Determining an Economic Value of High Assurance for Commodity Software Security

Virgil D. Gligor Adrian Perrig David Basin

CMU
Pittsburgh

ETH
Zurich

CDIS
KTH Stockholm
May 25, 2023

Outline

Why not high assurance? (~ 3 min)

Why revisit high assurance? (~ 10 min)

The need for selective high assurance (~ 10 min)

A Challenge (~ 2 min)

Illustrating a value of selective high assurance (~ 5 min -> …)

What is high-assurance commodity software?

Commodity software

general purpose software available for purchase by anyone on the open market;

e.g., *high volume sales & low cost*; <u>*not*</u> special-purpose software for government apps.

High assurance

mathematical methods (e.g., formal logics, number theory, information theory) used to prove *security properties* of a *computer program* or *set of programs*

e.g., beyond Common Criteria EAL7, US TCSEC A1, for the past decade

An early example

- Trusted (aka Secure) Xenix Kernel with *few* property specifications and proofs in Prolog

- *properties*: *penetration resistance* (e.g., [GG91, 92])
  - *entry* & *return point* protection
  - *parameter checking* on kernel entry
  - *time-of-check-to-time-of-use* atomicity
  - *inability to control execution of* kernel functions
  - *independence of kernel programs*
  - . . .

Why not high assurance?

[Gligor SPW 2010]

1) **High opportunity cost**

There will always be *rapid innovation* in commodity software and this will always lead to ***low-assurance*** commodity software systems

(why rapid innovation? *~0 cost of entry in software market, ~0 regulation, 0 liability*)

2) **Large commodity software systems;** aka., "*giants*" [Lampson 2004]

There will always be "***giants***" whose security properties that are *unknown* or *hard to prove*

[Lampson ACSAC 2000, CACM 2009]

3) **Defenders are rational:** *high assurance everywhere* (near perfection) is *impractical*

balance recovery cost(breach) **x probability**(breach) versus **Cost**(**prevention**);
i.e., versus **Cost**(**high assurance**)

balance tilted *away* from high assurance

Why revisit high assurance?

1. Two trends:

recovery cost(breach) **has *increased* substantially**

- cost of recovery from cybercrime ~ **1% of Global GDP**
- *average* cost/breach = **$4.24 M**; if *zero-trust* architectures, **$3.28M**; if AI/ML, **$2.9M**
- **10% Y/Y** recovery-cost *increase* (as of) 2021

formal verification cost has *decreased* dramatically

2013 - $362/SLoC - **seL4**
2014 - $128/SLoC - **Ironclad Apps**
2020 - $225/SLoC - **I/O separation kernel** (at $300K/PY)
2020 - **$40/SLoC** - **EverCrypt** libraries ~ **1/9 of 2013 cost**

=> formal verification of 50K – 70 SLoC "***wimps***" is practical

---

balance is beginning to tilt *towards* high assurance

Why revisit high assurance?

2) Selective high assurance: select & *isolate* small security-critical "*wimps*" of a "giant"
& *formally prove* their *security properties*

Notation

b = no. of selected attacks to be countered by formal verification of "giant's" *source code*

C_b(verification) = ***one-time*** **cost** of verifying *at most* **b** "wimps"

C_b(recovery) = *minimum* ***recurrent annual cost*** of recovery from **b** breaches of a "giant"

C_b(recovery)·***m***·***n*** = ***market cost*** for recovery by ***m*** defenders and ***n*** > 1 years

How can a producer recoup C_b(verification)?

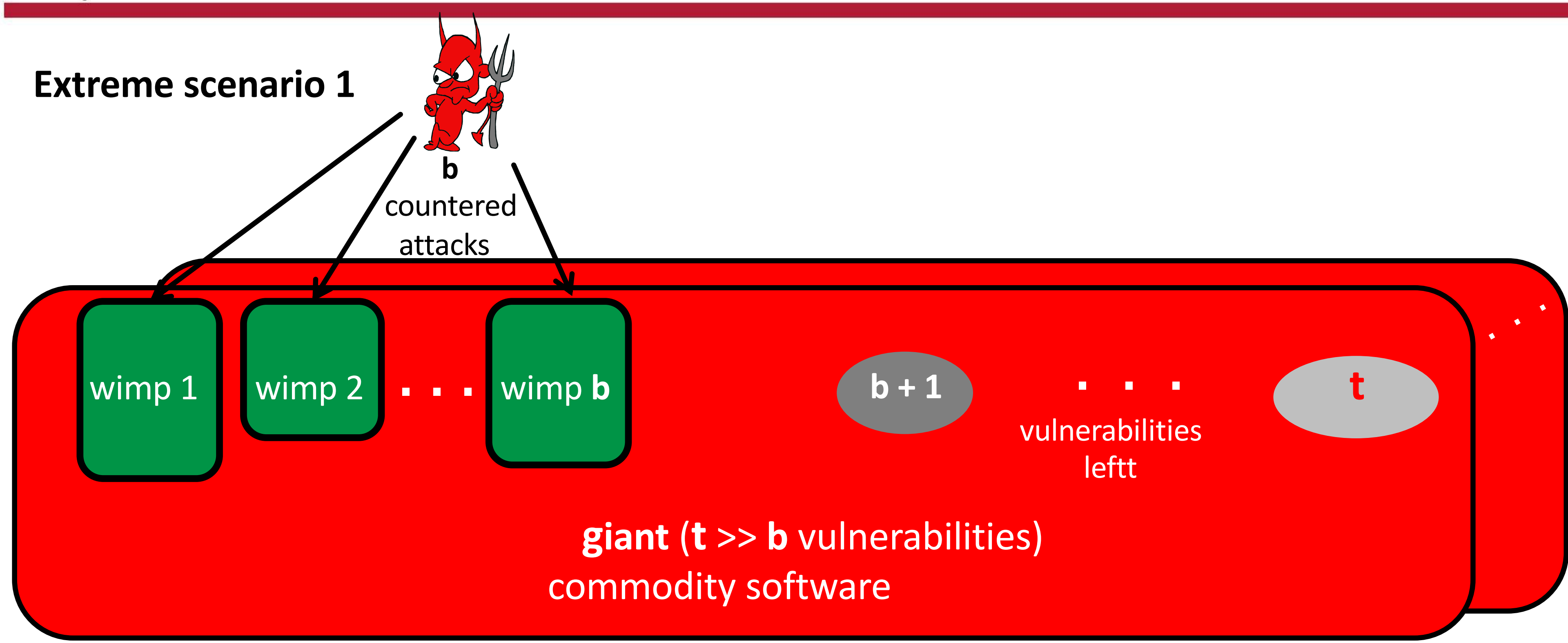
Producers & Defenders "know" the balance: C_b(verification) ≤ C_b(recovery)**x**1

Example: **b** = 3 security breaches in 42 attacks/year of US companies [VB2022]

C_3(recovery) = 3·$**2.9M** = **$8.7M**/year (already **$8.9M** in 2012)

Producer's selection:
b = 3 "wimps," size ≤ 72.5K SLoC => C_3(verification) **≤** 3·72.5K SLoC·**$40**/SLoC = **$8.7M**

Extreme scenario 1



$(m \geq 1, n)$ **-> producer** recoups cost C_b(verification) ≤ C_b(recovery)**x**1

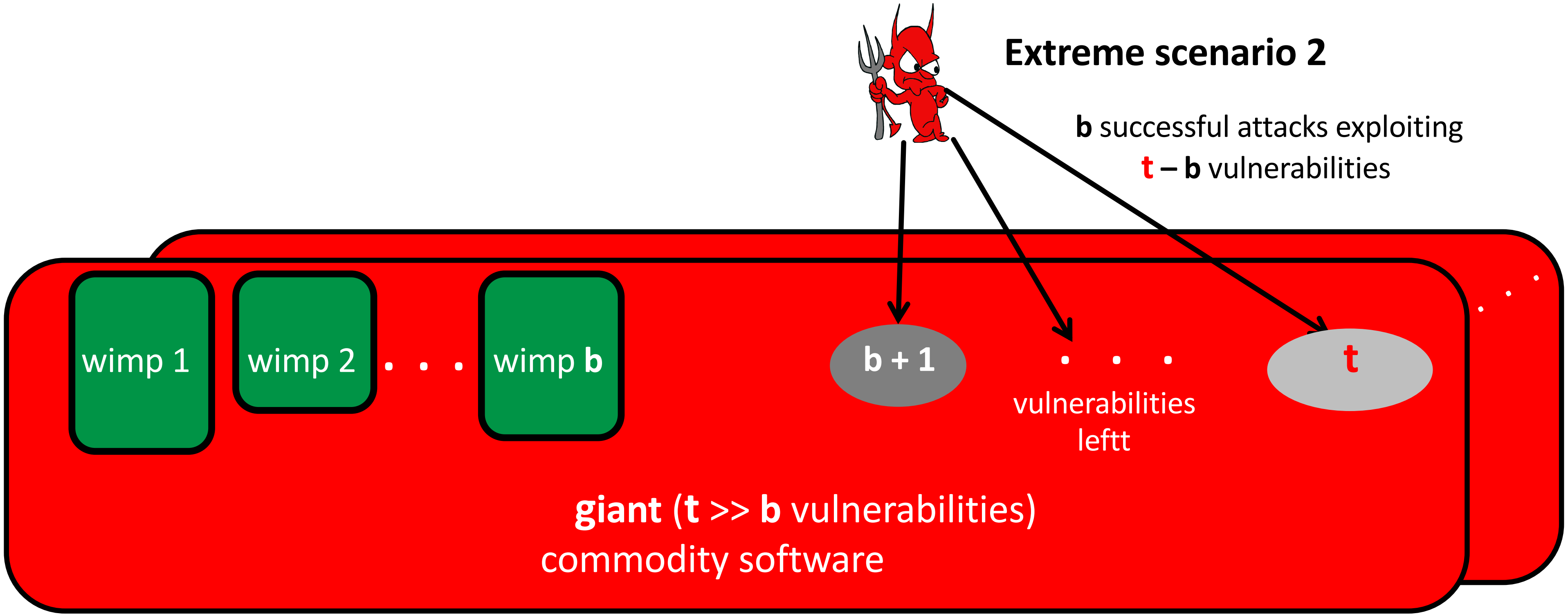
~~another choice~~: **defender$_{first}$ -> producer: 0**

defender$_{first}$'s cost: C_b(recovery) in year **1**,..., C_b(recovery) in year **n**

risk: low assurance fixes & limited deterrence

producer ~~**controlled**~~ ***m*** >> 1 **=> producer** increases price by C_b(verification)/***m*** = ***negl***.(***m***)]

Extreme scenario 2



($m \geq 1, n$) **->** **producer** recoups cost C_b(verification) **≤** C_b(recovery)**x**1

find *smallest* ϵ: $(1-\epsilon)\cdot C_b(\text{recovery})\cdot \boldsymbol{m}\cdot\boldsymbol{n} + C_b(\text{verification}) \leq C_b(\text{recovery})\cdot\boldsymbol{m}\cdot\boldsymbol{n}$

=> smallest $\epsilon > \mathbf{1}/(\boldsymbol{m}\cdot\boldsymbol{n})$. Then, setting $\epsilon = \mathbf{b}/(\mathbf{t}\text{-}\mathbf{b})$ => $\boxed{\mathbf{t} < \mathbf{b}(\boldsymbol{m}\cdot\boldsymbol{n} + \mathbf{1})}$

producer guesses $\boldsymbol{m} > \mathbf{m_0}$ and increases price by C_b(verification)/$\boldsymbol{m}$

producer recoups C_b(verification) in *all attacks* between **Extreme scenarios 1** & **2**

Estimation of (**t**, $\mathbf{m_0}$, $\mathbf{n_0}$): producer needs *a <u>very small</u>* market ($\mathbf{m_0}$, $\mathbf{n_0}$): *recoup cost*

Problem: surveys *cannot* reveal **t**, max no. of vulnerabilities/"giant"
s, no. of "giants"/defender
r, no. of responders/defender (typically **r** = 1)
m, no. of different defenders using same "giant" & ***n*** > 1

Example 1: **V** = total no. of *un-remediated vulnerabilities* of *all* defenders, *all* **R** responders

$V = t \cdot s \cdot R/r \Rightarrow t = r \cdot V / s \cdot R \;\&\; r/s \leq 1 \Rightarrow \boxed{t \leq \lceil V/R \rceil}$

e.g., **b** = 3, **R** = 47%×634, **V** = (1 – 46%) ×1.1M

=> **t** = 1994, $\mathbf{n_0} = 2$, $\mathbf{m_0} = 332$ / $\mathbf{n_0} = 7$, $\mathbf{m_0} = 95$ < 5.3% all companies on US stock exchanges

Example 2. **v** = lower bound on no. of *un-remediated vulnerabilities/defender*
s = average no. of applications/defender (another survey)

b = 3, $t \geq v/s$, **s** = 200, **v** ≥ 50,000

=> **t** ≥ 250, $\mathbf{n_0} = 2$, $\mathbf{m_0} = 42$ / $\mathbf{n_0} = 7$, $\mathbf{m_0} = 12$

The need for selective high assurance

Start with

expected **Cost**(defender) **=** **recovery cost**(breach) **x** **probability**(breach)

Show that *minimization of*

- **probability**(breach) &
- **recovery cost**(breach)

=> ***selective high assurance***

Minimize probability

- separate ***D****eterrence* from ***A****ssurance* [Lampson2009, GW2011]

  => separate probability(breach) into two ***non-independent*** components

  - ***A****ssurance* **->** **probability_A**(breach)

  - ***Deterrence*** **->** **probability_D**(breach)

- minimize

  - **probability** = ***min***(**probability_A, probability_D**) ≤ upper limit

Separation

probability$_A$(breach)

1 → 0

breach-prevention
assurance

low → **high**

- *security functions*
  verification, monitoring,
  recommendations, etc.

- *operational assurance*
  principles;
  e.g., least privilege,
  separation of duty,…

- *correctness assurance*
  models, design,
  implementation;
  proofs,
  security testing, etc.

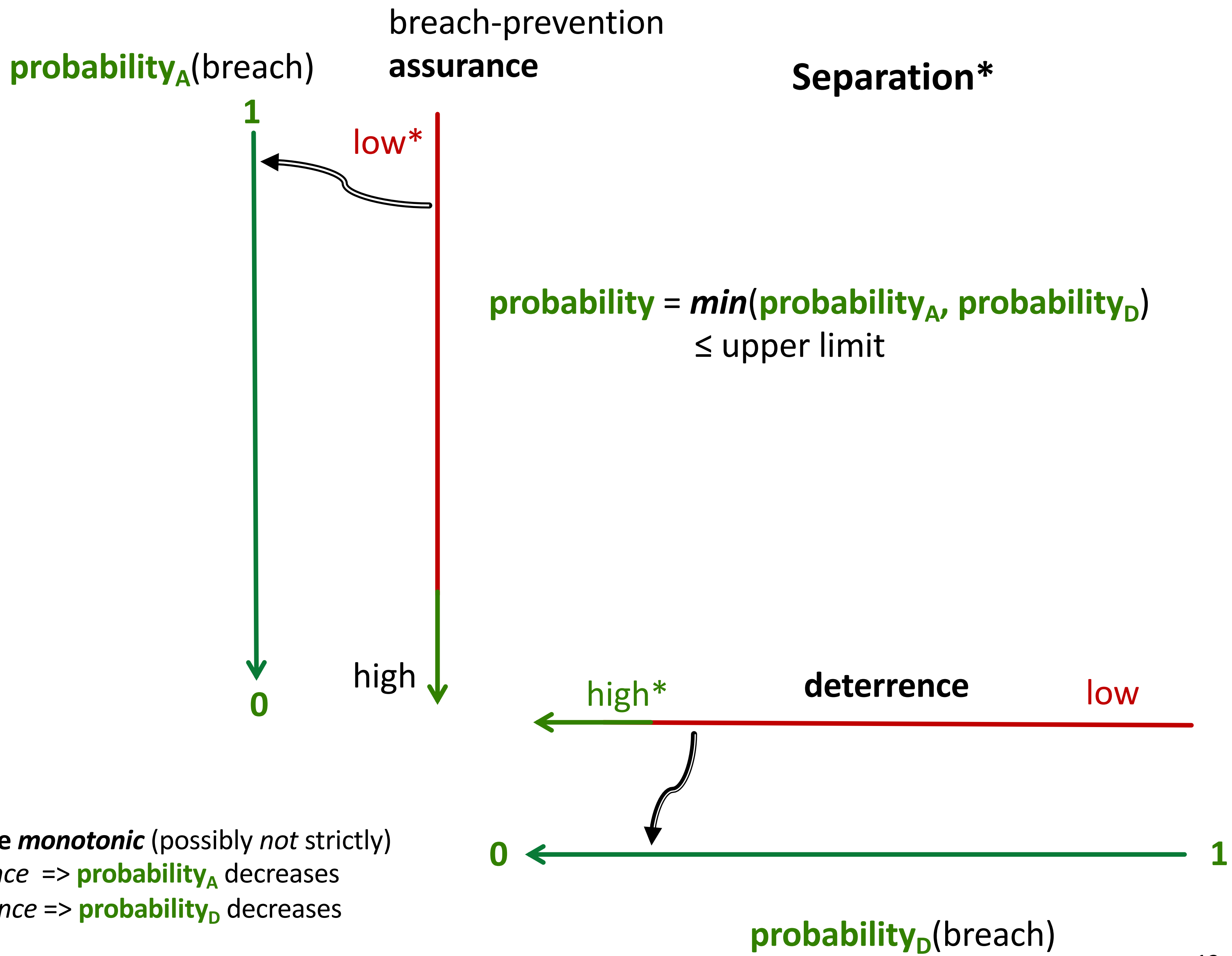
high ← **deterrence** ← **low**

- increased attack cost, detection & response
  audit & punishment, name & shame, etc . . .

0 ← 1

probability$_D$(breach)

Separation*



mappings "~>" are *monotonic* (possibly *not* strictly)
- increase ***Assurance*** => **probability$_A$** decreases
- increase ***Deterrence*** => **probability$_D$** decreases

"In fact, real-world security depends mainly on deterrence, and hence on the possibility of punishment." [Lampson, CACM 2009]

Defenders are rational [Lampson ACSAC 2000, CACM 2009]

- low assurance: **$probability_A$(breach) -> 1**

- **high** deterrence: **$probability_D$(breach) -> 0**

$$probability_D = \mathit{min}(probability_A, probability_D)$$

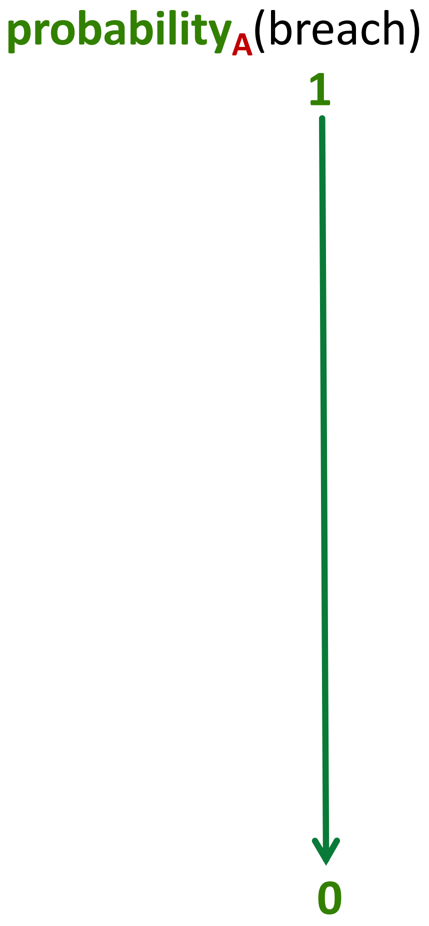
However, defenders must "assume breach" [NSA 2001, VB2022]

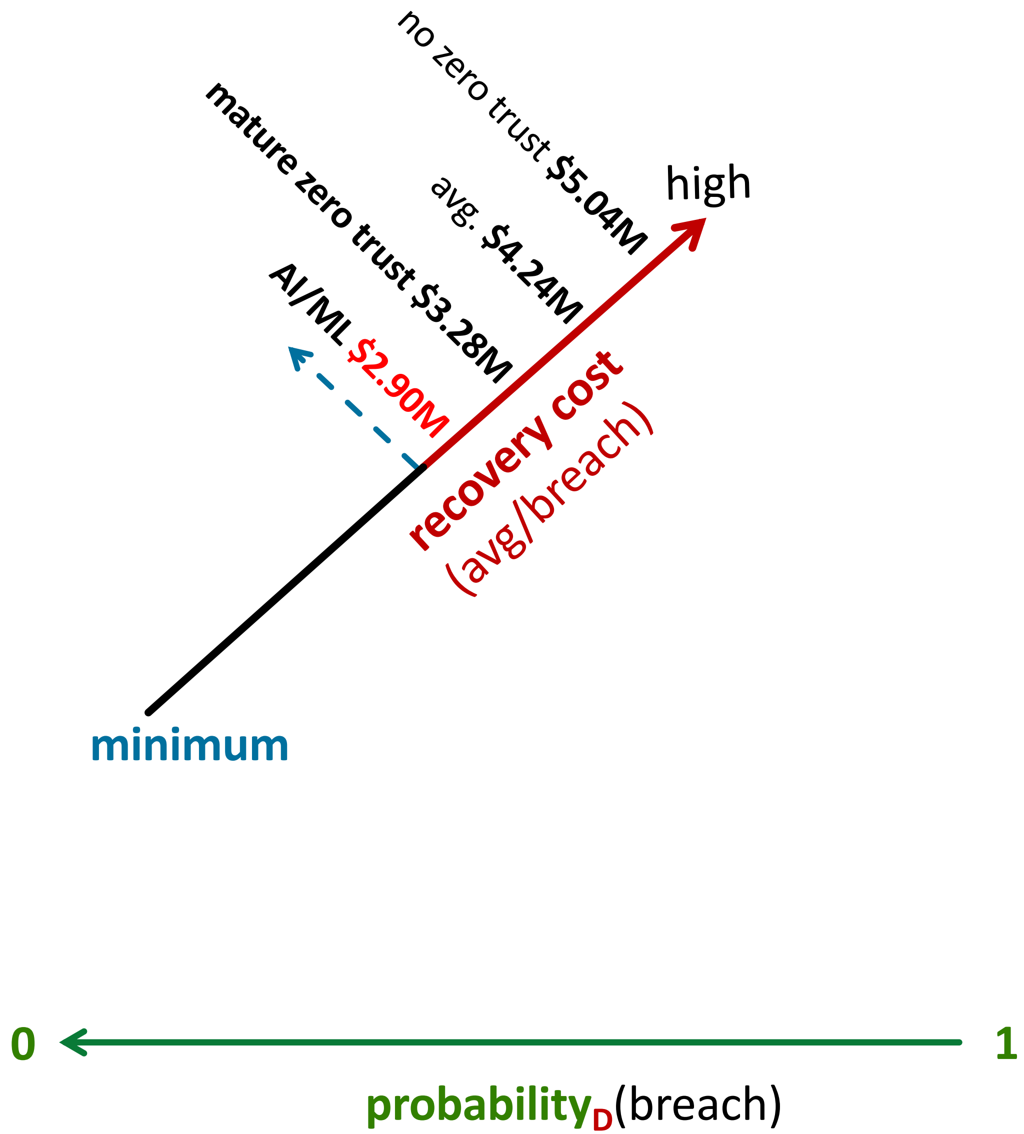
e.g., there is *no deterrence* of *state-sponsored attackers*

=> **$probability_D$(breach) -> 1**

=> **Cost**(defender) = **recovery cost**(breach) x **1**

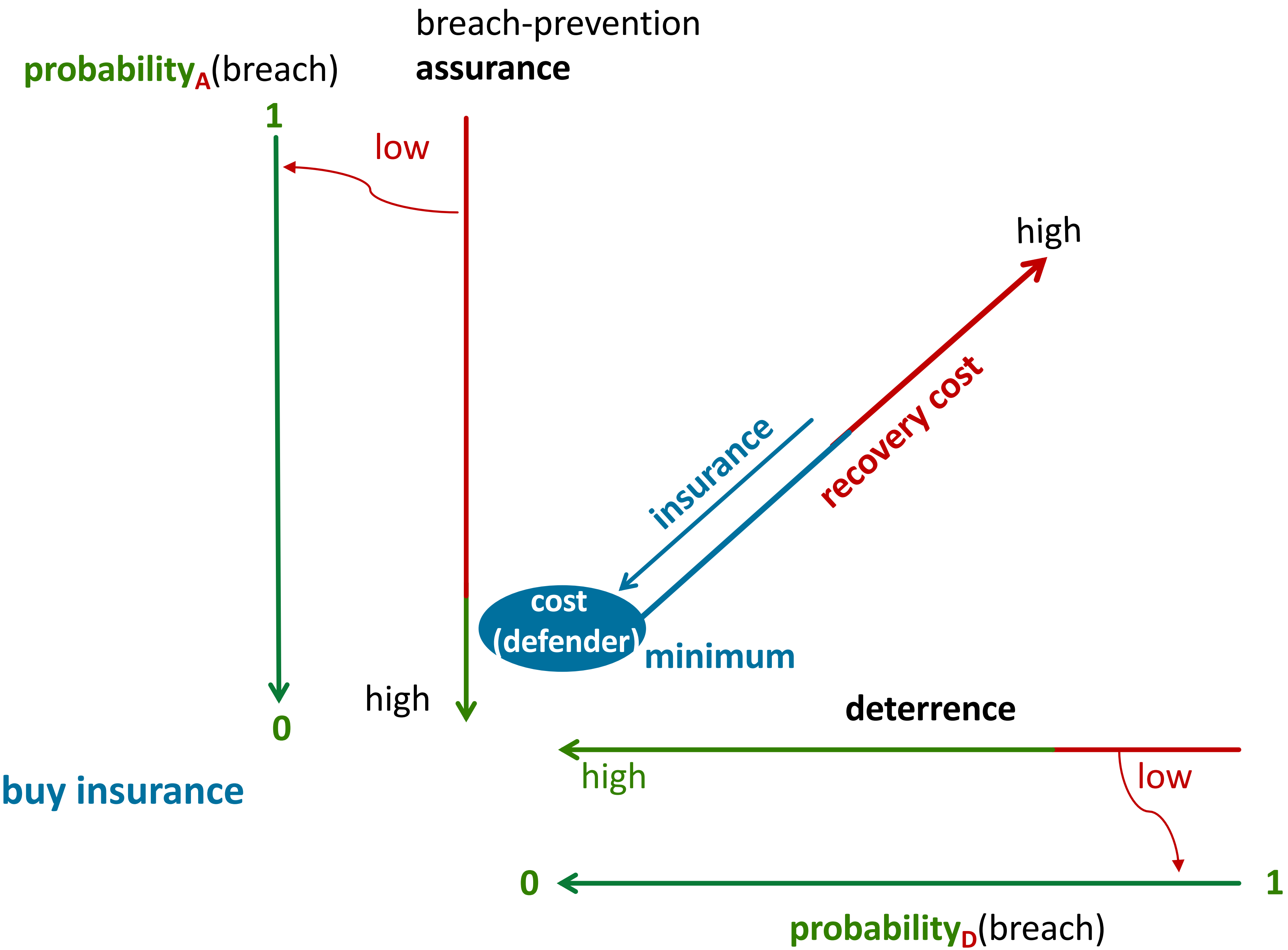
Minimize **recovery cost**



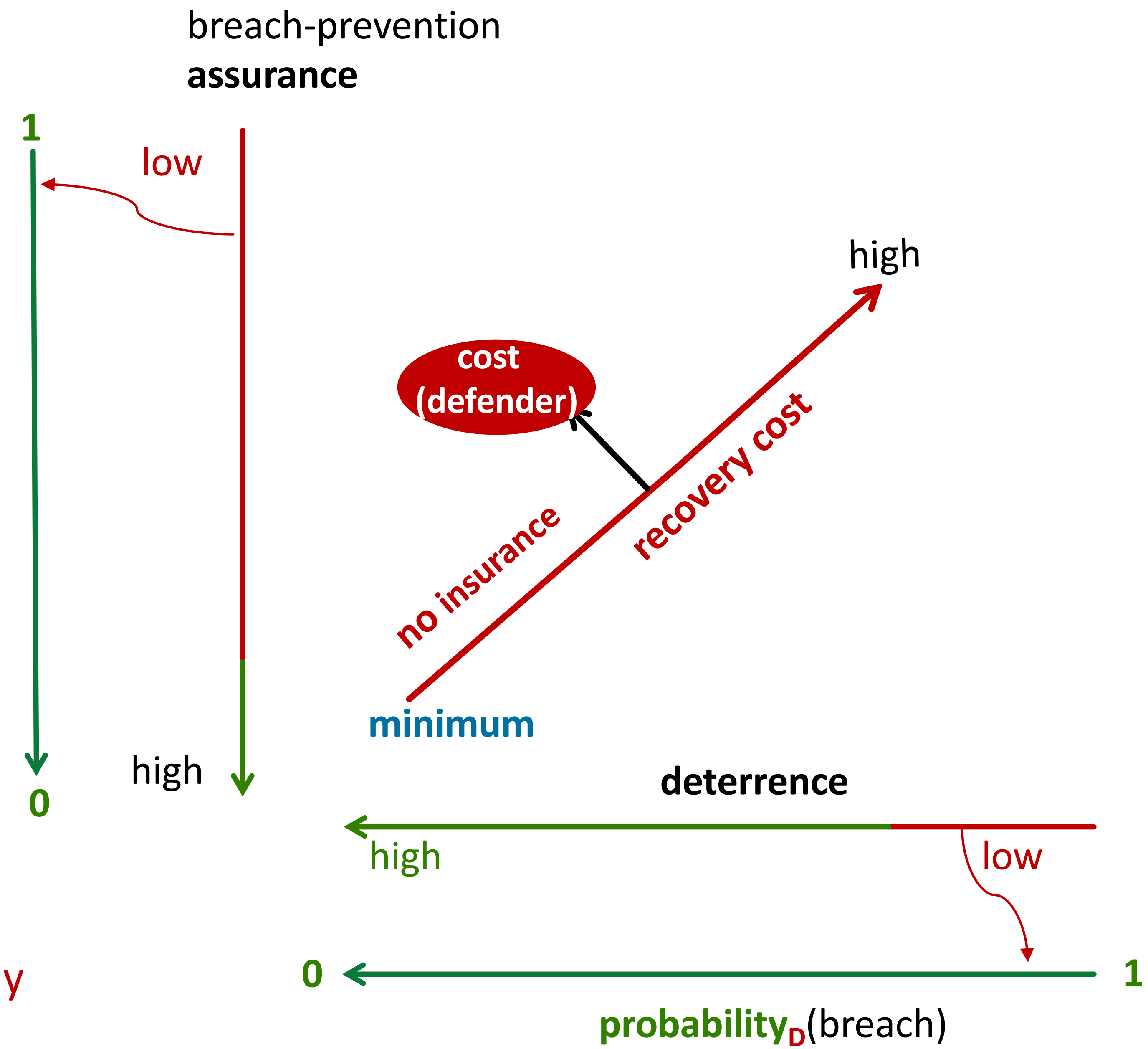


Example: IBM Cost of Data Breaches (2022)

Problem: recovery cost ≠ minimum



Solution: **buy insurance**



Problem:
many (> 50%) defenders
can afford but cannot buy
insurance

No alternative is left?

“assume breach” mindset [NSA2021, VB2022]

- **Assurance** – **low**: **probability$_A$**(breach) **-> 1**
- **Deterrence** – **low**: **probability$_D$**(breach) **-> 1**

- **Recovery cost** – **high**: **no insurance**

What is left is . . .

- **Selective High Assurance** => **probability$_A$**(breach) ≤ **upper limit**

- **Deterrence** - low: **probability$_D$**(breach) -> **1**

- **recovery cost** - high: no insurance

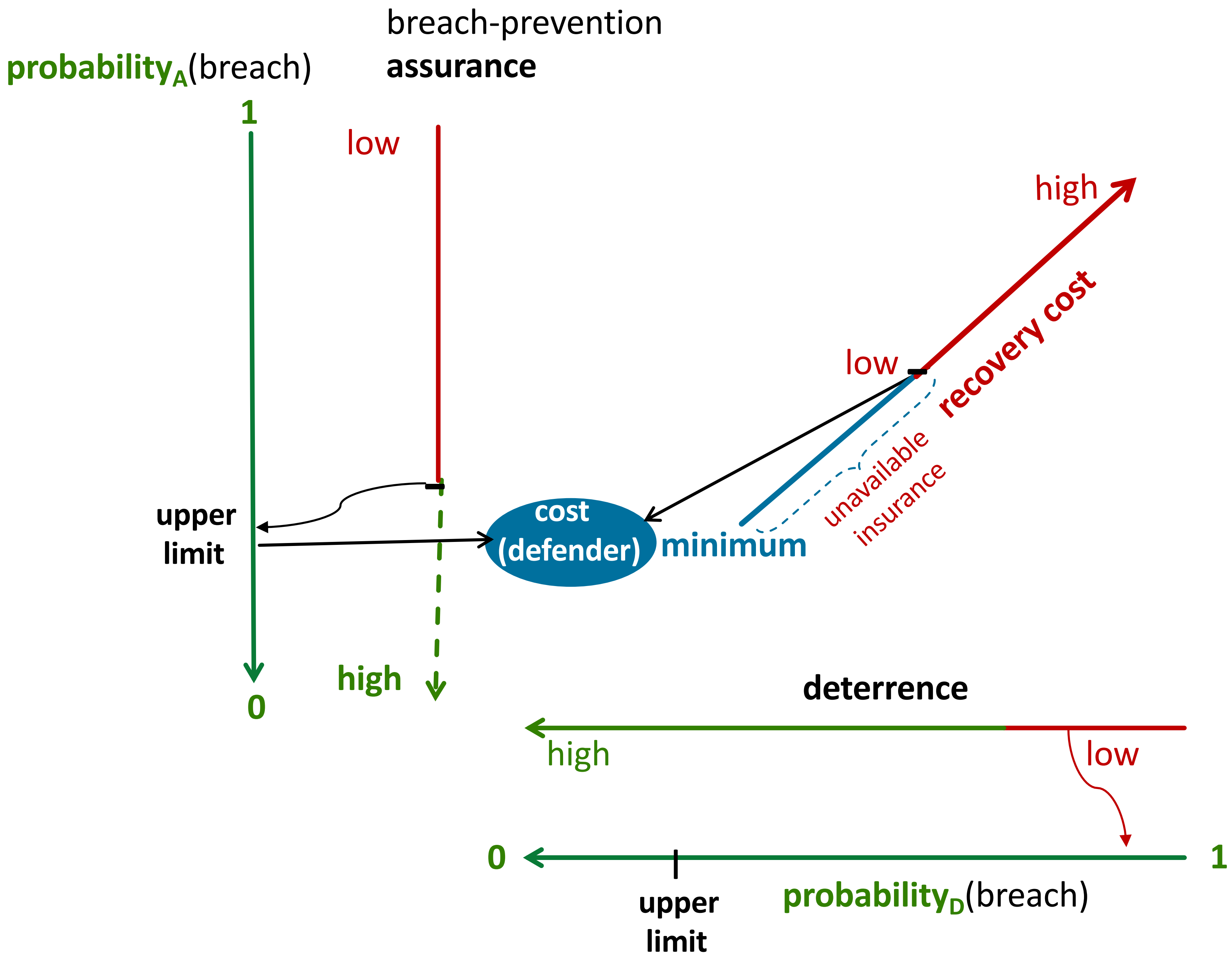
Goal

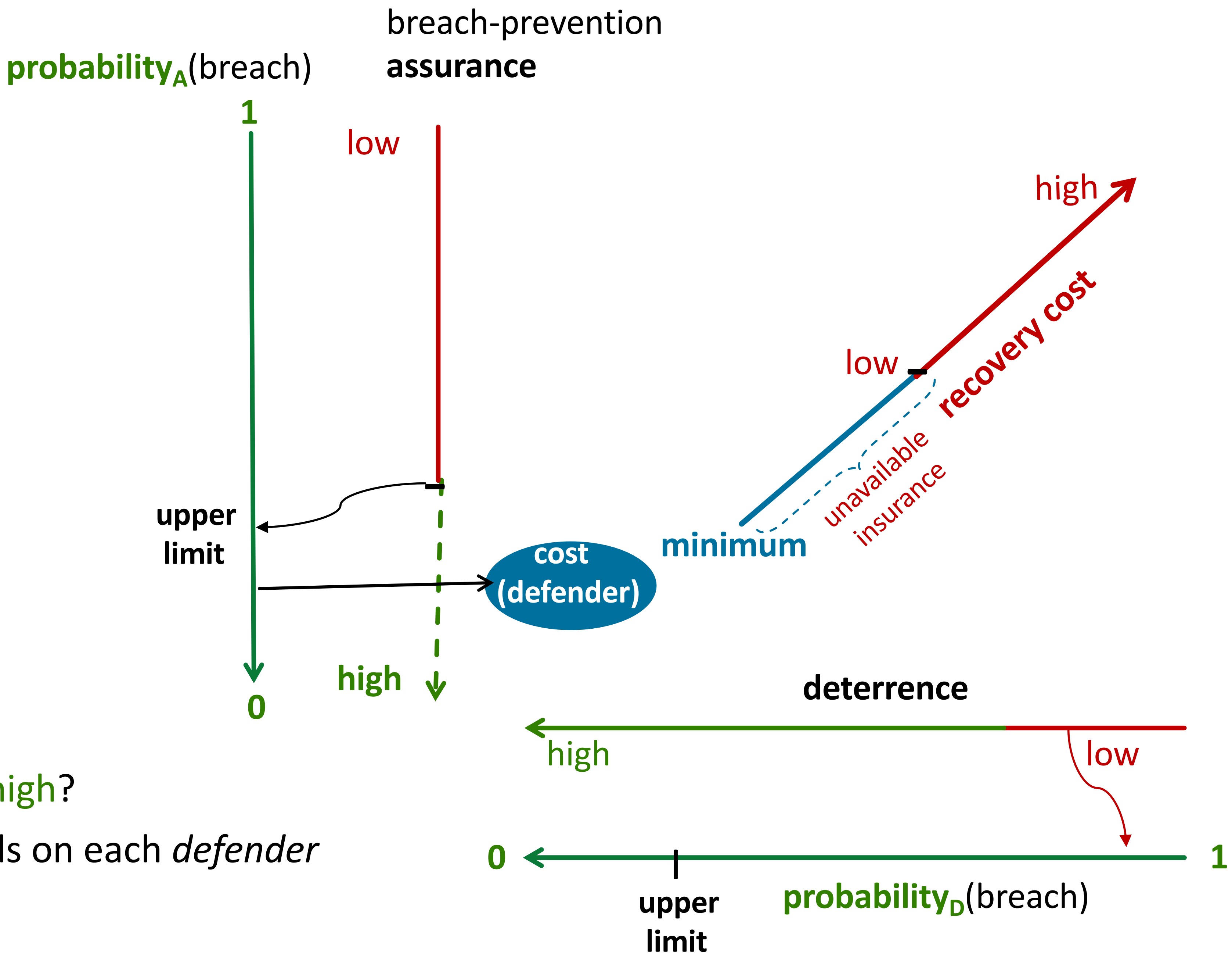
expected **Cost**(defender) = **probability$_A$**(breach) **x** **recovery cost**(breach)

≍ **insurance cost**(breach)

probability$_A$(breach) ≍ **insurance cost**(breach)/ **recovery cost**(breach)

≤ **upper limit**


breach-prevention
assurance
probability$_A$(breach)
1
low
high
recovery cost
low
upper
limit
cost
(defender)
minimum
unavailable
insurance
0
high
deterrence
high
low
0
1
upper
limit
probability$_D$(breach)



Question:

how high is high?

. . . it depends on each *defender*

A Challenge

Find a *<u>defender-independent</u> value* of high assurance for commodity software systems

e.g., find a *lower-bound value* that *depends only on the commodity software itself*

Goal: for *a <u>selected</u> set of breaches* of a *commodity software system*
selective high assurance **=>** ***probability***$_A$(breach) ≤ **upper limit**

Addressing the Challenge: Hypothesis of Formal Methods

Formal methods **=>** no vulnerability **=>** no security breach **=>** ***probability***(breach) = **0**

A Hypothesis Interpretation: Selective High Assurance

In general, for a *set of attacks* against a *selected source code*,
formal methods **=>** *attacks* are countered in *source code*
=> ***probability***(breach in <u>selected *source code*</u>) ≤ **upper limit (-> 0)**

How to do it?

For a *set of CVEs/CWEs* referring to the unverified *source code*,
formal methods **=>** *CVE/CWEs exploits <u>do not exist</u>* in verified source code
=> ***probability***(CVE/CWE exploit exists in *<u>source code</u>*) **= 0**

Illustrating a Value of Selective High Assurance (SHA) – Steps

1. Select a software system – i.e., SCION as the first example -- with:
   - *formally code-level verified security properties*,
   - substantial size and complexity,
   - internet-facing interfaces,
   - known attack surfaces

2. Select from over 200K vulnerabilities of CVE/CWE databases (MITRE, NIST) those
   - *that are countered by the formally verified protocols and services* of SCION
   - others that are related to selected ones; e.g., similarities and dependencies

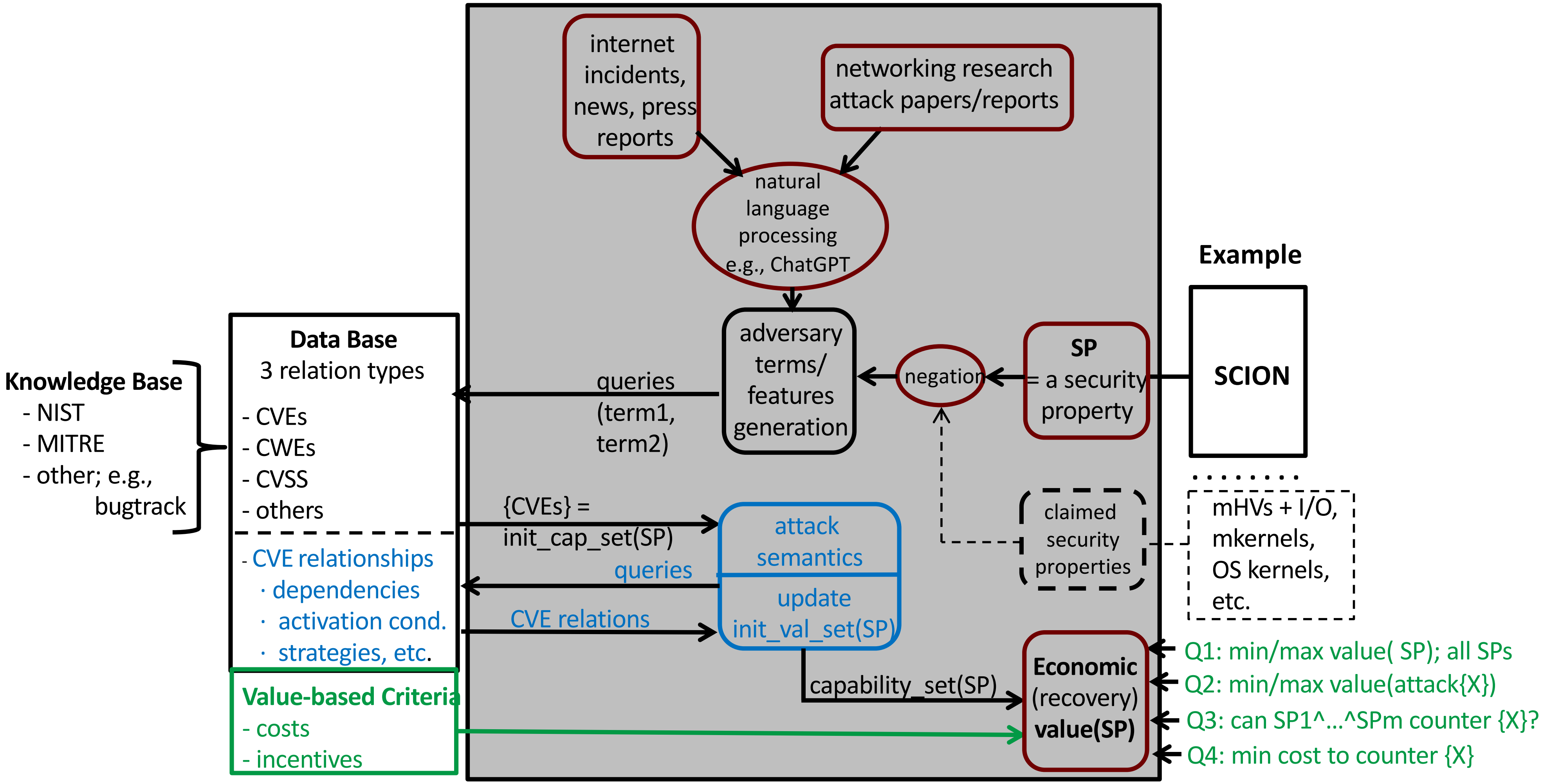
Define as many exploits (e.g., published and unknown attacks) as possible for the CVEs found in Step 2. Illustrate different attacker and defender values

Determine an industry-sanctioned *average cost of recovering from breaches caused by these exploits* in the *ordinary Internet*

The value obtained represents an *average lower bound of the formal methods* that enables SCION to counter those breaches.

Illustrating a Value of Selective High Assurance – Project Overview

NUS (Zhenkai Liang) - **ETH** (A. Perrig and D. Basin) – **CMU** (V. Gligor)



Q & A