Invicro
A Konica Minolta Company

# Security Features for iPACS® 2020

Original Document Release: 30 April 2020
Document Version 1.0

# Security Features for iPACS 2020

## Objective

This document describes security-related features in the current stable version of iPACS 2020.

## Table of Contents

## HTTP Secure (HTTPS)

While the standard Hypertext Transfer Protocol (HTTP) transmits data in plain text, iPACS uses Transport Layer Security (TLS) (1.2 or higher—use of TLS 1.1 can be enabled if required but is disabled by default) which encrypts all communication and allows authentication of the server.

Non-secure, older versions of the Secure Sockets Layer (SSL) protocol (in 2.0 and 3.0) as well as TLS 1.0 and 1.1, are disabled by default. TLS ensures users are connecting to the server they intend – for instance, it only sends login credentials to the expected authority by checking that the server address matches the encryption certificate provided by the server. These certificates are typically provided by customer IT groups, but can be provided by Invicro for installations hosted by Invicro. The full encryption of all communication protects login credentials as well as confidential imaging data and patient information.

HTTPS is based on public key cryptography, which means clients are not required to install pre-shared keys, rather keys for encryption are provided by the server and cryptographically signed by a public trust center.

All iPACS cloud systems hosted by Invicro employ HTTPS or can be upgraded at the customer's request.

## Automatic Redirect from HTTP to HTTPS

By default, iPACS will automatically redirect (301 – Moved Permanently) access to the HTTP port of an iPACS to the secure version, HTTPS, so users will always use the security optimized version without any extra effort.

## HTTP Strict Transport Security (HSTS)

Many modern browsers (e.g., Chrome, Firefox, Opera, Internet Explorer, Edge) support the HSTS Internet Engineering Task Force (IETF) standard that makes sure browsers always use the secure HTTPS version of iPACS once the user has visited this version for the first time. Similar to the automatic redirect, this feature ensures the end-user utilizes the secure version, while allowing non-HTTPS-capable web clients to access the service. Moreover, HSTS protects against protocol down-grade attacks.

## Perfect Forward Secrecy (PFS)

The web server used by iPACS is, by default, configured to provide PFS, which ensures that a session key derived from a set of long-term public and private keys will not be compromised if one of the (long-term) private keys is compromised in the future. Consequently, disclosure of the private key of the server used to decrypt messages at some point does not allow decryption of earlier recorded sessions.

## Reverse Proxy – Demilitarized Zone (DMZ) Support

For added security, the iPACS can also be configured to work with reverse proxy web servers, which are able to greater control web traffic. These backend servers act as single access points or gateways in server farms and forward all HTTP(S) traffic from external users to an iPACS server on a DMZ or internal customer network.

## Firewall

By default, all non-used ports of an iPACS server are protected by Linux' iptables firewall, limiting access to the server to only the services exposed explicitly. Additional restrictions (e.g., limiting access to specific source networks) can be added by the customer's IT, or via Invicro support.

Additionally, services used by iPACS only bind to public network devices where required, while using the localhost loopback or Unix sockets where possible.

Additional infrastructure firewalls are available depending on the deployment environment. These can make sure configuration or software errors do not result in opening ports for unauthorized sources. Such firewalls cannot be changed or shutdown from the Linux instance, even with root permissions. For instance:

- AWS: All AWS instances use very strict security groups, which are virtual firewalls, managed via the AWS console.
- Hetzner: Infrastructure firewalls, managed in the Robot console are available.
- New Haven DC: Infrastructure firewalls, managed by the IT team are available.

## Secure Password Storage (salted SHA-1 or SHA-256)

All passwords stored in iPACS internal user database are maintained as salted hashes. Consequently, any compromise of the user database does not immediately lead to an exposure of user passwords.

The passwords are not stored themselves; rather, a one-way hash using the Secure Hash Algorithm SHA-256 that produces a 256-bit checksum of the password is used. On login, the user password is hashed again and if the checksums match, iPACS assumes the password is correct. Additionally, the hashes are salted by adding a random number to the password, which makes applying pre-calculated hash dictionaries (rainbow tables) much more difficult.

The iPACS log files never contain user passwords.

## Secure LDAP (LDAPS) or Active Directory Secure Central User Management & Single-Sign-On

As an alternative or in addition to the internal user database, iPACS can also authenticate users against a Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory (AD) service, including full encryption of the communication between iPACS and the LDAP/AD server over a TLS Layer. Thus, login credentials are securely transmitted from the end-user to the final authentication server. Password expiry warnings and password changes via iPACS are also supported, as are multiple LDAP/AD servers for a high-availability solution.

## Single Sign-On (SSO) via OpenID Connect (OAuth 2.0)

Starting in version 2020, iPACS can additionally be configured to delegate authentication to an OpenID Connect server, such as Okta, Auth0 or Microsoft's Azure AD (including Office 365). This allows for central user management similar to LDAP/AD, with the additional benefit that users only have to login once per session and thus can more quickly get access to different web services, while still benefiting from the highest security features offered by the OpenID Connect provider (e.g., risk-based logins, MFA).

## Multi-Factor Authentication (MFA)

In addition to the standard password (something the user knows), iPACS 2.01 or higher can be configured to require two-factor authentication (2FA)(something the user has), which use time-dependent one-time tokens (TOTP) generated on the user's smart phone (supported by iOS, Android, etc.), which renew every 30 seconds. This dramatically increases security over a single factor, since phishing attacks no longer work.

For convenience and to increase acceptance, the TOTP tokens can be replaced by allowing access from trusted networks (IP based second factor), such as the company intranet. Additionally, devices with successful MFA authentication can be remembered for a specified number of days. The MFA setting can be configured to be optional, mandatory for all users, or mandatory for specific groups, such as all admin users.

## Password Complexity Rules

iPACS follows best-practices to increase password complexity and protect against dictionary attacks. Namely, all passwords must satisfy the following requirements:

- Contain eight or more characters
- Contain at least one lower case letter (a-z)
- Contain at least one upper case letter (A-Z)
- Contain at least one non-alphanumeric character (e.g., ?, %, *, $)
- Contain at least one digit (0-9)

iPACS includes the ability to configure a password expiration policy (e.g., the user must change their password every X days).

iPACS also includes the ability to configure a password duplication interval to prevent previously used passwords from being employed again (e.g., you can't reuse a password you've used in the past X days).

## Containerization Using Docker

iPACS services are encapsulated into separate Docker containers with well-defined interfaces between them. Similar to multiple virtual machines running on one physical host, the containers also allow running services that are walled off from each other and, most importantly, from the host operating system. As such, a break-in to the most exposed web application container does not automatically also compromise the underlying host operating system. In contrast to virtual machines, the container overhead is minimal and does not impede performance of the system.

## Cross Site Request Forgery (CSRF) – Request Token

Once the user has logged in to an iPACS successfully, requests from the IP address and with the secure session cookie could be used to execute commands on an iPACS even if the user has not explicitly authorized them (e.g., a user could be misguided into clicking a link or otherwise requesting an iPACS resource). As a result, this can trigger an action on the server that a user is unaware (e.g., deleting a file). To avoid these CSRF attacks, iPACS protects potentially dangerous operations by issuing one-time request tokens that must be submitted with the request, and the request is only executed if a valid token is found. Since an attacker cannot identify these tokens, this is an effective method to avoid this attack vector.

## Cross-Site Scripting (XSS)

Another frequent attack on websites is XSS, where the attacker attempts to embed its own code into the website (e.g., by producing error messages or posting text which contain raw code). To avoid this issue, the underlying web Model, View and Controller (MVC) framework used by iPACS automatically escapes potentially dangerous code in all error messages and makes it easier for developers to escape such code in all output templates. Successful implementation of XSS counter-measures is tested using website auditing software, such as Acunetix, NetSparker, or HCL AppScan. Additionally, such checks are part of the automated unit tests performed nightly for iPACS builds.

Invicro provides support for customers to perform their own web scans in their own environment and Invicro can provide example reports of our cloud-based test servers using one of the above auditing software.

## SQL Injection

Similar to XSS, an attacker could try to inject raw SQL code into the communication of iPACS with the backend relational database. This is countered by iPACS' database abstraction layer which provides an object relational mapping and automatically quotes all user-input parameters bound to an SQL query. In a few cases, raw SQL must be used and developers must take extra precautions to quote and properly handle bind-values manually. SQL injections are tested nightly by automated unit tests as well as during the web-site vulnerability tests described above.

## Snapshots (AWS or New Haven DC)

In virtual deployments at Amazon Web Services (AWS) or the New Haven DC, snapshots of an iPACS instance can be created nightly or at a higher frequency, as requested. Such snapshots are application consistent, by quiescing the MySQL database as well as the filesystem. As such, these snapshots are immediately ready to use - no database or filesystem recovery required.

## Encryption at Rest (AWS or New Haven DC)

On all AWS instances, as well as optionally for instances in the New Haven DC, the data volume is or can be encrypted. On AWS, the encryption built into the Elastic Block Store (EBS) is used, while in the New Haven DC, Linux Unified Key Setup (LUKS) encryption is offered.