IMPLEMENTING S4

SOURCE-CODE SECURITY SCANNING SERVICES

V0.5 2007 - DRAFT
BY DINIS CRUZ

TABLE OF CONTENTS

INTRODUCTION

These documents contain a mixture of: blue-prints, road maps, architectural overviews, templates and guidelines.

They were designed to help you (the reader) to kick start the provision of Source-code Security Scanning Services (S4) to external or internal clients[1].

The target audience of this document are:

- Security teams of large enterprises who have been mandated to implement such service
- Lead architects or senior developers of software houses who want to use S4[2] to reduce the number of vulnerabilities in their product and enforce security coding standards
- Security Consulting or Security Managed services companies who want to:
  - provide S4 to their clients (new revenue stream), or
  - improve an existent manual S4

Due to the complexity and interconnections of the applications to analyze, source-code security scanning is a non trivial process. Maximum care must be observed when adding new activities to that application's Software Development Lifecycle (SDL), since most development teams are heavily constraint by massive deadlines, features request, bugs to fix, etc…

It is highly recommended to use a 'baby step' approach to this S4 endeavour, in order to ensure that a solid, scalable and robust service is implemented. Although is very tempting to involve as soon as possible[3] the application's developers, it will fail in 99% of the cases. The recommended approaches are based on the gradual deployment of S4 in a phased, controlled and highly organized way, so that when developers are exposed to its results, they immediately see its value and embrace its beneficial capabilities.

HUMAN + TOOL

One theme that is presence throughout this document is the concept that the most effective and efficient method to perform source-code security analysis is to use both brains and automation (i.e. a '*person using a software application*' that is 'a *human using a tool*').

[1] The concept of *clients* is used to reefer to groups or companies that will use S4's services. These can be external (in the case of external commercial service providers) or internal (in the case of large companies which will integrate S4 as part of their managed security services)

[2] S4 will be used throughout this document to reefer to *Source-Code Security Scanning Services*

[3] With the objective to identify the vulnerabilities as soon as possible in the SDL

Large categories of security vulnerabilities can be easily be discovered and understood by non-application-security professionals (like developers or system architects) but certain common highly-exploitable and dangerous issues can only be discovered by experienced security professionals.

One analogy is performing agriculture tasks manually, versus doing it with a tool (cow or tractor)

vs

vs

Likewise you can fly like this,

vs

vs

It is important to note that the security skills and experience of the person (i.e. human) using the tool has a large impact on the assurance of final results, and due to a multitude of factors[4], the less-powerful combination of *'Developer + Tool'*[5] will not be able to successfully discover a large number of serious vulnerabilities in popular high profile enterprise and commercial applications (which are only discoverable by the *'Security Consultant + Tool'* combination.

THESE DOCUMENTS ARE DESIGNED AS TEMPLATES

This series of documents are provided as templates and it is expected that they will be customized and changed by the organization adopting them.

[4] Massive use of poorly understood Frameworks (from a security point of view), lack of security education, lack of security requirements, lack of successful malicious attacks

[5] Less powerful when compared with the best-of-both worlds mode of *'Security Consultant + Tool'* (not when compared with *manual source-code security reviews* or external *penetration testing*)

COPYRIGHT AND RIGHT OF USE

This information is provided under the XXX (TBD) license which grants all editing, reuse and modifying rights to its readers. It is expected that large sections of this document will be directly reused by its readers in their internal or external facing documents.

Credit and acknowledgement to Ounce and this document is appreciated but NOT mandatory.

NOTE ON VENDOR AGNOSTIC TONE

Although this document was created by Ounce Labs[6] Consultants and reflects the collective experience and wisdom of implementing these services on multiple clients, in order to increase the document's usability, all efforts were made to make the text vendor agnostic (i.e. without direct references to Ounce software and Ounce Labs).

S4 IMPLEMENTATION PACK STRUCTURE

This implementation pack is made 4 for distinct sections:

- S4 - Service Definition
- S4 – Engagement Model
- Implementing S4
- Security Consultant's guide to Ounce

The first two documents ('*S4 – Service Definition*' and '*S4-Engagement Model*') are designed contain the final definition and engagement model of the S4.

The documents should be approved by senior management and distributed[7] to existent or potential S4 clients.

The remaining two are designed to be used by the S4 team during and after service implementation.

[6] Ounce Labs is a commercial vendor of a source-code security scanning software called *Ounce* (see http://www.ouncelabs.com for more details)

[7] After being customized for the specific S4 deployment case

S4 - SERVICE DEFINITION

The *[COMPANY NAME] Source-code Security Scanning Service* (S4) offers the provision of source-code security analysis services to organizations or groups[8] who wish to gain a better understanding (and visibility) of the security vulnerabilities that might exist in their mission critical applications.

The objective of the S4[9] service is to have a positive impact on the Application's risk profile.

To achieve that objective S4 will:

- identify vulnerabilities,
- aid business owners & solution architects to understand its impact,
- help with remediation,
- confirm issues closure,
- train developers and
- write '*best practices'* guideline documents.

S4 makes extensive use of commercial Source-code Security Scanning software, which automates the code scanning process and exposes multiple interfaces for information reporting.

SERVICES PROVIDED

The following services are provided by S4:

- **Security Engagements** – S4 team performs a security engagement on client supplied source code.

  There are 5 different types of Security engagements available (Depending on the resources provided by the client and the application's risk profile):

  - *T1: Low Hanging Fruit - Partial,*
  - *T2: Low Hanging Fruit*
  - *T3: Normal Assessment*
  - *T4: Security Review*
  - *T5: Comprehensive Security Review*

[8] Also referred as *Clients*

[9] S4 will be used throughout this document to reefer to *Source-Code Security Scanning Services*

- **Consulting & Support**– S4 security team consultants help external (to S4) teams with their Security reviews, Threat Modelling and SDL integration. There are several types of engagement available
    - *'first couple days' engagement* (S4 only participates on the first couple days of an engagement in order to ensure that everything is ok)
    - *2nd level security review* (after the development teams have performed their analysis)
    - *Remote Support* – remote support to teams that are doing source code scanning on their applications

- **Vulnerability remediation assistance**

- **Training** – S4 consultants will perform regular knowledge transfers and train developers & solution architects on:
    - the chosen source-code scanning software
    - how to break applications (aka security testing)
    - how to write secure applications (aka defending applications)

- **Management Reports**
    - Monthly distributed to top level management and owners of affected applications
- **'*Best Practices*' documents**

SLA & KPI

In order to provide the maximum value to the business, the S4 service will have the following SLAs (Service Level Agreement):

SERVICE LEVEL AGREEMENT

- Find the most dangerous vulnerabilities in the source code analyzed (within the time allocated)
- Provide N[10] security review slots and N consulting slots per month[11]

[10] Number of slots depends on the number of resources available

[11] For applications that match the selection criteria

- Provide email support on XXX[12] Time zone with a response time of 2 days.
- Answer to new project requests within 5 days
- Deliver report to business owners 5 days after project engagement completed
- Provide monthly management reports (by the 5th working day of the following month)
- Re-test applications with 10 days of request received and updated source code drop

And be measured by the following KPIs (Key Performance Indicators):

KEY PERFORMANCE INDICATORS

- Number of engagements
- Number of groups or organizations using Source-code Security Scanning internally
- Number of Vulnerabilities (High and Medium) discovered
- Number of Vulnerabilities (High and Medium) Resolved or Mitigated
- Number of individuals trained
- Number of documents created and distributed

SELECTION CRITERIA

[This section is more relevant to internal security teams, unless the external consultant's were hired to perform S4 engagements on applications that match particular criteria]

The following criteria will be used to select applications that automatically are entitled to S4 engagements:

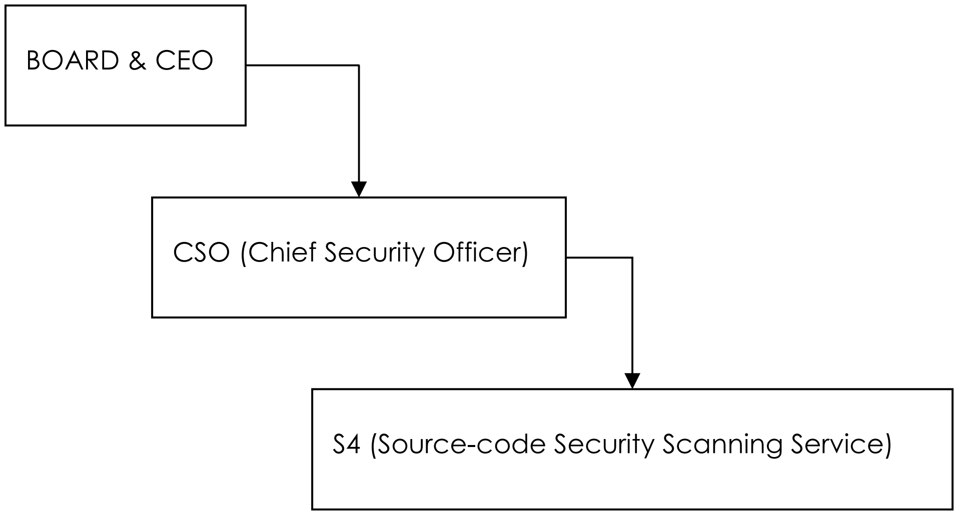
- Applications that are exposed to the Internet
- Mission Critical Applications which have been marked with a 111 CIA rating (1 for Confidentiality , 1 for Integrity and 1 for Availability)
- Applications that are sold to final customers
- Applications which have regulatory security requirements (for example PCI)

[12] Geographical location of the support team

S4 IN ORGANIZATIONAL CHART

[This section is more relevant to internal security teams. For external consultants it might be relevant to exposed WHO hired them

S4 is part of the CSO security initiatives and the following ORG Chart shows the chain of command:


BOARD & CEO
CSO (Chief Security Officer)
S4 (Source-code Security Scanning Service)

S4 FUNDING

[this section is more relevant to internal security teams, since external consultants will basically be charging for these services]

This service is independently funded by the CSO budget which means that the cost of software used and hired Security Consultants are provided at NO cost to the scanned application's owners.

The following are some exceptions where application owners will incur in extra costs:

- Applications profile does not match the *Selection Criteria*[13].
- Need to have an external party performing the test[14]
- Test deadline not compatible with the S4 delivery capability
- Large project which needs more resources than the ones made available by S4

[13] As defined at the *Selection Criteria* section of this document

[14] Some regulatory compliance requirements mandate that tests are provided by fully independent external parties

NON S4 USE OF AUTOMATED SOURCE-CODE SECURITY SCANNING SOFTWARE

In the cases there development teams have the resources to use the same Source-code security scanning software used by S4, the license will be:

i) centrally provided by S4 (based on previous negotiations with the commercial vendor)

ii) independently purchased

SOFTWARE LICENSING ENGAGEMENT MODELS

There are two licensing models available for the chosen commercial Source-code Security Scanning Software:

- Single application scan short term license (usually for 1 to 2 weeks time period)
- Multi application long term license (usually 1 to 3 years)

REMEDIATION PRACTICES

Part of S4's unique value to the business is its capability to provide detailed remediation information and guidance to developers. This will enable those developers to gain a much better understanding of the root causes of the coding or design practices that created the identified vulnerabilities.

REMEDIATION ADVISE

The remediation advice will be based on the *Knowledge Base* provided by the chosen source-code security scanning software and (when required) on custom recommendations written by the S4 Security Consultants.

VULNERABILITIES ARE BUGS

In order to simplify the management of vulnerabilities disclosed, whenever possible, High Risk vulnerabilities will be treaded as 'High Priority' Bug, and (if possible) automatically feed into the existing bug tracking solution.

ISSUE TRACING

S4 will keep track of the issues discovered and provide monthly reports on the outstanding open issues (i.e. not resolved).

An S4 issue can be closed in two ways:

- The application owners can ask for a retest once major issues are closed[15]
- When S4 performs an engagement on the next release of the application

THE WAF (WEB APPLICATION FIREWALL) OPTION

In the cases where security vulnerabilities have been discovered in mission-critical applications but source code changes are not possible in the desired time-scales[16] , a valid alternative is to use the list of vulnerabilities discovered by S4 to create WAF (Web Application Firewalls) rules which are specifically designed to protect against the vulnerabilities discovered.

This is the best use of WAFs since they are being used to 'fix' specific issues and can be deployed in very non-intrusive and impact-less way[17].

Note that it is the Security Consultant responsibility to write and test the WAF rules, since only he/she has the technical skills to confirm that the deployed rules are indeed fixing the application (and not only stooping one variation of the attacks)

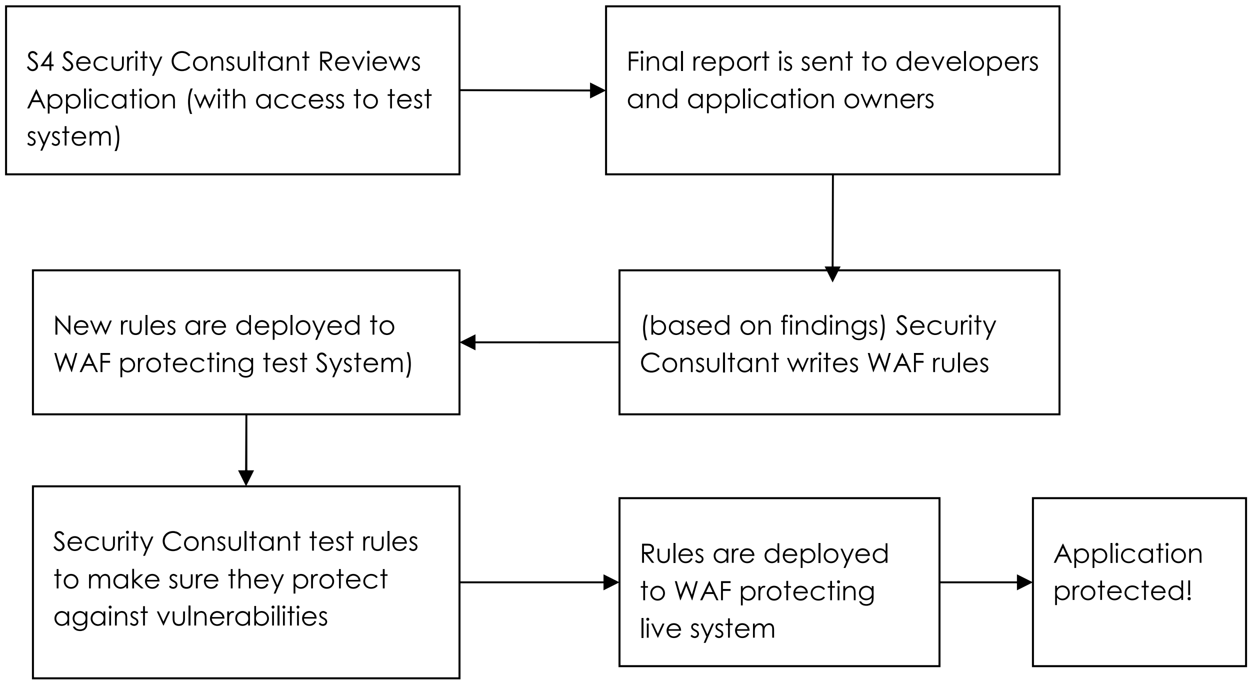
[15] Due to the exposure of S4 reports to senior management there is considerable internal pressure on application owners to address the security issues marked with a High Risk rating

[16] Some reasons for the inability to quickly patch applications with known vulnerabilities:

- Long patch cycles due to extended QA,
- non availability of original source code,
- development team focused in developing new version,
- lack of attacks doesn't justify development effort,
- original development brief didn't include security requirements so fixing the existing code base would be a very expensive exercise

[17] The common problem with WAFs is caused by the attempt to use WAFs to protect an entire application using their 'learning' modes and application-wide rules. This quickly becomes unmanageable in complex (i.e. real-world) applications, due to the size and complexity of the rules created, and (probably more importantly) the fact that even with those rules the application will be vulnerable to certain types of vulnerabilities. Using WAFs in 'virtual patching mode' is a the best-of-both-world situation where maximum value is obtained with minimum business impact

The following diagram shows the workflow from '*S4 engagement*' to '*Application protected*'.


S4 Security Consultant Reviews Application (with access to test system)
Final report is sent to developers and application owners
(based on findings) Security Consultant writes WAF rules
New rules are deployed to WAF protecting test System)
Security Consultant test rules to make sure they protect against vulnerabilities
Rules are deployed to WAF protecting live system
Application protected!

REPORT WRITING

Depending on the number of findings and the profile of the application different reports will be produced:

- High level report with one detailed example per vulnerability type
- Full reports containing detailed information of all vulnerabilities discovered (risk rating, category, file affected and line number)
- Partial reports filtered by Risk rating, Vulnerability type, APIs, etc…
- XML files which can be imported into development environment plug-ins[18] in order to allow developers to fully understand the reported issue (these can also be filtered according to a wide rage of criteria)

[18] Provided by the Source-code Security Scanning software

MANAGEMENT REPORTS, BEST PRACTICES AND GUIDELINES DOCUMENTS

The S4 team is uniquely positioned to provide high-value reports that measure and quantify the security risk profile of the applications tested.

Targeted at top level management, monthly reports will be created, containing high-level analysis of the issues discovered during the previous month and the accumulative status of outstanding issues.

Maximizing its access to real-world vulnerabilities in real-world code, S4 will identify common problem areas (for example the top 10 most common vulnerabilities) and produce documentation targeted at educating all relevant parties about them.

SECURE CODING STANDARDS

Once established and fully operational, S4 will work with application architects and system owners on the development of common *Secure Coding standards and guidelines*.

These will explicitly define best practices for common development requirements and non-acceptable practices (for example banned APIs)

INTEGRATION WITH OTHER SECURITY SERVICES

Whenever possible and required, S4 will integrate its data and resources with other existing internal security services (for example automatic network scanning or global vulnerability security tracking systems)

RISK RATING

In order to ensure a common standard in the classification of the issues discovered, the following table contains the criteria used when rating an exploitable security issue.

Risk rating	Description	Example of vulnerabilities that match this category
High	{TDB}	{TDB}
Medium	{TDB}	{TDB}
Low	{TDB}	{TDB}
Info	{TDB}	{TDB}

SDL INTEGRATION

There are 3 places where source code scanning can have the most impact in an organization SDL:

- Regular Scans during development (during source repository check-in or weekly)[19]
- Comprehensive security reviews on product / application milestore (usually tied in with QA)
- Developer Training

It is out of scope of this document to provide in depth descriptions about this topic (see Ounce document (PDF) : "*Source Code Vulnerability Testing in the SDL*")

[19] See "*Developer 'Big Red Button' modes*" section in this document

S4 – ENGAGEMENT MODEL

This section covers the practical elements for engaging S4, and is designed to help prospective S4 users to understand how S4 works (i.e. its engagement model).

Titles in this section:

- S4 Engagement Types
- Project Brief
- Security Handling confidential information
- The need for compileable source code
- The need for Test environment
- Security consultant and developer workflows
- ‘Human’ vs ‘Human + Tool’
- Outsourcing Security Tests
- S4 incident response capabilities
- Training
- S4 contact details
- References for further reading

{… add small description to each title … }

S4 ENGAGEMENT TYPES

The type of service provided on each engagement depends on the resources provided by the client.

The 5 engagement types available are:

- ‘Low Hanging Fruit’ - Partial
- ‘Low Hanging Fruit’
- Normal Assessment
- Security Review
- Comprehensive Security Review

The following table provides a high level view of the requirements of each engagement type:

ID	Engagement Type	Partial Source code provided	Full Source Code provided		Test environment provided (aka Black Box)	# man days allocated [20]	Minimum Skills required	Assurance of results [21] [22]
			Application to test	All major dependencies[23]				
T1	**‘Low Hanging Fruit’ - Partial**	X				1 to 2	Developer	Moderate
T2	**‘Low Hanging Fruit’**		X			1 to 2	Developer	Moderate
T3	**Normal assessment**		X	X		3 to 5	Developer	Moderate
T4	**Security Review**		X	X		3 + 1 day per 50 KLOC[24]	Security Consultant	High
T5	**Comprehensive Assessment**		X	X	X	3 + 2 day per 50 KLOC	Security Consultant	Highest

[20] The time required to create the source-code scanning environment is excluded from this value

[21] Without a live system to test the vulnerabilities discovered, in some cases, it will be hard for the security consultant to commit 100% that an issue is an remotely exploitable vulnerability

[22] The time allocated to an engagement also has a direct impact on the results assurance

[23] For example the Data and Presentation APIs (note that APIs that belong to J2EE or .NET Framework, are not included in this list (i.e. are not expected to be made available))

[24] KLOC = K Lines Of Code = 1000 Lines Of Code

SELECTING AN ENGAGEMENT TYPE

The factors that define which *engagement type* are available:

- Amount of source-code provided
- Availability of test environment
- Number of man days allocated to the project[25]
- Skill of testers
- Profile of target application

LIMITATIONS OF EACH ENGAGEMENT

When selecting an engagement it is important to understand their limitations. As an example, the following table describes the % of security issues that are expected to be discovered with each one:

ID	Engagement Type	Minimum Skills required [26]	Assurance of results	% of existent security issues expected to be discovered
T1	**'Low Hanging Fruit' Partial**	Developer	Moderate	**10% - 30%**
T2	**'Low Hanging Fruit'**	Developer	Moderate	**20% - 40%**
T3	**Normal assessment**	Developer	Moderate	**30% - 50%**
T4	**Security Review**	Security Consultant	High	**40% - 60%**
T5	**Comprehensive Assessment**	Security Consultant	Highest	**80% - 90%**

[25] The number of days available for each engagement depend on the profile of the target application, the number of resources available at S4 and the budget available (since S4 can always hire temporary extra resources)

[26] Although under normal circumstances S4 will deliver these 5 types of engagements using a Security Consultant (not a Developer), the assurance level and the % of issues discovered will still be low in the first 3 types ('*Low Hanging Fruit' – partial*, '*Low Hanging Fruit*' and '*Normal Assessment*')

WHEN SOURCE-CODE SECURITY SCANNING IS INTEGRATED IN THE SDL

In the cases where source-code scanning is integrated an application's SDL, the affected development team will have direct access to the chosen source-code security scanning software, and (depending on the resources available) will perform their analysis on a regular basis:

Their engagement type will be similar to S4's:

- *'Low Hanging Fruit'* or *'Normal Assessment'* - depending on the number of days allocated
- *'Security Review'* – in the cases were a member of the development team has security experience and knowledge.

Important Node: The fact that individual development teams are able to scan their code independently does NOT automatically imply that those applications are except to S4 engagements [27].

Mission critical applications still require a *'Comprehensive Review'* type engagement, although with a smaller number of days allocated (when compared with applications without internal use of source-code scanning tools)

PROJECT BRIEF

The '*Project Brief'* is a document jointly created by S4 and organization requesting the assessment.

This document is originally distributed to prospective clients who are expected to fill-in as much information as possible during the first 'official' engagement request.

The Project Brief contain the following sections:

- Project details
- Contact details
- Engagement Type (& Resources made available)
- Code Compilation requirements
- NDA
- Timescales and deadlines
- Report
- Sign-off criteria

[27] Assuming that these applications conform with the *Section Criteria* as described in this document

PROJECT DETAILS

- Project Name
- Project description
- Security requirements
- Technical details of applications to review
    - Existing Modules
        - Lines of source code in each
    - External dependencies
        - Is source code available for these projects
    - Documentation
- Total size of source code to analyze
- Project Phase (Development, Pre-Release, Post-Release)
- All security related documentation:
    - Thread Modeling
    - Previous security reports (Pen-test, source code scans)
    - Application Security documentation (explaining the security features)
    - Attack surface mapping (all the inputs points)

CONTACT DETAILS

- Business owner
- Lead Architect
- Lead Developers
- Security team contact
- IDS[28] team contact (if relevant)

[28] IDS = Intrusion Detection System

ENGAGEMENT TYPE (& RESOURCES MADE AVAILABLE)

For every application reviewed under the current engagement list the engagement type and the resources made available

CODE COMPILATION REQUIREMENTS

- Confirm all that will be required for the Security Consultants to be able to compile the source-code provided:
  - Development environment
  - Build environment
  - External dependencies
- If possible also explicitly define the required steps to build a local 'test environment'
  - Database required
  - Location of 'test data database creation' scripts
  - Open Ports

NDA

- Clarify the need or not to have an NDA in place
- See NDA example at …..

TIMESCALES AND DEADLINES

- Project start
- Project end
- Regular updates schedule (if required)
- Report delivery date

REPORT

- List expected report contents
- Define what happens if High Security vulnerabilities are discovered during engagement (i.e. when the application owner be notified ASAP?)

SING-OFF CRITERIA

- Completion Criteria
- Success Criteria

SECURELY HANDLING CONFIDENTIAL INFORMATION

SOURCE CODE DROPS

It is very important to securely retrieve and store the provided source-code (or source-code repository account details).

REPORTS WITH VULNERABILITY INFORMATION

In the cases where High security vulnerabilities were discovered, S4 will make all possible efforts to securely deliver the report (& exported XML files) to the relevant parties[29].

SUPPORTED ENCRYPTION METHODS:

There are 5 encryption methods currently supported by S4:

- Internal email system when using '*email encryption*' mode
- TrueCrypt
- PGP

[29] Since these findings can be very valuable in the hands of malicious attackers.

- WinZip (latest version)
- SSL

THE NEED FOR COMPILEABLE SOURCE CODE

A very important requirement to perform source code analysis is the need to have access to all relevant source code.

Source-code analyzers behave like compilers where they use compiler technology to model the code under review and understand the code and data flows.

For this reason it is critical for these tools to have access to the developer's development tool environment[30] and all dependencies.

It is the responsibility of the target application owner (who is engaging with S4) to ensure that all source-code provided can be compiled by the S4 team.

THE NEED FOR TEST ENVIRONMENT

In order to validate the results discovered during the source-code analysis it is very important that the security consultant is given access to the target application's test environments (i.e. fully operational version of the target application with test data)

The Security Consultant will also use this access to the live system to ensure that it has total visibility to the application's attack surface, input validation layers, and authentication / authorization frameworks.

It is the responsibility of the target application owner (who is engaging with S4) to ensure that a test environment populated with test data made available to S4 Security Consultants, and, that all required account details have been provided[31].

[30] For example Visual Studio, Eclipse, Rational or compilation scripts (ant, maven, nmake)

[31] In order to properly effectuate authorization tests on the provided test environment, S4 Security Consultants will need access to 2 different accounts per major authorization role. For example if there are 3 role-groups (Viewers, Editors and Admin), then S4 will need 6 different accounts (2 x Viewers, 2 x Editors and 2 x Admins)

SECURITY CONSULTANT AND DEVELOPER WORKFLOWS

Although it is very common to obtain high quality results using the out-of-the-box rule set[32] , the maximum value is obtained when the software is used by knowledgeable and experience Security Consultants.

'BIG RED BUTTON' OR 'JUST LOOK AT WHAT YOU UNDERSTAND AND IGNORE THE REST'

The 'Big Red Button' is a common analogy to represent the mode where the user only presses 1 button and immediately gets the desired results (without the need for major customizations or changes).

One could also think of this mode as '*Just look at what you understand and ignore the rest*'. This is usually the method recommended to developers without application security experience or time[33].

The objective is to maximize the developer's resource (and attention span) by only 'asking' him/her to spend valuable resources (i.e. time) addressing issues which they can understand.

It is important to note that these tools are designed to catch ALL types of security issues, from the simple to the more complicated. This creates a wealth of information which is very valuable to the Security Consultants but overwhelming to Developers.

Using an *airplane* analogy:

What the developer wants to see:

What the Security Consultant wants to see:

This is why it is highly recommended for developers to run the tool with selected filters applied so that they are only exposed to information they can understand.

For Security Consultants, the equivalent case of 'Big Red Button' occurs when not enough time is allocated to perform an analysis[34].

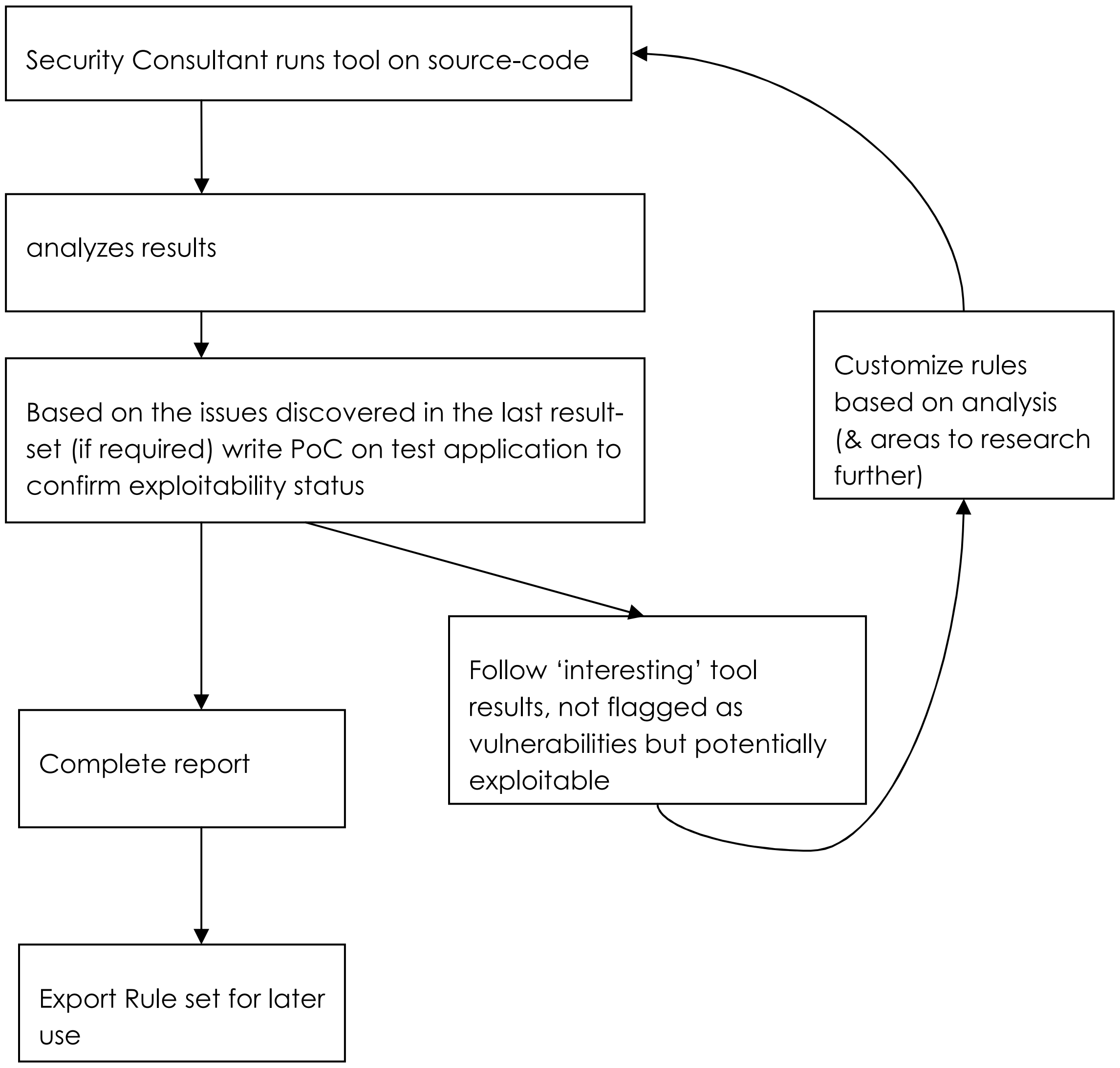
[32] Specially if the target application is using common frameworks who have already been mapped (by the Source-code Security Scanning software vendor) AND the application has not been securely reviewed before

[33] Since it is unrealistic to expect a developer to fully understand all data generated by the chosen tool

[34] see 'Low Hanging Fruit' mode in *S4 Engagement Types* section in this document

SECURITY CONSULTANT WORKFLOW

The following diagram shows a normal workflow of a security consultant when using the tool (the key concept is the constant refining of the rule-set until the tool has maximum visibility to the application's specific design, architecture and security sub-system[35]):


Security Consultant runs tool on source-code
analyzes results
Based on the issues discovered in the last result-set (if required) write PoC on test application to confirm exploitability status
Complete report
Export Rule set for later use
Follow 'interesting' tool results, not flagged as vulnerabilities but potentially exploitable
Customize rules based on analysis (& areas to research further)

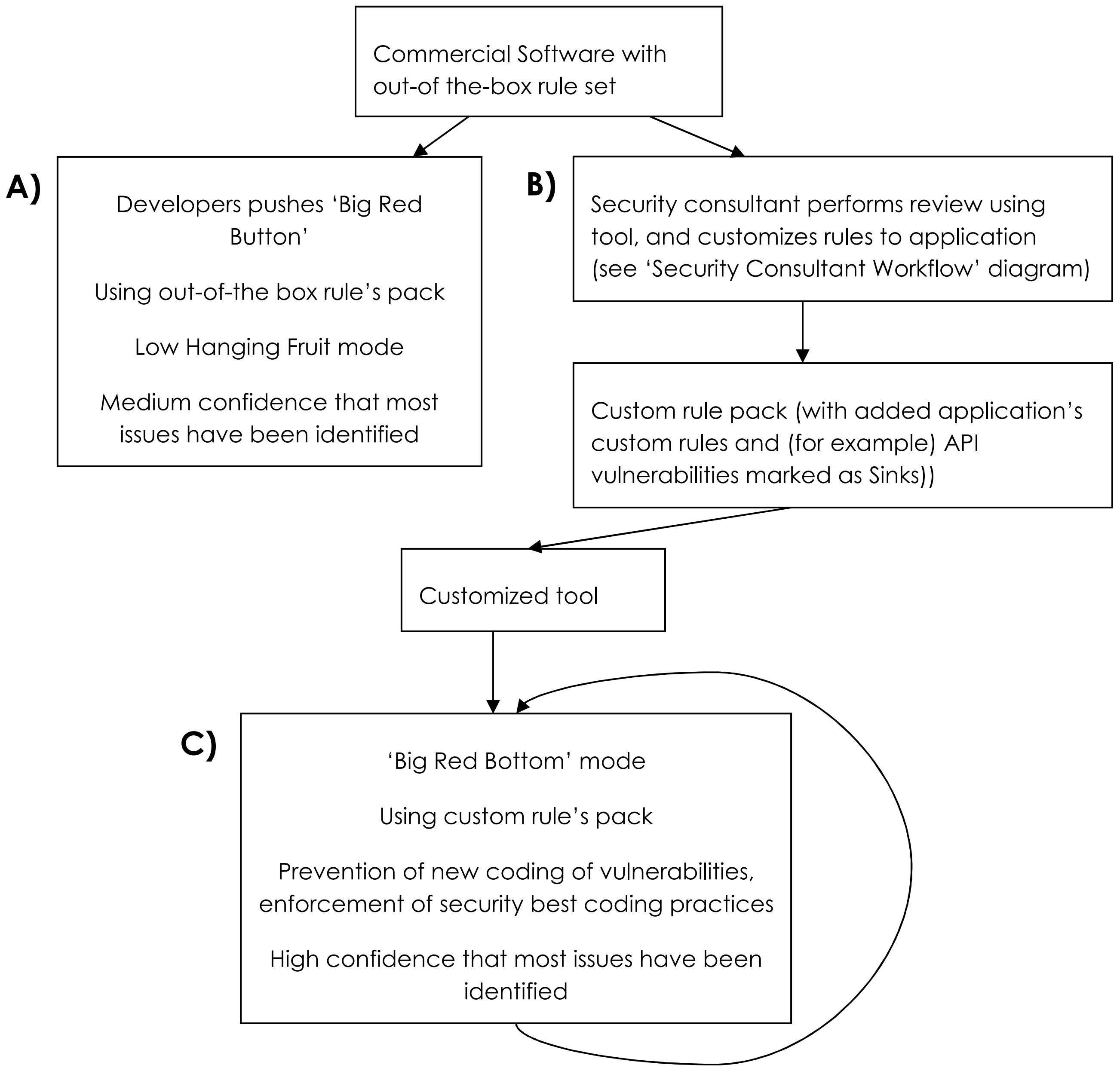
[35] Although it would be ideal if this customizations could be 100% automatically detected by the tools, given the complexity and interconnections of targeted applications, in 2007 there are no tools with those capabilities. All tool vendors are making dramatic advances in their scanning capabilities and with wide spread usage of these technologies it is expected that in a couple generations Source-code Security Scanning software will reach a high level of automation where the customization work required today will be dramatically reduced.

DEVELOPER 'BIG RED BUTTON' MODES

The following diagram shows the two different 'Big Red Button' modes available to developers (which do not have the same security knowledge and experience as Security Consultants).

The 3 scenarios shown are:

- A) Developer uses tool with out-of-the-box rule set
- B) Security Consultant uses tool in engagement and creates custom rule-set
- C) Developer uses tool with customized rule-set



Commercial Software with out-of the-box rule set
A)
Developers pushes 'Big Red Button'
Using out-of-the box rule's pack
Low Hanging Fruit mode
Medium confidence that most issues have been identified
B)
Security consultant performs review using tool, and customizes rules to application (see 'Security Consultant Workflow' diagram)
Custom rule pack (with added application's custom rules and (for example) API vulnerabilities marked as Sinks))
Customized tool
C)
'Big Red Bottom' mode
Using custom rule's pack
Prevention of new coding of vulnerabilities, enforcement of security best coding practices
High confidence that most issues have been identified

The main advantage of the B) → C) model is its scalability since the number of security consultants available is very small when compared with the number of security engagements required.

In this model, only the first scan needs to be performed by the Security Consultant with subsequent ones performed by the Developers themselves.

HUMAN VS 'HUMAN + TOOL'

One area of confusion in the industry is the value proposition of source-code scanning software when compared with a security consultant (aka: a *human*) that performs manual (or scripted) reviews.

The problem lies in the fact that it is claimed that tools can replace humans, where the real case is that the more knowledgeable and experienced the human is, the more value can be extracted from these tools.

Following the theme of the previous analogies shown throughout this document, the following pictures show two methods to go to the North Pole:

or

Using powerful tools does make a massive difference in the speed, quality and effort required to achieve the desire objectives[36].

Other analogies:

- Notepad vs Elipse vs Visual Studio
- Cooking on Fire vs Aga
- Swimming vs Sail Boat vs Motor Boat
- Hand Writing vs Type Writter vs Computer
- Abacus vs Calculator

[36] As with the north pole journey, the 'manual' (by foot) approach will still deliver the same result as the automatic (by car). The difference is in the time and effort required to get there.

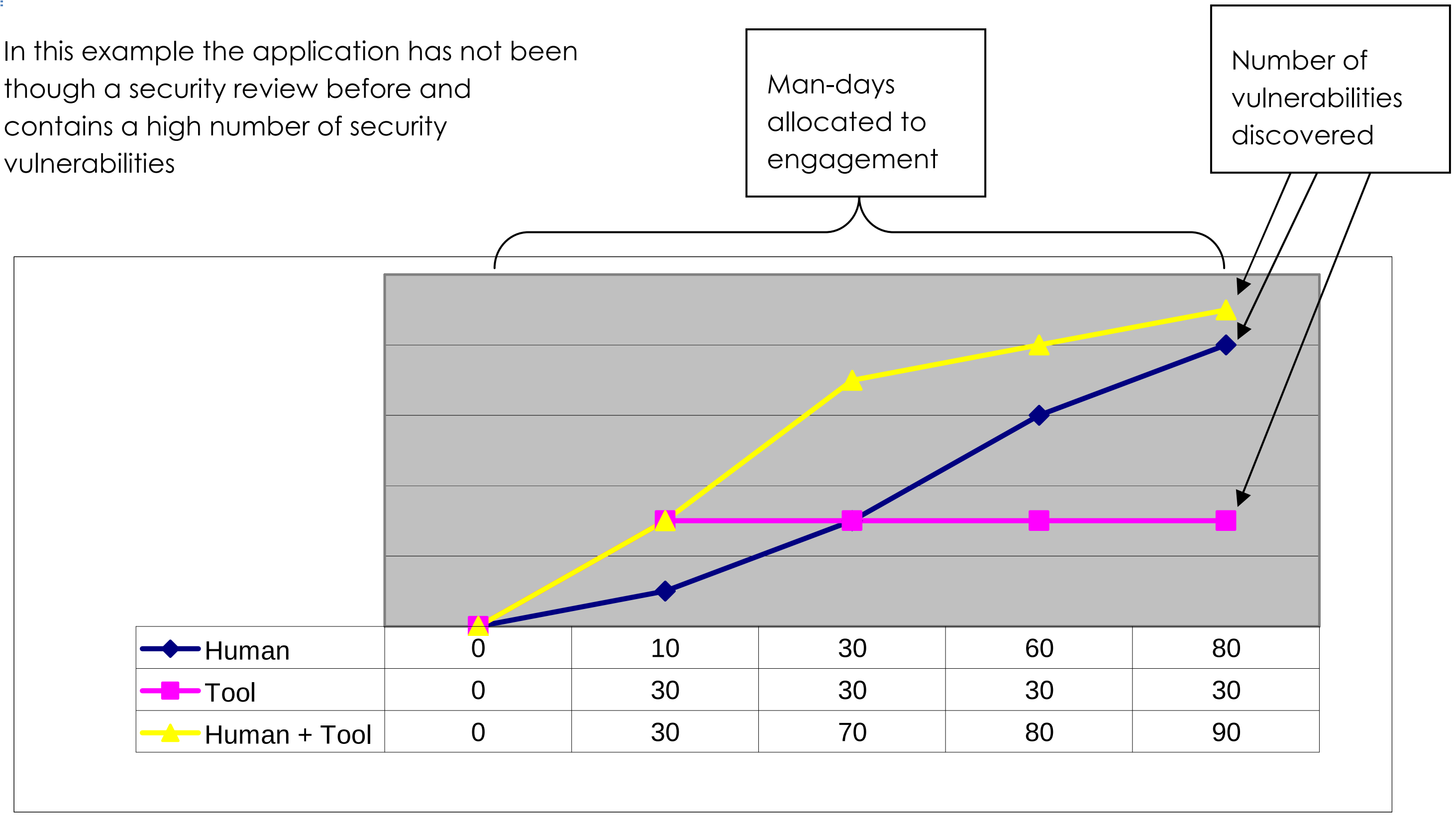
‘HUMAN + TOOL’ PERFORMANCE

The following graphs show projected numbers of vulnerabilities discovered (during a certain period of time) using 3 different methodologies:

- **Human** – Security Consultant analyzing source-code without the recourse of specialized tools (only home grown scripts)
- **Tool** – Use of source-code security scanning software out-of-the-box with no customization or changes to the default settings
- **Human + Tool** – ‘Best of both worlds’ scenario where the Security Consultant uses his brain to understand the data provided by the tool (which is recursively customized based on the results discovered)[37].

INSECURE APPLICATION

In this example the application has not been though a security review before and contains a high number of security vulnerabilities



Human	0	10	30	60	80
Tool	0	30	30	30	30
Human + Tool	0	30	70	80	90

(vertical axis: Number of vulnerabilities discovered , Horizontal axis: time spent on engagement)

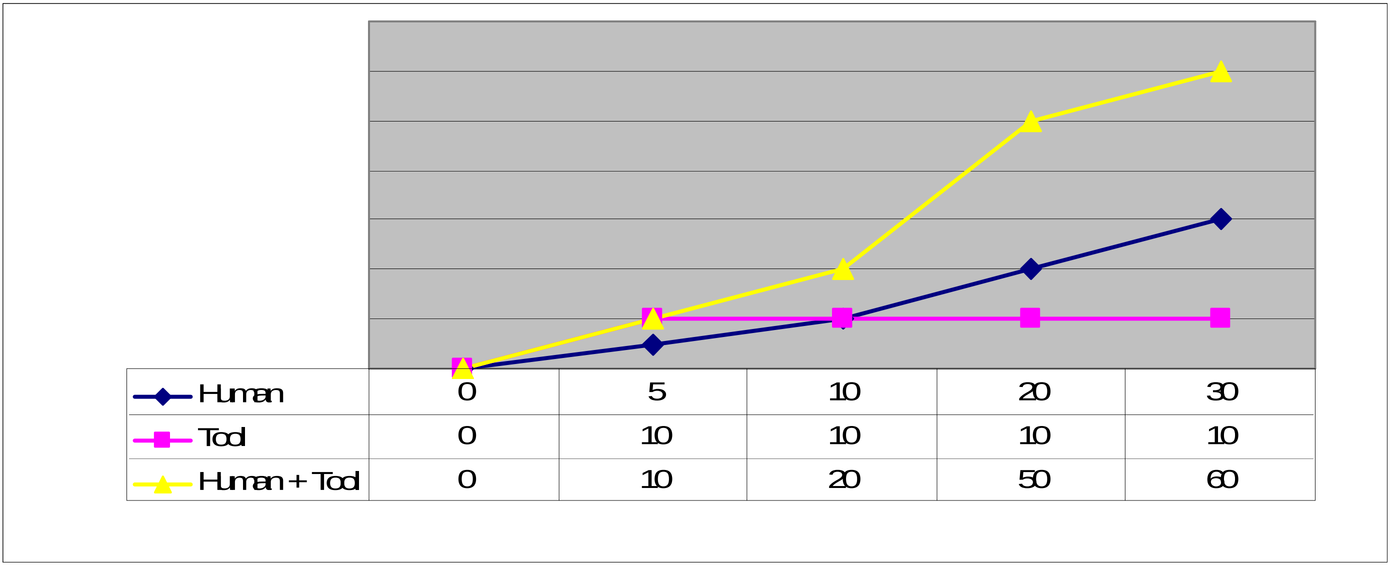
Some observations on this scenario:

[37] See diagram in ‘Security Consultant Workflow’ section in this document

- The *'Tool'* has as initial edge against the *'Human'* but with time the *'Human'* discovered a higher number since the number of issues highlighted by the *'Tool'* is static versus the increasing number discovered by the *'Human'*
- The 'Human + Tool' combination is the most efficient since it benefits from the initial higher number of issues discovered by the *'Tool'* and the constant customizations. This allows for the number of issues discovered to peek much earlier when compared with the *'Human'* results
- Given enough time, the *'Human'* will discover almost has any issues as the ones discovered by *'Human + Tool'*
- In this graph the best ROI value is the 70% mark of the *'Tool + Human'*

SECURE APPLICATION (BUT NO SDL INTEGRATION)

In this next example, the application has been through a security review process before, the developers feel they understand security, but source-code security scanning is not integrated into the SDL



Human	0	5	10	20	30
Tool	0	10	10	10	10
Human + Tool	0	10	20	50	60

(vertical axis: Number of vulnerabilities discovered , Horizontal axis: time spent on engagement)

Some observations on this scenario:

- The application is much 'harder to break' and all three approaches take longer to produce meaningful results
- The *'Tool'* only discovers 10% of the issues
- The *'Human'* has a very slow start and take considerable time before being real productive and start to discover considerable numbers of vulnerabilities on the application

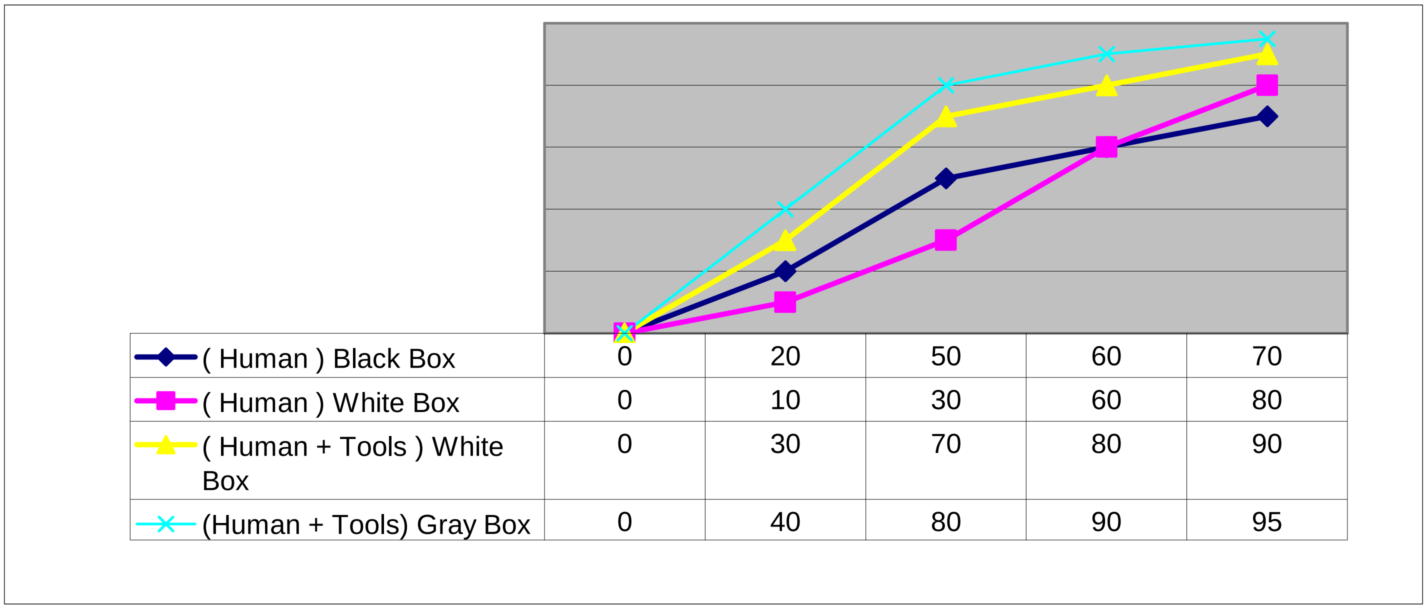
- The *'Human + Tool'* also has a slow start (benefiting from the 10% initial results provided by the out-of-the-box *'Tool'*), but when the 'Human' driving the 'Tool' gains an understanding of the application design & Business logic (and customizes the rule-set to the application's vulnerable spots) the number of findings grows at a much higher rate. Another big advantage that the 'Human + Tool' combination has is the fact that once a particular type of vulnerability is discovered and it's 'formula' is understood, it is possible to write 'pattern based signatures' and scan the entire code-based for similar exploitable variations.

- The total number of findings in this scenario was capped at 60% since with *hardened large applications* it is unlikely that 90% or more of the vulnerabilities could be discovered unless a very unrealistic number of days are allocated to its analysis.

- In this graph the best ROI value is the 50% mark of the *'Tool + Human'*

WHITE BOX VS BLACK BOX VS GRAY BOX

The following chart, shows a comparison between external test (black box) and Source code analysis (White box) in a *'Human'* and a *'Humans + Tools'* scenario:



◆ (Human) Black Box	0	20	50	60	70
■ (Human) White Box	0	10	30	60	80
▲ (Human + Tools) White Box	0	30	70	80	90
× (Human + Tools) Gray Box	0	40	80	90	95

(vertical axis: Number of vulnerabilities discovered , Horizontal axis: time spent on engagement)

Some observations on this scenario:

- The *'Black Box'* (which is the normal penetration testing model) is the least efficient since it takes longer to reach a high number of vulnerabilities (and it doesn't reach the same count as the others (given the same amount of time allocated)

- See previous example for comments on the *'(Human) – White Box'* and '(Human + Tools)

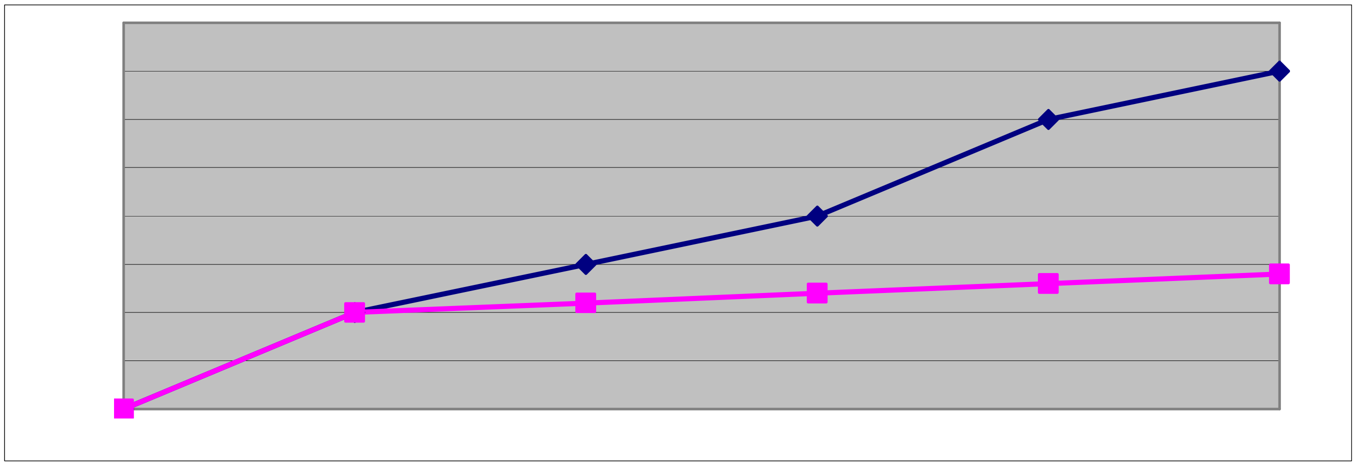
- The *'Human + Tools (Gray Box)'* (I.e. with both access to the source code and live test environment) is the most efficient since the 'Human' is able to quickly confirm the vulnerabilities discovered during source-code analysis and it able to ensure that the entire application's attack surface is tested

VALUE EXTRACTED BY DEVELOPER VS SECURITY CONSULTANT

This final graph shows the difference in value extracted from the source-code scanning software when used by developers and when used by a Security Consultant.

When the tool is used by a developer there is high gain in the beginning of the engagement and minor gains from there onwards (due to the developer's lack of customization and fully understanding of the results shown)

When the tool is used by a security consultant, there is a constant increase in value extracted from the tool since due to the constant customization of the tool during the engagement



OUTSOURCING SECURITY TESTS

In the cases where external parties are commissioned to provide security assessment services to internal applications (either with or without access to the source-code), the S4 team should be involved before and after the engagement (i.e. creating the brief and reviewing the results[38]).

[38] The result review and normalizing phase is very important in order to ensure that the risk rating of the vulnerabilities discovered by external parties is consistent with the criteria used by S4 in engagement reports and in management reports

This will ensure that the work commissioned and delivered matches the high quality standards of service provided by S4.

Some important points to take in consideration:

- The objective should not be to show the buyer how good the pen test company is but to increase the level of security of the target application
    - The odds are already against the penetration testing company[39]
    - Provide as much information and resources as possible since the *'no information provided test'* (such as *'here is an IP address and see what you can do'*) is only testing a subset of the possible attack vectors
- The best test environment is the gray box model (with both source code and test environment)
    - If source code cannot be provided or if there is no budget available for analysis by the external company, a good compromise is to give the external company access to a *'Low Fruit Report'*[40] of the target application (as created by the chosen Source-code Security Scanning Software)
- Evaluate the company's testing strategy and confirm their analysis methodology:
    - Manually: from the 1st line till the last one, or
    - Manually: Focusing on the attack surface, or
    - Automatically: Using XYZ tool out of the box, or
    - Automatically: Using XYZ tool with consultant XYZ
- Confirm WHO will be performing the assessment since not all security consultants are equal. One should be hiring individuals and not companies

S4 INCIDENT RESPONSE CAPABILITIES

{... unique set of skills that exist in S4 ...}

{...note: when triggered, must re-schedule current engagements (and do some damage control on the current expectations ...}

{... have plan ready (out of scope of this document) for this situations ...}

[39] Since these companies will only have a limited amount of time to assess an application and are not able to use all attack-vectors available to a real-world attack (for example: Advanced Fuzzing, Social Engineering, Compromise of internal clients (via browser exploits) and Denial of Service)

[40] See 'S4 Engagement Types' section in this document

{... Previously define who will do what ...}

{... next steps will depend on the existence or not of WAF (if one is not in place now is not the best time to do it ...}

{... Should S4 Security Consultants change code and fix the problem? (only on last resourt) ...}

TRAINING

S4 provides the following training services both client-site and remotely[41]:

- 1 Day training on source-code security scanning software
- 1 Day training on Application Security
- 1 hour presentation on source-code security scanning software
- 2 hour presentation on vulnerabilities discovered during S4 engagement

S4 CONTACT DETAILS

S4 can be contacted using the following mediums:

- Email: support@s4.{URI_Address}
- Website: s4.{URI_Address} (WIKI based)
- Phone : + (00) 0000 1111 2222
- Postal Address: ....
- PGP KEY: ....

REFERENCES AND FURTHER READING

- OWASP website, OWASP Testing Guide, OWASP Code Review project
- Ounce White papers
- Secure Code book , SANS
- Microsoft & Java best practices documents, Blogs

[41] using WebEx technology

IMPLEMENTING S4

This section contains practical information/advice on how to implement a S4 in an organization.

This is an internal document (usually not for distribution to S4 clients) and focused on the HOW and not of the WHAT or WHY.

S4 IMPLEMENTATION TIMELINE

The following is a common sequence of events from initial mandate to S4 deployment.

- Mandate from Senior management to implement a Source-code Security Scanning Service
- Contact with multiple vendors to understand tools capabilities
- Create evaluation and success criteria
- *Simple PoC*[42]: Evaluation of 2 or 3 products, and select 1 for further research
- *Extended PoC*
    - Central engagements to selected applications
        - Live and remote presentations of findings
        - Delivery of report and recommendations
    - One or two deployments of the selected software on selected development groups
        - Training of software usage

- Create *Extended PoC* report focusing on the advantages and disadvantages of the tool (i.e. what can be done today with the state of the selected software)
- Finish *S4 Service Definition* and *S4 Engagement Model* documents and define required budget
- Negotiate contracts with chosen software vendor
- Obtain senior management approval
- Implement S4
    - Set-up team (Hiring as required)
    - Set-up S4 environment (servers, email address, web hosting for wiki website)
    - Train team on Software and S4 engagement model
    - Perform two or three complete engagements (from beginning to end) and adjust *S4 Service Definition* and *S4 Engagement Model* documents if required

- Launch S4 by announcing it to the organization

[42] PoC = Proof of Concept

ASSET ENUMERATION

Create (or obtain if available) an inventory of potential target application with:

- Application name, brief description and technologies used
- Business owners & Points of contacts
- Importance to the business
- CIA[43] rating

THE BUSINESS CASE FOR S4

{... don't put a lot of info here, just give a top level view of the issue and point to external resources that cover this in details (from OWASP to Ounce articles...}

ROI

{... put here screenshots from Ounce ROI XCL document ...}

BUILDING TEAM

To be successful the Source Code Security Service (S4) must build a string team with the following skill set:

- Commercial White Box (Source Code review) and Black Box (Penetration Testing) experience
- Commercial Development experience (required for providing remediation assistance to developers)
- Writing skills (required for report writing but also used to create Best Practices and Guidelines documents)
- Classroom and Web Based training experience
- Consulting experience
- Ability to Promote change throughout an organization

[43] CIA = Confidentiality Integrity and Availability

HIRING S4 MEMBERS

- Background checks
- Check Open source projects created and participation
- Participation on conferences and on online communities like OWASP

PROVISIONING REQUIREMENTS

- Laptops:
    - *{… Spec …}*
- Servers:
    - *{… Spec …}*
- Virtual Server environments (VMWare or Virtual PC)
    - *{… Spec …}*
- Network bandwidth
    - *({… Spec …}*
- Hard-disk requirements:
    - *({… Spec …}*

REASONS FOR S4 EXISTENCE

It is very important to understand why companies and organizations do something about security and specifically why S4 was created[44]:

- Was there a successful (or almost successful) attack?
- Is the objective to prevent damage if attacked?

  For example:

    - Financial
    - Brand value

[44] since these answers have direct impact on S4's corporate positioning and clout

  - Loss of customers

- Are there regulatory requirements that affect S4 business owners? (SOX, PCI, Data Protection, etc…)
- Is there client pressure?
- Are the competitive advantages in having more 'secure' applications?

S4 POWER

Another very important factor to take in consideration is how much power this S4 service will has.

Answering the following questions will provide this information:

- Who provided political, financial and organization support to S4? (is it a C-Level executive or is it a mid level department
- Can S4 stop products or applications from being published or released?
- Is an S4 review mandatory for all mission critical projects?
- What are the penalties for 'creating'[45] High vulnerabilities? For example using:
  - Known 'banned' APIs,
  - Known Insecure designs or
  - Known Insecure coding practices, or
  - Unknown insecure coding practice or insecure designs.
- What the rewards for 'creating' secure code (& performing Threat Models of the target application)
- Can S4 create documents containing:
  - Lists of banned APIs
  - Secure Coding practices
  - Secure Design patterns

[45] '*creating*' in this context means '*writing programming code into an application that contains a security vulnerability*'

SCOPING A SOURCE-CODE REVIEW

A common problem faced by teams responsible for performing source code security scanning services is *'How to scope a project?'* (i.e. answering the client's question *'How long do you need to securely review my application code-code and tell me if it is secure or not?'*).

Unfortunately there are no hard rules, but the following are some pointers based on Ounce Consultants experience:

- A good number to start is: 3 man-days + 2 man-days per 50 kloc[46]
- Using Tools (like Ounce) can reduce the amount of time required by 20% to 40%
- Add 10% of the time calculated for report writing
- Add 10% of the time calculated for project management and meetings with client
- It usually tales two client engagements to analyze a large application, since only after the first engagement the project brief is completed. This project brief which will contain the correct number of lines of code to review and the correct application's architecture and interconnections diagram[47].

EXPECTED VULNERABILITY COUNT BEFORE AND AFTER S4

{... insert graph showing a common line-chart of the number of vulnerabilities known on an application vs the lines of code (with milestones for the before and after S4 involvement) ...}

OTHER POPULAR SDL

The following are popular SDL architectures:

- Microsoft SDL
- OWASP CLASP
- Cigital Security TouchPoints

[46] Kloc = thousand lines of code

[47] When compared with the (incorrect) kloc count and application's architecture diagrams originally provided by the client

SECURITY CONSULTANT'S GUIDE TO OUNCE

This section contains information targeted at Security Consultants that regularly use Ounce in their security engagements and wish to maximize its capabilities.

GLOSSARY

Brief descriptions of the main terms used throughout this section

- S4 – Source Code Security Services
- Tainted data – {... malicious data ...}
- Smart Trace – *{…description …}*
- Source – *{…description …}*
- Sink & Lost Sink – *{…description …}*
- Vulnerabilities, Type I & Type II – *{…description …}*
- VBUG – Xml Files containing detailed information about the vulnerabilities discovered
- Bundles – *{…description …}*

TECHNIQUES TO AUGMENT OUNCE SCANNING CAPABILITIES

The following list contains proven techniques and methodologies used to increase Ounce's understanding and visibility of the target's application:

- 'Taintometer' - *{... explanation of concept...}*
    - o *{… 3 different Taint methods / techniques…}*
        - ▪ Sinks
        - ▪ Sources
        - ▪ Callbacks (the radioactive pill) – *{… Used to handle: Web Services, Remoting, Ajax …}*
    - o Add rules to mark Database inputs as sinks (and see what ends up in there)
    - o Mark sensitive data as sources of Tainted data
    - o Mark exceptions as sources of tainted data

  - Map all configuration's files entry points in Ounce[48]
  - Mark validation routines as Sinks (to see what ends up in there)
- Handle the lost-sinks first
- Deal with each API at the time and convert its findings (ie. Vulnerabilities) into custom rules
- For non-common types of vulnerabilities (usually initially flagged as a 'Validation Required' vulnerability) figure out it's 'repeatable pattern' and re-scan the code base looking for all instances and variations
- Use Black-Box testing techniques to:
  - Confirm findings
  - Find attack surface (i.e. cross check what can be requested on the browser[49] with the *Ounce's* detected input points)
  - Find new vulnerabilities, which will then be converted into a 'repeatable pattern' so that all instances (and variations) can be scanned for
- Understand the business model of the attackers & create an Threat Mode for the target application
  - Follow the identified assets using Ounce (by marking them sources of Tainted data)
- Introduce extra sinks (i.e vulnerabilities) on the strategic locations in order to analyze the types of data that arrive there
- If unit-tests are available feed tainted-data into the methods they invoke (since they should be mapping to the application's attack surface)

COMMON SCENARIOS THAT OCCUR AFTER THE FIRST SCAN

- Total visibility of all dependencies APIs, lots of findings, no lost-sinks
- Non Findings, lots of Types IIs
- Mixed bag of results

{... for each one of these, talk about the strategies and recommended course of action ...}

[48] web.xml in J2EE and web.config in .NET

[49] In the cases where the target is an Web Application

{... on the lost sinks and dependencies, it is very important to check how much Ounce 5.0 understands them (for sinks, check the list of Lost-Sinks methods to gain an idea of what is missing (on the inputs the bet way is to use a live instance of the application under test (for example website) and confirm that the all available (and testable) inputs points are visible by Ounce 5.0 (this is basically mapping the 'attack surface') ...}

HOW TO KNOW WHEN YOU ARE DONE

In order to answer the clients though question 'Is my application secure', it is important to know when good enough coverage has been achieved so that the Security Constant is comfortable is saying Yes or No[50].

Note that the following comments only apply to 'hardened' applications of which there are no High vulnerabilities to report.

When using Ounce, the lack of High Findings (as Vulnerabilities, Type I or Type II) cannot be automatically be assumed to mean that the application is secure. One can only make that assumption when:

- The attack surface is all mapped, there are no Lost Sinks and all vulnerabilities are mapped & resolved (i.e. Moved to bundles)
  - In .Net ort Java the attack surface tends to be defined by the config files: web.xml or web.config
  - Double check extra '*Validation and Encoding*' and '*this method is not susceptible to Taint*' rules, since they could be eliminating real vulnerabilities.
- The output routines have been mapped and *Ounce* is able to identify them as sinks (for example custom Java Tag Libs)
- The actions described in the '*Techniques to augment ounce scanning capabilities*' section of this document have been fully explored

[50] Ironically it is much easier to say *"Your application is massively insecure and here is the list of High Security vulnerabilities"* than to say *"Your application is secure and I don't have any High Security Vulnerabilities to report"*. The issue is that in the first case (when the application is very insecure), the Security Consultant has no responsibility to find ALL issues since considerable amounts of time were spent in documenting the issues identified. In the second case (when the application is secure) the pressure is on the Security Consultant, since he/she is now making a strong statement about the application's security profile (and if the Security Consultant is not comfortable that the entire application was thoroughly analyzed and all possible attack vectors have been covered, the Security Consultant has some hard choices to make in the terminology used on the report)

- The application's business logic is well understood by the Security Consultant and *Ounce* been used to check the possible attack paths
- All unique variations of High, Medium and Low Vulnerabilities and Type I and Type II have been reviewed and cleared

CATEGORIES OF VULNERABILITIES

- Remotely Exploitable – *{… description …}*
- Internal 'by design' (on APIs) – *{… description …}*
- Internal 'not exploitable' – *{… description …}*

{… Which ones to fix depends on the 'appetite' of the client for risk …}

USING OUNCE FEATURES

BUNDLES

{… explain how I use Bundles ...}

SMART AUDIT

{… mention that how useful it is ...}

CUSTOM RULES

{... how to use them ...}

PBSA – REGEXES

{... how to use it ...}

OUNCE BETA TOOLS

{... List the ones that I have at the moment and the ones under development ...}

HOW TO HANDLE

- Web Services - *{... put mini case study here ...}*
- MVC Frameworks like Spring AJAX - *{... put mini case study here ...}*
- Data APIs
- Client-Server apps, Stand alone applications, Appliances and embedded systems

FINDING BUSINESS LOGIC ISSUES

{... for example adding extra sinks on the database calls to see what is happening ...}

{... a more advanced analysis is one that does this multiple rounds (one for each level of Authorization ...}

FINDING IMPROPER USE OF APPLICATION ASSETS

{... for example marking them as tainted and seeing where they end up in ...}

{… add case study …}

DIFFERENT ISSUES FOR DIFFERENT LANGUAGES

{... each language tends to have different issues ... describe the mains ones…}

- Asp.NET & J2EE
- C# & Java
- C++
- ASP Classic
- VB 6

CURRENT KNOWN ISSUES WITH OUNCE SCANNING ENGINE

- *Ounce* needs help in identifying input points (for example web services)
- No taint propagation intro global variables
- Lost Sinks when faced with multiple possible 'Interface implementing' methods
- *{… add additional major analysis engine issues…}*

COMMERCIAL INFORMATION ON OUNCE

This section contains commercial information on Ounce Labs software and consulting services

OUNCE CONSULTING SERVICES

The following are services provided by Ounce Consulting services

- PoCs
- Training
- First couple days in External security engagement
- Consulting
    - Source-code security scanning
    - SDL integration
    - S4 implementation and deployments

OUNCE REQUIREMENTS

- Supported Platforms: Windows, Linux, Solaris
- Database used: MySql (soon also Oracle)

RFP

This section contains information that is usually requested to Ounce when companies send an RFP to the different tool vendors

RFP CONTENTS

{... questions that are usually in the RFP ...}

OUNCE RFP ANSWERS

…