

Attackers



Sensitive Information in Software Engineering

Requirements Gathering

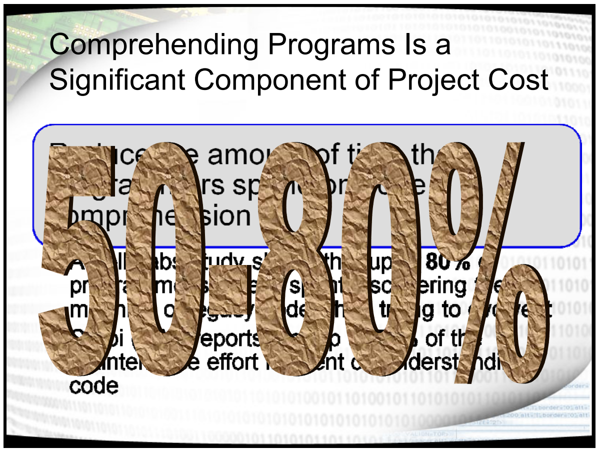
Design

Development

Testing

Maintenance and Evolution

Sensitive Information Is Handled at Every Step of Software Development Lifecycle

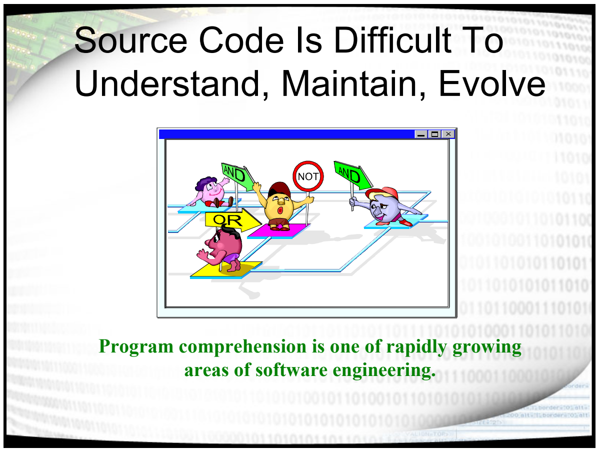


Comprehending Programs Is a Significant Component of Project Cost

Reduce the amount of time that programmers spend on code comprehension

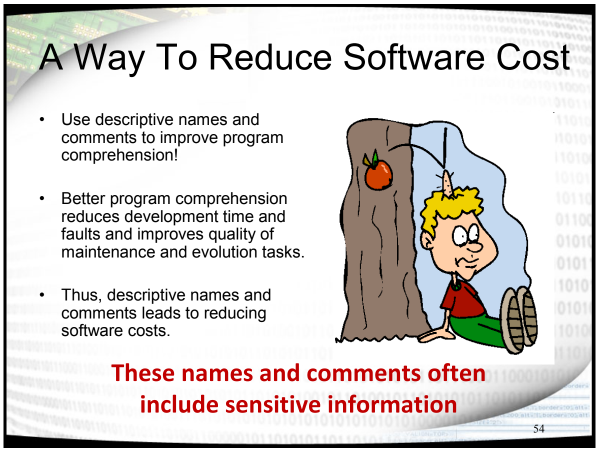
50-80%

A Bell Labs study shows that up to **80 %** of programmers' time is spent discovering the meaning of legacy code when trying to evolve it [illegible] reports that up [illegible] of the maintenance effort is spent on understanding code



Source Code Is Difficult To Understand, Maintain, Evolve

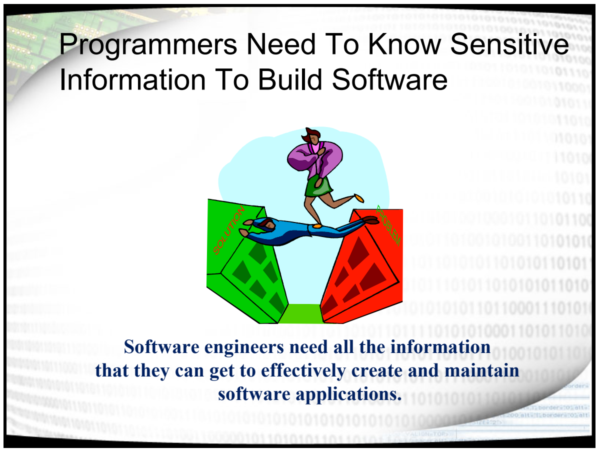
Program comprehension is one of rapidly growing areas of software engineering.



A Way To Reduce Software Cost

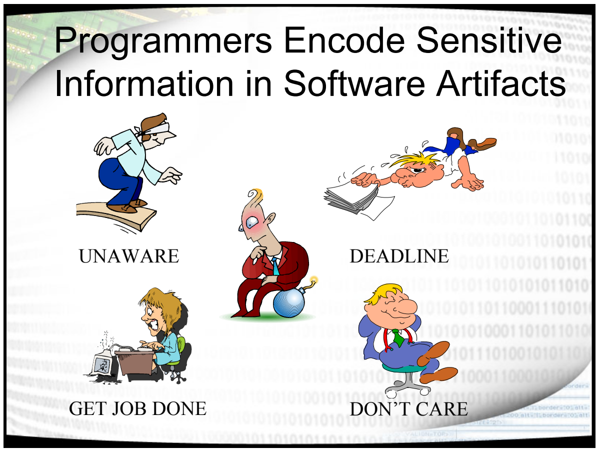
- Use descriptive names and comments to improve program comprehension!
- Better program comprehension reduces development time and faults and improves quality of maintenance and evolution tasks.
- Thus, descriptive names and comments leads to reducing software costs.

These names and comments often include sensitive information



Programmers Need To Know Sensitive Information To Build Software

Software engineers need all the information that they can get to effectively create and maintain software applications.



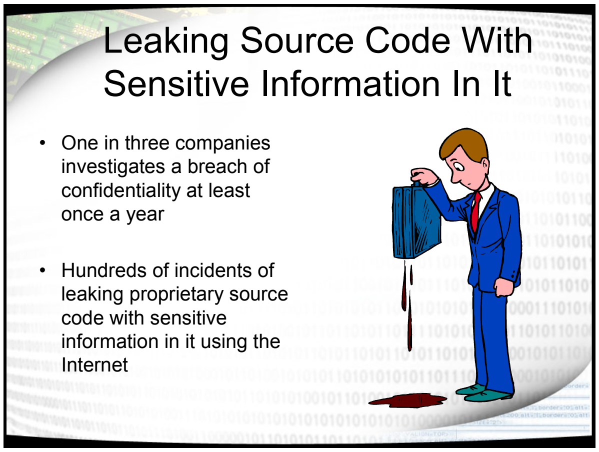
Programmers Encode Sensitive Information in Software Artifacts

UNAWARE

DEADLINE

GET JOB DONE

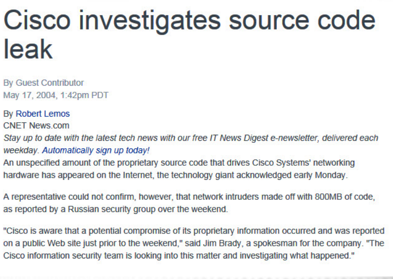
DON'T CARE



Leaking Source Code With Sensitive Information In It

- One in three companies investigates a breach of confidentiality at least once a year
- Hundreds of incidents of leaking proprietary source code with sensitive information in it using the Internet

Source Code Leaks Are Serious



Cisco investigates source code leak

By Guest Contributor
May 17, 2004, 1:42pm PDT

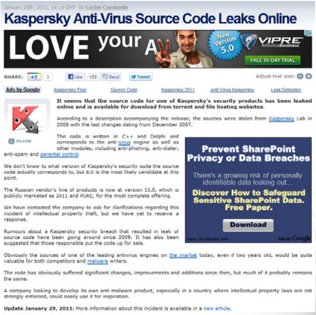
By Robert Lemos
CNET News.com
Stay up to date with the latest tech news with our free IT News Digest e-newsletter, delivered each weekday. Automatically sign up today!

An unspecified amount of the proprietary source code that drives Cisco Systems' networking hardware has appeared on the Internet, the technology giant acknowledged early Monday.

A representative could not confirm, however, that network intruders made off with 800MB of code, as reported by a Russian security group over the weekend.

"Cisco is aware that a potential compromise of its proprietary information occurred and was reported on a public Web site just prior to the weekend," said Jim Brady, a spokesman for the company. "The Cisco information security team is looking into this matter and investigating what happened."

Source Code Leaks Are Serious



January 28th, 2011, 14:19 GMT · By Lucian Constantin

Kaspersky Anti-Virus Source Code Leaks Online

It seems that the source code for one of Kaspersky's security products has been leaked online and is available for download from torrent and file hosting websites.

According to a description accompanying the release, the sources were stolen from Kaspersky Lab in 2008 with the last changes dating from December 2007.

The code is written in C++ and Delphi and corresponds to the anti-virus engine as well as other modules, including anti-phishing, anti-dialer, anti-spam and parental control.

We don't know to what version of Kaspersky's security suite the source code actually corresponds to, but 8.0 is the most likely candidate at this point.

The Russian vendor's line of products is now at version 11.0, which is publicly marketed as 2011 and PURE, for the most complete offering.

We have contacted the company to ask for clarifications regarding this incident of intellectual property theft, but we have yet to receive a response.

Rumours about a Kaspersky security breach that resulted in leak of source code have been going around since 2009. It has also been suggested that those responsible put the code up for sale.

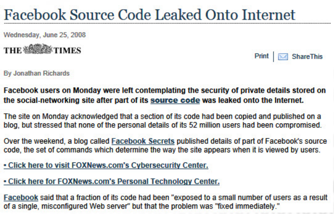
Obviously the sources of one of the leading antivirus engines on the market today, even if two years old, would be quite valuable for both competitors and malware writers.

The code has obviously suffered significant changes, improvements and additions since then, but much of it probably remains the same.

A company looking to develop its own anti-malware product, especially in a country where intellectual property laws are not strongly enforced, could easily use it for inspiration.

Update January 29, 2011: More information about this incident is available in a new article.

Source Code Leaks Are Serious



Facebook Source Code Leaked Onto Internet

Wednesday, June 25, 2008

THE TIMES

Print | ShareThis

By Jonathan Richards

Facebook users on Monday were left contemplating the security of private details stored on the social-networking site after part of its source code was leaked onto the Internet.

The site on Monday acknowledged that a section of its code had been copied and published on a blog, but stressed that none of the personal details of its 52 million users had been compromised.

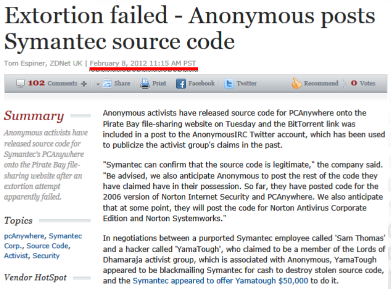
Over the weekend, a blog called Facebook Secrets published details of part of Facebook's source code, the set of commands which determine the way the site appears when it is viewed by users.

• Click here to visit FOXNews.com's Cybersecurity Center.

• Click here for FOXNews.com's Personal Technology Center.

Facebook said that a fraction of its code had been "exposed to a small number of users as a result of a single, misconfigured Web server" but that the problem was "fixed immediately."

Source Code Leaks Are Serious



Extortion failed - Anonymous posts Symantec source code

Tom Espiner, ZDNet UK | February 8, 2012 11:15 AM PST

Summary

Anonymous activists have released source code for Symantec's PCAnywhere onto the Pirate Bay file-sharing website after an extortion attempt apparently failed.

Topics

pcAnywhere, Symantec Corp., Source Code, Activist, Security

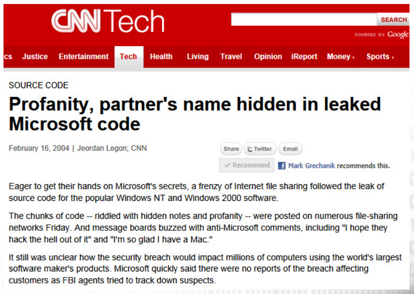
Vendor HotSpot

Anonymous activists have released source code for PCAnywhere onto the Pirate Bay file-sharing website on Tuesday and the BitTorrent link was included in a post to the AnonymousIRC Twitter account, which has been used to publicize the activist group's claims in the past.

"Symantec can confirm that the source code is legitimate," the company said. "Be advised, we also anticipate Anonymous to post the rest of the code they have claimed have in their possession. So far, they have posted code for the 2006 version of Norton Internet Security and PCAnywhere. We also anticipate that at some point, they will post the code for Norton Antivirus Corporate Edition and Norton Systemworks."

In negotiations between a purported Symantec employee called 'Sam Thomas' and a hacker called 'YamaTough', who claimed to be a member of the Lords of Dhamaraja activist group, which is associated with Anonymous, YamaTough appeared to be blackmailing Symantec for cash to destroy stolen source code, and the Symantec appeared to offer Yamatough $50,000 to do it.

Source Code Leaks Are Serious



SOURCE CODE

Profanity, partner's name hidden in leaked Microsoft code

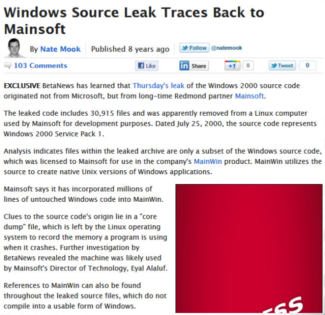
February 16, 2004 | Jeordan Legon; CNN

Eager to get their hands on Microsoft's secrets, a frenzy of Internet file sharing followed the leak of source code for the popular Windows NT and Windows 2000 software.

The chunks of code -- riddled with hidden notes and profanity -- were posted on numerous file-sharing networks Friday. And message boards buzzed with anti-Microsoft comments, including "I hope they hack the hell out of it" and "I'm so glad I have a Mac."

It still was unclear how the security breach would impact millions of computers using the world's largest software maker's products. Microsoft quickly said there were no reports of the breach affecting customers as FBI agents tried to track down suspects.

Who Leaked MS Source Code?



Windows Source Leak Traces Back to Mainsoft

By Nate Mook | Published 8 years ago

EXCLUSIVE BetaNews has learned that Thursday's leak of the Windows 2000 source code originated not from Microsoft, but from long-time Redmond partner Mainsoft.

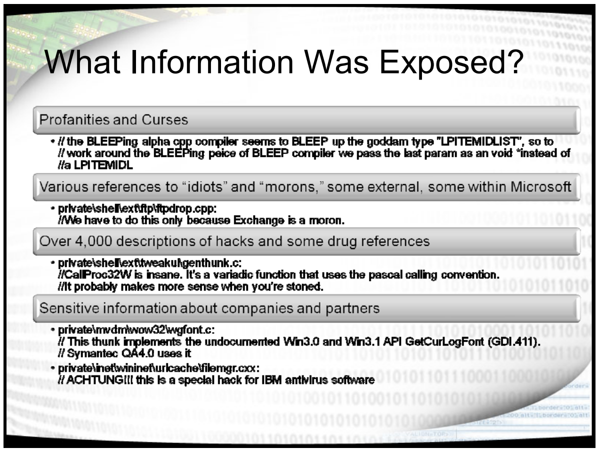
The leaked code includes 30,915 files and was apparently removed from a Linux computer used by Mainsoft for development purposes. Dated July 25, 2000, the source code represents Windows 2000 Service Pack 1.

Analysis indicates files within the leaked archive are only a subset of the Windows source code, which was licensed to Mainsoft for use in the company's MainWin product. MainWin utilizes the source to create native Unix versions of Windows applications.

Mainsoft says it has incorporated millions of lines of untouched Windows code into MainWin.

Clues to the source code's origin lie in a "core dump" file, which is left by the Linux operating system to record the memory a program is using when it crashes. Further investigation by BetaNews revealed the machine was likely used by Mainsoft's Director of Technology, Eyal Alaluf.

References to MainWin can also be found throughout the leaked source files, which do not compile into a usable form of Windows.



What Information Was Exposed?

Profanities and Curses

- // the BLEEPing alpha cpp compiler seems to BLEEP up the goddam type "LPITEMIDLIST", so to // work around the BLEEPing peice of BLEEP compiler we pass the last param as an void *instead of //a LPITEMIDL

Various references to "idiots" and "morons," some external, some within Microsoft

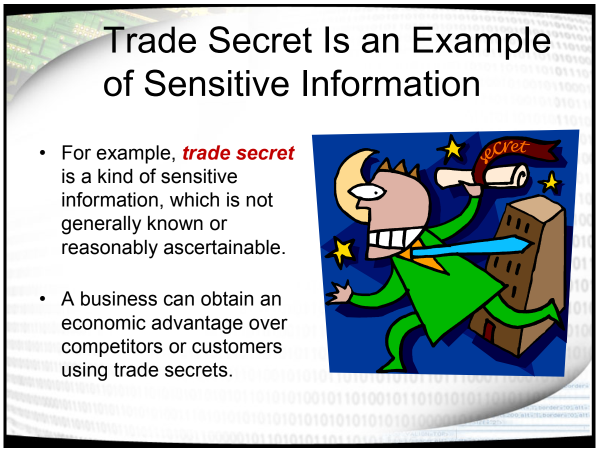
- private\shell\ext\ftp\ftpdrop.cpp:
  //We have to do this only because Exchange is a moron.

Over 4,000 descriptions of hacks and some drug references

- private\shell\ext\tweakui\genthunk.c:
  //CallProc32W is insane. It's a variadic function that uses the pascal calling convention.
  //It probably makes more sense when you're stoned.

Sensitive information about companies and partners

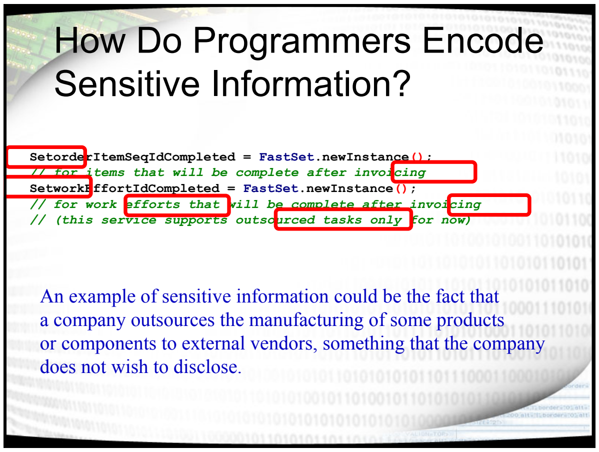
- private\mvdm\wow32\wgfont.c:
  // This thunk implements the undocumented Win3.0 and Win3.1 API GetCurLogFont (GDI.411).
  // Symantec QA4.0 uses it
- private\inet\wininet\urlcache\filemgr.cxx:
  // ACHTUNG!!! this is a special hack for IBM antivirus software



Trade Secret Is an Example of Sensitive Information

- For example, ***trade secret*** is a kind of sensitive information, which is not generally known or reasonably ascertainable.
- A business can obtain an economic advantage over competitors or customers using trade secrets.

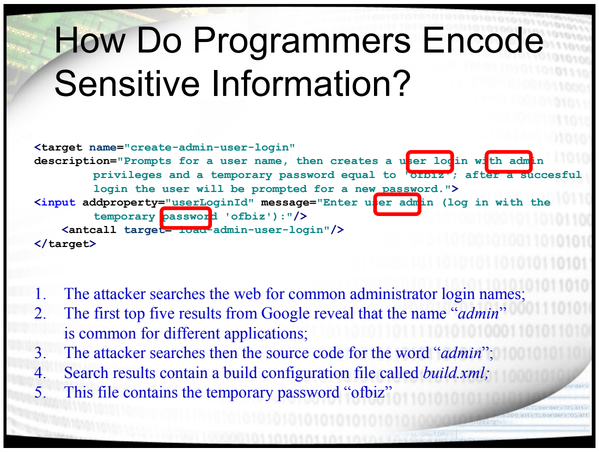
Attackers and Victims in the Globally Distributed Context



How Do Programmers Encode Sensitive Information?

```
SetorderItemSeqIdCompleted = FastSet.newInstance();
// for items that will be complete after invoicing
SetworkEffortIdCompleted = FastSet.newInstance();
// for work efforts that will be complete after invoicing
// (this service supports outsourced tasks only for now)
```

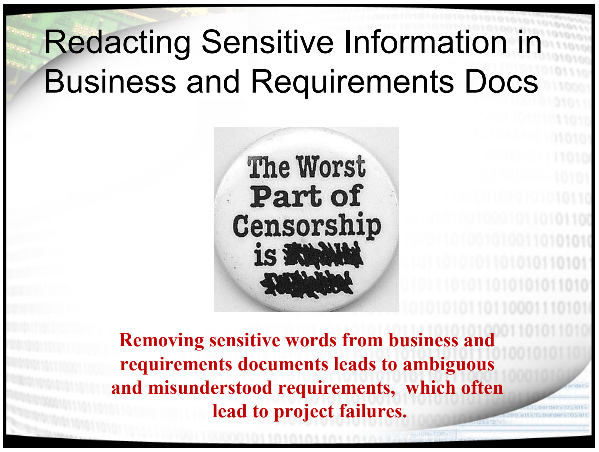
An example of sensitive information could be the fact that a company outsources the manufacturing of some products or components to external vendors, something that the company does not wish to disclose.



How Do Programmers Encode Sensitive Information?

```
<target name="create-admin-user-login"
description="Prompts for a user name, then creates a user login with admin
        privileges and a temporary password equal to 'ofbiz'; after a succesful
        login the user will be prompted for a new password.">
<input addproperty="userLoginId" message="Enter user admin (log in with the
        temporary password 'ofbiz'):"/>
   <antcall target="load-admin-user-login"/>
</target>
```

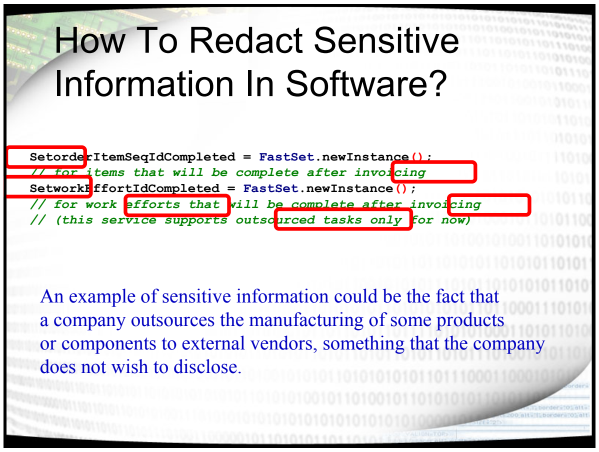
1. The attacker searches the web for common administrator login names;
2. The first top five results from Google reveal that the name “*admin*” is common for different applications;
3. The attacker searches then the source code for the word “*admin*”;
4. Search results contain a build configuration file called *build.xml*;
5. This file contains the temporary password “ofbiz”



Redacting Sensitive Information in Business and Requirements Docs

The Worst
Part of
Censorship
is [redacted]

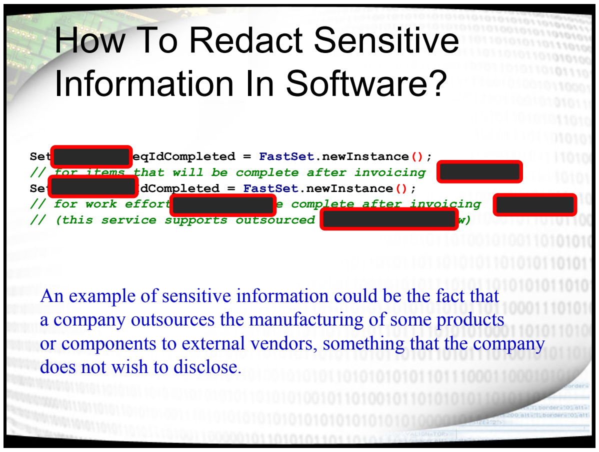
Removing sensitive words from business and requirements documents leads to ambiguous and misunderstood requirements, which often lead to project failures.



How To Redact Sensitive Information In Software?

```
SetorderItemSeqIdCompleted = FastSet.newInstance();
// for items that will be complete after invoicing
SetworkEffortIdCompleted = FastSet.newInstance();
// for work efforts that will be complete after invoicing
// (this service supports outsourced tasks only for now)
```

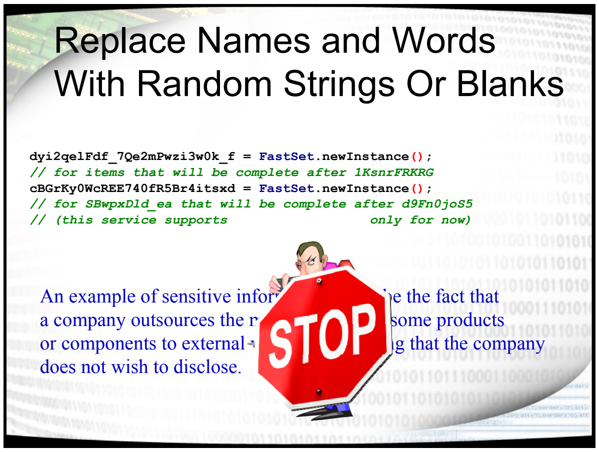
An example of sensitive information could be the fact that a company outsources the manufacturing of some products or components to external vendors, something that the company does not wish to disclose.



How To Redact Sensitive Information In Software?

```
Set[redacted]eqIdCompleted = FastSet.newInstance();
// for items that will be complete after invoicing [redacted]
Se[redacted]dCompleted = FastSet.newInstance();
// for work effort[redacted]e complete after invoicing [redacted]
// (this service supports outsourced [redacted]w)
```

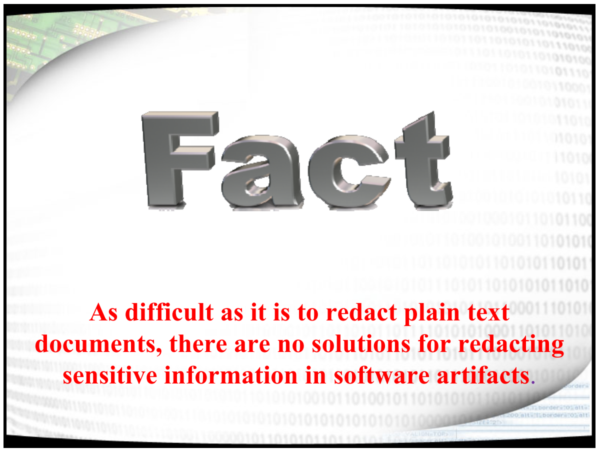
An example of sensitive information could be the fact that a company outsources the manufacturing of some products or components to external vendors, something that the company does not wish to disclose.



Replace Names and Words With Random Strings Or Blanks

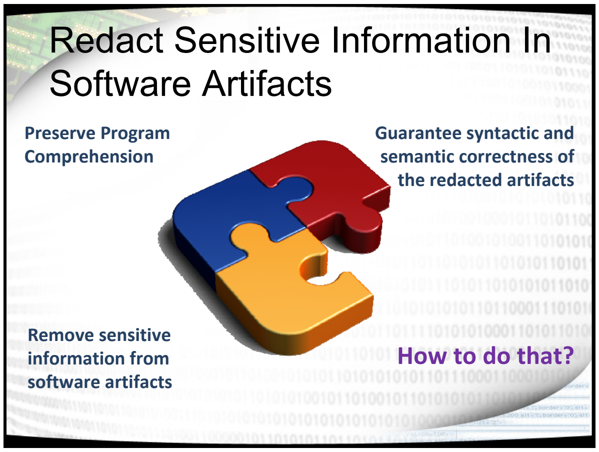
```
dyi2qelFdf_7Qe2mPwzi3w0k_f = FastSet.newInstance();
// for items that will be complete after 1KsnrFRKRG
cBGrKy0WcREE740fR5Br4itsxd = FastSet.newInstance();
// for SBwpxDld_ea that will be complete after d9Fn0joS5
// (this service supports                    only for now)
```

An example of sensitive infor e the fact that
a company outsources the m some products
or components to external g that the company
does not wish to disclose.



Fact

As difficult as it is to redact plain text documents, there are no solutions for redacting sensitive information in software artifacts.



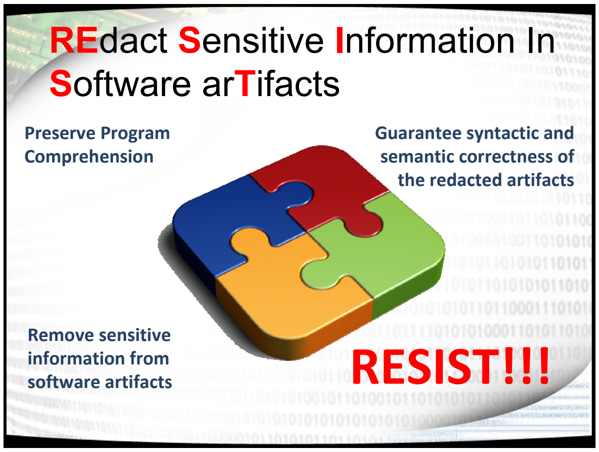
Redact Sensitive Information In Software Artifacts

Preserve Program Comprehension

Guarantee syntactic and semantic correctness of the redacted artifacts

Remove sensitive information from software artifacts

How to do that?



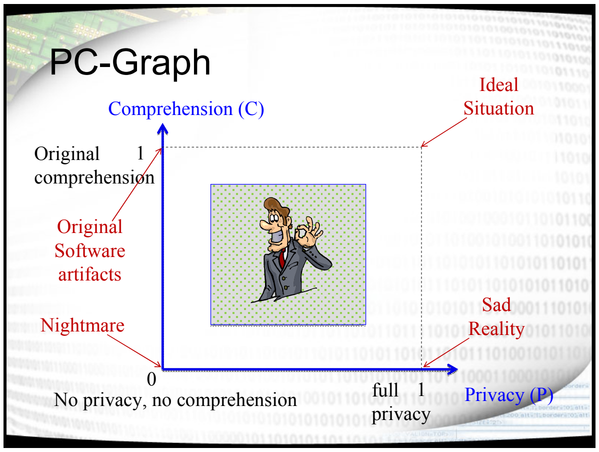
REdact Sensitive Information In Software arTifacts

Preserve Program Comprehension

Guarantee syntactic and semantic correctness of the redacted artifacts

Remove sensitive information from software artifacts

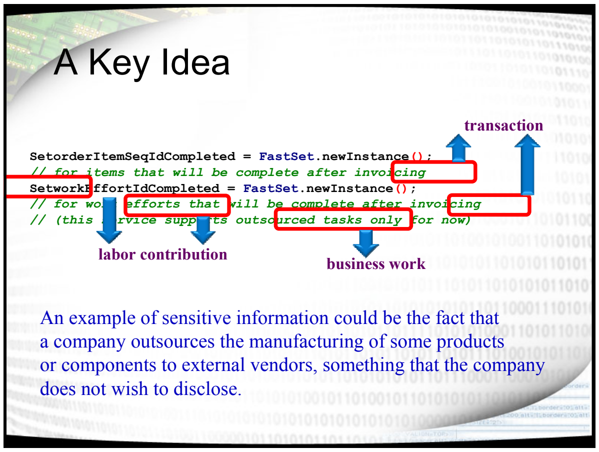
RESIST!!!



PC-Graph

Goals of RESIST

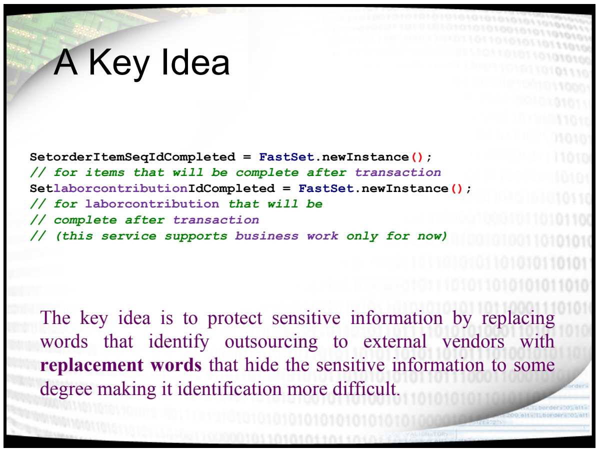
Desired Privacy
Enable stakeholders
to balance privacy
and utility of program
comprehension
Acceptable
program
comprehension



A Key Idea

```
SetorderItemSeqIdCompleted = FastSet.newInstance();
// for items that will be complete after invoicing
SetworkEffortIdCompleted = FastSet.newInstance();
// for work efforts that will be complete after invoicing
// (this service supports outsourced tasks only for now)
```

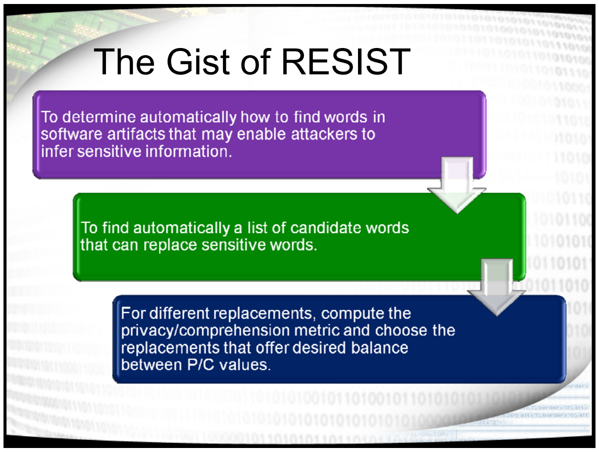
An example of sensitive information could be the fact that a company outsources the manufacturing of some products or components to external vendors, something that the company does not wish to disclose.



A Key Idea

```
SetorderItemSeqIdCompleted = FastSet.newInstance();
// for items that will be complete after transaction
SetlaborcontributionIdCompleted = FastSet.newInstance();
// for laborcontribution that will be
// complete after transaction
// (this service supports business work only for now)
```

The key idea is to protect sensitive information by replacing words that identify outsourcing to external vendors with **replacement words** that hide the sensitive information to some degree making it identification more difficult.

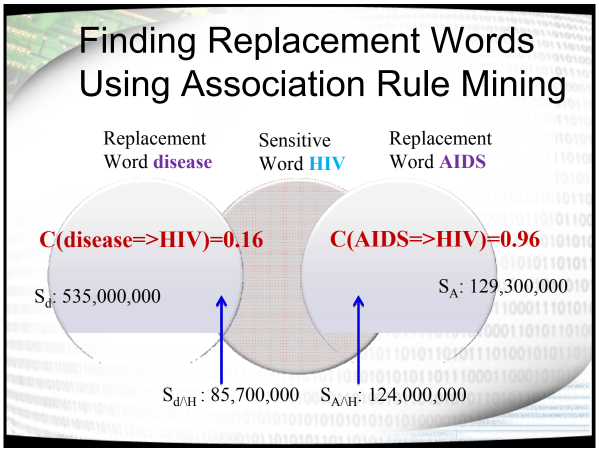


The Gist of RESIST

To determine automatically how to find words in software artifacts that may enable attackers to infer sensitive information.

To find automatically a list of candidate words that can replace sensitive words.

For different replacements, compute the privacy/comprehension metric and choose the replacements that offer desired balance between P/C values.



Finding Replacement Words Using Association Rule Mining

Replacement Word **disease**

Sensitive Word **HIV**

Replacement Word **AIDS**

C(disease=>HIV)=0.16

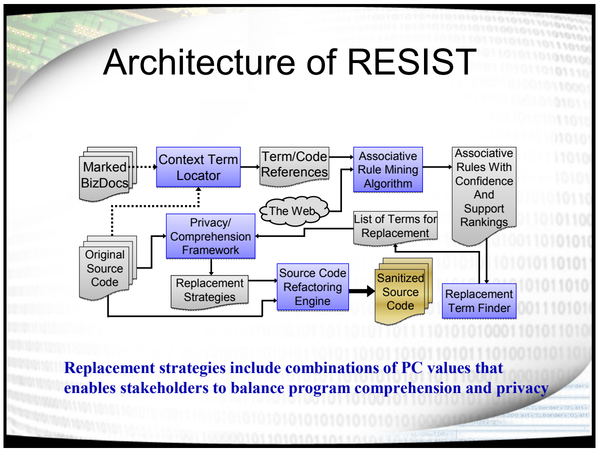
C(AIDS=>HIV)=0.96

S_d: 535,000,000

S_A: 129,300,000

$S_{d/\backslash H}$: 85,700,000

$S_{A/\backslash H}$: 124,000,000



Architecture of RESIST

Marked BizDocs

Context Term Locator

Term/Code References

Associative Rule Mining Algorithm

Associative Rules With Confidence And Support Rankings

The Web

List of Terms for Replacement

Privacy/ Comprehension Framework

Original Source Code

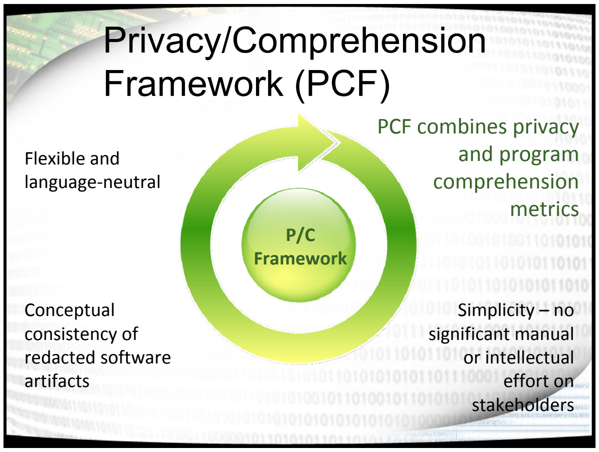
Replacement Strategies

Source Code Refactoring Engine

Sanitized Source Code

Replacement Term Finder

Replacement strategies include combinations of PC values that enables stakeholders to balance program comprehension and privacy



Privacy/Comprehension Framework (PCF)

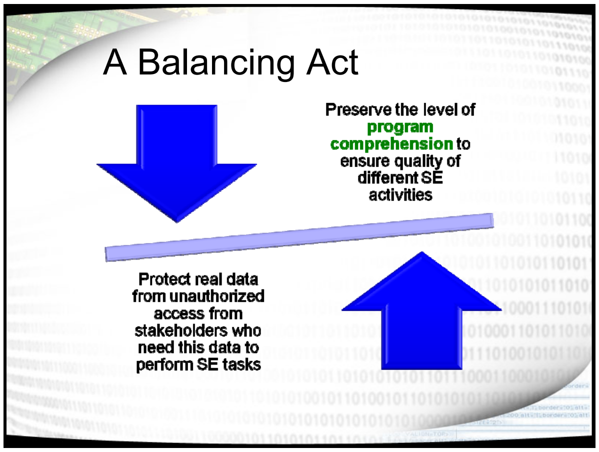
Flexible and language-neutral

PCF combines privacy and program comprehension metrics

P/C Framework

Conceptual consistency of redacted software artifacts

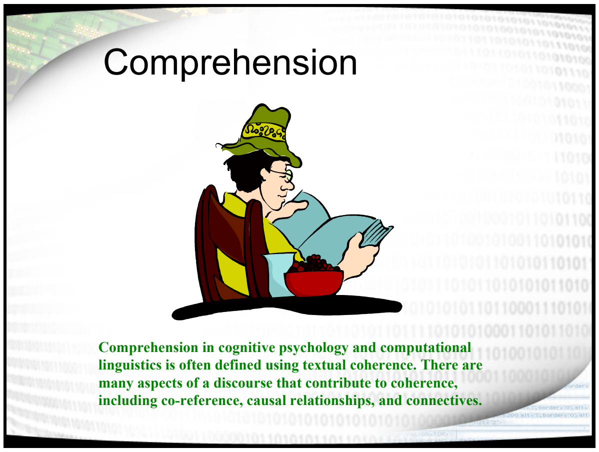
Simplicity – no significant manual or intellectual effort on stakeholders



A Balancing Act

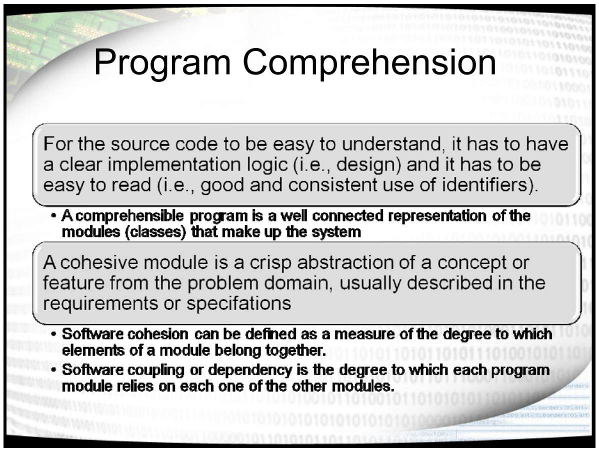
Preserve the level of **program comprehension** to ensure quality of different SE activities

Protect real data from unauthorized access from stakeholders who need this data to perform SE tasks



Comprehension

Comprehension in cognitive psychology and computational linguistics is often defined using textual coherence. There are many aspects of a discourse that contribute to coherence, including co-reference, causal relationships, and connectives.



Program Comprehension

For the source code to be easy to understand, it has to have a clear implementation logic (i.e., design) and it has to be easy to read (i.e., good and consistent use of identifiers).

- A comprehensible program is a well connected representation of the modules (classes) that make up the system

A cohesive module is a crisp abstraction of a concept or feature from the problem domain, usually described in the requirements or specifations

- Software cohesion can be defined as a measure of the degree to which elements of a module belong together.
- Software coupling or dependency is the degree to which each program module relies on each one of the other modules.

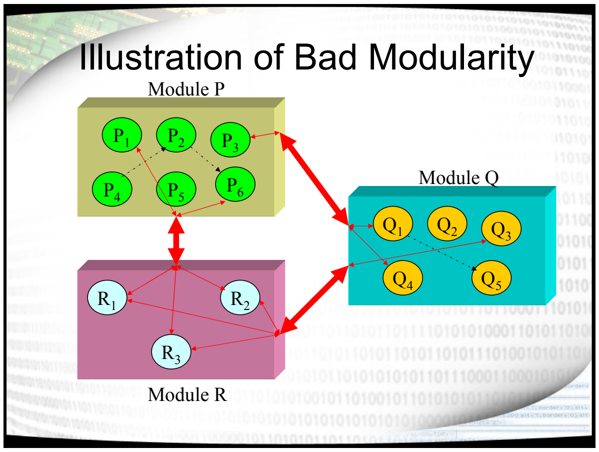


Illustration of Bad Modularity

Illustration of Good Modularity

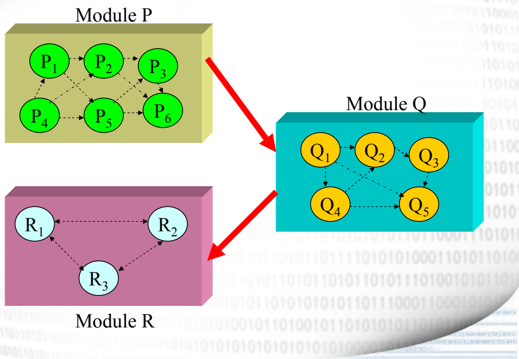
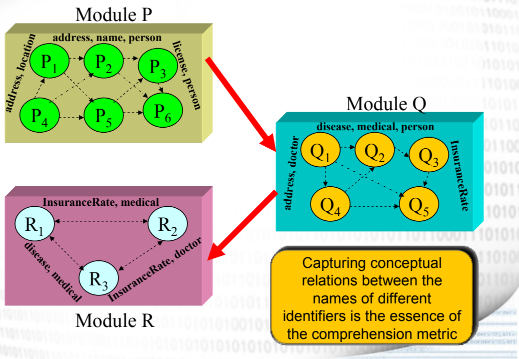


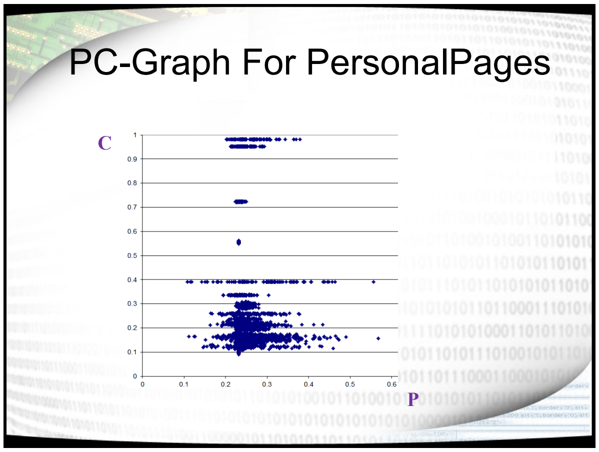
Illustration of Good Modularity



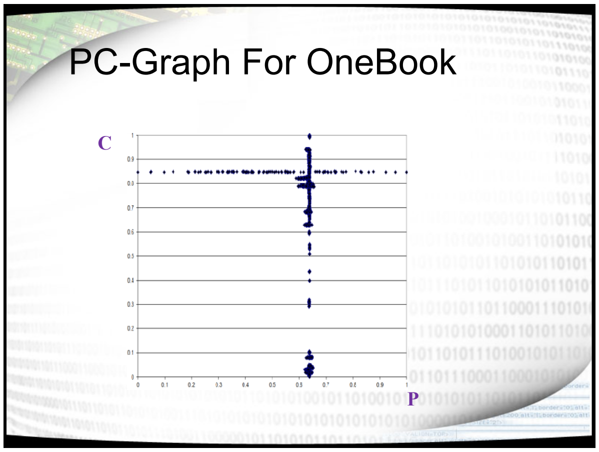
Capturing conceptual relations between the names of different identifiers is the essence of the comprehension metric

Program Comprehension Metric

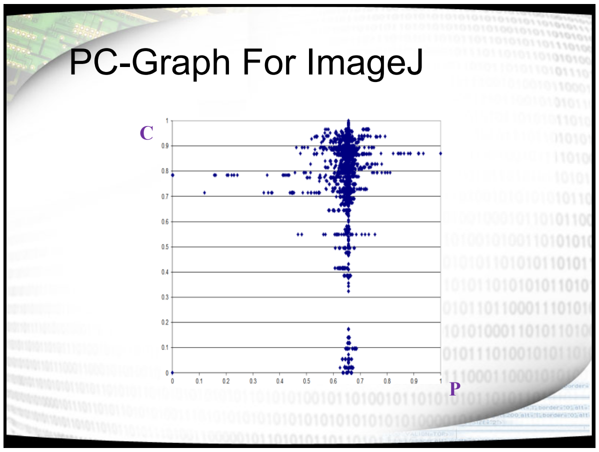
- Conceptual cohesion and coupling (C3) are based on the analysis of textual information in source code, expressed in comments and identifiers [Poshyvanyk'06]
  - C3 is the measure of the textual coherence of classes within the context of the entire system.
- We use Latent Semantic Indexing to analyze the textual information from source code and compute C3.



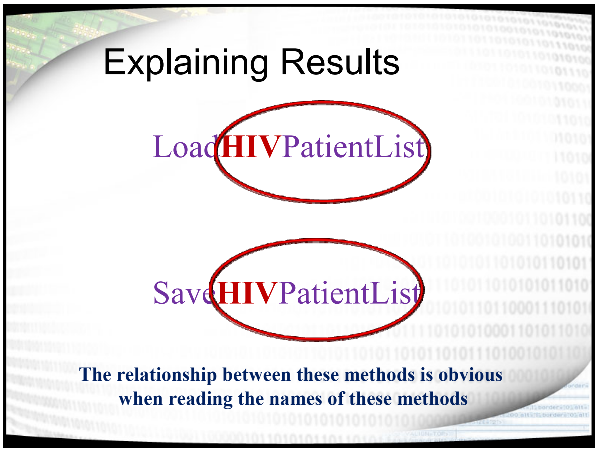
PC-Graph For PersonalPages



PC-Graph For OneBook



PC-Graph For ImageJ

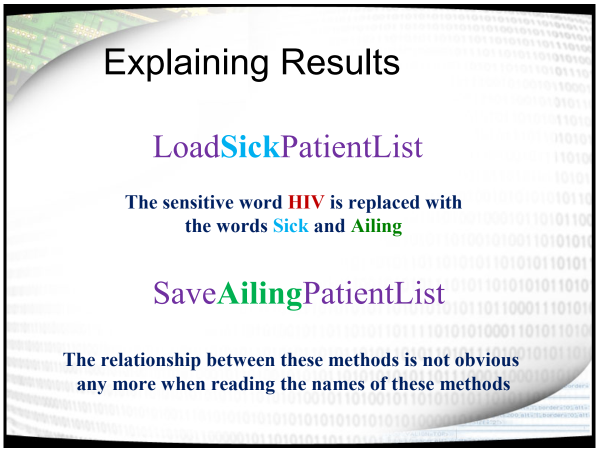


Explaining Results

LoadHIVPatientList

SaveHIVPatientList

The relationship between these methods is obvious when reading the names of these methods



Explaining Results

LoadSickPatientList

The sensitive word HIV is replaced with the words Sick and Ailing

SaveAilingPatientList

The relationship between these methods is not obvious any more when reading the names of these methods

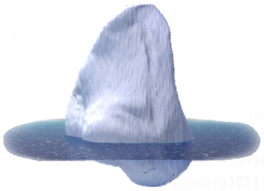
Looking Ahead

- Optimizing sensitive word replacement
- Ensuring privacy for evolving software
- Empirical evaluation
- Technology transfer

Existing Solutions to Protect Sensitive Information for SE tasks

- “*kb*-Anonymity: A Model for Anonymized Behavior-Preserving Test and Debugging Data” by Budi et al., PLDI’11
- “Better Bug Reporting With Better Privacy” by Castro et al., ASPLOS’08
- “Privacy and Utility for Defect Prediction: Experiments with MORPH” by Peters and Menzies, ICSE’12
- “Scrash: A system for generating secure crash information” by Broadwell et al., USENIX Security 2003.
- “Camouflage: Automated Anonymization of Field Data” by Clause and Orso, ICSE 2011.

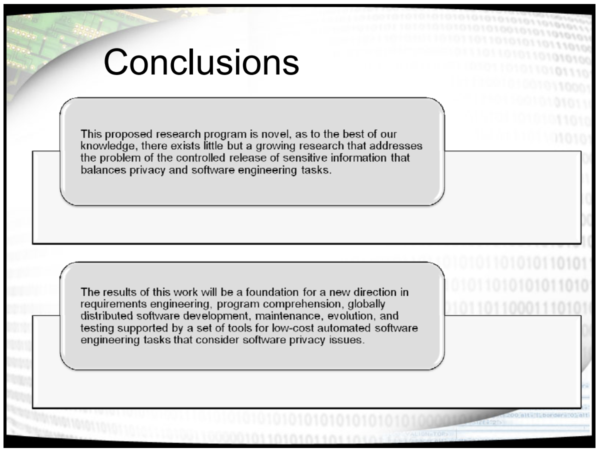
It Is Just the Tip of The Iceberg



Problems at the intersection of software development, distributed service provision, and data privacy to allow application owners to release their software artifacts to different service providers with guarantees that sensitive information is removed from the source code and these artifacts while preserving the utility of different software engineering tasks.

Looking Ahead

- Privacy ∩ program comprehension;
- Privacy ∩ distributed computing, including service-oriented apps;
- Privacy ∩ mining software repositories;
- Privacy ∩ performance engineering;
- Privacy ∩ fault tolerance;
- Languages that enables programmers to write code with controlled privacy.

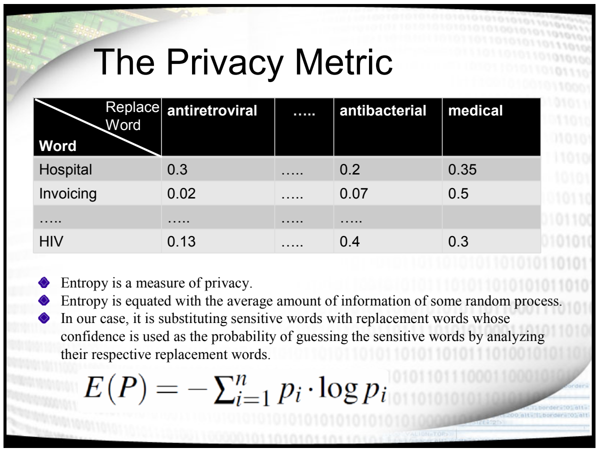


Conclusions

This proposed research program is novel, as to the best of our knowledge, there exists little but a growing research that addresses the problem of the controlled release of sensitive information that balances privacy and software engineering tasks.

The results of this work will be a foundation for a new direction in requirements engineering, program comprehension, globally distributed software development, maintenance, evolution, and testing supported by a set of tools for low-cost automated software engineering tasks that consider software privacy issues.

Email: drmark@uic.edu and denys@cs.wm.edu

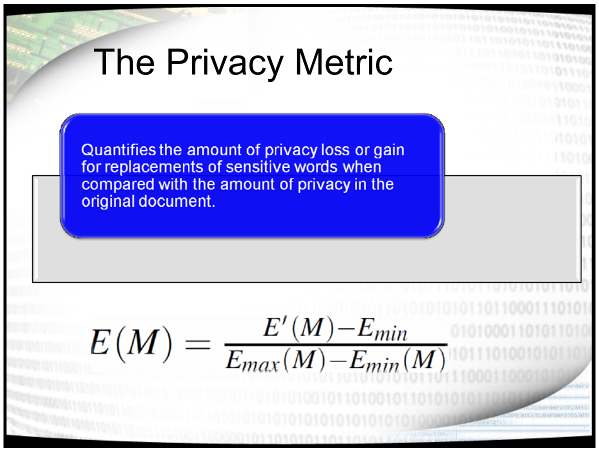


The Privacy Metric

Replace Word Word	antiretroviral	…..	antibacterial	medical
Hospital	0.3	…..	0.2	0.35
Invoicing	0.02	…..	0.07	0.5
…..	…..	…..	…..	
HIV	0.13	…..	0.4	0.3

- Entropy is a measure of privacy.
- Entropy is equated with the average amount of information of some random process.
- In our case, it is substituting sensitive words with replacement words whose confidence is used as the probability of guessing the sensitive words by analyzing their respective replacement words.

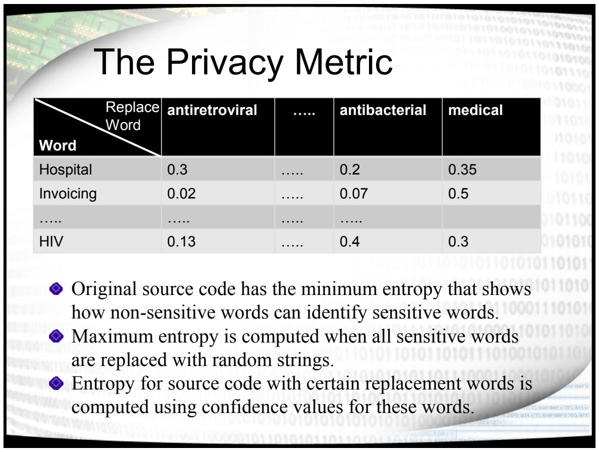
$$E(P) = -\sum_{i=1}^{n} p_i \cdot \log p_i$$



The Privacy Metric

Quantifies the amount of privacy loss or gain for replacements of sensitive words when compared with the amount of privacy in the original document.

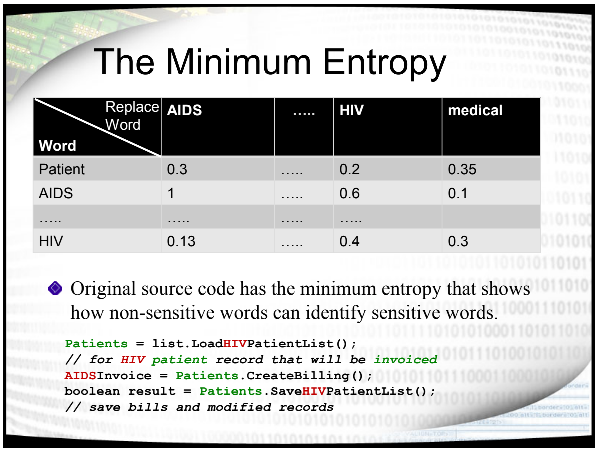
$$E(M) = \frac{E'(M)-E_{min}}{E_{max}(M)-E_{min}(M)}$$



The Privacy Metric

Replace Word Word	antiretroviral	…..	antibacterial	medical
Hospital	0.3	…..	0.2	0.35
Invoicing	0.02	…..	0.07	0.5
…..	…..	…..	…..	
HIV	0.13	…..	0.4	0.3

- Original source code has the minimum entropy that shows how non-sensitive words can identify sensitive words.
- Maximum entropy is computed when all sensitive words are replaced with random strings.
- Entropy for source code with certain replacement words is computed using confidence values for these words.

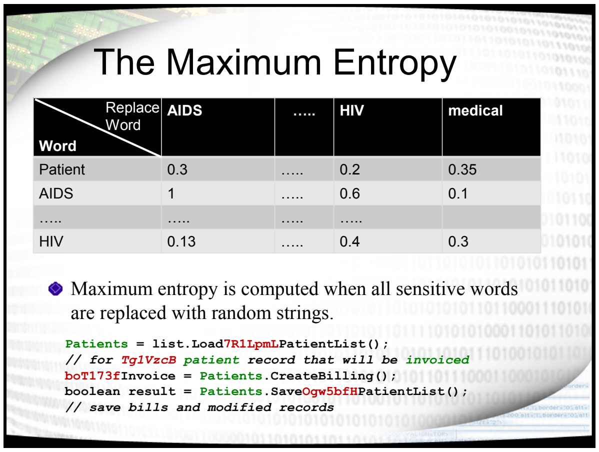


The Minimum Entropy

Replace Word Word	AIDS	…..	HIV	medical
Patient	0.3	…..	0.2	0.35
AIDS	1	…..	0.6	0.1
…..	…..	…..	…..	
HIV	0.13	…..	0.4	0.3

- Original source code has the minimum entropy that shows how non-sensitive words can identify sensitive words.

```
Patients = list.LoadHIVPatientList();
// for HIV patient record that will be invoiced
AIDSInvoice = Patients.CreateBilling();
boolean result = Patients.SaveHIVPatientList();
// save bills and modified records
```

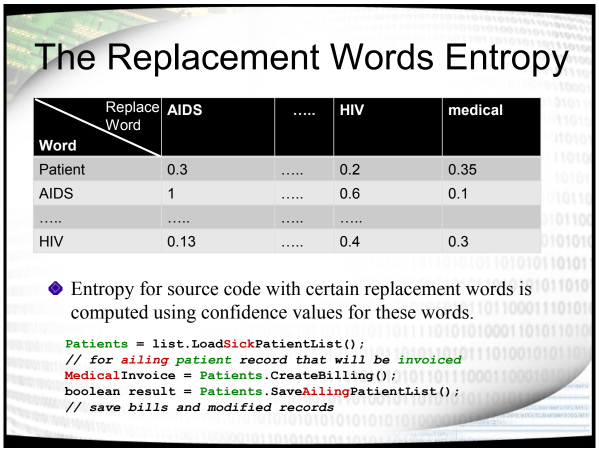


The Maximum Entropy

Replace Word Word	AIDS	.....	HIV	medical
Patient	0.3	.....	0.2	0.35
AIDS	1	.....	0.6	0.1
.....	.....	.....	.....	
HIV	0.13	.....	0.4	0.3

- Maximum entropy is computed when all sensitive words are replaced with random strings.

```
Patients = list.Load7R1LpmLPatientList();
// for Tg1VzcB patient record that will be invoiced
boT173fInvoice = Patients.CreateBilling();
boolean result = Patients.SaveOgw5bfHPatientList();
// save bills and modified records
```

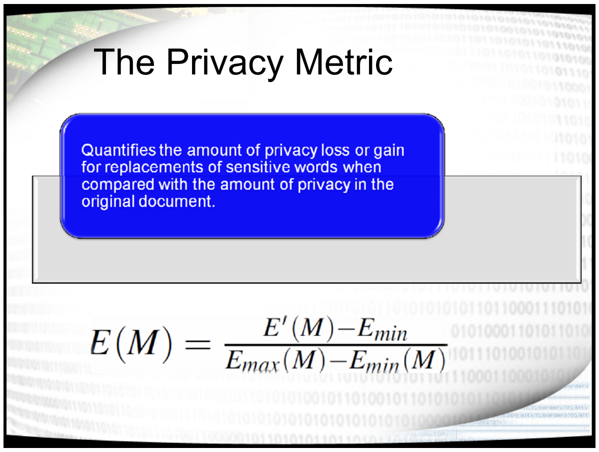


The Replacement Words Entropy

Replace Word Word	AIDS	.....	HIV	medical
Patient	0.3	.....	0.2	0.35
AIDS	1	.....	0.6	0.1
.....	.....	.....	.....	
HIV	0.13	.....	0.4	0.3

- Entropy for source code with certain replacement words is computed using confidence values for these words.

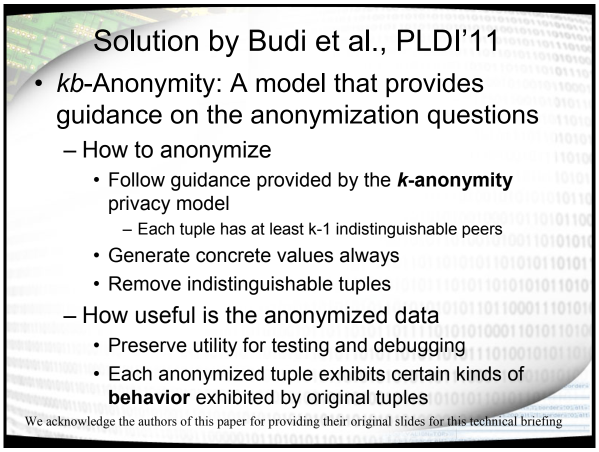
```
Patients = list.LoadSickPatientList();
// for ailing patient record that will be invoiced
MedicalInvoice = Patients.CreateBilling();
boolean result = Patients.SaveAilingPatientList();
// save bills and modified records
```



The Privacy Metric

Quantifies the amount of privacy loss or gain for replacements of sensitive words when compared with the amount of privacy in the original document.

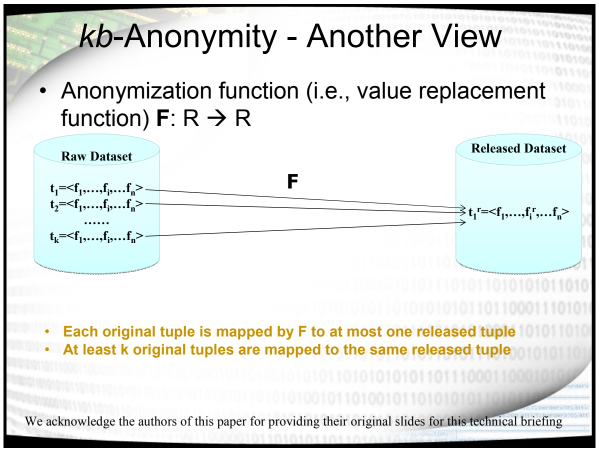
$$E(M) = \frac{E'(M) - E_{min}}{E_{max}(M) - E_{min}(M)}$$



Solution by Budi et al., PLDI’11

- *kb*-Anonymity: A model that provides guidance on the anonymization questions
  - How to anonymize
    - Follow guidance provided by the ***k*-anonymity** privacy model
      - Each tuple has at least k-1 indistinguishable peers
    - Generate concrete values always
    - Remove indistinguishable tuples
  - How useful is the anonymized data
    - Preserve utility for testing and debugging
    - Each anonymized tuple exhibits certain kinds of **behavior** exhibited by original tuples

We acknowledge the authors of this paper for providing their original slides for this technical briefing



kb-Anonymity - Another View

- Anonymization function (i.e., value replacement function) **F**: R → R

Raw Dataset

t_1=<f_1,…,f_i,…f_n>

t_2=<f_1,…,f_i,…f_n>

……

t_k=<f_1,…,f_i,…f_n>

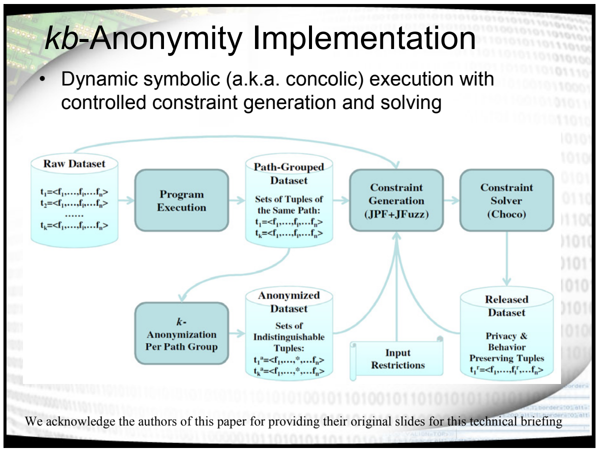
F

Released Dataset

t_1^r=<f_1,…,f_i^r,…f_n>

- **Each original tuple is mapped by F to at most one released tuple**
- **At least k original tuples are mapped to the same released tuple**

We acknowledge the authors of this paper for providing their original slides for this technical briefing



kb-Anonymity Implementation

- Dynamic symbolic (a.k.a. concolic) execution with controlled constraint generation and solving

Raw Dataset

t_1=<f_1,…,f_i,…f_n>

t_2=<f_1,…,f_i,…f_n>

……

t_k=<f_1,…,f_i,…f_n>

Program Execution

Path-Grouped Dataset

Sets of Tuples of the Same Path:

t_1=<f_1,…,f_i,…f_n>

t_k=<f_1,…,f_i,…f_n>

Constraint Generation (JPF+JFuzz)

Constraint Solver (Choco)

k-Anonymization Per Path Group

Anonymized Dataset

Sets of Indistinguishable Tuples:

t_1^a=<f_1,…,*,…f_n>

t_k^a=<f_1,…,*,…f_n>

Input Restrictions

Released Dataset

Privacy & Behavior Preserving Tuples

t_1^r=<f_1,…,f_i^r,…f_n>

We acknowledge the authors of this paper for providing their original slides for this technical briefing