The Thing About Protecting Data Is, You Have To Protect Data

Andy LoPresto

Hortonworks DataFlow
06/27/16

HORTONWORKS®
POWERING THE FUTURE OF DATA™

Contents

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Agenda

Introduction

Common Security Risks

Examples

NiFi Security Features

Roadmap

Questions

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Agenda

Introduction

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Introduction

Andy LoPresto

- Hortonworks DataFlow Team — Security
- @yolopey
- alopresto@hortonworks.com
  - PGP Fingerprint: 9B51 8C0D A489 0D3C 2F5B 0F2B **125A 4E68 51BF 2B79**
- alopresto@apache.org
  - Committer, Apache NiFi
  - PGP Fingerprint: 70EC B3E5 98A6 5A3F D3C4 BACE **3C6E F65B 2F7D EF69**

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Agenda

Introduction

Common Security Risks

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

If a security issue is challenging, people often convince themselves they can make up for it elsewhere, and they move on

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

TLS Issues

- Configuration Issues
- Certificate Validation
  - Requests for “passwordless keystores/truststores”
- Hostname Verification
- Legacy Protocols/Cipher Suites
  - Logjam (Weak DH)
  - SSLv3
- Downgrade Attacks
  - POODLE
  - FREAK

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Authentication & Authorization Challenges

- Identity Stores
  - Duplication of data/management effort
- Infrastructure Overhead for PKI
- Single Sign-On

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Plaintext Data on Disk

- Configuration (Sensitive Values)
- Untrusted Infrastructure/OS
- Persisted Data
- Logs

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Data Obfuscation

- Weak Cryptographic Hash Functions (Crackable)
- Encoding (Reversible)

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Weak Encryption

- Legacy or Vulnerable Cipher Algorithms
  - (3)DES
  - RSA w/ small key size
  - ECB mode of operation
  - CTR with repeated nonce
- Password-Based Encryption (PBE)
  - Bad passwords
  - Bad Key Derivation Functions (KDF)

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Lack of Data Assurance

- Application Data Trust
- Unauthenticated Encryption

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Data Leakage

- Communication With External Services
  - Fat-finger one IP or hostname, and you broadcast your data to the Internet
- Poor Anonymization
  - Other products report “anonymous” statistics outside of organization using Google Analytics

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

"It turns out there's a significant flaw in the approach. Because both the medallion and hack numbers are structured in predictable patterns, *it was trivial to run all possible iterations through the same MD5 algorithm* and then compare the output to the data contained in the 20GB file. Software developer Vijay Pandurangan did just that, and *in less than two hours he had completely de-anonymized all 173 million entries*."

http://arstechnica.com/tech-policy/2014/06/poorly-anonymized-logs-reveal-nyc-cab-drivers-detailed-whereabouts/

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Agenda

Introduction

Common Security Risks

Examples

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

When I hear “No one will ever think of that”, I hear “I didn’t think of that.”

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

TLS Issues

- How confident are you that the apps/libraries you use correctly implement and verify TLS connections?

11

I've had to do something like this when using commons-httpclient to access an internal https server with a self-signed certificate. Yes, our solution was to create a custom TrustManager that simply passed everything (logging a debug message).

This comes down to having our own SSLSocketFactory that creates SSL sockets from our local SSLContext, which is set up to have only our local TrustManager associated with it. You don't need to go near a keystore/certstore at all.

So this is in our LocalSSLSocketFactory:

```
static {
    try {
        SSL_CONTEXT = SSLContext.getInstance("SSL");
        SSL_CONTEXT.init(null, new TrustManager[] { new LocalSSLTrustManager() }, null);
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException("Unable to initialise SSL context", e);
    } catch (KeyManagementException e) {
        throw new RuntimeException("Unable to initialise SSL context", e);
    }
}

public Socket createSocket(String host, int port) throws IOException, UnknownHostException {
    LOG.trace("createSocket(host => {}, port => {})", new Object[] { host, new Integer(port)

    return SSL_CONTEXT.getSocketFactory().createSocket(host, port);
}
```

Along with other methods implementing SecureProtocolSocketFactory. LocalSSLTrustManager is the aforementioned dummy trust manager implementation.

share edit flag

answered May 13 '09 at 17:25

53k • 14 • 97 • 97

6 ▲ If you disable all trust verification, there's little point using SSL/TLS in the first place. It's OK for testing locally, but not if you want to connect outside. – Bruno Dec 23 '11 at 13:26

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Weak Encryption

- Legacy or Vulnerable Cipher Algorithms
  - (3)DES
  - RSA w/ small key size
  - ECB mode of operation
  - CTR with repeated nonce
- Password-Based Encryption (PBE)
  - Bad passwords
  - Bad Key Derivation Functions (KDF)

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

CRACKING ENCRYPTION

- Aldrin wants to send an encrypted message to Yolanda
- Andy is curious and wants to break it
- Andy intercepts encrypted message (public WiFi MitM, DNS poisoning, etc.)

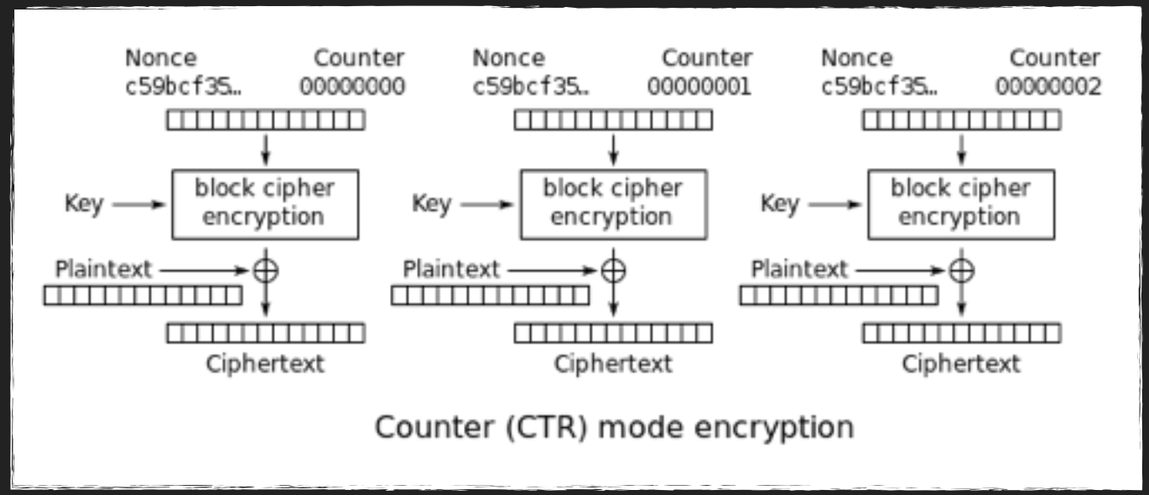
THESE ARE ALL DUMB EXAMPLES

Ok, fine

- Encrypting data with AES-CTR
- Well-known, well-documented, widely-used mode of operation
- Fast (hardware-optimized) and can operate as a parallelized stream cipher

SO FAR SO GOOD

- Counter mode works by combining an “Initialization Vector” (or “nonce”) and a “counter” value and then encrypting that result with the key
- This forms the keystream, and that keystream is simply XOR with the plaintext to create the cipher text



YOU HAD ONE JOB

- … as long as you *never* use the same nonce and key twice

SEE THE FLAW IN ACTION

- First encrypt message with AES-CTR 256 using key & IV

```
Plaintext  : DO NOT RESTART KAFKA SERVER | key: This is a bad keyThis is a bad k |
iv: This is a bad IV

Key hex: 546869732069732061206261642064206b6579546869732069732061206261642064206b (64) | IV
hex: 54686973206973206120626164204956 (32)

Cipher text: ycxZx2PeQHeMcuKjqAkHIm64tgEoM0si7Uew

Cipher text (hex): c9cc59c763de40778c72e2a3a80907226eb8b60128334b22ed47b0
```

CHOSEN PLAINTEXT ATTACK

```
Plaintext 1  : DO NOT RESTART KAFKA SERVER

Cipher text 1: ycxZx2PeQHeMcuKjqAkHIm64tgEoM0si7Uew | size: 36

Cipher text 1 (hex):
c9cc59c763de40778c72e2a3a80907226eb8b60128334b22ed47b0 | size: 54

Plaintext 2  : THIS IS ANDYS OTHER MESSAGE

Cipher text 2: 2csw2gzDMwWIb/K7qX1oPWe7r2BFJV0j+kWn | size: 36

Cipher text 2 (hex):
d9cb30da0cc33305886ff2bba97d683d67bbaf6045255d23fa45a7 | size: 54
```

XOR CIPHER TEXTS

Cipher Text 1 (hex):
c9cc59c763de40778c72e2a3a80907226eb8b60128334b22ed47b0

Cipher Text 2 (hex):
d9cb30da0cc33305886ff2bba97d683d67bbaf6045255d23fa45a7

XOR Cipher Text (hex):
1007691d6f1d7372041d101801746f1f090319616d161601170217

START DRAGGING CRIB

```
Crib: 'MESSAGE' | Hex: 4d455353414745
Position: 00 | CT len: 54 | Crib len: 14 | Crib: 'MESSAGE'
1007691d6f1d7372041d101801746f1f090319616d161601170217
4d455353414745
______________________________________________________
5d423a4e2e5a36 ]B:N.Z6

Position: 02 | CT len: 54 | Crib len: 14 | Crib: 'MESSAGE'
1007691d6f1d7372041d101801746f1f090319616d161601170217
  4d455353414745
______________________________________________________
  4a2c4e3c5c3437 J,N<\47

...
```

START DRAGGING CRIB

We know that “MESSAGE” is the last 7 characters of one plaintext…

```
Position: 40 | CT len: 54 | Crib len: 14 | Crib: 'MESSAGE'
1007691d6f1d7372041d101801746f1f090319616d161601170217
                                          4d455353414745
------------------------------------------------------
                                          20534552564552  SERVER

Crib results: 'MESSAGE'
0=]B:N.Z6
2=J,N<\47
...
40= SERVER
```

PARTIAL RECOVERY

- We have recovered a portion of the other plaintext that looks valuable
- If we knew one of the whole plaintexts…

COMPLETE RECOVERY

XOR Cipher Text: 1007691d6f1d7372041d101801746f1f090319616d161601170217

Crib: 'THIS IS ANDYS OTHER MESSAGE' | Hex: 544849532049532041 4e445953204f54484552204d455353414745

Position: 00 | CT len: 54 | Crib len: 54 | Crib: 'THIS IS ANDYS OTHER MESSAGE'

1007691d6f1d7372041d101801746f1f090319616d161601170217

544849532049532041 4e445953204f54484552204d455353414745

444f204e4f5420524553544152 54204b41464b4120534552564552 DO NOT RESTART KAFKA SERVER

Crib results: 'THIS IS ANDYS OTHER MESSAGE'

0=DO NOT RESTART KAFKA SERVER

EVEN EASIER WITH FORMATTED PLAINTEXT

```
Plaintext 1  : Format: Phil message one | size: 24
Cipher text 1: y+wL5E3+WgWZSd+O2jBCGlyfmiUoD2AV | size: 32
Cipher text 1 (hex): cbec0be44dfe5a059949df8eda30421a5c9f9a25280f6015 | size: 48
Plaintext 2  : Format: Andy message one | size: 24
Cipher text 2: y+wL5E3+WgWIT9Kb2jBCGlyfmiUoD2AV | size: 32
Cipher text 2 (hex): cbec0be44dfe5a05884fd29bda30421a5c9f9a25280f6015 | size: 48
XOR Cipher Text: 00000000000000011060d15000000000000000000000000
Position: 16 | CT len: 48 | Crib len: 8 | Crib: 'Andy'
000000000000000011060d150000000000000000000000
                416e6479
________________________________________
                5068696c Phil
```

REMEMBER AUTHENTICATION?

- Attacker was able to recover plaintext but doesn't have the key, so can't encrypt anything and send to Yolanda, right?

PWNED

```
Plaintext 1  : DO NOT RESTART KAFKA SERVER

Cipher text 1: ycxZx2PeQHeMcuKjqAkHIm64tgEoM0si7Uew | size: 36

Cipher text 1 (hex):
c9cc59c763de40778c72e2a3a80907226eb8b60128334b22ed47b0 | size: 54

Plaintext 2  : PLEASE

XOR Plaintext  : 1403650f1c11

XOR Cipher text: ddcf3cc87fcf

Replaced cipher text:
ddcf3cc87fcf40778c72e2a3a80907226eb8b60128334b22ed47b0

Malicious CT B64: 3c88yH/PQHeMcuKjqAkHIm64tgEoM0si7Uew

Recovered: PLEASE RESTART KAFKA SERVER
```

AEAD/HMAC TO THE RESCUE

- All decryption appliances should require HMAC verification ***BEFORE*** even attempting to decrypt cipher text
- Attacker does not know shared secret and would not be able to generate valid MAC
- Yolanda would reject message without even reading

Agenda

Introduction

Common Security Risks

Examples

NiFi Security Features

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Authentication & Authorization

- Remote Identity Stores
  - LDAP/AD
  - Kerberos
- Infrastructure Overhead for PKI
  - Client Certificates
  - Step-by-step guide on Hortonworks Community Connection
- Pluggable Authorizer
  - Integrates with Apache Ranger
  - Does not currently support local username/password storage, so nothing to “crack”
  - Swappable implementations

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

Extensive Encryption Options

- Key Derivation Functions
  - *NiFi Legacy*
  - *OpenSSL PKCS#5 v1.5 EVP_BytesToKey*
  - PBKDF2
  - Bcrypt
  - Scrypt
  - Raw Key
- Algorithms
  - *DES, 3DES*, Blowfish, etc.
  - AES w/ CBC, CTR, and GCM
- Other Protocols
  - OpenPGP key/password

* Certain "*unsafe*" configurations not available by default; deprecation and validation warnings if password length is too short in conjunction with weak algorithms

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Communication Endpoint Assurance

- Site-to-Site (S2S) Communication
  - TLS Mutual Authentication
  - High TLS protocols & cipher suites
- Cluster Communication
  - TLS Mutual Authentication
  - High TLS protocols & cipher suites
- Communication With External Services
  - TLS Mutual Authentication available

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Agenda

Introduction

Common Security Risks

Examples

NiFi Security Features

Roadmap

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

“As practitioners, we want non-experts to choose apps based on what we care about the most, security. But that’s not how people work.”

Jessy Irwin

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Protected Configuration Values

- Pluggable Architecture
  - Simple Encryption
  - HSM Integration
  - Datastore Integration (Vault, Keywhiz, etc.)
- Key/token Read in at Bootstrap
  - NiFiProperties can resolve sensitive values for internal services
  - Raw passwords/keys not stored on filesystem

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

TLS Protocol/Cipher Suite Controls

- Automatic Protocol/Cipher Suite Upgrades
  - Integration with Mozilla TLS Labs tools
  - Plug n' Play Simplicity
    - Admin enables feature in configuration
    - "Set it and forget it"
      - High/medium/low
      - Automatically sets and continuously analyzes supported cipher suites to ensure compatibility & risk minimization
- Isolation of SSLContextService(s)
  - UI/API connection could require client certificate & TLSv1.2
  - Processor connection to legacy system could allow TLSv1

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Encrypted Logs

- NiFi Provides Alternatives to Log Grepping
  - Provenance Data
  - Processor Stats
- Pluggable Log Interceptors
  - Accept messages, key/value pairs
  - Sensitive keys are registered in a lookup
  - Values for these keys are encrypted (using log-specific key) before output
- Log File Encryption
  - More difficult to debug, but more complete protection
  - Batching/latency trade-off

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Searchable Protected Logs

- NiFi Provides Alternatives to Log Grepping
  - Provenance Data
  - Processor Stats
- Pluggable Log Interceptors
  - Accept messages, key/value pairs
  - Sensitive keys are registered in a lookup
  - Values for these keys are hashed (using standard CHFs) before output
- Searching logs only requires executing hash function on query to generate “real” search term

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Encrypted Repositories

- Transparent Data Encryption (TDE)
  - Provenance Repository
  - FlowFile Repository
  - Content Repository
  - User Database
- Performance Tradeoff
  - More necessary for untrusted hardware/OS infrastructure
  - Current mitigations include OS-level access controls and Full Disk Encryption (FDE)

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Sensitive FlowFile Attributes

- Control Visibility of Specific Attributes
  - Data at rest (FlowFile & Provenance Repositories)
  - Data in motion (Communicated to external services)
  - *Data in play (Live data in application)*

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Provenance

- Cryptographic Signatures
  - Individual events (record signature)
  - Lineage graph (chained signatures)
- Visibility Controls
  - Mark provenance events as sensitive
  - Sensitive processors/processor types can mark all child events as sensitive
  - Individual/inherited access controls

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Agenda

Introduction

Common Security Risks

Examples

NiFi Security Features

Roadmap

Questions

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Learn, Share at Birds of a Feather

Streaming, DataFlow & Cybersecurity

Thursday June 30

6:30 pm, Ballroom C

© Hortonworks Inc. 2011 – 2016. All Rights Reserved

HORTONWORKS

Thank You

© Hortonworks Inc. 2011 – 2016. All Rights Reserved