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

Advanced Threat Protection · Incident Response · Special Projects · Research

Christopher Abad, Gabriel Acevedo, Derek Soeder

Cylance Labs Division, Cylance, Inc.

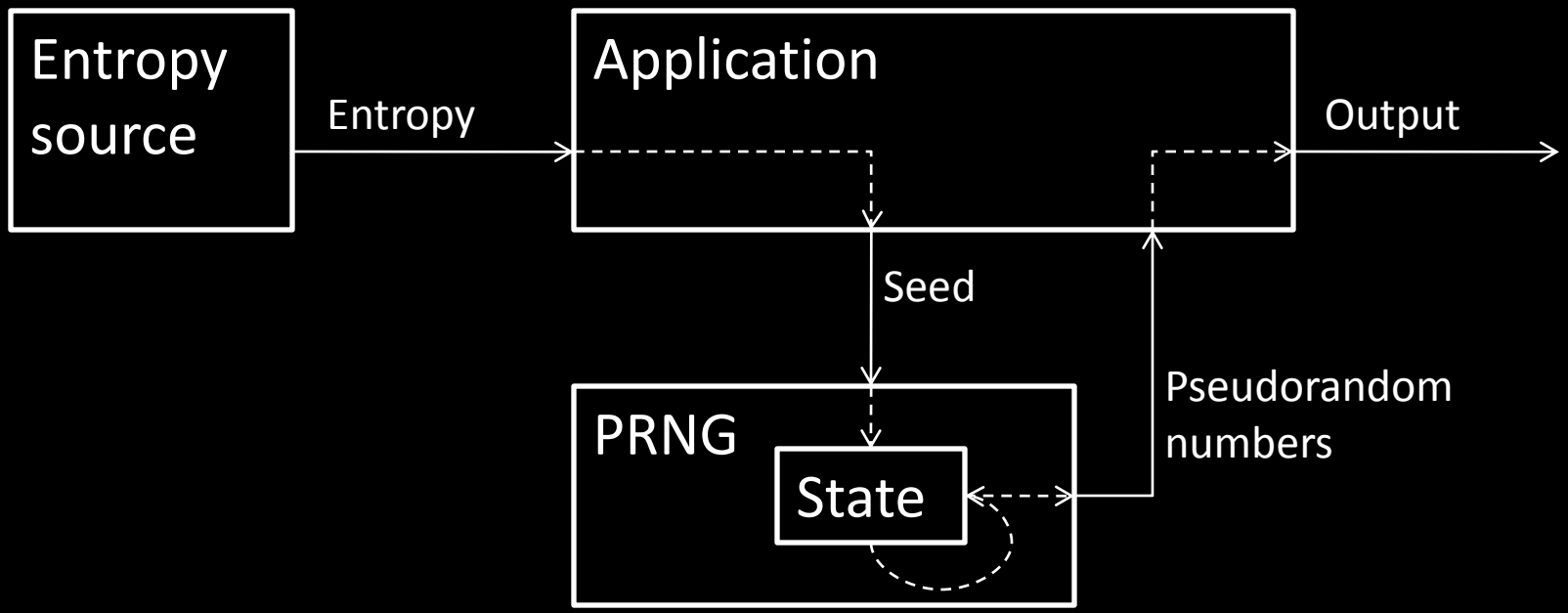
"The Science of Security"

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
Seed
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

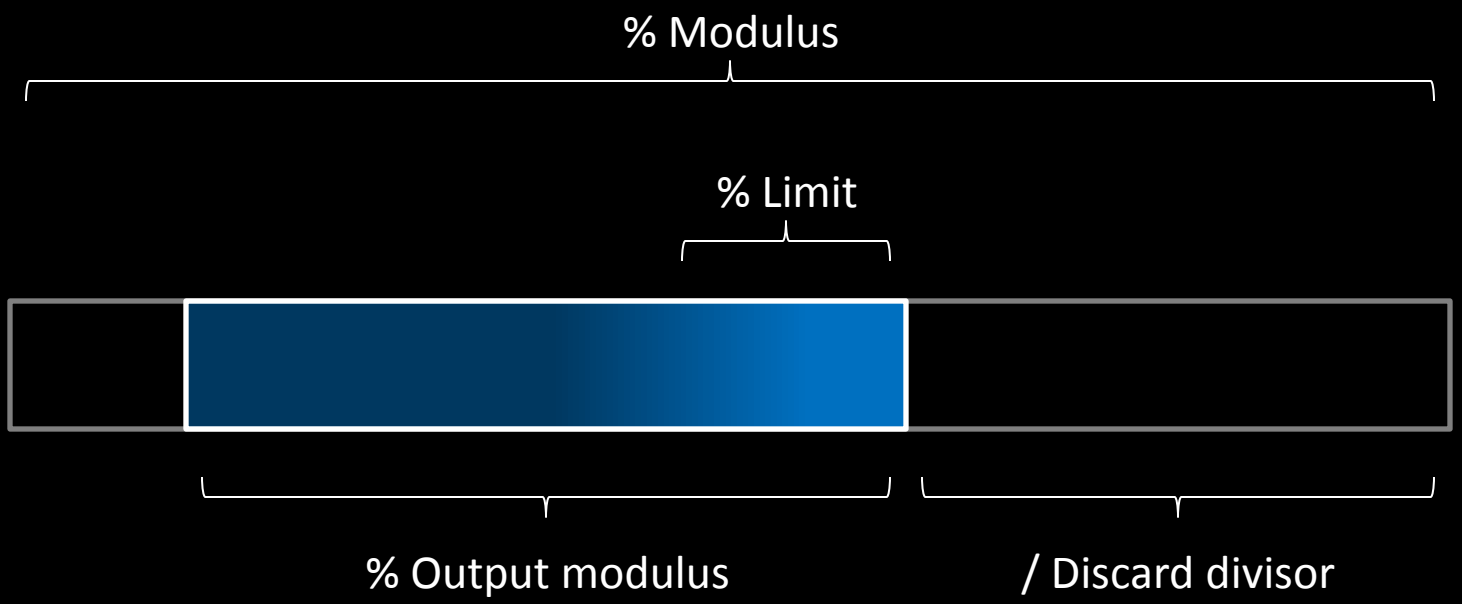
Some states might not map to a seed

About PRNGs

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

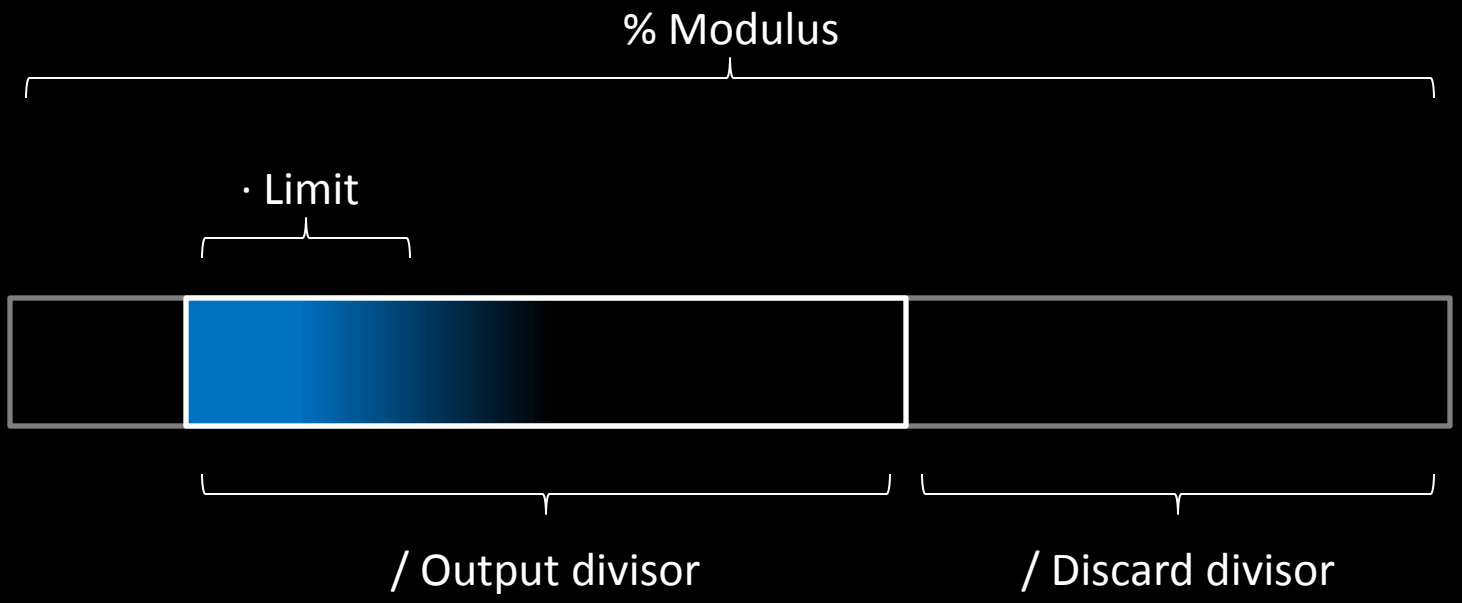
About PRNGs

- Modular (take-from-bottom)



About PRNGs

- Multiplicative (take-from-top)



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

PRNGs by Example

- Linear congruential generator (LCG)
- Array-based
- Miscellaneous

PRNGs by Example

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

- A = multiplier C = increment M = modulus
  D = discard divisor
  R = output modulus (RAND_MAX + 1)

PRNGs by Example

- LCG examples:

PRNG	A	C	M	D	R
MSVCRT	214013	2531011	2^{32}	2^{16}	2^{15}
Java	0x5DEECE66D	11	2^{48}	2^{16} 2^{17}	2^{32} 2^{31}
BSD libc	16807	0	2147483647	1	2147483647
VBScript	0xFD43FD	0xC39EC3	2^{24}	1	2^{24}
MSSQL/PHP	40014 40692	0 0	2147483563 2147483399	1.000 000 012 324 788 164	2147483563

PRNGs by Example

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

- At most M^N possible states, > possible seeds

PRNGs by Example

- Array-based examples:

PRNG	N	Sep	Index ±	M	D	Operation
.NET	55	21	+1	2147483647	1	$a_k = (a_k - a_{k+Sep}) \% M$
glibc (3)	31	3	+1	2^{32}	2	$a_{k+Sep} = (a_k + a_{k+Sep}) \% M$
PureBasic	17 17	10	-1	2^{32}	1	$x = rotr(a_k, 13) + a_{k+Sep}$ $a_k = rotr(b_k, 5) + b_{k+Sep}$ $b_k = x$

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: $s_i = 18273 \cdot (s_{i-1} \% 2^{16}) + (s_{i-1} / 2^{16})$
      $t_i = 36969 \cdot (t_{i-1} \% 2^{16}) + (t_{i-1} / 2^{16})$
    - Output: $x_i = (2^{14} \cdot (s_i \% 2^{18}) + (t_i \% 2^{18})) / 2^{32}$
  - 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
  - {Output, alphabet} → Seed(s)
  - {Seed, alphabet} → Next/previous output
  - {Seed, ±n} → Seed for n^{th} next/previous state

The Tool: Prangster

- Benchmarks

PRNG	Full naive brute-force		ABCDEFGH from A..Z	ABCDEFGHIJKLMNOP from A..Z	ABCDEFGHIJKLMNOP ABCDEFGHIJKLMNO P from A..Z
BSD libc	26 seconds		1 second	1 second	1 second
Java	96 days		20 minutes	2 seconds	< 1 second
MSVCRT	63 seconds		< 1 second	< 1 second	1 < second
V8	19,856 years (Full state)	145 seconds (Half state)	< 1 second	< 1 second	1 < second

Demonstration

Questions?

Thank you!

Derek Soeder
dsoeder@cylance.com

Christopher Abad
cabad@cylance.com

Gabriel Acevedo
gacevedo@cylance.com