Agenda

• Why we’re here
• Data protection basics
• What is “enterprise data protection”?
• Why data protection is difficult and scary
• The five Ws of data protection
• Key management: the “other” gotcha
• A realistic approach to enterprise data protection
Enterprise Data Protection In Sixty Minutes
Why We’re Here

• Data protection is on many folks’ minds these days
  – CxOs, CISOs are saying “Gotta protect stuff now!”

• Breaches are in the news
  – Heartland, Target, Home Depot...even RSA!

• Many sites have implemented several point solutions
  – Different platforms, different problems...not interoperable!

• DLP (data leakage prevention) is not foolproof
  – If it’s leaked but protected, you care a whole lot less!

• The h4xx0rs are out there...
  ...and they’re getting smarter and more creative

• Internal breaches are increasing
  – Gartner et al. agree: 70%++ breaches are internal
Encryption vs. Tokenization

• **Encryption** we’re all (sort of) familiar with:
  – using an algorithm (*cipher*)
  – plus a secret value (*key*)
  – to transform data (*plaintext*)
  – into another format (*ciphertext*)
  – so it is no longer readable without *decryption*

• **Tokenization** is another approach to data protection
  – Replaces values with randomly generated values
  – Values (typically) stored in database, database lookups required
Encryption and Tokenization

• The goal of both encryption and tokenization:
  – Make important data useless to anyone not authorized to read it!

• Note: Data protection tends to talk of data as “messages”
  – Stored data may not go anywhere, but same principles apply

THE MISSILE LAUNCH CODE IS XYZZY123plover

MV*U24AT2HaIKUewzqWPzvL
XaT9UGM!\zj(`iwPO...
Encryption Types: Symmetric

• Symmetric encryption means same key is used to encrypt and decrypt
  – Means both parties need access to the same keys
• Many varieties (algorithms):
  – DES, TDES, AES, Twofish, RC4, CAST5, IDEA, Blowfish...
• Can be strong and also fairly high-performance
  – “Strength” determined by key length in bits as well as algorithmic integrity
Stream vs. Block Stream Encryption

• Symmetric encryption comes in two flavors:
  – *Stream* ciphers transform the key as they progress, processing one chunk (bit, byte, whatever) at a time
  – *Block* ciphers use fixed keys every block (blocksize=keysize)

• Difference matters little in practice
  – Stream generally faster, but requires more key complexity
  – Many block ciphers have modes that effectively operate like stream ciphers
  – Most data protection products use block ciphers
Asymmetric *aka* Public Key Encryption

• Asymmetric encryption means what it sounds like:
  – Different keys needed to encrypt and decrypt
  – Each entity has two keys: *public* and *private*
  – Invented in 1970s (Diffie-Hellman, RSA, UK government)

• Makes key distribution much easier:
  – I can publish my public key safely
  – You encrypt using public key, I decrypt using my private key

• Downside is performance
  – Symmetric algorithms are typically *much* faster—public key often too expensive for application data protection
  – Requires significant data layout/application changes
Asymmetric Encryption Uses

• Some use cases are ideal for public key encryption
  – Hassle-free (public) key exchange makes some things easy
  – A key is a key, so either (private/public) usable for encryption or decryption, provided “other” used for opposite function

• Better yet, encrypt twice: my private, your public
  – You and I can email each other our public keys
  – I encrypt with my private, your public
  – You decrypt with your private, my public

• You now know the data was encrypted by me, I know only you could decrypt it
  – Provided neither of us has exposed our private keys!
Hybrids: Key “Wrapping”

• Because asymmetric encryption is expensive, hybrid solutions are attractive:
  – Sender generates random symmetric key
  – Encrypts actual data (‘payload’) using that symmetric key
  – Encrypts symmetric key using target’s public key
  – Sends encrypted symmetric key with data

• To decrypt:
  – Key decrypted using (expensive) asymmetric (private key)
  – Payload decrypted using cheaper symmetric algorithm
Cryptographic Hashes and Digests

- Related to encryption: cryptographic hashes aka digests
  - Functions that convert variable-length input to fixed-length output
  - Any change to original data changes the hash
  - Used in digital signatures, as checksums, etc.

- Good hashes (SHA-1/2/3, MD4/5) have these properties:
  - Easy to compute for given data
  - Infeasible to reconstruct data from hash
  - Infeasible to modify data without changing hash
  - Collisions (same hash from different data) very rare

- A good way to represent data without leakage risk
  - Frequently used for things like verifying downloads
Digital Signatures

- Digital signatures are also related to cryptography
  - Generated from the data using public/private-like key pairs
  - Result is a hash-like blob

- Signatures prove data authenticity and integrity
  - Authenticity: Data is from who it says it's from
  - Integrity: Data has not been tampered with (since signing)

- Implements important concept: non-repudiation
  - Means sender cannot (reasonably) say "I didn’t sign that"

- Frequently used for things like secure email
  - Avoids problems due to forged mail
Message Authentication Codes (MACs)

• A MAC (Message Authentication Code) is a keyed hash
  – Created using a hash function plus a secret key
  – Verify both data integrity and authenticity

• Different from digital signatures: same secret key used by creator/reader
  – Thus more like symmetric encryption, where digital signatures are more like public key encryption

• Generally faster to generate than digital signatures
  – MAC sent along with data
  – Receiver re-generates MAC against data, confirms match
  – Useful for verifying transactions
What is “Encryption Strength”? 

• **Encryption strength** refers to the likelihood that an attacker can “break” encrypted data
  
  – Typically tied to bit length of encryption key
  – Exponential: 128-bit key is $2^{64}$ times as strong as 64-bit
  – See “Understanding Cryptographic Key Strength” on youtube.com/user/VoltageOne for a good discussion/illustration

• The encryption community is collaborative
  
  – Research, algorithms are all published and peer-reviewed
  – Cryptographers look for weaknesses in their own and each others’ work
More About “Encryption Strength”

• Cryptographers “cheat” in favor of *attacker* when analyzing
  – Make assumptions like “attacker has multiple known examples of encrypted data and matching plaintext”
  – Also assume they’ll know plaintext when they find it, and that the encryption algorithm is known

• “Weaknesses” reported are often largely theoretical—only NSA could really exploit
  – Huge amounts of time, brute-force computing power required
More About “Encryption Strength”

• This “cheating” ensures encryption strength is real*
  – This approach increases security for all
  – By the time an algorithm is accepted as a standard and implemented in products, confidence is high
  – Even if a weakness is later discovered, it’s likely largely theoretical/impractical for most to exploit

• Makes it easy to spot the charlatans
  – Companies whose proprietary algorithms are not peer-reviewed
  – Also look for claims like “unbreakable encryption”, or focus on key length rather than standards-based cryptography

* Well, as real as the smartest minds in the business can make it!
Symmetric Encryption Examples

• DES: Data Encryption Standard
  – Selected as standard by US government in 1976
  – Block cipher, uses 56-bit keys
  – Considered insecure: as of 1999, “breakable” in < 24 hours

• TDES: Triple DES
  – What it sounds like: DES applied three times
  – Uses two or three different keys
  – Thus at least $2^{112}$-bit key strength (168-bit with three keys)
  – Considered secure, though relatively slow
More Encryption Algorithm Examples

• AES: Advanced Encryption Standard
  – Symmetric, adopted as US standard in 2001, reasonably fast
  – 128-, 192-, or 256-bit keys
  – Ubiquitous and proven: rarely any reason to use anything else

• Blowfish, Twofish, Serpent...
  – Symmetric, similar to AES in strength; mostly a bit slower
  – Algorithms are public domain (as is AES)

• Diffie–Hellman (ECDH), RSA, IBE
  – Asymmetric, much slower than common symmetric
  – RSA used in SSL; IBE most common email encryption technology
Format-Preserving Encryption

- **Format-Preserving Encryption** is another choice
  - Data encrypted with FPE has *same format* as input
  - Encrypted SSN still 9 digits; name has same number of characters; credit card number has same number of digits...

<table>
<thead>
<tr>
<th>Name</th>
<th>SS#</th>
<th>Credit Card #</th>
<th>Street Address</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>James Potter</td>
<td>385-12-1199</td>
<td>5421 9852 8235 6981</td>
<td>1279 Farland Avenue</td>
<td>77901</td>
</tr>
<tr>
<td>Ryan Johnson</td>
<td>857-64-4190</td>
<td>5587 0806 2212 0139</td>
<td>111 Grant Street</td>
<td>75090</td>
</tr>
<tr>
<td>Carrie Young</td>
<td>761-58-6733</td>
<td>5348 9261 0695 2829</td>
<td>4513 Cambridge Court</td>
<td>72801</td>
</tr>
<tr>
<td>Brent Warner</td>
<td>604-41-6687</td>
<td>4929 4358 7398 4379</td>
<td>1984 Middleville Road</td>
<td>91706</td>
</tr>
<tr>
<td>Anna Berman</td>
<td>416-03-4226</td>
<td>4556 2525 1285 1830</td>
<td>2893 Hamilton Drive</td>
<td>21842</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Name</th>
<th>SS#</th>
<th>Credit Card #</th>
<th>Street Address</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>James Cqvzgk</td>
<td>161-82-1292</td>
<td>5184 2292 5001 6981</td>
<td>289 Ykzbpoi Clpppn</td>
<td>77901</td>
</tr>
<tr>
<td>Ryan lounrfo</td>
<td>200-79-7127</td>
<td>5662 9566 7734 0139</td>
<td>406 Cmxto Osfalu</td>
<td>75090</td>
</tr>
<tr>
<td>Carrie Wntob</td>
<td>095-52-8683</td>
<td>5774 6343 6896 2829</td>
<td>1498 Zejojbbx Pqkag</td>
<td>72801</td>
</tr>
<tr>
<td>Brent Gzhqlv</td>
<td>178-17-8353</td>
<td>4974 7815 8270 4379</td>
<td>8261 Saicbmeayqw Yotv</td>
<td>91706</td>
</tr>
<tr>
<td>Anna Tbluhm</td>
<td>525-25-2125</td>
<td>4288 0276 0003 1830</td>
<td>8412 Wbbhalhs Ueyzg</td>
<td>21842</td>
</tr>
</tbody>
</table>
Format-Preserving Encryption

• Format-Preserving Encryption benefits:
  – Avoids database schema changes
  – Minimizes application changes
  – In fact, most applications can operate on the protected data:
    Fewer than 10% of applications need actual data

• Another HP Security Voltage technology
  – Invented by Voltage Security, based on work at Stanford
  – Mode of AES, NIST draft standard SP800-38G (or Google “nist ffx”)
  – Peer-reviewed, proven technology
FPE: Cross-Platform Capable

• ASCII/EBCDIC issues go away
  – Data converted to Unicode before encryption/decryption
  – Results in native host format (ASCII or EBCDIC)
  – Possible because character sets are deterministic (FPE!)

• Encrypt/decrypt *where the data is created/used*
  – Avoids plaintext data ever traversing the network
Tokenization for Data Protection

• Confusion abounds re tokenization vs. encryption
  – Some QSAs think tokenization is better—”no encryption key”
  – Cryptographers see the database index *itself* as a key

• Problems with traditional tokenization approaches
  – Sounds simple, easy to cobble together a solution
  – Problems: replication, backup, collisions
  – Result: early success, with dramatic failure after rampup
  – “Shell game”: moves the problem to one critical database
Choosing Tokenization

• When to choose tokenization over encryption
  – When QSA prefers it unilaterally
  – When regulations/policies require key rollover (PCI et al.)
  – As ever, “Choose the right tool for the right use case”

• Why not choose tokenization over encryption?
  – Transaction volume, network latency, database replication costs

• Enter HP Security Voltage Secure Stateless Tokenization technology!
  – Avoids all these—no database: random table instantiated once
FPE for Data Masking

• Application testing needs realistic datasets
  – Fake sample datasets typically too small, not varied enough

• Best bet: Use production data...**but:**
  – Test systems may not be as secure
  – Testing staff should not have full access to PII!

• Answer: Use FPE to mask (anonymize) test data
  – With FPE, protected production data is already usable for test
  – No extra steps required!
Implementing Data protection
What is “Enterprise data protection”?

• A scalable, manageable data protection plan
  – Standards-based, provably secure

• Applies across multiple data sources (databases etc.)
  – Not just point solutions for specific data sources

• Cross-platform
  – Everyone has multiple platforms nowadays

• Includes key management
Data Protection Is Difficult

• Lots of different technologies
  – Hardware-based, software-based, hardware-assisted
  – DES, TDES, AES, Blowfish, Twofish, CAST, PGP, GPG ...
• Companies have lots of data in lots of places
  – Much of it probably of unknown value/use
  – The sheer volume is daunting
• Difficult to imagine how to get started
  – Easier to stick your head in the sand and hope it goes away
Data Protection Is Scary

• Most of us don’t understand the technologies
  – Math classes were a looong time ago

• It changes constantly
  – We hear “DES has been broken, use AES”
  – What does that mean? Is DES useless? Is AES next to fall?

• Lots of snake-oil salesmen in data protection
  – www.meganet.com touts “unbreakable encryption”

• Easy to decide data protection is unapproachably complex
  – Like buying your first house, or doing your own taxes...

• Yes, if you get it wrong, you **will** lose data!
  – Another reason prompting avoidance behavior...
The Five Ws of Data Protection

• **Why** protect data?
• **What** should be protected?
• **Where** should it be protected?
• **When** should it be protected?
• **Who** should be able to protect/access (decrypt/detokenize) it?
• **How** will you protect it?
Why Protect Data?

• Every company has data to protect
  – NPPI, PII, or just PI
  – Customer information
  – Internal account information
  – Intellectual property
  – Financial data

• Every company moves data around
  – Backup tapes
  – Networks
  – Laptops
  – Flash drives
  – Data for test systems
Why Protect Data?

• Different media have different issues
  – Very few backup tapes get lost...but it does happen
  – Networks get compromised fairly regularly
  – Laptops are lost or stolen every day
  – Flash drives are disposable nowadays

• Different media types mean different levels of risk
  – Deliberate, targeted network breaches are obvious concern
  – Missing backups probably won’t be read
  – Missing laptops probably won’t be analyzed for PII
  – Found flash drives are probably given to the kids
Why Protect Data?

• Breaches happen! Identity Theft Resource Center says:
  – Not improving...and what about undetected/small ones?
  – Can you afford to bet your job/business?

• Data protection is **not** a luxury
  – Claimed cost per compromised card is $154–$215!!! *
  – Target breach: 40M++; Heartland : 130M; TJX: 94M cards
  – Do the math...

* Source: Ponemon Institute
  $154 = negligent inside
  $215 = malicious/criminal act
Why Protect Data?

• Data breach sources:
  - 73%: external
  - 18%: insiders
  - 39%: business partners
  - 30%: multiple parties

• But insider breaches far more expensive:
  - External attack costs averages $57,000
  - Insider attacks average $2,700,000!

Source: Verizon Business Data Breach Investigations Report
Why Protect Data?

• Commonalities:
  – 66%: victim unaware data was on system
  – 75%: not discovered by victim
  – 83%: not “highly difficult”
  – 85%: opportunistic
  – 87%: avoidable through “reasonable” controls

• Causes:
  – 62%: attributed to “significant error”
  – 59%: from hacking or intrusions
  – 31%: used malicious code
  – 22%: exploited vulnerability
  – 15%: physical attacks

“It could happen to anyone!” (and does)
Why Protect Data?

• The law is catching up with the reality
  – PCI DSS (Payment Card Industry Data Security Standard)
  – Red Flag Identity Theft Rules (FACTA)
  – GLBA (Gramm-Leach-Bliley Act)
  – SB1386 (California)
  – Directive 95/46/EC (EU)
  – HIPAA
  – etc.

• PCI DSS not only requires data protection, but also:
  – Restrict cardholder data access by business need-to-know
  – This is called separation of duties
What To Protect?

• Everything! (Well, maybe not...)
  – Performance, usability, cost are barriers
  – Partners likely use different data protection technology
  – Changing every application that uses the data is prohibitive

• No single answer
  – Laptops, flash drives: at least PII, probably all data
  – Backup tapes: all data
  – Whole-database encryption possible but not a good answer
What To Protect?

• Whole database encryption fails on several counts
  – Can impose unacceptable performance penalty
  – Prevents data compression, using more disk space etc.
  – Violates separation of duties requirements
  – Better to just protect the PII (whatever that is)!

• What about referential integrity and other data relationships?
  – Database 1 and database 2 both use SSN as key
  – If you protect them, protected SSNs better match!
  – Else must decrypt every access, and indexes useless
Typical Data Protection Approaches

- Whole disk/database/filesystem encryption
  - Encrypt all data in DB—slows all applications
  - No granular access control, no separation of duties
  - No security of data within applications

- Column data protection solutions
  - Protect data via DB API or stored procedure
  - Major DB type/version dependencies
  - No data masking support and poor separation of duties

- Traditional application-level data encryption
  - Protect data itself via complex API
  - Requires DB schema/application format changes
  - High implementation cost plus key management complexity

- Lookaside database (aka “Tokenization”)
  - CC# indexed, actual CC# in protected DB
  - Requires online lookup for every access
  - Requires major rearchitecting; scope issues
Where To Protect?

• Different question than “what”:
  – Data at rest and in motion

• Data at rest
  – “Brown, round, and spinning” (disks of all types)
  – On tape (backup or otherwise)

• Data in motion
  – Traversing the network
Where To Protect?

- Data in motion particularly troublesome
  - How do you know if it’s been sniffed as it went by?
- Data at rest *somewhat* easier
  - Intrusion detection systems fairly effective (if installed and configured, and if someone actually checks the logs)
  - ESMs very effective on z/OS (if administered correctly)
- Different issues, thus different criteria!
When To Protect?

• Ideally, data is protected as it’s captured
  – By the data entry application, or the card swipe machine

• In reality, it’s often done far downstream
  – The handheld the flight attendant just used—is it protecting?
  – Did last night’s restaurant protect your credit card number?
  – If the data goes over a wireless network, is it WEP? WPA?

• “Doing it right” is harder: more touchpoints
  – Easier (if less effective) to say “Just protect at the database”
  – Avoids interoperability issues (ASCII/EBCDIC, partners)
Who Can Protect/Decrypt?

• Usual question is: Who can decrypt?
  – Who should have the ability to decrypt PII?

• Should your staff have full access to all data?
  – Many unreported (or undetected) internal breaches occur

• What if someone leaves the company?
  – How do you ensure their access is ended?

• What if an encryption key is compromised?
  – Can you revoke it, so it’s no longer useful?

• PCI DSS et al. require these kinds of controls
  – This is a big deal—not trivial to implement
How Will You Protect Data?

• Hardware? Software?
  – Many options exist for both

• Is a given solution cross-platform?
  – If not, you **must** reprotect when data moves

• AES? TDES? Symmetric? Public/private key?
  – Many, **many** choices exist—too many!
How Will You Protect Data?

• Different issue: How do you get from here to there?
  – 100M++ data records—how to protect without outage?
  – “Customer database down next week while we protect”?!

• What about data format changes?
  – Protected data usually larger than original
  – Does not compress well (typically “not at all”)
  – Database schema, application fields expect current format
  – Can you change everything that touches the data?
    (Should you need to?)
Key Management

• “Encryption is easy, key management is hard”
  – Ultimately, encryption is just some function applied to data
  – To recover the original data, you need key management

• Three main key management functions:
  1. Give encryption keys to applications that must protect data
  2. Give decryption keys to users/applications that correctly authenticate according to some policy
  3. Allow administrators to specify that policy: who can get what keys, and how they authenticate
Key Management

• Key servers generate keys for each new request
  – Key server must back those up—an ongoing nightmare
  – What about keys generated between backups?
  – Maybe punch a card every time a key is generated...

• Alternative: Derive keys on-the-fly
  – No key database, no ongoing backups/synchronization

• What about distributed applications?
  – How do you distribute keys among isolated networks?
  – What about keys for partners who receive encrypted data?
  – “Allow open key server access” not a good answer
  – Suggest it, watch network security folks’ heads explode
Getting There From Here: A Realistic Approach
A Realistic Approach

• Investigate data protection, now or soon
  – Better now than after a breach

• Understand that choices have far-reaching effects
  – Data tends to live on for a very long time

• Expect to use multiple solutions
  – Backups, laptops, databases all have different requirements
  – “Right” answer differs
  – E.g., for backups, hardware-based encryption; for customer database, column-based data protection
1. Classify data by degree of sensitivity
   • This is harder than it sounds!
2. Analyze risks: Security costs
   • How secure can you afford to be?
3. Implement solution (remediation)
   • \textbf{Must} be a gradual process
4. Use compensating controls sparingly
   • By definition, they’re suboptimal
5. Goal: persistent data protection everywhere
   • Best achieves regulatory compliance
Key Steps

• **Key:** Involve stakeholders across the enterprise
  – “No database is an island”: multiple groups use the data
  – Partners, widespread applications need access too...

• **Key:** Find a “starter” application
  – Generating test data from production is a good beachhead
  – If you “get it wrong”, you haven’t lost anything “real”

• **Key:** Designate data by sensitivity:
  - **Red:** Regulated (legally required to be protected)
  - **Yellow:** Intellectual property or other internal (unregulated)
  - **Green:** Public
  – Each requires a different level of isolation/protection
Proof of Concept

• Protect a representative database
  – “Database” could be RDBMS, .csv, flat file...

• Update application(s) that access it
  – You know what all your applications do, right? 😊

• Validate performance, usability, integrity
  – Protection is not free: may see significant performance hit

• Demonstrate to other groups
  – Invite discussion, counter-suggestions

• Once (if!) project approved, request executive mandate
  – Otherwise, some groups may simply not participate
Finishing the Job

• Doing *all* databases/applications takes time
  – Expect glitches
  – Perhaps most difficult: understanding data relationships
  – Table A and Table B seem unrelated, but aren’t

• Lather, rinse, repeat...
  – Each database will have its own issues/surprises
Making Intelligent Choices
Approaches and Options
Choose a Hardware Solution?

• Many choices: encrypting disk arrays, tape drives...
  – Minimal performance impact

• Biggest issue: hardware only protects hardware!
  – Layers “above” all unprotected
  – Use case determines value

• Also: key management
  – Usually proprietary
  – Difficult to integrate
  – Separation of Duties?
Or a Software Solution?

• Data protection software products fall into two categories
  1. SaaS/SOA/SOAP (web services) remote server-based
  2. Native (data protection performed on local machine)

• Some are very narrow “point” solutions
  – E.g., platform-specific, or file encryption only

• Do you want to manage dozens of products? (Hint: “No”)
  – Enterprise solution is cross-platform, solves multiple problems

• Web services is good for low-volume/obscure platforms
  – Native solutions perform better, avoid network vagaries
Questions to Ask

• Security-related questions
  – Are algorithms strong, peer-reviewed?
  – Does solution support hardware security modules/assists?
  – Is key management part of the solution?

• Operational/deployment questions
  – Is implementation cost reasonable?
  – Is implementation under your control?
  – Is product multi-platform?
Summary
Conclusion

• Data protection is not a luxury, not optional today
• Many solutions exist
• Different data/media require different solutions

• A complex topic, but one that can be mastered!
Data Protection Resources

- InfoSecNews.org: email/RSS feed of security issues
  infosecnews.org/mailman/listinfo/isn
- HP Security Voltage security, cryptography, and usability blog
  www.voltage.com/blog
- Bruce Schneier’s CRYPTO-GRAM monthly newsletter
  schneier.com/crypto-gram.html
- RISKS Digest: moderated forum on technology risks
  catless.ncl.ac.uk/risks
- US Computer Emergency Response Team advisories
  us-cert.gov/cas/signup.html
- Track breaches: privacyrights.org and databreachtoday.com and
datalossdb.org and idtheftcenter.org
  (links checked 2015-07-22)
Questions/Discussion

Phil Smith III
703.476.4511
phsiii@hp.com