KAUNAS UNIVERSITY OF TECHNOLOGY

INFORMATICS FACULTY

Software engineering

Project work report

KAUNAS 2011

Table of Contents

Product documentation

Requirements documentation

Aim of the project:

Develop an open source alternative to commercial TeamViewer product. Ours product will support these basic features:

- Remote desktop access
- Remote mouse control
- Remote keyboard control
- Ability to view remote computer file system without interfering

Project members:

- Justas Šalkevičius - developer
- Julius Rentas - developer
- Kęstutis Vaškevičius - website / developer
- Kastytis Venckys - documentation

Product overview and use cases

- Simple
- Easy to use
- Lightweight

Host use case diagram:

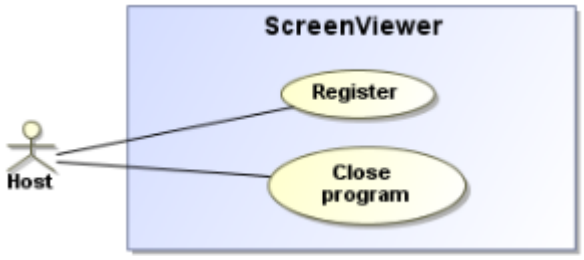


Figure 1 Host Use case diagram

Host can:

- can register account
- close the program

Viewer use case:

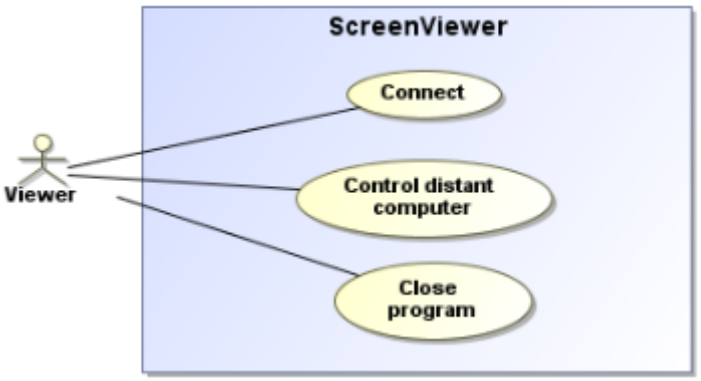


Figure 2 Host Use case diagram

Viewer can:

- Connect to the Host computer
- Control remote desktop
- Close program

Architectural requirements

- Identify possible challenges
- Choose most suitable programming language
- Choose what tools and software will be needed for project realization

Functional requirements

- Show program GUI
- Allow to choose to be: Host or Viewer
- Connect to the internet and host computer
- Show remote desktop in program window
- Allow to control remote mouse
- Allow to control remote keyboard
- Ability to view remote computer file system without interfering
- Allow to close program

Non-functional requirements

- Simple GUI
- Quick response
- Small resource usage

Technical requirements:

Minimum:

- 1GHz x86-x64 architecture CPU.
- 512 MB of RAM.
- Integrated GPU.
- TCP/IP Network connection.
- Mouse and keyboard.
- 25 MB of available hard-disk space

Recommended:

- 1GHz x86-x64 architecture CPU.
- 1 GB of RAM for XP users and 2GB of RAM for W7 users.

- Integrated GPU.
- TCP/IP Network connection.
- Mouse and keyboard.
- 30 MB of available hard-disk space

Environmental requirements

- Program must connect to computers that have external IP address.

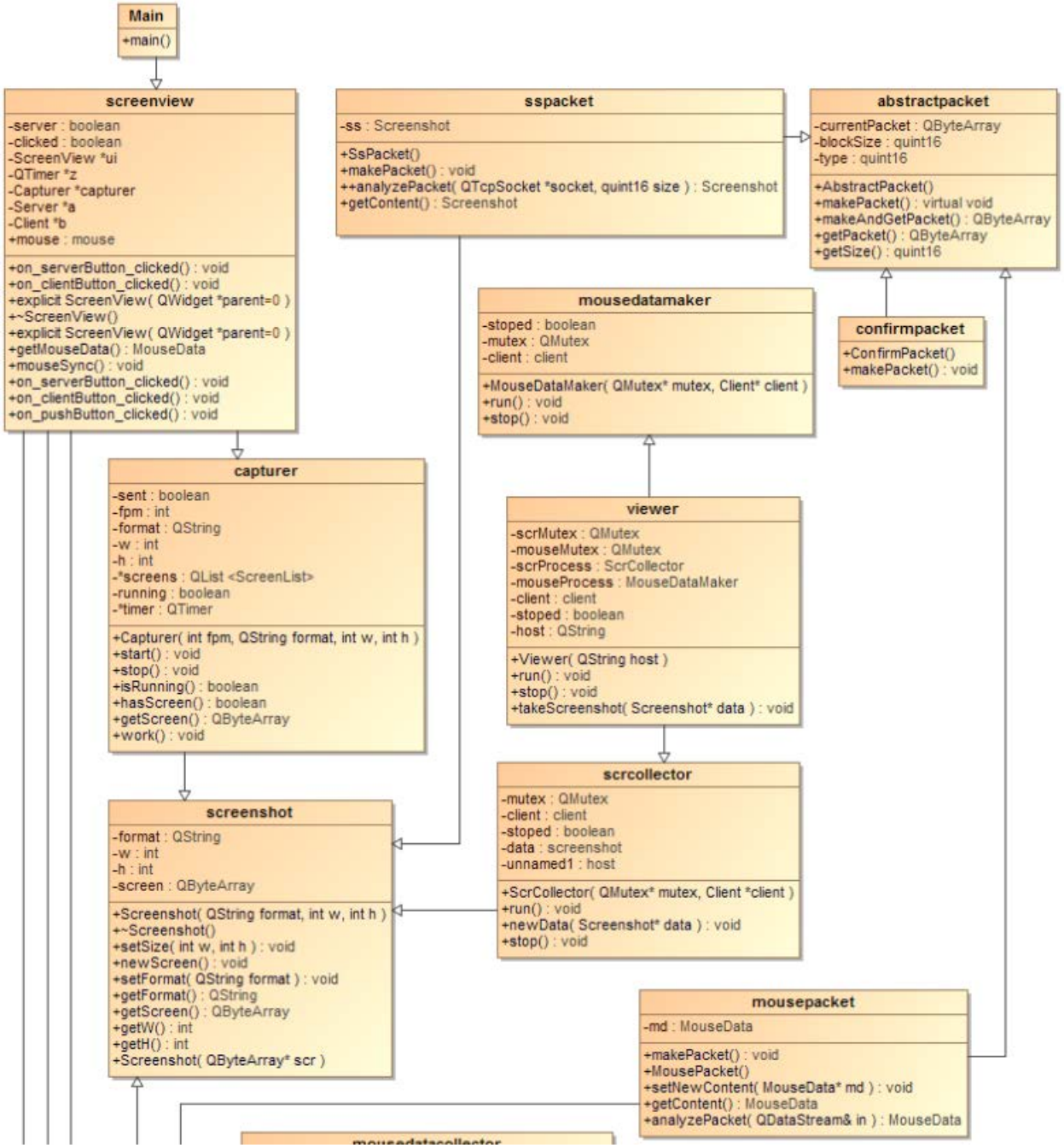
Interaction requirements (e.g. how the software should work with other systems)

- Friendly work with other internet programs
- Antivirus programs shouldn’t consider program as a threat
- OS firewall must allow program to connect to the internet

Architecture/Design documentation

Architecture design

- UML classes



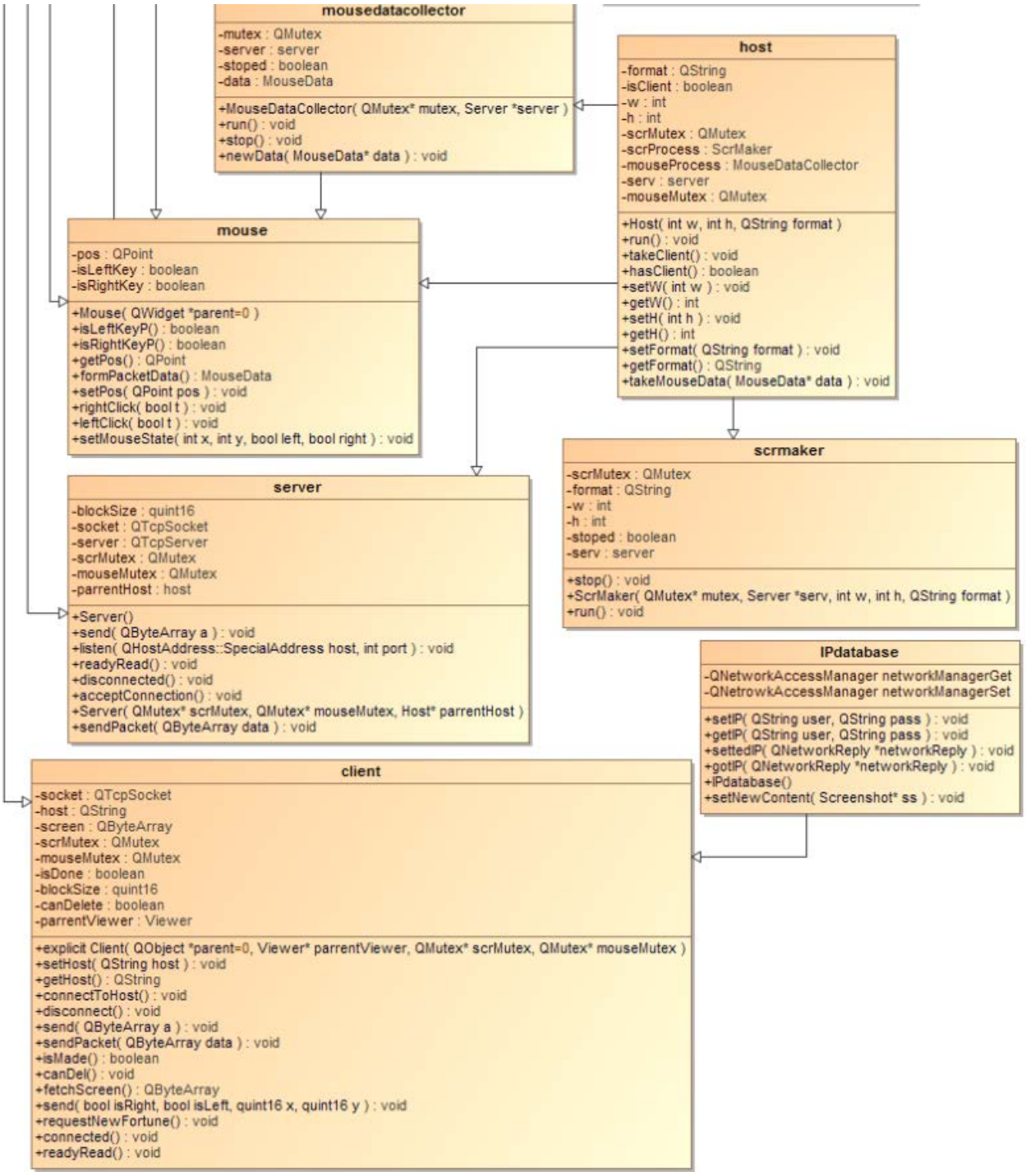


Figure 3 ScreenViewer UML classes

Interface design

- GUI example:

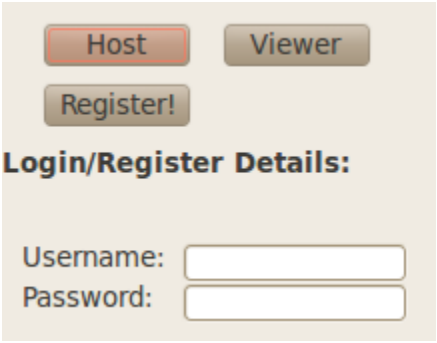


Figure 4 ScreenViewer window

- Host button – allow to control your computer
- Viewer button – Connect to remote desktop
- Register button – Create your account
- Username – your chosen name
- Password – your made-up password

Technical documentation

Patents, licenses

- Project uses the MIT license:

Tools

- **Qt** uses standard C++ but makes extensive use of a special code generator (called the Meta Object Compiler, or moc) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform API for file handling.
- **PHP** is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications. PHP can be deployed on most web servers and as a standalone interpreter, on almost every operating system and platform free of charge. There is also commercial

software such as RadPHP, a rapid application development framework for the PHP language.

- **Apache** is developed and maintained by an open community of developers under the auspices of the **Apache Software Foundation**. The application is available for a wide variety of operating systems, including Unix, FreeBSD, Linux, Solaris, Novell NetWare, AmigaOS, Mac OS X, Microsoft Windows, OS/2, TPF, and eComStation. Released under the Apache License, Apache is open-source software. **Apache** supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl, and **PHP**.
- **MySQL** is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. Free-software-open source projects that require a full-featured database management system often use **MySQL**. For commercial use, several paid editions are available, and offer additional functionality. **MySQL** is a popular choice of database for use in web applications, and is a central component of the widely used LAMP web application software stack—LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python".

Methods

- **Extreme Programming (XP)** is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. This method has several potential drawbacks, including problems with unstable requirements, no documented compromises of user conflicts, and a lack of an overall design specification or document. (*http://en.wikipedia.org/wiki/Extreme_Programming*)
- **Pair programming** is an agile software development technique in which two programmers work together at one workstation. One types in code while the other reviews each line of code as it is typed in. The person typing is called the **driver**. The person reviewing the code is called the **observer** (or **navigator**). The two programmers switch roles frequently. While reviewing, the observer also considers the strategic direction of the work, coming up with ideas for improvements and likely future problems to address. This frees the driver to focus all of his or her attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide. (*http://en.wikipedia.org/wiki/Pair_programming*)
- Test-driven development (**TDD**) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, then

produces code to pass that test and finally refactors the new code to acceptable standards. (*http://en.wikipedia.org/wiki/Test-driven_development*)

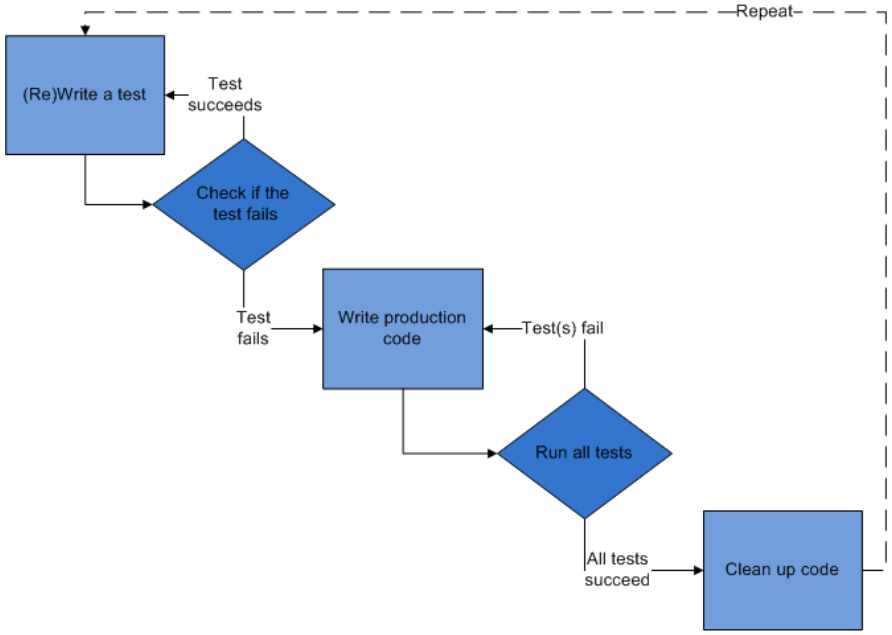


Figure 5 TDD method diagram

Testing

Static analysis

- **Static program analysis** (also **Static code analysis** or **SCA**) is the analysis of computer software that is performed without actually executing programs built from that software (analysis performed on executing programs is known as dynamic analysis). In most cases the analysis is performed on some version of the source code and in the other cases some form of the object code. The term is usually applied to the analysis performed by an automated tool, with human analysis being called program understanding, program comprehension or code review.
- The sophistication of the analysis performed by tools varies from those that only consider the behavior of individual statements and declarations, to those that include the complete source code of a program in their analysis. Uses of the information obtained from the analysis vary from highlighting *possible coding errors* (e.g., the lint tool) to formal methods that mathematically prove properties about a given program (e.g., its behavior matches that of its specification).
- It can be argued that software metrics and reverse engineering are forms of static analysis. In fact deriving software metrics and **static analysis** are increasingly deployed together, especially in creation of embedded systems, by defining so called *software quality objectives*.

Testing tool

- **Cppcheck** is an open source static code analyzer tool for **C/C++** programming languages. It's a versatile tool that can check non-standard code.
- Cppcheck supports a wide variety of static checks that may not be covered by the compiler itself. These checks are static analysis checks that can be performed at a source code level. The program is directed towards static analysis checks that are rigorous, rather than heuristic in nature.

Cppcheck checks for these errors:

64-bit portability

Check if there are 64-bit portability issues:

- assign address to/from int/long

Auto Variables

A pointer to a variable is only valid as long as the variable is in scope. Check:

- returning a pointer to auto or temporary variable
- assigning address of an variable to an effective parameter of a function
- returning reference to local/temporary variable
- returning address of function parameter

Boost usage

Check for invalid usage of Boost:

- container modification during BOOST_FOREACH

Bounds checking

Out of bounds checking

Class

Check the code for each class:

- Missing constructors
- Are all variables initialized by the constructors?
- Warn if memset, memcpy etc. are used on a class
- Are there unused private functions
- 'operator=' should return reference to self
- 'operator=' should check for assignment to self
- Constness for member functions

Exception Safety

Checking exception safety:

- Throwing exceptions in destructors
- Throwing exception during invalid state
- Throwing a copy of a caught exception instead of rethrowing the original exception

Match assignments and conditions

Match assignments and conditions:

- Mismatching assignment and comparison => comparison is always true/false
- Mismatching lhs and rhs in comparison => comparison is always true/false
- Detect matching 'if' and 'else if' conditions

Memory leaks (address not taken)

Not taking the address to allocated memory.

Memory leaks (class variables)

If the constructor allocates memory then the destructor must deallocate it.

Memory leaks (function variables)

Is there any allocated memory when a function goes out of scope?

Memory leaks (struct members)

Don't forget to deallocate struct members.

Non reentrant functions

Warn if any of these non reentrant functions are used:

- Asctime
- crypt
- ctermid
- ctime
- ecvt
- fcvt
- fgetgrent
- fgetpwent
- fgetspent
- gcvt
- getgrent
- getgrgid
- getgrnam
- gethostbyaddr
- gethostbyname
- gethostbyname2
- gethostent
- getlogin
- getnetbyaddr
- getnetbyname
- getnetgrent
- getprotobyname
- getpwent
- getpwnam
- getpwuid
- getrpcbyname
- getrpcbynumber
- getrpcent
- getservbyname
- getservbyport
- getservent
- getspent
- getspnam
- gmtime
- localtime
- rand
- readdir
- strtok
- tempnam
- tmpnam
- ttyname

Null pointer

Null pointers:

- null pointer dereferencing

Obsolete functions

Warn if any of these obsolete functions are used:

- bcmp
- bcopy
- bsd_signal
- fcvt
- gcvt
- bzero
- ecvt
- ftime

- getcontext
- gethostbyaddr
- gethostbyname
- getwd
- index
- makecontext
- pthread_attr_getstackaddr
- pthread_attr_setstackaddr
- rindex
- scalbln
- swapcontext
- ualarm
- usleep
- vfork
- wcswcs

Other

Other checks:

- Assigning *bool* value to pointer (converting *bool* value to address)
- bad usage of the function 'sprintf' (overlapping data)
- division with zero
- using fflush() on an input stream
- scoped object destroyed immediately after construction
- assignment in an assert statement
- *sizeof* for array given as function argument
- *sizeof* for numeric given as function argument
- incorrect length arguments for 'substr' and 'strncmp'
- invalid usage of output stream. For example: std::cout << std::cout;'
- wrong number of arguments given to 'printf' or 'scanf;'
- C-style pointer cast in cpp file
- redundant if
- bad usage of the function 'strtol'
- unsigned division
- Dangerous usage of 'scanf'
- passing parameter by value
- Incomplete statement
- check how signed char variables are used
- variable scope can be limited
- condition that is always true/false
- unusual pointer arithmetic. For example: "abc" + 'd'
- redundant assignment in a switch statement
- redundant strcpy in a switch statement
- look for 'sizeof sizeof ..'
- look for calculations inside sizeof()
- assignment of a variable to itself
- mutual exclusion over || always evaluating to true
- exception caught by value instead of by reference
- Clarify calculation with parentheses

- using increment on boolean
- comparison of a boolean with a non-zero integer
- comparison of a boolean expression with an integer other than 0 or 1
- suspicious condition (assignment+comparison)
- suspicious condition (runtime comparison of string literals)
- suspicious condition (string literals as boolean)
- duplicate break statement
- unreachable code
- testing if unsigned variable is negative
- testing is unsigned variable is positive
- using bool in bitwise expression
- Suspicious use of ; at the end of 'if/for/while' statement.
- incorrect usage of functions from ctype library.
- optimization: detect post increment/decrement

STL usage

Check for invalid usage of STL:

- out of bounds errors
- misuse of iterators when iterating through a container
- mismatching containers in calls
- dereferencing an erased iterator
- for vectors: using iterator/pointer after push_back has been used
- optimisation: use empty() instead of size() to guarantee fast code
- suspicious condition when using find
- redundant condition
- common mistakes when using string::c_str()
- using auto pointer (auto_ptr)
- useless calls of string functions

Uninitialized variables

Uninitialized variables:

- using uninitialized variables and data

Unused functions

Check for functions that are never called.

UnusedVar

UnusedVar checks

- unused variable
- allocated but unused variable
- unred variable
- unassigned variable
- unused struct member

Unit testing

- **Unit testing** is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit could be an entire module but is more commonly an individual function or procedure. In object-oriented programming a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionally by white box testers during the development process.
- Ideally, each test case is independent from the others: substitutes like method stubs, mock objects, fakes and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper) to being formalized as part of build automation.

Unit testing library (tool)

- The **QTestLib** framework, provided by Nokia, is a tool for unit testing Qt based applications and libraries. **QTestLib** provides all the functionality commonly found in unit testing frameworks as well as extensions for testing graphical user interfaces.

QTestLib Features

QTestLib is designed to ease the writing of unit tests for **Qt** based applications and libraries:

Feature	Details
Lightweight	QTestLib consists of about 6000 lines of code and 60 exported symbols.
Self-contained	QTestLib requires only a few symbols from the Qt Core library for non-gui testing.
Rapid testing	QTestLib needs no special test-runners; no special registration for tests.
Data-driven testing	A test can be executed multiple times with different test data.
Basic GUI testing	QTestLib offers functionality for mouse and keyboard simulation.
Benchmarking	QTestLib supports benchmarking and provides several measurement back-ends.
IDE friendly	QTestLib outputs messages that can be interpreted by Visual Studio and KDevelop.
Thread-safety	The error reporting is thread safe and atomic.
Type-safety	Extensive use of templates prevents errors introduced by implicit type casting.
Easily extendable	Custom types can easily be added to the test data and test output.

Tests

```
Starting /home/kestutis/programming/KTU/screenviewer/src/scrview/untitled...
********* Start testing of TestScreenshot *********
Config: Using QTest library 4.6.2, Qt 4.6.2
PASS   : TestScreenshot::initTestCase()
PASS   : TestScreenshot::testNew()
PASS   : TestScreenshot::testResize()
PASS   : TestScreenshot::testFormat()
PASS   : TestScreenshot::cleanupTestCase()
Totals: 5 passed, 0 failed, 0 skipped
********* Finished testing of TestScreenshot *********
********* Start testing of TestCapturer *********
Config: Using QTest library 4.6.2, Qt 4.6.2
PASS   : TestCapturer::initTestCase()
PASS   : TestCapturer::testFpm()
PASS   : TestCapturer::testCommon()
PASS   : TestCapturer::testStartStop()
PASS   : TestCapturer::cleanupTestCase()
Totals: 5 passed, 0 failed, 0 skipped
********* Finished testing of TestCapturer *********
********* Start testing of TestSsPacket *********
Config: Using QTest library 4.6.2, Qt 4.6.2
PASS   : TestSsPacket::initTestCase()
PASS   : TestSsPacket::testEmptyMake()
PASS   : TestSsPacket::testMake()
PASS   : TestSsPacket::testMakeAndGet()
```

```
PASS   : TestSsPacket::testSetNewContent()
PASS   : TestSsPacket::testPacketSize()
PASS   : TestSsPacket::cleanupTestCase()
Totals: 7 passed, 0 failed, 0 skipped
********* Finished testing of TestSsPacket *********
********* Start testing of TestMousePacket *********
Config: Using QTest library 4.6.2, Qt 4.6.2
PASS   : TestMousePacket::initTestCase()
PASS   : TestMousePacket::testEmptyMake()
PASS   : TestMousePacket::testMake()
PASS   : TestMousePacket::testMakeAndGet()
PASS   : TestMousePacket::testSetNewContent()
PASS   : TestMousePacket::testPacketSize()
PASS   : TestMousePacket::cleanupTestCase()
Totals: 7 passed, 0 failed, 0 skipped
********* Finished testing of TestMousePacket *********
********* Start testing of TestConfirmPacket *********
Config: Using QTest library 4.6.2, Qt 4.6.2
PASS   : TestConfirmPacket::initTestCase()
PASS   : TestConfirmPacket::testMake()
PASS   : TestConfirmPacket::testMakeAndGet()
PASS   : TestConfirmPacket::testPacketSize()
PASS   : TestConfirmPacket::cleanupTestCase()
Totals: 5 passed, 0 failed, 0 skipped
********* Finished testing of TestConfirmPacket *********
/home/kestutis/programming/KTU/screenviewer/src/scrview/untitled exited with code 0
```

Threat modeling and risk analysis

Threat Analysis and Modeling v3.0 (tool)

Building a secure application and requires an understanding of the threats against that application. The challenge has been the difficulty in adopting threat modeling practice for software application development. The Microsoft Application Consulting; Engineering (ACE) team developed a process that allows non-security subject matter experts to produce feature-rich threat models.

The process:

- Provides a consistent methodology for objectively identifying and evaluating threats to applications
- Translates technical risk to business impact
- Empowers a business to manage risk
- Creates awareness among teams of security dependencies and assumptions

Microsoft Application Threat Modeling is a critical security activity, enabling effective application risk management during the SDLC and beyond.

Microsoft Threat Analysis & Modeling tool facilitates creation and assimilation of threat models. Now from entered non-security already-known data a feature-rich threat model can be produced. Along with automatically identifying threats, the tool can produce such valuable security artifacts as:

- Data access control matrix
- Component access control matrix
- Subject-object matrix
- Data flow
- Call flow
- Trust flow
- Attack surface
- Focused reports

Threats

A threat is defined as an undesired event, a potential occurrence, often best described as an effect that might damage or compromise an asset or objective. It may or may not be malicious in nature.

Threats

DDOS
Risk Rating: 9
Risk Response: Reduce
Confidentiality: No
Integrity: No
Availability: Yes

Task Items
☑ Request size should be constrained
☑ Consider performance as a requirement
☑ Implement appropriate exception handling
☑ Display generic error messages
☑ DDOS ip blocking

Viewer leaves PC on
Risk Rating: 3
Risk Response: Avoid
Confidentiality: Yes
Integrity: No
Availability: No

Collects TCP packets
Risk Rating: 6
Risk Response: Accept
Confidentiality: Yes
Integrity: Yes
Availability: No

Task Items
☑ Use cryptographically strong keys
☑ Use well-known cryptographic algorithms
☑ Salt the hash value with a unique random bits

Hacks SQL DB
Risk Rating: 3
Risk Response: Avoid
Confidentiality: Yes
Integrity: Yes
Availability: Yes

Task Items
☑SQL Wildcard queries should be avoided
☑Perform context sensitive HTML encoding
☑Untrusted input should be validated (Managed)
☑Use parameterized SQL statement

Changesd packets data
Risk Rating: 3
Risk Response: Avoid
Confidentiality: Yes
Integrity: Yes
Availability: No

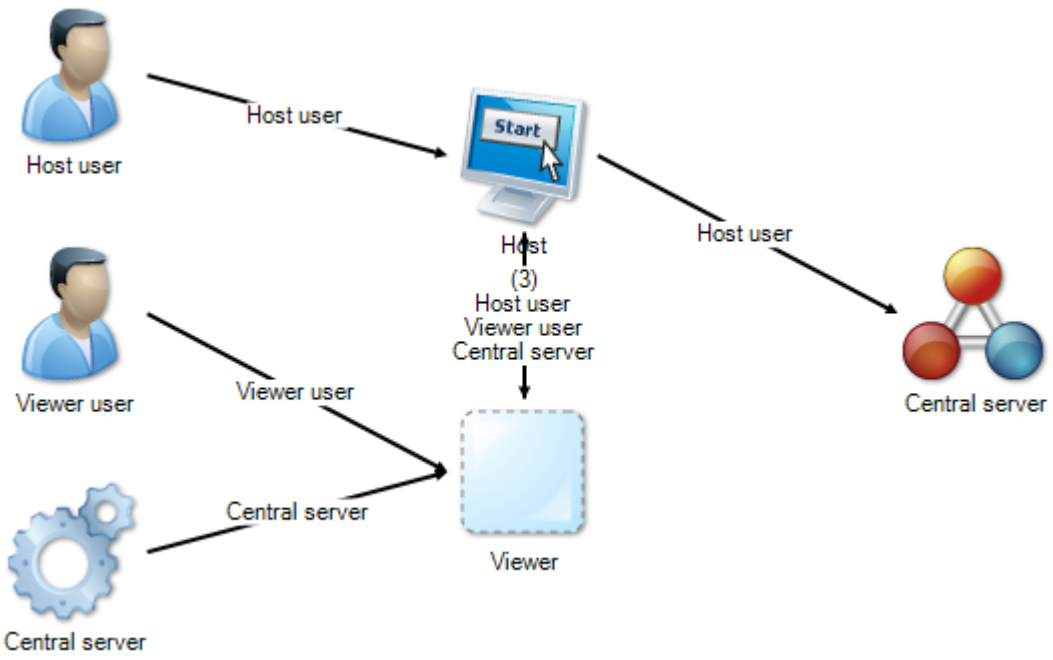
Task Items
☑Use cryptographically strong keys
☑Use well-known cryptographic algorithms
☑Utilize platform to store secret key
☑Utilize IPSec with Encryption
☑Salt the hash value with a unique random bits

Brute force password
Risk Rating: 2
Risk Response: Avoid
Confidentiality: Yes
Integrity: Yes
Availability: Yes

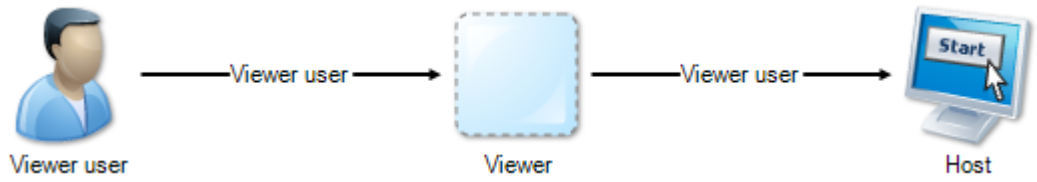
Task Items
☑Enforce password complexity requirement

Trust flows

Connect action trust flow



Remote control from viewer action trust flow



Remote host control action trust flow

User documentation

User tutorial

Host tutorial

1. Start ScreenViewer program.
2. Create account by entering your username and password in opened program window.
3. Press Host button.
4. Wait for a viewer to connect to your computer.

Note for 2: Please be sure that your computer is connected to the internet and OS firewall is not blocking program.

Viewer tutorial

1. Start ScreenViewer program.
2. Enter Host‘s username and password in opened window.
3. Press View button.
4. Control remote destop.

Note for 2: Please be sure that your computer is connected to the internet and OS firewall is not blocking program.