A Security Solution For Wireless IP Networks

EPFL Semester Project

Jean-Philippe Pellet
jean-philippe.pellet@epfl.ch

14th April, 2005

Outline

Wireless IP Network Security Solutions

- Wired Equivalent Privacy (WEP)
  - Default security solution
  - Not secure
    (Fluhrer et al., 2001)
- Wi-Fi Protected Access (WPA)
  - New standard
  - Drivers not always available
  - Implementation: huge work, testing difficult
- Virtual Private Network (VPN)
  - Family of security solutions with same principles
  - Secure IP traffic through untrusted nets
  - Multiple implementations; open source projects

Wireless IP Network Security Solutions

- Wired Equivalent Privacy (WEP)
  - Default security solution
  - Not secure
    (Fluhrer et al., 2001)
- Wi-Fi Protected Access (WPA)
  - New standard
  - Drivers not always available
  - Implementation: huge work, testing difficult
- Virtual Private Network (VPN)
  - Family of security solutions with same principles
  - Secure IP traffic through untrusted nets
  - Multiple implementations; open source projects

Wireless IP Network Security Solutions

- Wired Equivalent Privacy (WEP)
  - Default security solution
  - Not secure
    (Fluhrer et al., 2001)
- Wi-Fi Protected Access (WPA)
  - New standard
  - Drivers not always available
  - Implementation: huge work, testing difficult
- Virtual Private Network (VPN)
  - Family of security solutions with same principles
  - Secure IP traffic through untrusted nets
  - Multiple implementations; open source projects

Outline

Security Solution Requirements

- Data encryption
- Client authorization
- Scalability to hundreds of clients
- Cross-platform (Linux, Mac OS X, Windows)
- User-friendly GUI
- Royalty-free/open source software

Possible VPN Solutions

- IPSec (Secure Internet Protocol)
  - Very flexible, a lot of implementations
  - Proprietary extensions: not always compatible
- Point-to-Point Tunneling Protocol (PPTP)
  - Microsoft's VPN/tunneling protocol
  - Security flaws in the past
- VTun
  - Open source VPN project
  - Proprietary protocol, security flaws
- OpenVPN
  - Open source VPN project
  - Relies on OpenSSL
  - Runs on all target platforms

Possible VPN Solutions

- IPSec (Secure Internet Protocol)
  - Very flexible, a lot of implementations
  - Proprietary extensions: not always compatible
- Point-to-Point Tunneling Protocol (PPTP)
  - Microsoft's VPN/tunneling protocol
  - Security flaws in the past
- VTun
  - Open source VPN project
  - Proprietary protocol, security flaws
- OpenVPN
  - Open source VPN project
  - Relies on OpenSSL
  - Runs on all target platforms

Possible VPN Solutions

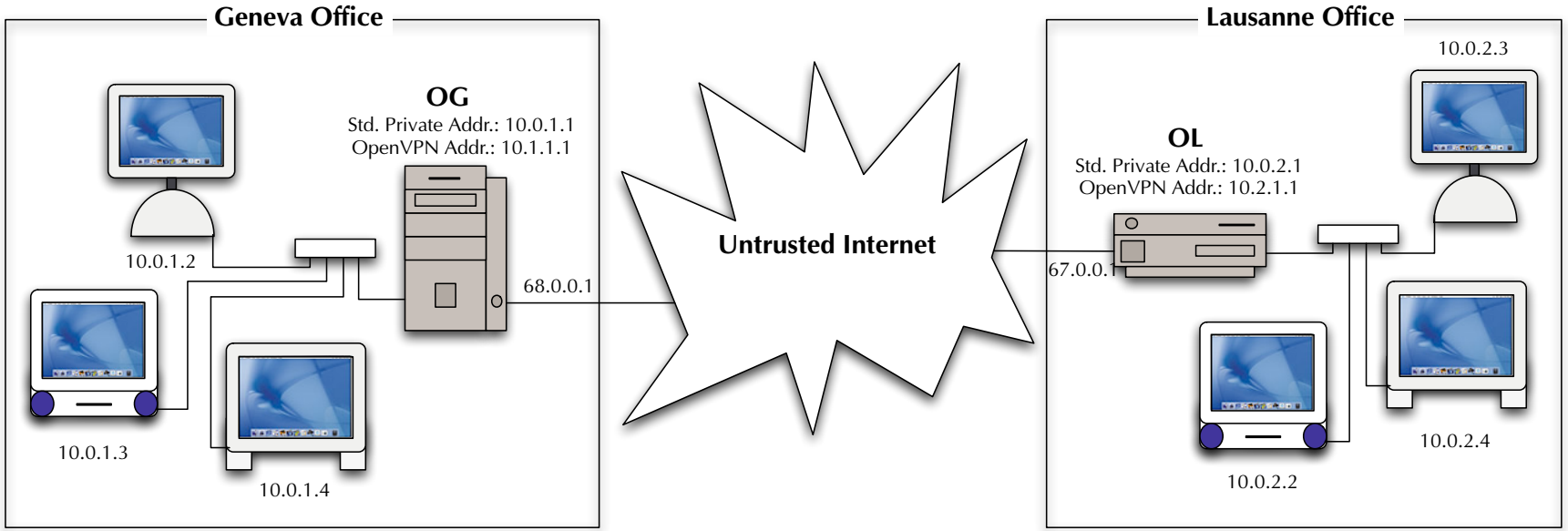
- IPSec (Secure Internet Protocol)
  - Very flexible, a lot of implementations
  - Proprietary extensions: not always compatible
- Point-to-Point Tunneling Protocol (PPTP)
  - Microsoft's VPN/tunneling protocol
  - Security flaws in the past
- VTun
  - Open source VPN project
  - Proprietary protocol, security flaws
- OpenVPN
  - Open source VPN project
  - Relies on OpenSSL
  - Runs on all target platforms

Possible VPN Solutions

- IPSec (Secure Internet Protocol)
  - Very flexible, a lot of implementations
  - Proprietary extensions: not always compatible
- Point-to-Point Tunneling Protocol (PPTP)
  - Microsoft's VPN/tunneling protocol
  - Security flaws in the past
- VTun
  - Open source VPN project
  - Proprietary protocol, security flaws
- OpenVPN
  - Open source VPN project
  - Relies on OpenSSL
  - Runs on all target platforms

Outline

How a Traditional VPN Works I



How a Traditional VPN Works II

- Tunnel mode: two communicating peers
- Public & local IP addresses
- Listen for traffic on special OpenVPN IP address
- Traffic encrypted & forwarded to other peer
- At other peer: traffic decrypted & forwarded on subnet
- Routing tables must be modified OpenVPN machine's address
- $\Rightarrow$ Seamless secure subnet interconnection

How a Traditional VPN Works II

- Tunnel mode: two communicating peers
- Public & local IP addresses
- Listen for traffic on special OpenVPN IP address
- Traffic encrypted & forwarded to other peer
- At other peer: traffic decrypted & forwarded on subnet
- Routing tables must be modified OpenVPN machine's address
- $\Rightarrow$ Seamless secure subnet interconnection

OpenVPN Encryption Modes I

Data tunneled can be encrypted by OpenVPN:

- No encryption at all
  - Only tunneling. Not what we want.
- Encryption based on a pre-shared key mechanism
  - How WEP works.
- Encryption with TLS-based mechanism
  - TLS: Transport Layer Security. Successor of SSL, Secure Sockets Layer
  - Public certificates + random numbers $\Rightarrow$ session key
  - Session keys periodically renegociated

OpenVPN Encryption Modes I

Data tunneled can be encrypted by OpenVPN:

- No encryption at all
  - Only tunneling. Not what we want.
- Encryption based on a pre-shared key mechanism
  - How WEP works.
- Encryption with TLS-based mechanism
  - TLS: Transport Layer Security. Successor of SSL, Secure Sockets Layer
  - Public certificates + random numbers $\Rightarrow$ session key
  - Session keys periodically renegociated

OpenVPN Encryption Modes I

Data tunneled can be encrypted by OpenVPN:

- No encryption at all
  - Only tunneling. Not what we want.
- Encryption based on a pre-shared key mechanism
  - How WEP works.
- Encryption with TLS-based mechanism
  - TLS: Transport Layer Security. Successor of SSL, Secure Sockets Layer
  - Public certificates + random numbers $\Rightarrow$ session key
  - Session keys periodically renegociated

OpenVPN Encryption Modes II

- Pre-shared key mechanism:
  - Same key used by both peers
  - Key cannot be renegociated
  - Scales badly
- TLS-based dynamic key exchange:
  - Private/public key pairs
  - Public certificates are exchanged
  - Mutual authentication
  - Scales well. Chosen for the rest of the project

OpenVPN Encryption Modes II

- Pre-shared key mechanism:
  - Same key used by both peers
  - Key cannot be renegociated
  - Scales badly
- TLS-based dynamic key exchange:
  - Private/public key pairs
  - Public certificates are exchanged
  - Mutual authentication
  - Scales well. Chosen for the rest of the project

OpenVPN TLS Server Mode

- Client/server architecture
- `tls-server` and `tls-client` OpenVPN modes
- OpenVPN 2: new server mode:
  - Creates tunnels with multiple clients
  - Dynamically assign VPN addresses
  - Scales better

Outline

Signed Certificate

- Certificates must be signed by Certification Authority (CA)
- Unsigned certificates $\Rightarrow$ connection fails
- $\Rightarrow$ Access Control (one of our requirements)
- CA freely chosen:
  - Trusted third part
  - One of the peers
    - OpenSSL functions can do this

OpenVPN & Target Platforms

- Linux
  - + LZO Library
  - ⇒ `configure`, `make`, `make install`
- Mac OS X
  - + LZO Library
  - + Third-party TUN/TAP driver
  - ⇒ `configure`, `make`, `make install`
  - OpenVPN must be run as root to allocate TUN/TAP device
- Windows
  - Precompiled binaries, GUI installer
  - Windows 2k/XP security warning (unsigned TAP driver)

OpenVPN & Target Platforms

- Linux
  - + LZO Library
  - ⇒ `configure`, `make`, `make install`
- Mac OS X
  - + LZO Library
  - + Third-party TUN/TAP driver
  - ⇒ `configure`, `make`, `make install`
  - OpenVPN must be run as root to allocate TUN/TAP device
- Windows
  - Precompiled binaries, GUI installer
  - Windows 2k/XP security warning (unsigned TAP driver)

OpenVPN & Target Platforms

- Linux
  - + LZO Library
  - ⇒ `configure`, `make`, `make install`
- Mac OS X
  - + LZO Library
  - + Third-party TUN/TAP driver
  - ⇒ `configure`, `make`, `make install`
  - OpenVPN must be run as root to allocate TUN/TAP device
- Windows
  - Precompiled binaries, GUI installer
  - Windows 2k/XP security warning (unsigned TAP driver)

Outline

Running OpenVPN

- Command-line tool
- Runtime options:
  - Passed as arguments
  - Stored in config file:
    `openvpn --config` *`configfile.ovpn`*
- Distinguish two configs:
  - Server config
    - Accept several clients
    - Distribute addresses
    - Check clients' availability
  - Client config
    - Connect to server
    - Inform user of errors
    - Modify routing tables to use tunnel

Running OpenVPN: Server Configuration I

```
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh.pem
server 10.0.5.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway"
client-to-client
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status.log
```

Running OpenVPN: Server Configuration II

- More commands needed on server side
- Ensure packets coming out of the tunnel are forwarded

1. `echo 1 > /proc/sys/net/ipv4/ip_forward`
2. `iptables -A FORWARD -i tun+ -j ACCEPT`
3. `iptables -t nat -A POSTROUTING -j MASQUERADE`

Running OpenVPN: Client Configuration

```
client
dev tun
proto udp
remote 192.168.0.1 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client.crt
key client.key
reneg-sec 3600
comp-lzo
verb 1
```

Outline

Motivation & Requirements

- Goal:
  - Hide command-line aspect
  - Make connection as simple as Click & Connect
  - Better integration with OS
- Requirements:
  - Shell interaction capabilities
  - Platform independence
  - Standalone executable

Motivation & Requirements

- Goal:
  - Hide command-line aspect
  - Make connection as simple as Click & Connect
  - Better integration with OS
- Requirements:
  - Shell interaction capabilities
  - Platform independence
  - Standalone executable

Considered IDEs

- REALbasic
  - Modern, OO BASIC implementation
  - Standalone binaries for Mac OS, Win32 and Linux/x86
  - Extended shell support, but only for Mac OS X
- C# & .Net
  - Basic shell support
    ⇒ not enough to interact with OpenVPN
  - Not cross-patform
  - IDE not free
- Java
  - Cross-platform
  - Good shell interaction support
  - Not standalone: require JRE
    ⇒ Bundlers/Wrappers for JAR files needed

Considered IDEs

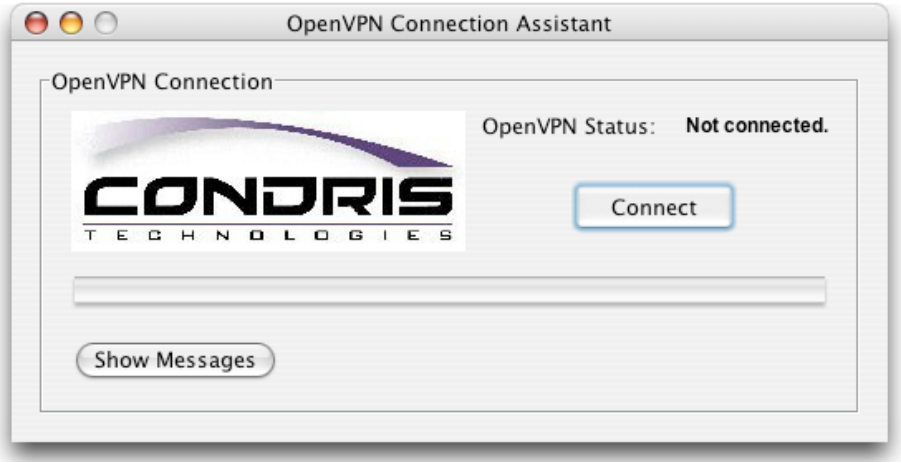
- REALbasic
  - Modern, OO BASIC implementation
  - Standalone binaries for Mac OS, Win32 and Linux/x86
  - Extended shell support, but only for Mac OS X
- C# & .Net
  - Basic shell support
    $\Rightarrow$ not enough to interact with OpenVPN
  - Not cross-patform
  - IDE not free
- Java
  - Cross-platform
  - Good shell interaction support
  - Not standalone: require JRE
    $\Rightarrow$ Bundlers/Wrappers for JAR files needed

Considered IDEs

- REALbasic
  - Modern, OO BASIC implementation
  - Standalone binaries for Mac OS, Win32 and Linux/x86
  - Extended shell support, but only for Mac OS X
- C# & .Net
  - Basic shell support
    $\Rightarrow$ not enough to interact with OpenVPN
  - Not cross-patform
  - IDE not free
- Java
  - Cross-platform
  - Good shell interaction support
  - Not standalone: require JRE
    $\Rightarrow$ Bundlers/Wrappers for JAR files needed

Interface Requirements

- Familiar Look-and-feel
- Single-window interface
- Possibility to see OpenVPN's output
- Disconnect on close
- Logging capabilities: keep track of errors



Outline

GUI Assistant Implementation I

- OpenVPN process $\longrightarrow$ `java.lang.Process` object
- Output monitored by separate Java thread:
  - Monitor progression
    - Same successful connection messages
    - Update progress bar accordingly
  - Detect error messages
    - Visual feedback on failure
    - More descriptive error descriptions
  - Log output on failure

GUI Assistant Implementation II

- Error messages/connection messages can change with future versions of OpenVPN
  - Store them in external file
  - Load them dynamically at run time
  - $\Rightarrow$ No need to recompile the Java code
- Labels, captions, titles, messages: internationalization
  - Store them in a external file
  - Good solution: Java's `properties` files
  - Loaded automatically according to current locale
  - Can reside in a JAR file

GUI Assistant Implementation II

- Error messages/connection messages can change with future versions of OpenVPN
  - Store them in external file
  - Load them dynamically at run time
  - $\Rightarrow$ No need to recompile the Java code
- Labels, captions, titles, messages: internationalization
  - Store them in a external file
  - Good solution: Java's `properties` files
  - Loaded automatically according to current locale
  - Can reside in a JAR file

GUI Assistant Implementation III, Platform Caveats

- Windows
  - Default gateway can be lost
  - ⇒ Save and restore it: `route PRINT`, `route ADD`
  - Processes cannot be terminated from Java code
  - ⇒ `taskkill` (WinXP) or `kill` tools
  - Check availability for previous Windows versions
- Mac OS X
  - Run as root to use TUN/TAP driver

GUI Assistant Implementation III, Platform Caveats

- Windows
  - Default gateway can be lost
  - ⇒ Save and restore it: `route PRINT`, `route ADD`
  - Processes cannot be terminated from Java code
  - ⇒ `taskkill` (WinXP) or `kill` tools
  - Check availability for previous Windows versions
- Mac OS X
  - Run as root to use TUN/TAP driver

GUI Assistant Deployment: Installation, JVMs

JVM available? If no, install/provide one. Options are:

- InstallAnywhere
  - Bundles JVMs
  - $\Rightarrow$ professional-looking, platform-tailored installers
  - Too high price: $2999
- JSmooth (for Windows only)
  - Wraps JAR file into EXE
  - Looks for installed JVMs; can launch bundled JVM
  - GNU Public Licence
- JarBundler (for Mac OS X only)
  - Wraps JAR file into OS X application
  - Native OS X behaviour for Java app
  - Mac OS X-specific options

GUI Assistant Deployment: Installation, JVMs

JVM available? If no, install/provide one. Options are:

- InstallAnywhere
  - Bundles JVMs
  - $\Rightarrow$ professional-looking, platform-tailored installers
  - Too high price: $2999
- JSmooth (for Windows only)
  - Wraps JAR file into EXE
  - Looks for installed JVMs; can launch bundled JVM
  - GNU Public Licence
- JarBundler (for Mac OS X only)
  - Wraps JAR file into OS X application
  - Native OS X behaviour for Java app
  - Mac OS X-specific options

GUI Assistant Deployment: Installation, JVMs

JVM available? If no, install/provide one. Options are:

- InstallAnywhere
  - Bundles JVMs
  - $\Rightarrow$ professional-looking, platform-tailored installers
  - Too high price: $2999
- JSmooth (for Windows only)
  - Wraps JAR file into EXE
  - Looks for installed JVMs; can launch bundled JVM
  - GNU Public Licence
- JarBundler (for Mac OS X only)
  - Wraps JAR file into OS X application
  - Native OS X behaviour for Java app
  - Mac OS X-specific options

Outline

Architectural Context I

- Headless Linux Server
- Configured through JSPs
- JSPs affect Hibernate DB
- Changes in settings:
  - New config files
  - Start/stop service
- Wanted: automate file generation

Architectural Context II

- A JSP $\widehat{=}$ a `Section` in DB
- A section is linked to:
  - Multiple files
  - Multiple services
- Config file made from:
  - Values from different sections
  - Values not in sections
  - File name not constant

Sketchy Algorithm

When changing section S_0:

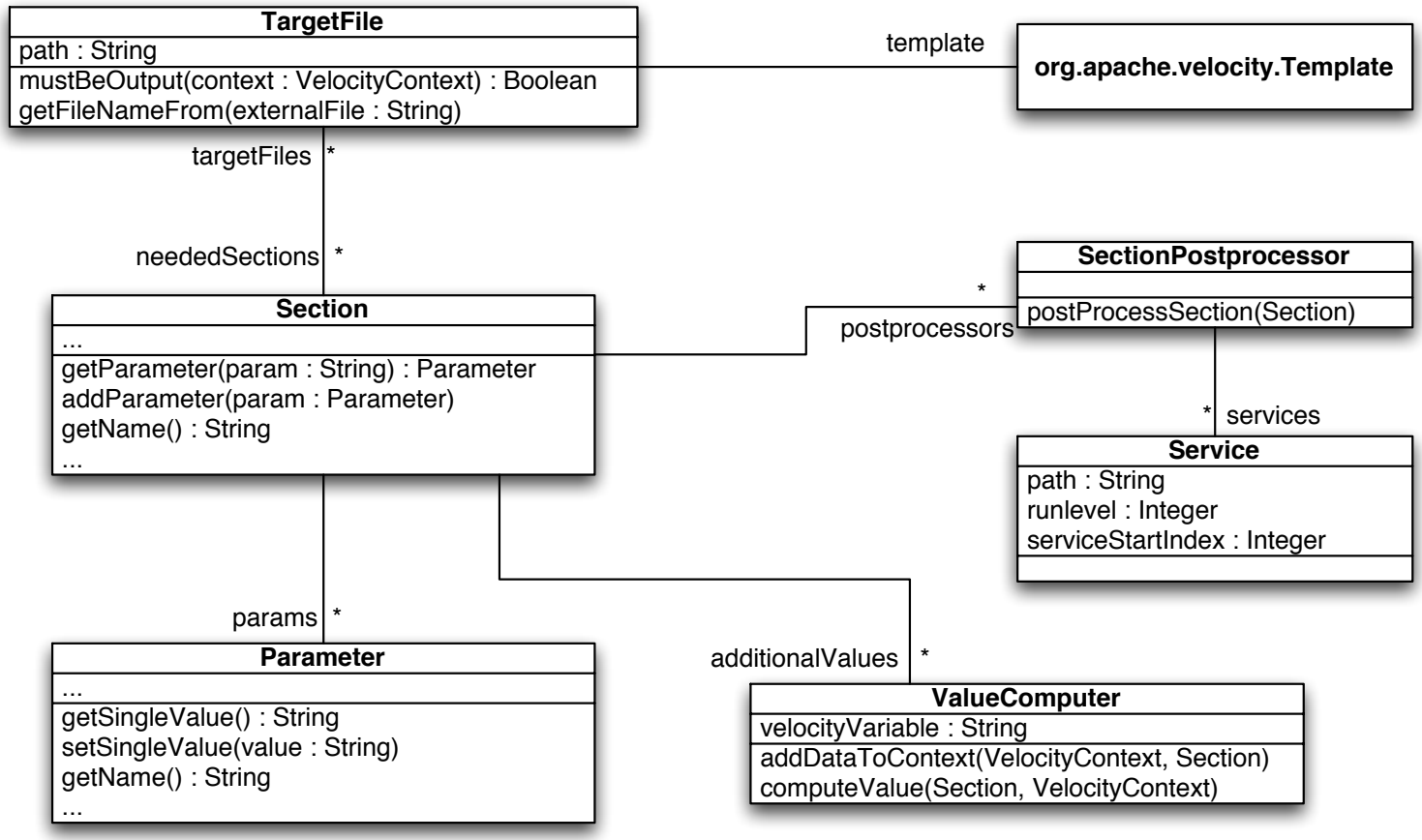
- Let F = all files linked to S_0
- Let $S = \bigcup_{f \in F}$ [all sections needed by f]
- For each $f \in F$:
  - Retrieve needed values v_i from S
  - Generate additional needed values w_i using v_i
  - Generate f
- Start/stop all services linked to S_0 using values from S_0

Outline

Velocity Framework

- Template + `VelocityContext` object ⇒ output file
- `VelocityContext` object:
  - Key/value map
  - Keys are strings
  - Values are any Java object
- Template:
  - Text file
  - Uses the `VelocityContext`'s values
  - Invokes Java methods using reflection
  - Simple language to manipulate Velocity variables

UML Class Diagram



Implementation Issues

- Needs to replace system files
  $\Rightarrow$ Must be run as root
- Not enough to start or stop Linux service
  (not persistant across reboot)
- $\Rightarrow$ Deal with symlinks in `/etc/init.d/rcx.d/`
- To include templates in JAR file:
  Use alternate resource loader

Summary I: What We Have Done So Far

- Secure Wireless Network
  $\Rightarrow$ VPN solution
  $\Rightarrow$ OpenVPN
  - Run mode
  - Config file
  - GUI Connection Assistant
    - Java implementation
    - Platform-dependant deployment
- Configuration File Generation
  - Java implementation with Velocity

Summary I: What We Have Done So Far

- Secure Wireless Network
  $\Rightarrow$ VPN solution
  $\Rightarrow$ OpenVPN
  - Run mode
  - Config file
  - GUI Connection Assistant
    - Java implementation
    - Platform-dependant deployment
- Configuration File Generation
  - Java implementation with Velocity

Summary I: What We Have Done So Far

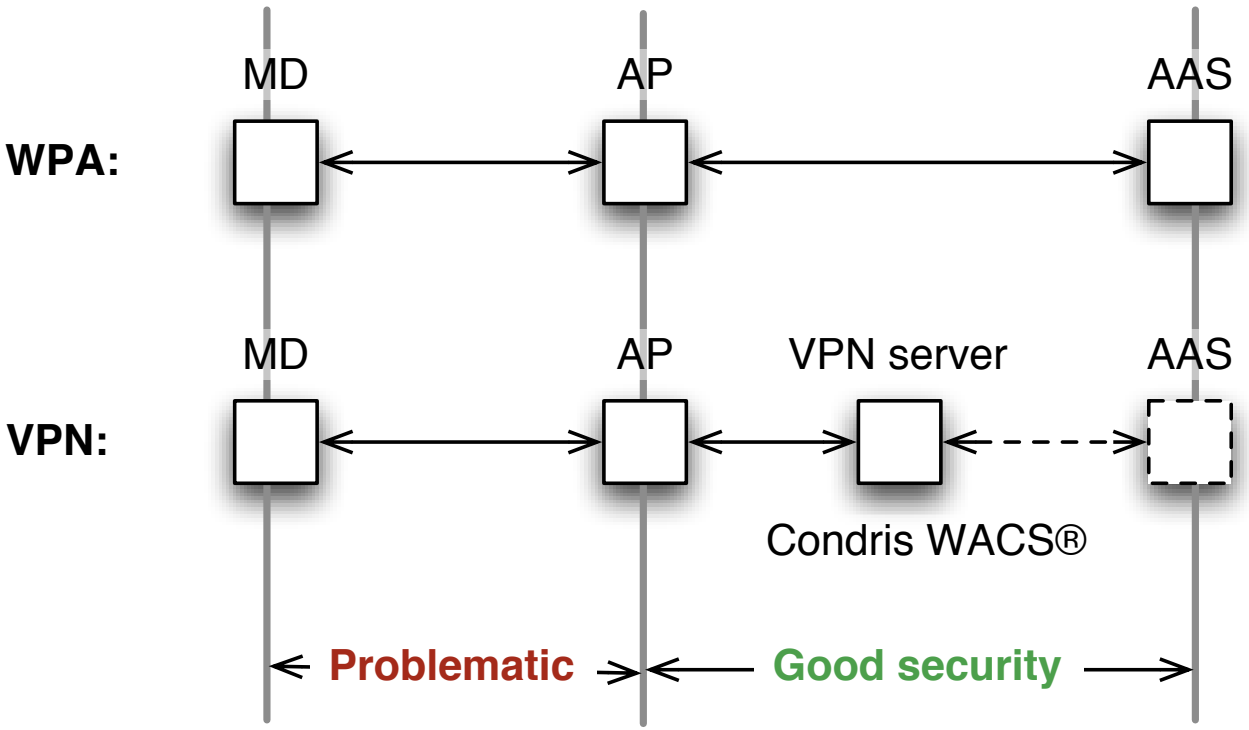
- Secure Wireless Network
  $\Rightarrow$ VPN solution
  $\Rightarrow$ OpenVPN
  - Run mode
  - Config file
  - GUI Connection Assistant
    - Java implementation
    - Platform-dependant deployment
- Configuration File Generation
  - Java implementation with Velocity

Summary I: What We Have Done So Far

- Secure Wireless Network
  $\Rightarrow$ VPN solution
  $\Rightarrow$ OpenVPN
  - Run mode
  - Config file
  - GUI Connection Assistant
    - Java implementation
    - Platform-dependant deployment
- Configuration File Generation
  - Java implementation with Velocity

Summary II: Concluding Discussion

- VPN solution scope $\neq$ WPA's



Thanks for your attention!