Security Remodeling:
10 Ways to Retrofit Your Transactional System

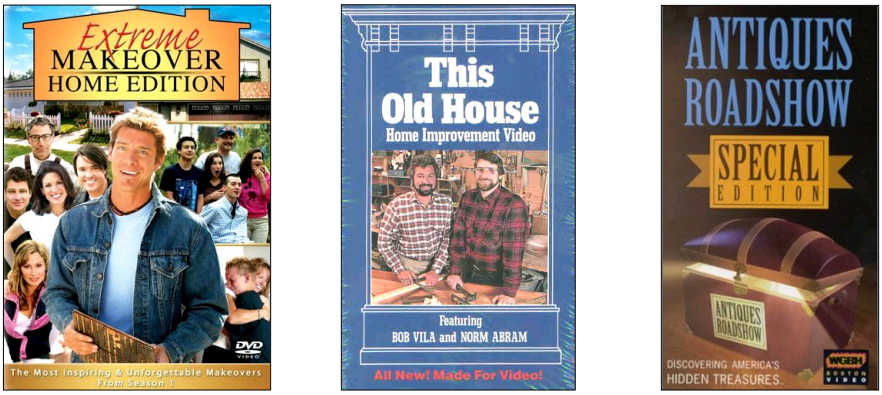
Benjamin Jun | Cryptography Research, Inc. | 27 October 2008 | DEV-105

CRYPTOGRAPHY RESEARCH

RSA CONFERENCE EUROPE 2008

System Life Cycle

RSA CONFERENCE EUROPE 2008



ABC, Inc.; This Old House Ventures, Inc.; PBS

Today's Talk

RSA CONFERENCE EUROPE 2008

- Our focus is client-server, transaction based systems
  - Living with legacy design decisions
- Consider incremental changes that can:
  - Reduce your vulnerability cross section
  - Improve the reviewability of your design
  - Contain the impact of a breech

> *"You go to war with the Army you have. They're not the Army you might want or wish to have at a later time"*
> – Former US Defense Secretary Donald Rumsfeld

Cryptography Research

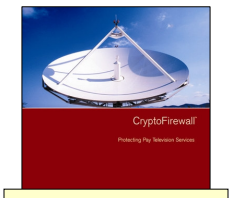
Who am I? What do I do?

RSA CONFERENCE EUROPE 2008

- Cryptography Research, Inc.
  - Solve complex fraud, piracy problems
  - R&D emphasis on applied security issues
- Design and Evaluation Services
- License Security Technologies

Cryptography Research

DPA:
Tamper Resistance

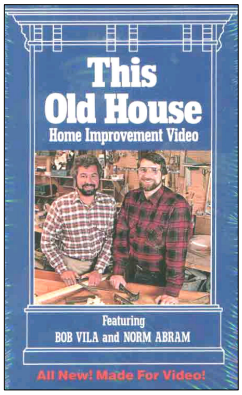


CryptoFirewall:
Pay-TV Security

Cryptography Research

Evaluating Retrofit Suggestions

- Check your foundation
  - Are your security concerns well-articulated?
  - Can you map your concerns to implementation? (Threat & risk assessment / SPF)
- Evaluating retrofit suggestions
  - Does this reduce my risk?
  - Can this focus future security efforts?
  - Do I need to better understand security needs/risks?
- Cautions
  - There may be faster paths to regulatory compliance
  - Can distract security team



1. Refresh Documentation (1/2)

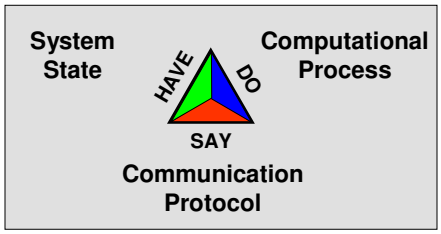
- Improves reviewability
  - Threats, design requirements, and architecture clearly stated
  - Enables resource investment on security critical areas
- Improves implementation robustness
  - Implementers do not need to infer required checks and functional restrictions
  - Handling of security conditions defined across all operating extremes

Security-sensitive specifications avoid:

- **Ambiguity**: pushes critical security decisions downstream
- **Complexity**: increases the # of potential surprises (bugs ≅ LOC^2)

1. Refresh Documentation (2/2)

- The best use of your time may be to provide security-relevant documentation of your legacy system
  - Perform code review in conjunction with your documentation
- Specs must precisely define these areas:
  - *Protocols*: Messages, sequences, bounds checking, preserved state across I/O
  - *Data structures*: Strict data structure definitions + access control requirements
  - *State machines*: Error, exit handling

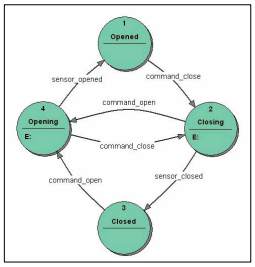


Cryptography Research, Inc.

Cryptography Research

2. Improve State Handling (1/2)

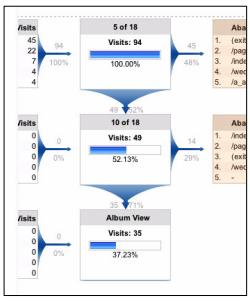
- Web applications carry a large amount of transaction state
  - Load balancing tools, application servers, don't help
- Security implications
  - Clients enter state with enhanced privileges
  - Transactions complete "improperly"
- Can we centralize/simplify state management?
  - Pay special attention to what is not specified well in your documentation
  - Can you improve error handling? Can you use a smaller number of well placed, robust error handlers?
  - Pay special attention to DB reads



Cryptography Research

2. Improve State Handling (2/2)

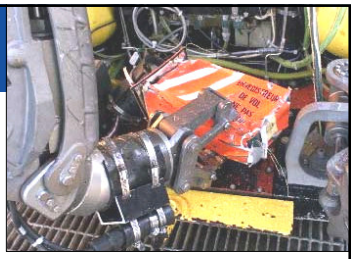
- Employ “transaction supervisor”
  - Single application API for all transactions
  - Secondary transaction control includes additional security checks and logging
- Harden existing transaction server
  - Add additional session state checks. Map to anticipated vulnerabilities/violations.
  - Example: “strict” build flag performs additional checks on transaction state



“Transaction Funnel” (Google analytics)

Cryptography Research

3. Build Audit Server (1/2)



Submersible Recovers Flight Data Recorder from Alaska 261 (US DOD)

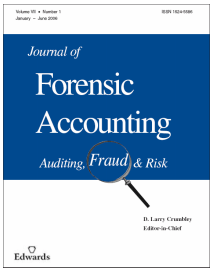
- Logging challenges
  - Can you get enough information?
  - Do you have too much information?
  - How do we parse what we have?
  - What about sensitive information in the log?
  - Boredom!
- Assumption: IDS/IPS already looking for “obvious” anomalies
  - …what of other application security requirements?

- Well placed audit servers improve "sloppy" environments
  - Financial systems, CCTV cameras, …
  - Goal: Build trusted log for problem detection / recovery
  - Can be more cost effective to unroll transactions after the fact
  - Protect against insider threats

Cryptography Research

3. Build Audit Server (2/2)

- Re-evaluate logs with problem recovery in mind
  - Take your auditor to lunch!
- Build log aggregation/collection tool
  - Select choke point and start with simple logging; integrate logs from different sources
  - Make "one-way" logs by encrypting with public key (GnuPG)



- Build log parsing & alert tools
  - Do not focus on alarms and anomalous behavior detection
  - Build easy to modify tools that can unroll critical sequences
    - Significant events in a single session, series of DB operations
    - Events during large spans of time

4. Protect Legacy DB Fields (1/2)

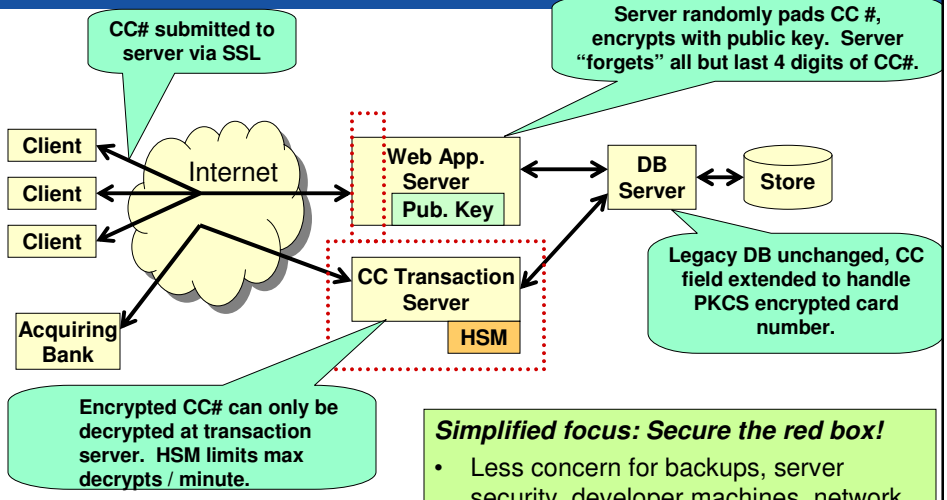
- Existing DB security tools
  - SQL firewall / live query linting tools
  - Disk volume encryption

```
BEGIN WORK;
UPDATE floor_inventory
SET quantity = quantity -
3 WHERE sku = '1927241';
COMMIT;
```

- But…
  - Narrower access rules hard to configure and verify outside of some PEN testing scenarios
  - DB contents still queryable, on development machines, etc.
  - Hard to modify legacy DB!

- Encrypt on a per-field basis
  - Encrypt sensitive fields on entry to legacy DB
  - Require decryption key to read data
  - Simplifies management

4. Protect Legacy DB Fields (2/2)



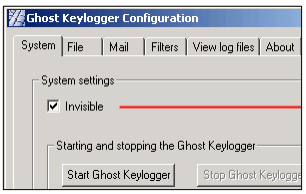
Simplified focus: Secure the red box!

- Less concern for backups, server security, developer machines, network nodes, inside jobs, etc.
- Max rate of DB compromise limited by velocity throttling

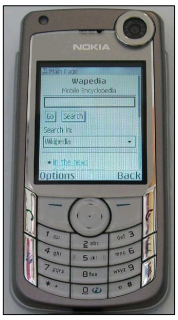
Cryptography Research

5. Client Environment Checking (1/2)

- Client environment can be a wild place!
  - Hostile code, keyloggers, data scrapers, tunnels, 'bots, TCP/IP redirectors, …
  - What's in your list of browser root keys?
- Patch management & NAC
  - Update client before granting access to network resources
  - General SW version checks
- The next battleground: Mobile platforms
  - Example: BBproxy attack with Blackberry tunnel to Intranet
  - Better configuration management needed



Ghost Keylogger, Sureshot Software



Cryptography Research

5. Client Environment Checking (2/2)

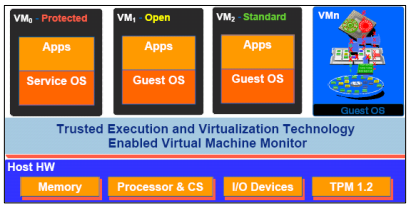
Downloadable Code

- Anti-virus profile updates
- Windows Update
- World of Warcraft "Warden" client

Improved Application Partitions

- Independent partitioned VMs
- API uses root of trust (TPM) to vouch for integrity of BIOS, OS, apps., …
- Challenging to get assurance on complex systems

McGraw & Hoglund
ISBN 978-0-321-46072-1



"Intel Trusted Execution Technology Overview"
2003

6. End-to-end Security

- Encapsulate private data before transmittal
  - Yahoo! uses javascript md5 and random challenge to protect password, prevent replay
  - Helpful when client cannot initiate SSL connection
  - Keeps passwords off edge servers, load balancing equipment, developer logs, …

Yahoo! login

- Encapsulate private data at client
  - TPM can seal/unseal data
  - Improve privacy and integrity of client-side data

```
TSS_RESULT Tspi_Data_Seal
(
   TSS_HENCDATA   hEncData,       // in
   TSS_HKEY       hEncKey,        // in
   UINT32         ulDataLength,   // in
   BYTE*          rgbDataToSeal,  // in
   TSS_HPCRS      hPcrComposite   // in
);
```

TCG Software Stack (TSS) Specification Version 1.10 (2003), p174

7. Obfuscation (1/2)

Password hashing code at http://login.yahoo.com

```
function hash(form,login_url) {
...
    if (valid_js()) {
      var passwd = form.passwd.value;
      var hash1 = MD5(form.passwd.value);
      var challenge = form[".challenge"].value;
      var hash2 = MD5(form.passwd.value) + challenge;
      var hash;
      if(form.passwd.value){
        hash=MD5(hash2);
      } else {
        hash="";
      }
      var js = 0;
```

Sep 2004

Feb 2006

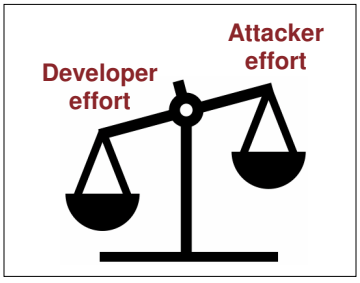
(https://a248.e.akamai.net/sec.yimg.com/lib/bc/bc_1.7.3.js)

```
function yzq4(r){var w=window;var d=w.yzq1;if(d==null)return;
if(typeof(d)==yzq5){var u="";if(d.s!=null)u+=d.s;
if(d.p!=null)u+=d.p;if(u.length>yzq6){w.yzq1=null;return;}d.s=d.p=nul
l;var z="";var s=0;var o=Math.random();var b;for(b in d){if(d[b]!=
null){if(u.length+z.length+d[b].length<=yzq6)z+=d[b];else
{if(u.length+d[b].length>yzq6){}else {s++;yzq2(u+z+"&Q="+s+"&O="+o);
z=d[b];}}}}if(s)s++;yzq2(u+z+"&Q="+s+"&O="+o);w.yzq1=null;}}function
yzq7(e){yzq4('l');}function yzq8(e){yzq4('u');}function yzq9(yzqa,
yzqb, yzqc){if (yzqc){var o=yzqc.toString();var m=yzqa;var
a=o.match(new RegExp("\\(([^\\)]*)\\)"));a=(a[1].length
>0?a[1]:"e");m=m.replace(new RegExp("\\([^\\)]*\\)","g") ,
"("+a+")");if(o.indexOf(m)<0){var b=o.indexOf("{");if
(b>0)o=o.substring(b,o.length);else return yzqc;o=o.replace(new
RegExp("([^a-zA-Z0-9$_])this([^a-zA-Z0-9$_])","g"),"$1yzq_this$2");
var s=m+";"+"var rv = f( "+a+",this);";var n="{"+"var a0 =
'"+a+"';"+"var ofb = '"+escape(o)+"' ;"+"var f = new Function( a0,
'yzq_this', unescape(ofb));"+s+"return rv;"+"}";return new
Function(a, n);}else return yzqc;}return yzqb;}
```

* not functionally
* identical

7. Obfuscation (2/2)

- General obfuscation goals
  - Limit understanding of code
  - Obscure data structures, instruction flow
- What obfuscation can do
  - Provide resiliency against automated attacks & 'bots
  - But tends to yield "brittle" security

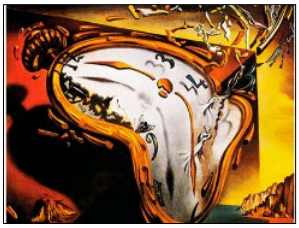


Go the other way!

- Apply with caution!
  - Can waste enormous amounts of time
  - Useless without facility for debugging & code review
  - If required, invest in tools for obfuscation and analysis

8. Prioritize

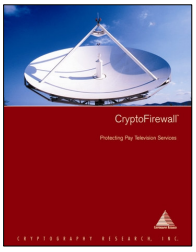
- Some things get more difficult with time…
  - Protocols + key management have incredible inertia
  - Corner case handling
  - Focus on what's hard to change later!
- Some reactive security elements are cost (and risk) effective to defer..
  - Expand risk management logic
  - Make infrastructure investments that enable reactive security



Explosion, Salvidor Dali

9. Retrofitting May Be the Wrong Idea!

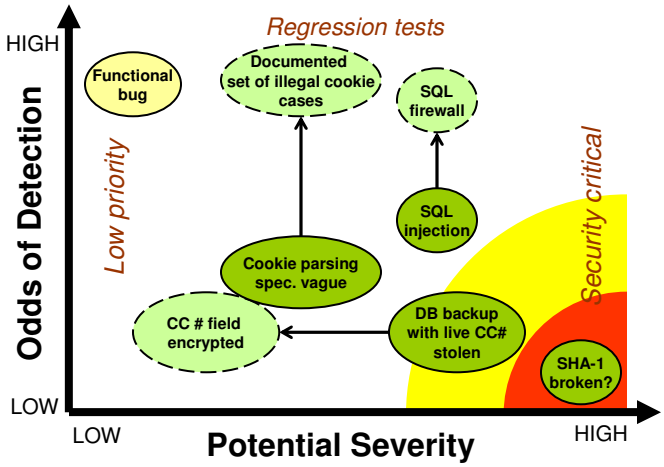
- "Extreme Makeover" may be required if…
  - System facing dedicated attackers
  - Legacy system a mess
  - You have resources to start over



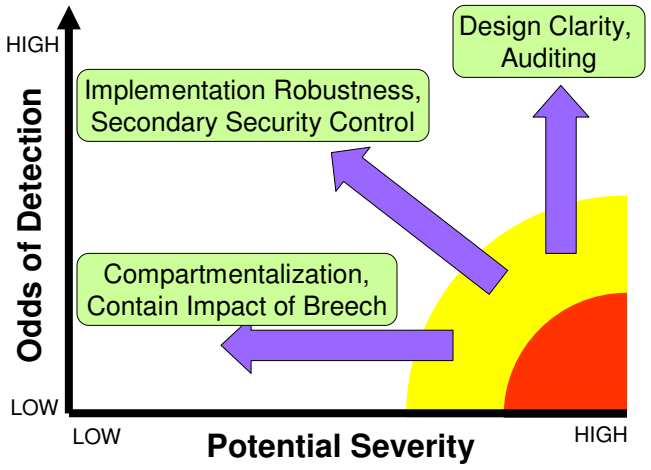
"CryptoFirewall"
Cryptography Research

- Alternative: Secondary security control
  - Security module to augment existing infrastructure
  - Design for smooth integration with legacy system
  - Goal: **>> 2X security**, **<< 2X operational cost**

10. Maximize Return on Effort (1/2)

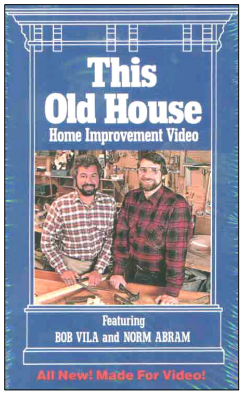


10. Maximize Return on Effort (2/2)



Conclusion

- Your system is only as strong as your weakest link!
- Use tools to…
  - Improve reviewability
  - Contain breech
  - Reduce vulnerability cross section
- Your charge: work efficiently!
  - Articulate your risks well
  - Map your implementation to your risks



This Old House Ventures, Inc.

RSA CONFERENCE EUROPE 2008

Contact Information

For more information, or to discuss how Cryptography Research can help with a security problem:

Benjamin Jun
ben@cryptography.com
415.397.0123
www.cryptography.com

CRYPTOGRAPHY RESEARCH

We're hiring!

If you are technically strong and want to work on challenging crypto and security problems, please send a resume!