We had a debug session with a customer today, where there was an issue with the Forward Proxy. While debugging the issue, we observed several unexpected problems that can have very serious stability and security implications.

1. The Node Manager sent a config file to the Node Server that contained a barren **&** character (in the keystore password, see #2). This should be impossible if the Node Manager uses proper XML serialization. Because the XML is invalid, the config message was not processed.
2. The keystore passwords for all certificates are coming unencrypted in the configuration message. By design, these should be encrypted fields, but there appears to be a regression.

[dhiraj] Checked NM side code appears to be unchanged since 2018. SensitiveFieldUtil is used to encrypt the context.

@Nullable

public static KeyStoreSpecXO getKeyStoreSpecsFromSettings(@Nullable KeyAndCertificateStore certificateStore) {

if (certificateStore == null) { return null; }

if (certificateStore.getCertStoreType() != KeyAndCertificateStore.KeyAndCertStoreType.JKS) { return null; }

KeyStoreSpecXO xo = *initKeyStoreSpec*(certificateStore);

SensitiveFieldUtil.setRedactable(certificateStore.getFileContents(), xo::setKeyStoreContent);

return xo;

}

SSN 566-33-4320

1. When trying to restart the proxy service from MS, we got into a state where the whole Proxy was down and the docker container had to be restarted by Ops. I don’t think we fully understand how we got into this state.
2. Even though we have Nginx TLS termination in front of the Reverse Proxy, we use the same private key and certificate in both. Theoretically, we don’t need to reuse the same public certificate in the Proxy if we know it’s fronted by Nginx. Note: this may not matter if we want to get away from using Nginx TLS termination for reverse proxy.
3. The config message contains both certificates (including private keys) managed by the customer (FP CA certificate), and certificates owned by CipherCloud (\*.ciphercloud.io certificate).  
     
   The whole config file, including the keystores and the passwords are logged at ERROR level when there’s an XML validation error (caused by #1). Because of #2, a savvy customer can today download the logs and literally get our \*.ciphercloud.io TLS private key. Note: hopefully this should be addressed when we redesign customer access to logs in multi-tenant deployments.  
     
   Conversely, it is possible for anyone with access to the logs (i.e. our Ops guys and any MS admin on a tenant) to get private keys and passwords for customer-uploaded certificates.
4. The password that is used to protect the **\*.ciphercloud.io** certificate/private key in production is the same password that’s hardcoded in our BaseTest and that every developer knows.

I think we need a redesign to make sure that config messages don’t have sensitive data (passwords) in the clear, even if the message is over tls; or an ironclad framework to prevent them from being logged by any library.

Additionally, we may consider rotating the \*.ciphercloud.io certificate. We should also revisit how we’re handling different passwords in our system, and specifically where we reuse passwords and/or leave them as defaults.