Black-Box Assessment of Pseudorandom Algorithms

Derek Soeder
dsoeder@cylance.com

Christopher Abad
cabad@cylance.com

Gabriel Acevedo
gacevedo@cylance.com
Agenda

• About PRNGs
• PRNGs by Example
• Attack Methodology
• The Tool: Prangster
• Demonstration
Who we are

Cylance
the sound of security

Advanced Threat Protection • Incident Response • Special Projects • Research

Christopher Abad, Gabriel Acevedo, Derek Soeder
Cylance Labs Division, Cylance, Inc.
“The Science of Security”
About PRNGs
About PRNGs

• Pseudorandom number generator
  • Deterministic, *appears* unpredictable
  • Designed for simplicity and performance
  • Not secure

• Cryptographically secure random number generator (CSRNG)
  • Accumulates entropy
  • Designed for security
About PRNGs

Entropy source → Entropy → Application → Output

Application → Seed → PRNG

PRNG → State → Pseudorandom numbers
About PRNGs

Seed
- Derived from “entropy” or supplied by application
- Initial internal state is derived from it

State
- Internal state of PRNG
- Transformed for each pseudorandom number generated

Some states might not map to a seed
About PRNGs

• Consuming pseudorandom numbers
  • Modular (“take-from-bottom”)
  • Multiplicative (“take-from-top”)

black hat USA 2013
About PRNGs

• Modular (take-from-bottom)

- % Modulus
- % Limit
- % Output modulus
- / Discard divisor

black hat USA 2013
About PRNGs

- Multiplicative (take-from-top)

- Limit

  - % Modulus

  - / Output divisor

  - / Discard divisor
About PRNGs

**Ordinal value**

- Pseudorandom number from PRNG, processed by application
- Used to select a symbol for pseudorandom output

**Symbol**

- One unit of pseudorandom application output, usually a byte or character
- Mapping from numbers to symbols is the “alphabet”
- Size of alphabet = “limit”
About PRNGs

• Alphabet
  • Decided by application
  • Pseudorandom numbers to symbols via alphabet is a generalized but common pattern

• Example:
  • abcdefghijklmnopqrstuvwxyz
  • ABCDEFGHIJKLMNOPQRSTUVWXYZ
  • 0123456789!@#$%^&*()_-+
  • ‘a’ = 0, ‘Z’ = 51, ‘*’ = 69 or 71, ‘=’ = 77, etc.
PRNGs by Example

- Linear congruential generator (LCG)
- Array-based
- Miscellaneous
PRNGs by Example

• Linear congruential generator (LCG)
  • Next state: $s_i = (A \cdot s_{i-1} + C) \mod M$
  • Output: $x_i = (s_i / D) \mod R$

• $A = \text{multiplier}$   $C = \text{increment}$   $M = \text{modulus}$
  $D = \text{discard divisor}$
  $R = \text{output modulus (RAND\_MAX + 1)}$
PRNGs by Example

- **LCG examples:**

<table>
<thead>
<tr>
<th>PRNG</th>
<th>A</th>
<th>C</th>
<th>M</th>
<th>D</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSVCRT</td>
<td>214013</td>
<td>2531011</td>
<td>$2^{32}$</td>
<td>$2^{16}$</td>
<td>$2^{15}$</td>
</tr>
<tr>
<td>Java</td>
<td>0x5DEECE66D</td>
<td>11</td>
<td>$2^{48}$</td>
<td>$2^{16}$</td>
<td>$2^{32}$ $2^{31}$</td>
</tr>
<tr>
<td>BSD libc</td>
<td>16807</td>
<td>0</td>
<td>2147483647</td>
<td>1</td>
<td>2147483647</td>
</tr>
<tr>
<td>VBScript</td>
<td>0xFD43FD</td>
<td>0xC39EC3</td>
<td>$2^{24}$</td>
<td>1</td>
<td>$2^{24}$</td>
</tr>
<tr>
<td>MSSQL/PHP</td>
<td>40014</td>
<td>0</td>
<td>2147483563</td>
<td>1.000 000 012 324 788 164</td>
<td>2147483563</td>
</tr>
</tbody>
</table>
PRNGs by Example

- Array-based
  - Array of N integers modulo M
  - Two indices with a fixed separation
  - $a_k = (a_k \pm a_{k+\text{Sep}}) \mod M$  
    $a_{k+\text{Sep}} = (a_{k+\text{Sep}} \pm a_k) \mod M$

- At most $M^N$ possible states, > possible seeds
**PRNGs by Example**

- **Array-based examples:**

<table>
<thead>
<tr>
<th>PRNG</th>
<th>N</th>
<th>Sep</th>
<th>Index ±</th>
<th>M</th>
<th>D</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET</td>
<td>55</td>
<td>21</td>
<td>+1</td>
<td>2147483647</td>
<td>1</td>
<td>$a_k = (a_k - a_{k+Sep}) \mod M$</td>
</tr>
<tr>
<td>glibc (3)</td>
<td>31</td>
<td>3</td>
<td>+1</td>
<td>$2^{32}$</td>
<td>2</td>
<td>$a_{k+Sep} = (a_k + a_{k+Sep}) \mod M$</td>
</tr>
<tr>
<td>PureBasic</td>
<td>17</td>
<td>10</td>
<td>-1</td>
<td>$2^{32}$</td>
<td>1</td>
<td>$x = \text{rotr}(a_k, 13) + a_{k+Sep}$</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$a_k = \text{rotr}(b_k, 5) + b_{k+Sep}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$b_k = x$</td>
</tr>
</tbody>
</table>
PRNGs by Example

• Array-based exhibit recurrence relations
  • .NET: \( x_{i+55} = x_i - x_{i+21} + \text{error} \)
  • glibc (3): \( x_{i+31} = x_i + x_{i+28} + \text{error} \)

• Error
  • Caused by interactions of “hidden” state
  • Stymies prediction
  • Can actually be useful
PRNGs by Example

• Miscellaneous

• Google V8: “multiply-with-carry”
  • Next state: \[ s_i = 18273 \cdot (s_{i-1} \mod 2^{16}) + (s_{i-1}/2^{16}) \]
  \[ t_i = 36969 \cdot (t_{i-1} \mod 2^{16}) + (t_{i-1}/2^{16}) \]
  • Output: \[ x_i = (2^{14} \cdot (s_i \mod 2^{18}) + (t_i \mod 2^{18})) / 2^{32} \]

• Perl: uses platform’s libc rand() / (RAND_MAX + 1)
Attack Methodology
Attack Methodology

• Identify pseudorandom output
• Collect samples
  • Isolate truly pseudorandom portion
  • Determine complete alphabet
  • Detect biases if possible
Attack Methodology

• Recover seed from output
  • Guess PRNG if not known
  • Guess alphabet
    • Usually the most obvious arrangement
    • Use biases/error if available

• Exploit
  • Forward/reverse prediction
  • Recover entropy
The Tool: Prangster
The Tool: Prangster

• Why?

• Functions
  • \(\{\text{Output, alphabet}\} \rightarrow \text{Seed(s)}\)
  • \(\{\text{Seed, alphabet}\} \rightarrow \text{Next/previous output}\)
  • \(\{\text{Seed, } \pm n\} \rightarrow \text{Seed for } n^{\text{th}} \text{ next/previous state}\)
The Tool: Prangster

- Benchmarks

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BSD libc</td>
<td>26 seconds</td>
<td>1 second</td>
<td>1 second</td>
<td>1 second</td>
</tr>
<tr>
<td>Java</td>
<td>96 days</td>
<td>20 minutes</td>
<td>2 seconds</td>
<td>&lt; 1 second</td>
</tr>
<tr>
<td>MSVCRT</td>
<td>63 seconds</td>
<td>&lt; 1 second</td>
<td>&lt; 1 second</td>
<td>1 &lt; second</td>
</tr>
<tr>
<td>V8</td>
<td>19,856 years (Full state)</td>
<td>145 seconds (Half state)</td>
<td>&lt; 1 second</td>
<td>&lt; 1 second</td>
</tr>
</tbody>
</table>
Questions?
Thank you!

Derek Soeder
dsoeder@cylance.com

Christopher Abad
cabad@cylance.com

Gabriel Acevedo
gacevedo@cylance.com