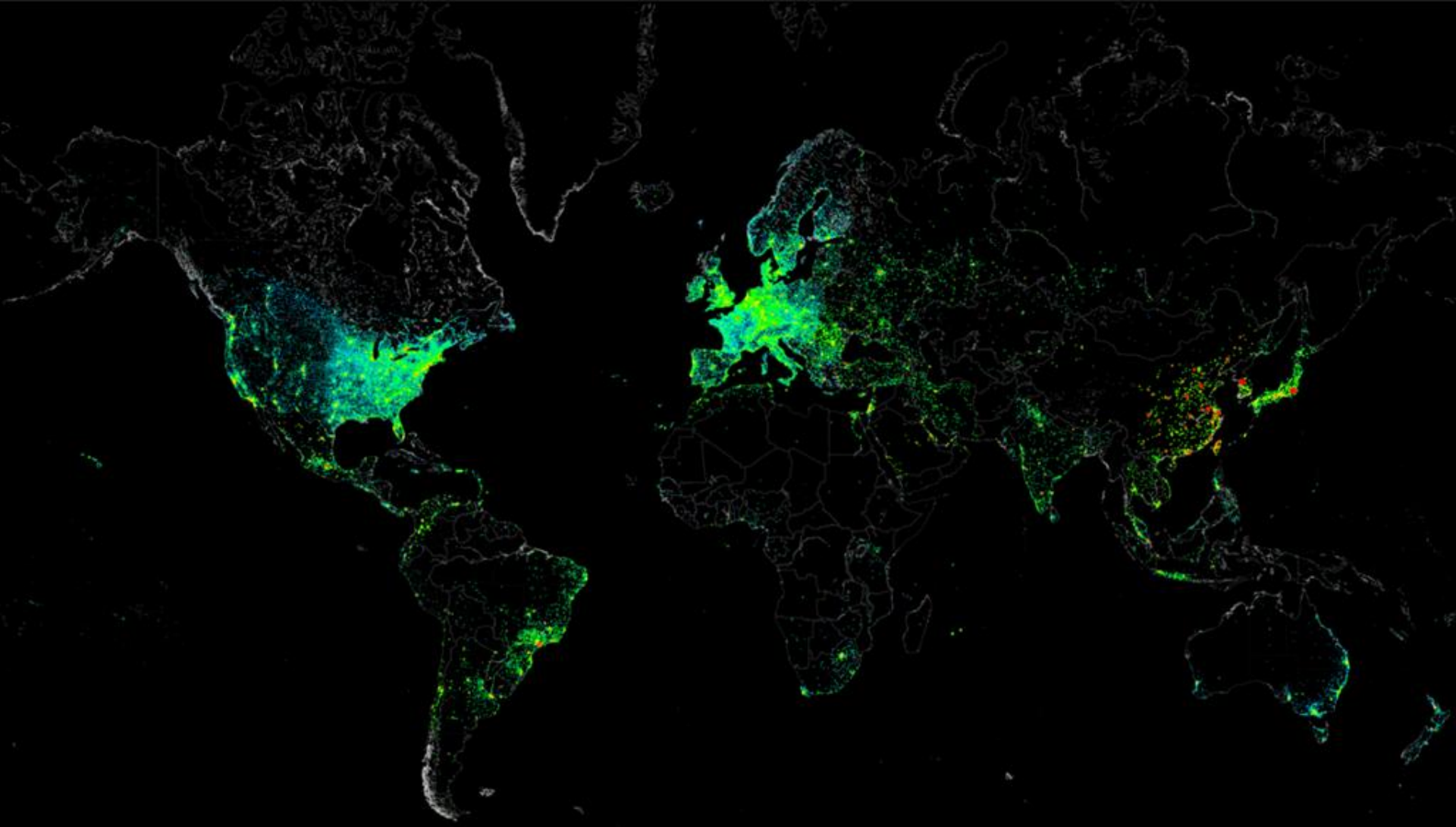
INFORMATION SECURITY WAR ROOM

USENIX SECURITY 2014

WHO AND WHAT

- Who
  - Prof. Sergey Bratus, Dartmouth College
  - Felix 'FX' Lindner, LangSec's & NATO's Zauberlehrling, otherwise Recurity Labs
- What
  - Options on the table, past and present
  - Who took them away and why
  - Current engagements with enemy forces
  - The final option



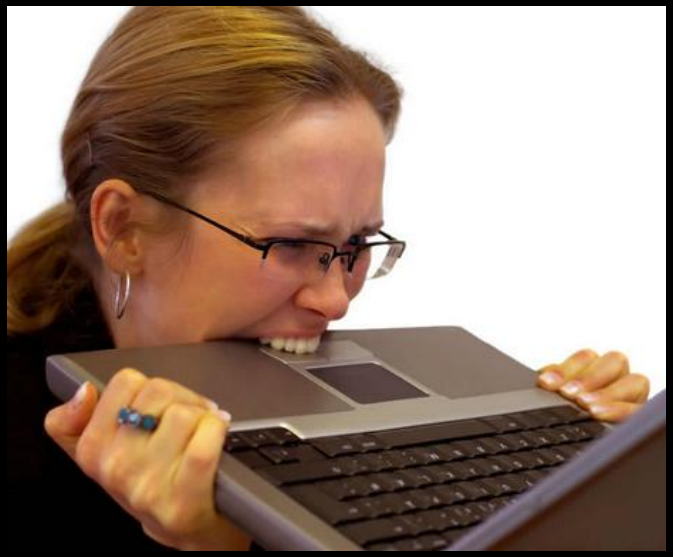
Where are the secure or otherwise correctly functioning computers?

ROOT CAUSE ANALYSIS

Map: Internet Census 2012 / Carna

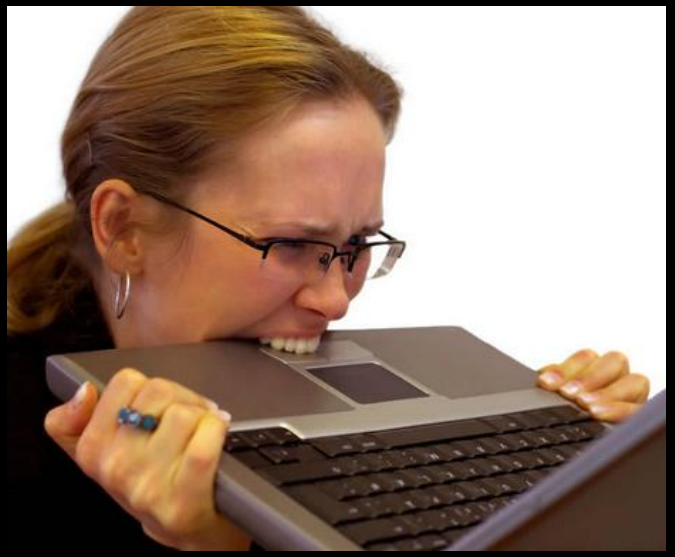
Drug Addiction

Drug Addiction



You Can't Do Anything

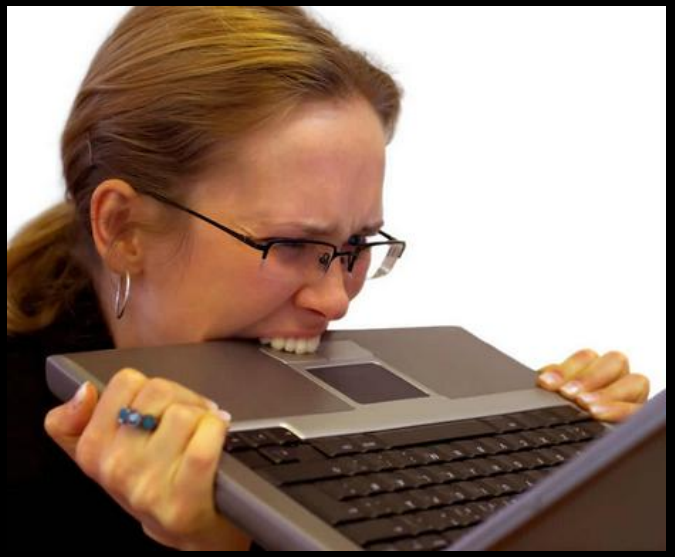
Drug Addiction



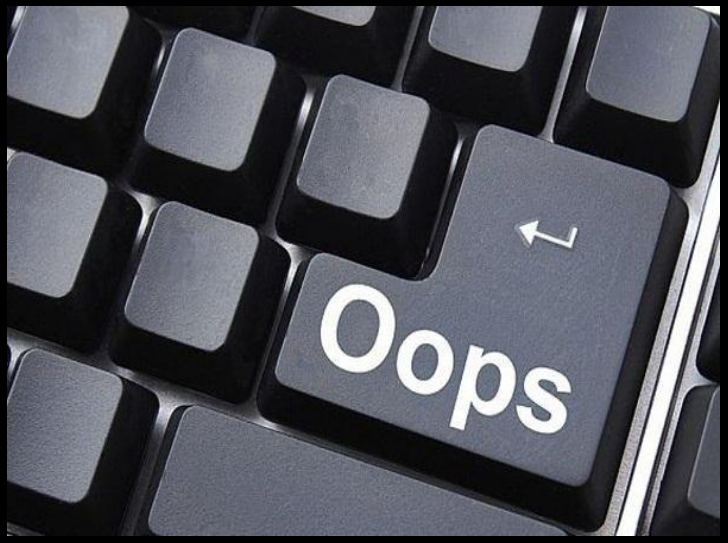
You Can't Do Anything

No Liability!

Drug Addiction

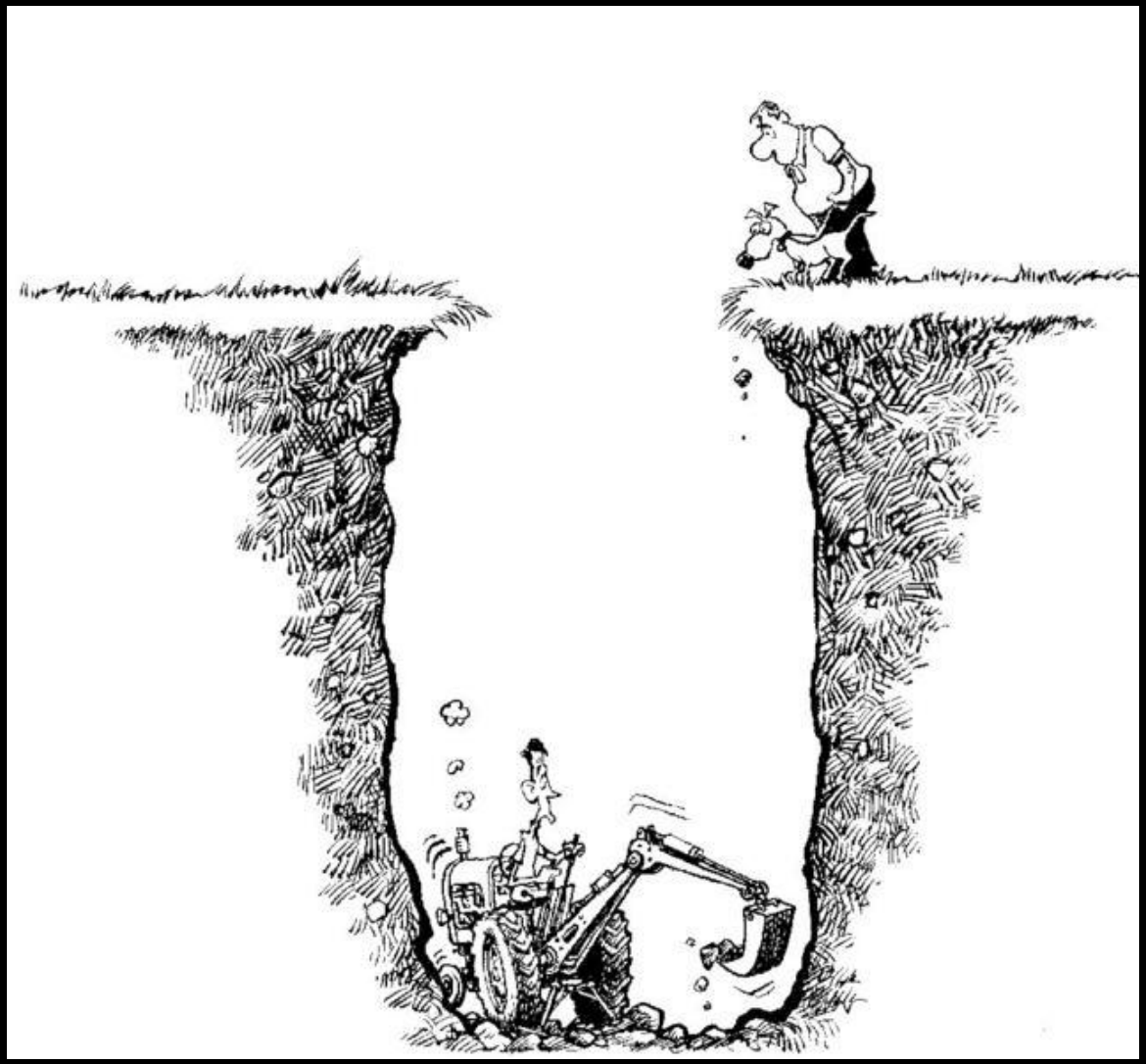


You Can't Do Anything



No Liability!

All You Have is Patches



Manual for Finding Yourself in a Hole

Chapter 1: Stop Digging

The Industry Got Your Back

The Industry Got Your Back

Just Believe!

The Industry Got Your Back

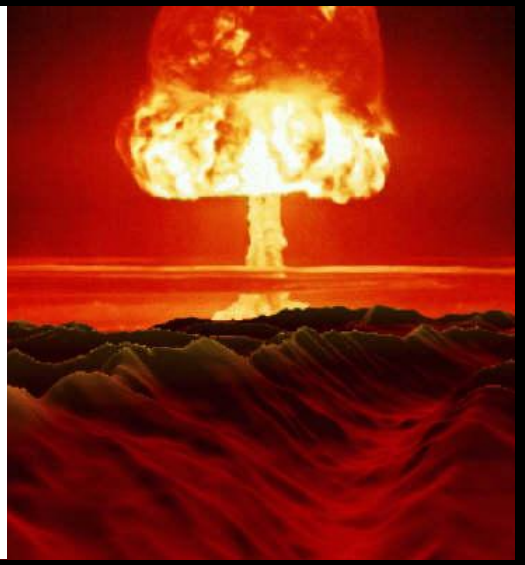
Just Believe!

A Business of Trust

The Industry Got Your Back

Just Believe!

A Business of Trust



Heartbleed? Sorry! That's Open Source.

GLOSSARY OF PARSING IN KERNEL CODE

Line Speed IPS: 500 protocol parsers, written in C, running in kernel

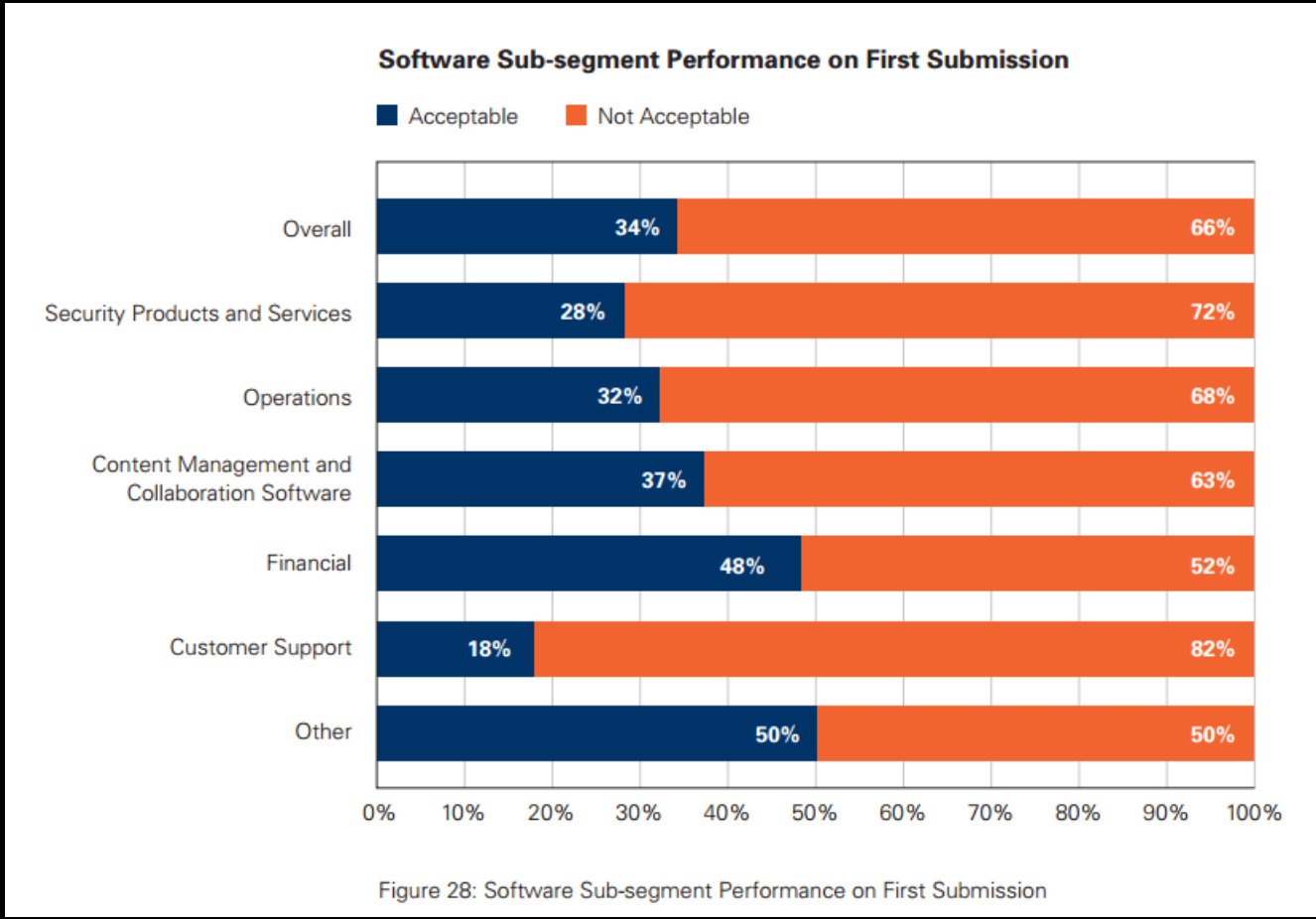


Figure 28: Software Sub-segment Performance on First Submission

"State of Software Security Report Volume 5", Veracode, April 8th, 2013

ORDNANCE QUALITY REPORT

Sir, reporting S.N.A.F.U., Sir!

Manual on Tool Clearance Regulations

Prerequisite Understanding Required

REGULATION IS EASY - AND #FAILS

```
OBTAIN "cyber security trends and implications" FROM regulator AS report
FOR EACH sector IN report:
  FOR EACH consumer_trend IN sector:
    SET vendor.fail TO unknown
    FOR EACH vendor MARKETING FOR consumer_trend:
      FOR EACH product MARKETED BY vendor:
        IF NOT product.isReported THEN
          SET vendor.illegal TO true
        END IF
        CALL BSI.LocalAgencyEquivalent WITH product RETURNING verdict
        IF verdict < acceptable_threshold THEN
          SET vendor.fail TO true
        END IF // fail
      END FOR // product
    END FOR // vendor
    IF vendor.illegal THEN
      CALL FTC.LocalAgencyEquivalent WITH vendor RETURNING prisoners
    ELSE IF vendor.fail EQUALS true THEN
      CALL FTC.LocalAgencyEquivalent WITH vendor RETURNING money
    ELSE
      SET vendor.fail TO false
    END IF
  END FOR // consumer_trend
END FOR // sector
```

Teufel mit dem Belzebub austreiben

When the cure is more harmful than the disease

Stephen Van Evera: Offense-Defense Theory

I. War will be more common in periods when conquest is easy, or is believed easy, than in other periods.

II. States that have, or believe they have, ***large*** offensive opportunities or ***defensive vulnerabilities*** will initiate and fight more wars than other states.

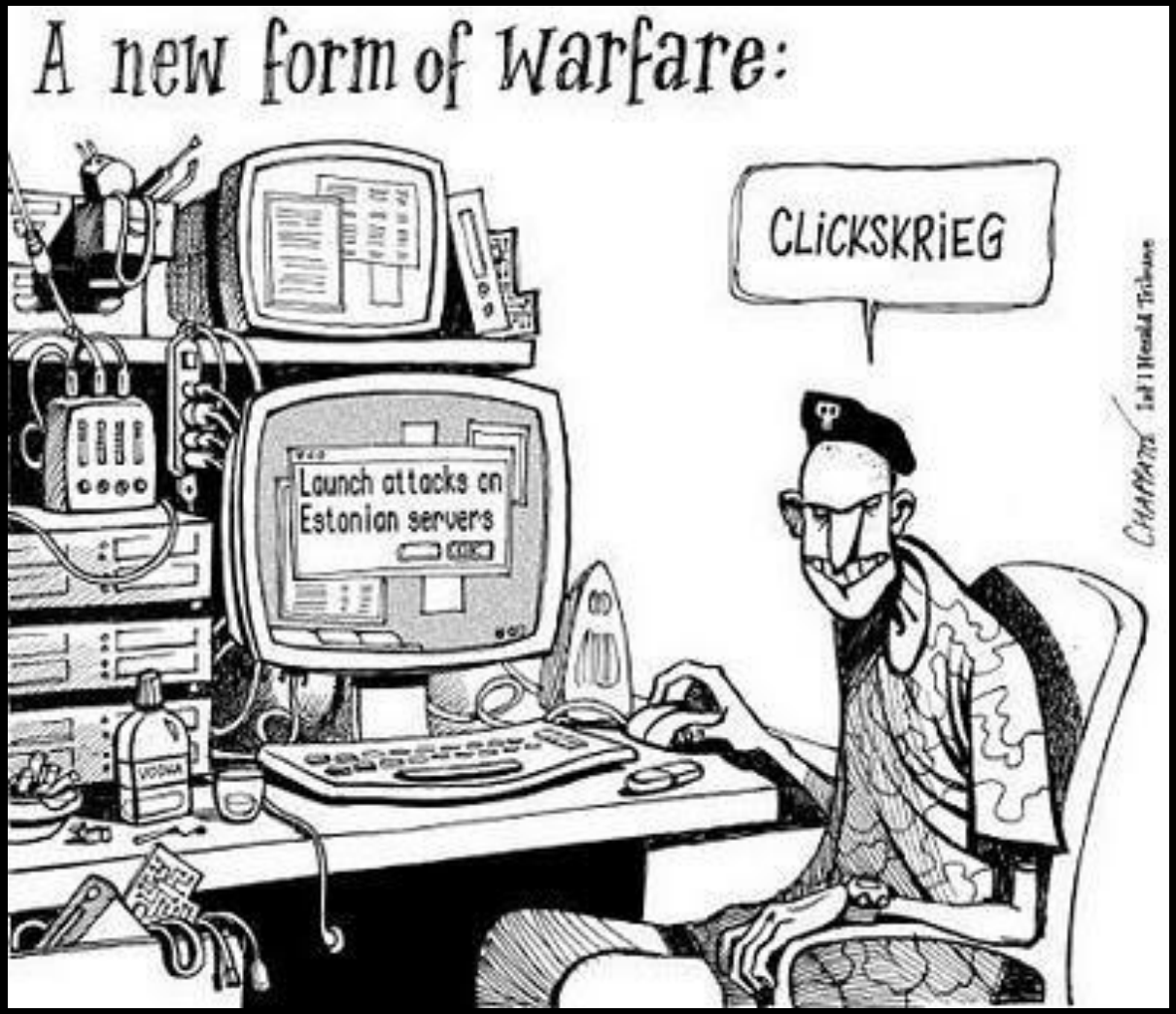
III. Actual examples of true imbalances are rare and explain only a moderate amount of history. However, false perceptions of these factors are common and thus explain a great deal of history.

Code of Conduct: 2011-09-12 UN proposal by China, Russia, Tajikistan and Uzbekistan

- ”Not to use ICTs including networks to carry out hostile activities or acts of aggression and pose threats to international peace and security. Not to proliferate information weapons and related technologies.“
- “cooperate in combating criminal and terrorist activities which use ICTs […]”
  - “[…] curbing dissemination of information which […] undermines other countries' political, economic and social stability, as well as their spiritual and cultural environment”
- “ensure the supply chain security of ICT products and services […]”
- “respect the rights […] and freedom of searching for, acquiring and disseminating information”
- “establishment of a multilateral, transparent and democratic international management of the Internet”
- “settle any dispute resulting from the application of this Code through peaceful means and refrain from the threat or use of force”

ADVANCED = IT WORKED
PERSISTENT = REPEATEDLY
THREAT = YOU DON'T KNOW WHY

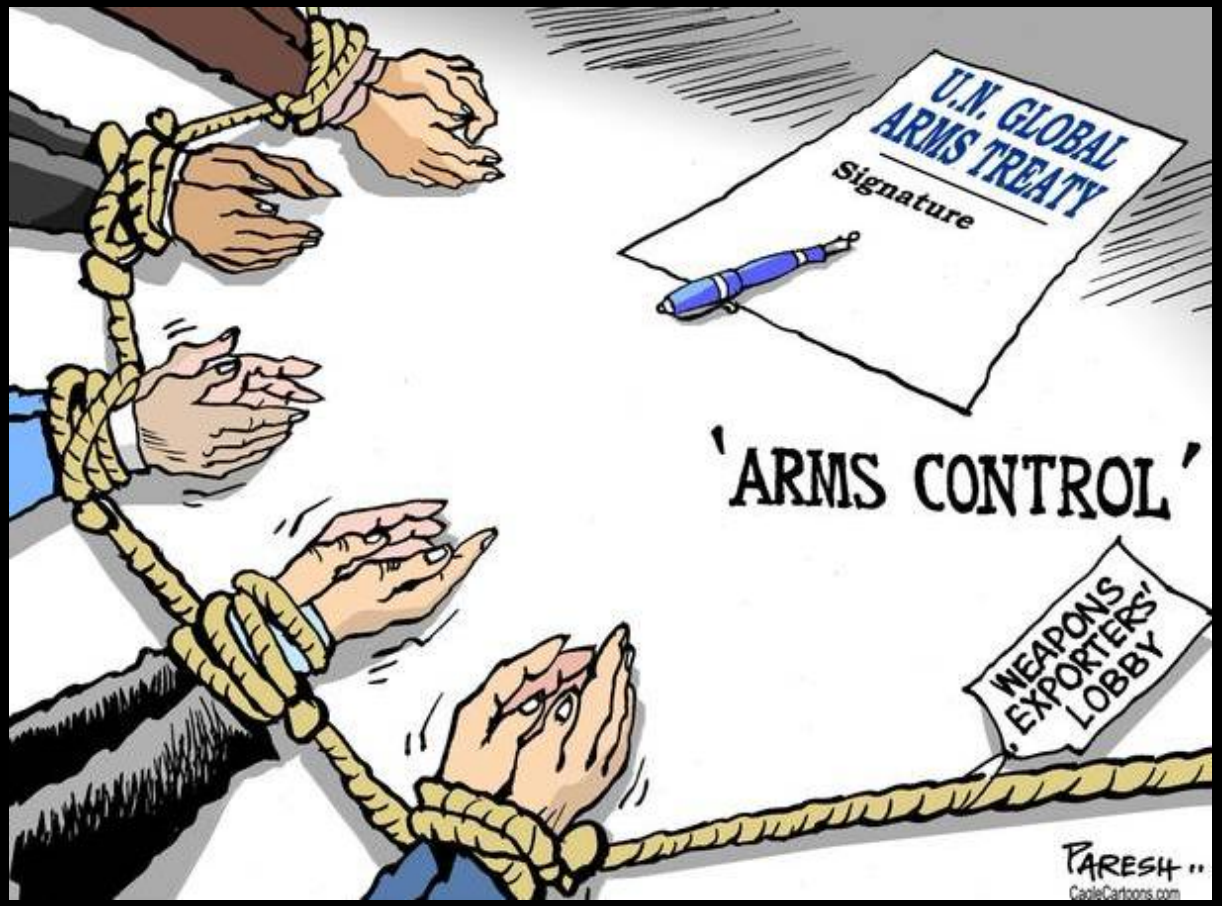
Side note: Either China has the coolest Metasploit UI known to man, or these soldiers are, in fact, playing a game.



RUSSIAN INTERPRETATION OF АПТ

Автомат Калашникова

Ботнет



MEMO ON THE HANDLING OF CYBER…THINGS

Business Model Preservation

WASSENAAR ARRANGEMENT – DEC 2013

Cat 4:

"Intrusion software": "Software" specially designed or modified to avoid detection by "monitoring tools", or to defeat "protective countermeasures", of a computer or network-capable device, and performing any of the following:

a) The extraction of data or information, from a computer or network-capable device, or the modification of system or user data; or

b) The modification of the standard execution path of a program or process in order to allow the execution of externally provided instructions.

WASSENAAR ARRANGEMENT – DEC 2013

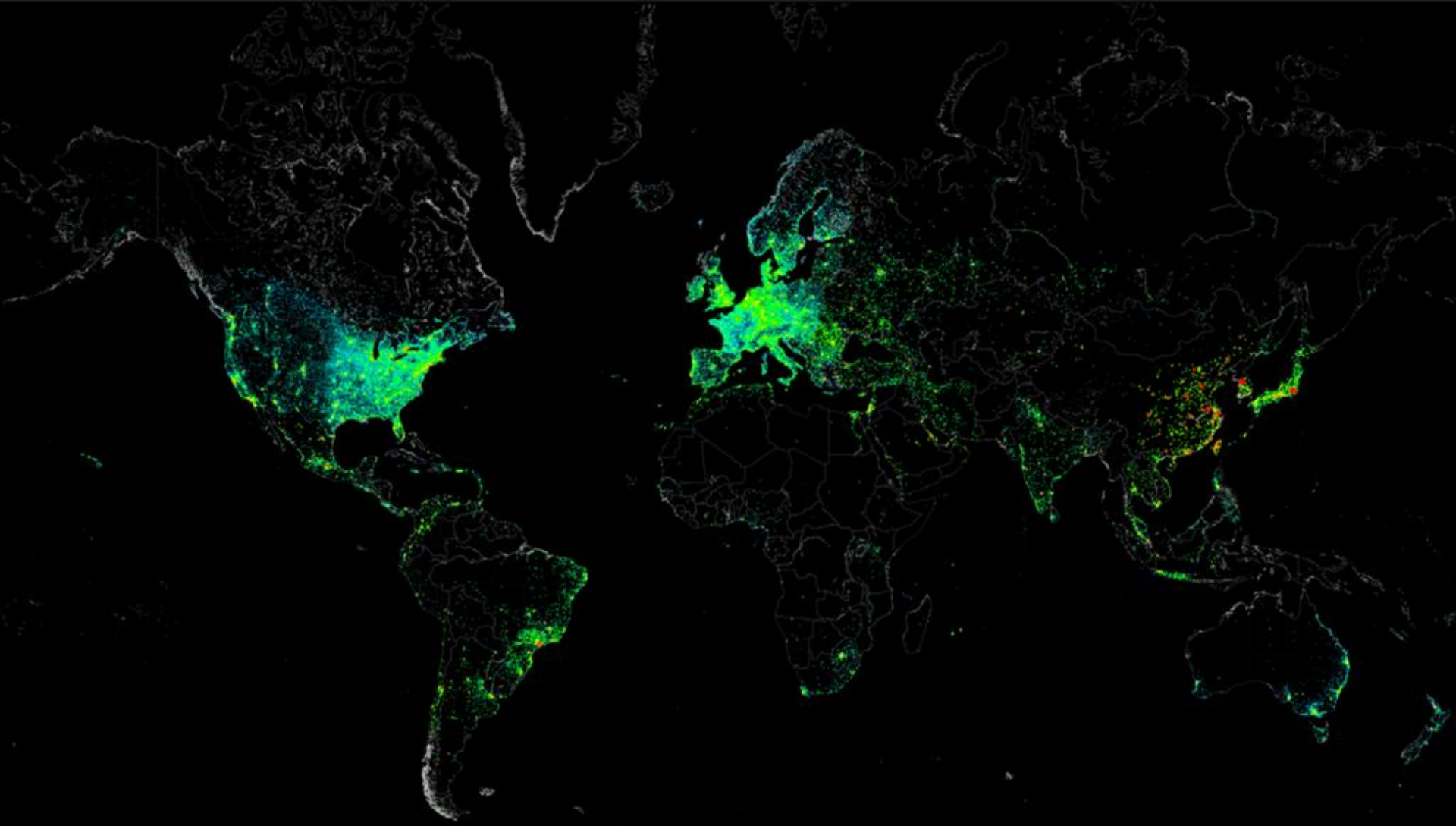
5. A. 1. j.

IP network communications surveillance systems or equipment, and specially designed components therefor, having all of the following:

1. Performing all of the following on a carrier class IP network (e.g., national grade IP backbone):
   a) Analysis at the application layer (e.g., Layer 7 of Open Systems Interconnection (OSI) model (ISO/IEC 7498-1));
   b) Extraction of selected metadata and application content (e.g., voice, video, messages, attachments); and
   c) Indexing of extracted data; and
2. Being specially designed to carry out all of the following:
   a) Execution of searches on the basis of 'hard selectors'; and
   b) Mapping of the relational network of an individual or of a group of people.

Note 5.A.1.j. does not apply to systems or equipment, specially designed for any of the following:

a. Marketing purpose;
b. Network Quality of Service (QoS);or
c. Quality of Experience (QoE).



Laws and regulations before textbooks? 90s Crypto Wars Reloaded.

BROKEN PHONE LEGISLATION

Map: Internet Census 2012 / Carna

OFFENSIVE RESEARCH

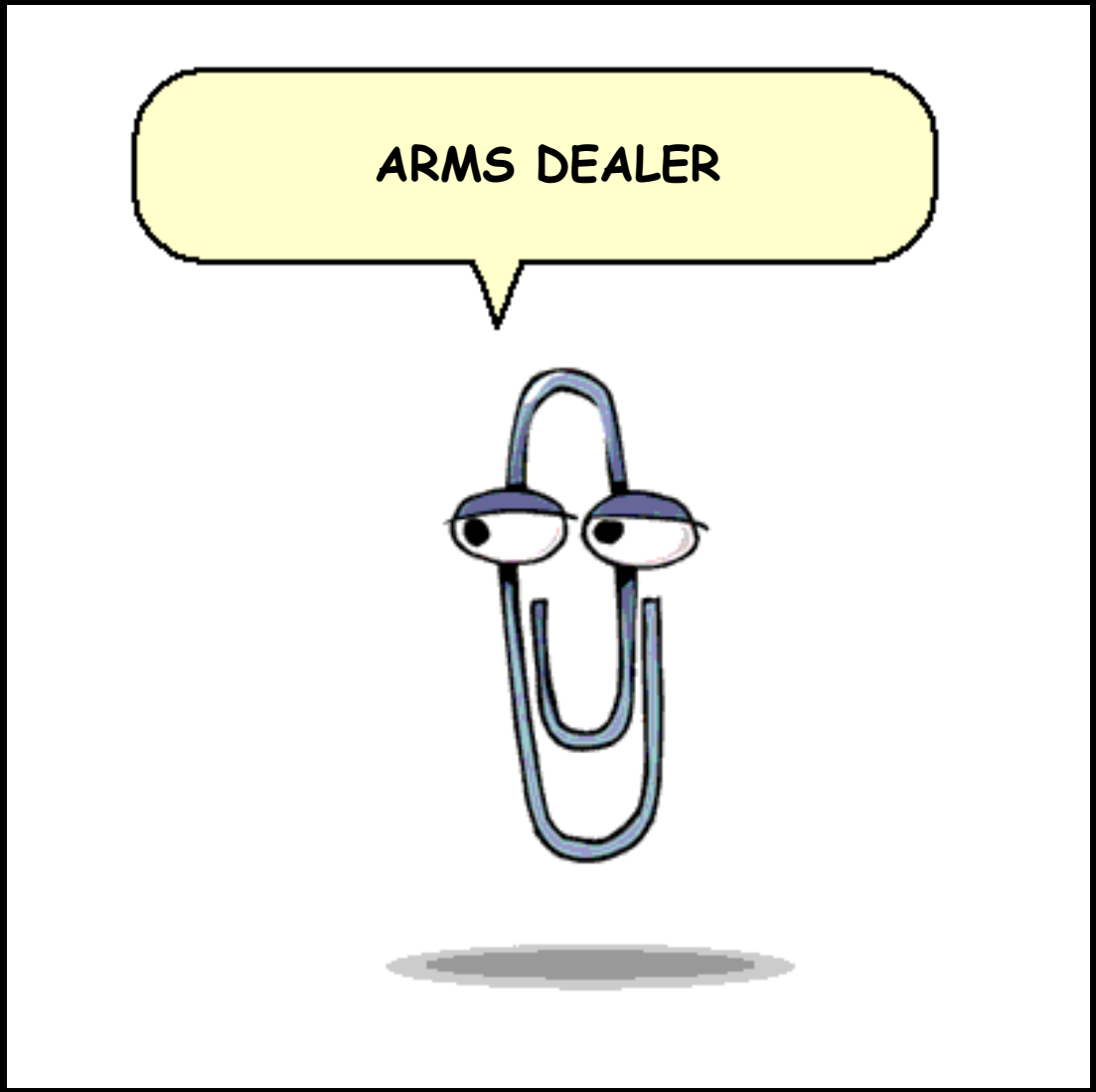
- Community call to action in August 2014 USENIX ;login:
  - https://www.usenix.org/system/files/login/articles/02_bratus.pdf
  - http://www.cs.dartmouth.edu/~sergey/drafts/why-offensive-security-needs-textbooks.pdf

PLAN ON HACKER RESEARCH

And the Extinction Thereof

HACKERS COST US MONEY

HACKERS GONE, MORE MONEY

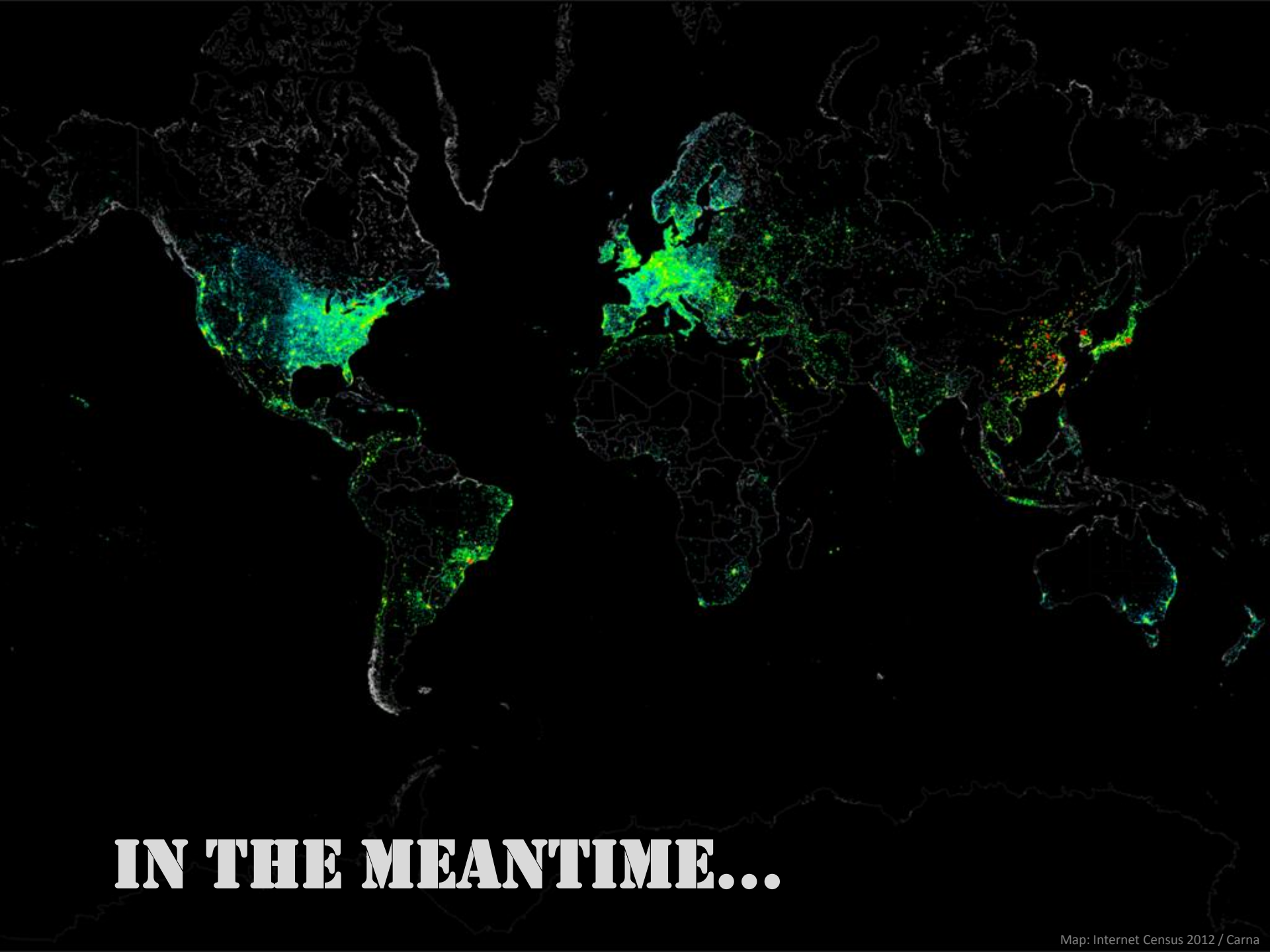


EXPERT (RETIRED) OPINION

"Our lobbyists say it must be illegal."

WHAT ABOUT THE ROOT CAUSE?

- Military always had harder purchase requirements
  - Their camping equipment is better than anyone's
  - They do camping like it is their job
  - Surprise: It is!
- The DoD spends 80 months for new software
- Military is the right entity to ask for product liability in software
  - They need measures
  - The suppliers need guidance to follow



Map: Internet Census 2012 / Carna

Real Security Makes Money

- Grammars scare people
  - Generating code automatically makes them wanting grammars
- Product Houses in enterprises know the cost of not having a grammar
  - Hence XML
- Providing a process to agree on interface grammars is providing a strong business incentive to the people that matter
  - If you save someone money, they don't ask how you did that, they just do as you tell them

IF NO MONEY IS SPENT SAFE SOME MONEY

- Given a set of strong data types…
- For each product group…
  1. List information to handle
  2. Strongly type all fields
  3. Send to all product groups
  4. Request signoff
  5. If not signed off, goto 1
- Generate code for all product groups: Nail it

1	Name	var String
2	SpargelSize	var int

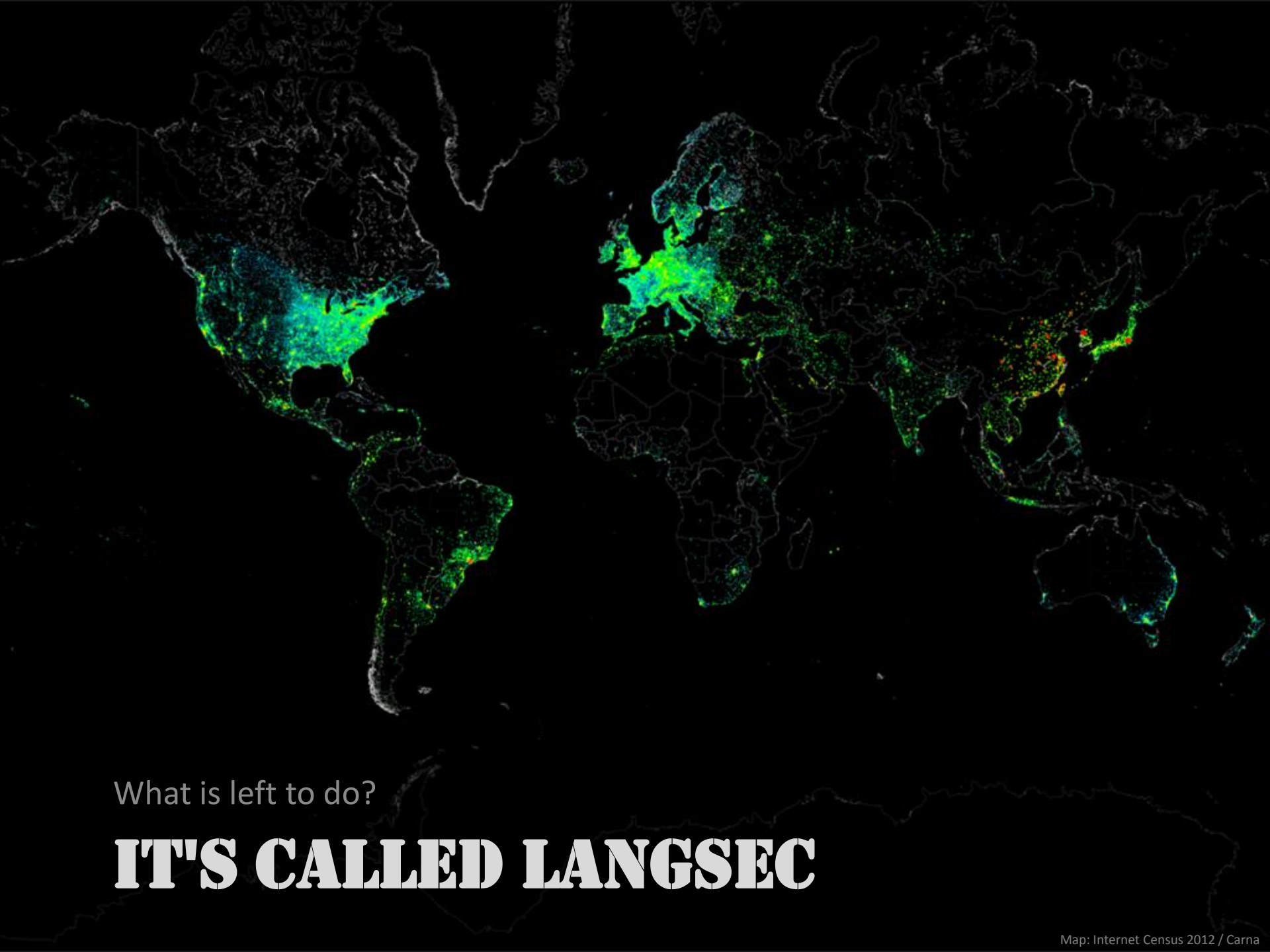
1	Name	var UTF8
2	SpargelSize	var unsigned int

1	Name	var UTF8
2	SpargelSize	unsigned int32

1	Name	UTF8[200]
2	SpargelSize	unsigned int32

The Agreement Machine Scales

- Standardization Committees have the same problem: n^2
- The standard solution is to agree on disagreement, also known as Vendor Extension
  - IPv6 is a prime example



What is left to do?

IT'S CALLED LANGSEC

Map: Internet Census 2012 / Carna

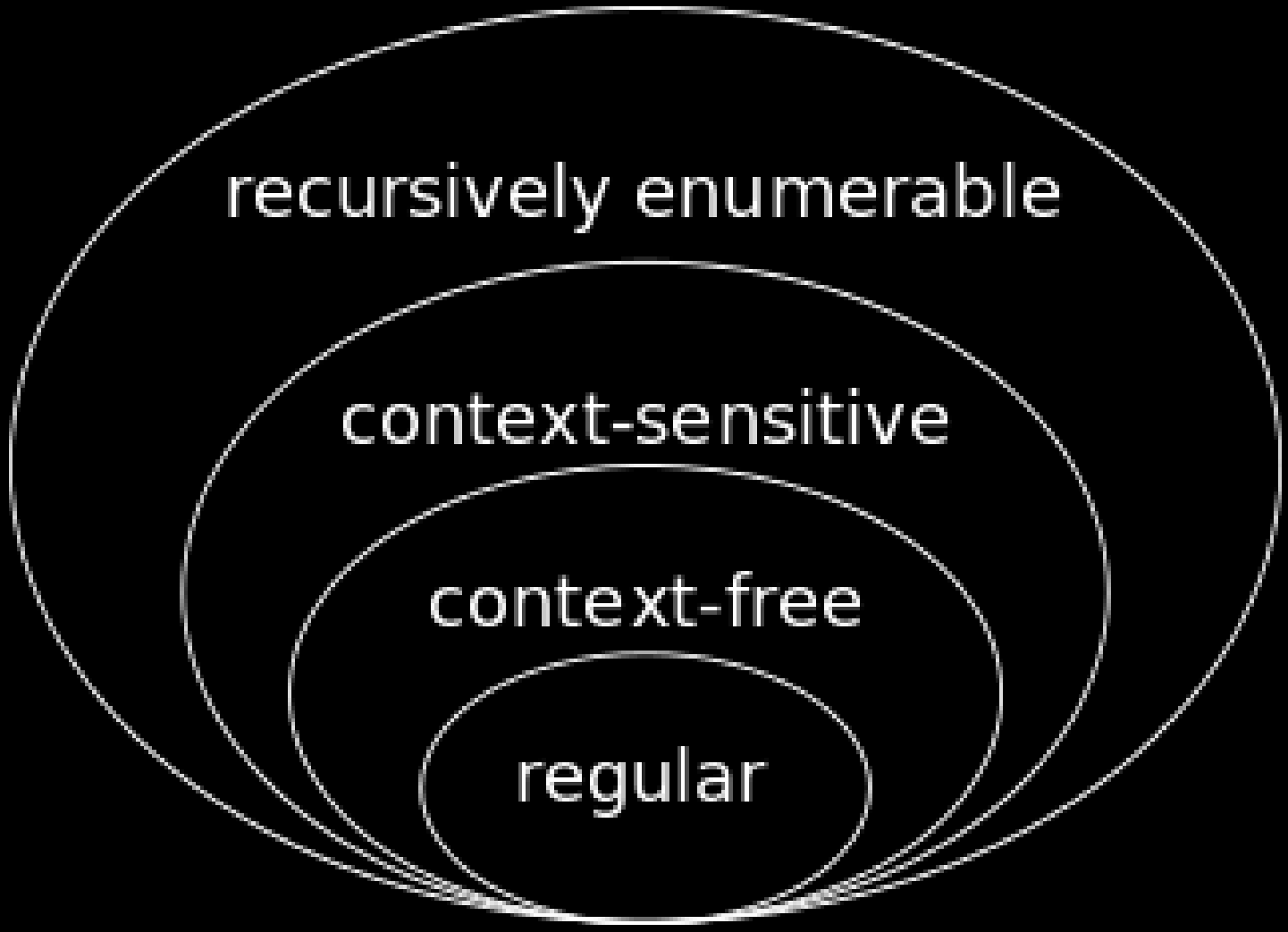
ENEMY AT THE ~~GATES~~
COMMUNICATION BOUNDARIES

- Len Sassaman's motivation for LangSec: cryptography **doesn't** solve secure communication in a world where all input-handling code is exploitable
- Problem: predicting effects of inputs on a given program
- We must model computations induced by inputs

"Security Applications of Formal Language Theory",
Len Sassaman, Meredith L. Patterson, Sergey Bratus, Michael E. Locasto, Anna Shubina
in IEEE Systems Journal, Volume 7, Issue 3, Sept. 2013
http://www.cs.dartmouth.edu/~sergey/langsec/papers/sassaman-jsys7-3.pdf

Enemy at the ~~Gates~~ Communication Boundaries

- Problem: predicting effects of inputs on a given program
- We must model computations induced by inputs
- Formal language theory connects “inputs” with computation via **recognition**
  - Classes of languages form a hierarchy w.r.t. computational power needed to recognize them (Chomsky hierarchy)



THE CHOMSKY HIERARCHY

A.k.a. "Chomsky-Schützenberger hierarchy"

EXPLOITS VS RECOGNIZERS

- Exploits have been exercising this connection!
- Web Apps: “where recursively nested object inputs are validated with a regexp”
  - Context-free language cannot be recognized with a finite state automaton. #Fail
- Many famous parser (memory corruptions) bugs are recognition bugs: a property of input is assumed, but not actually verified.

RULES OF [INPUT] ENGAGEMENT

- Inputs can only be trusted after they have been fully recognized [in formal language sense]
- Inputs that have no definition that can be so recognized cannot be trusted
- Communication boundary code that does not implement a recognizer for a well-specified formal language of inputs cannot be trusted

"A Patch for Postel's Robustness Principle" -- IEEE S&P Journal Mar/Apr 2012
http://langsec.org/papers/postel-patch.pdf

ANY INPUT IS A PROGRAM

- Everyday: code meets crafted input, becomes a machine for executing that input
- Input “becomes” code
  - System becomes “exploited” = runs unexpected computation it is trusted to NOT run
- In fact: **Any input is code**
  - “Everything is an interpreter” (Greg Morrisett)
- Remember WASSENAAR?
  - "The modification of the standard execution path of a program or process in order to allow the execution of externally provided instructions."
  - It follows that ***inputs are regulated arms***

"Exploitation is magic" (pwnie here)

- "Any sufficiently advanced technology is indistinguishable from magic" --A. Clarke
- "Any sufficiently complex input data is indistinguishable from bytecode; the code that must interpret it is indistinguishable from a virtual machine for that bytecode."--SB
- Corollary: parsing complex data is being (remotely) programmed

"Everything is an interpreter"

- Any system that receives inputs is driven by these inputs into state transitions
  - Regexp is a machine, string being matched is a program
  - Same for any finite state or pushdown automaton
  - Contents of tape are Turing machine's programs
  - Any code whatsoever is a machine for the data it reads

INVISIBLE MACHINES (1)

- Standard function prologues and epilogues are an **automaton**, distributed through code, data fragments on the **stack** are its **programs**
  - This automaton realizes the “control flow graph”
  - Programmed since Aleph One’s Phrack 49:14
  - Solar Designer, Newsham, gera, Nergal, and others discovered programming techniques
  - Known as ROP (Shacham) since 2007

A sketch of how exploit programming techniques developed can be found in http://langsec.org/papers/Bratus.pdf

INVISIBLE MACHINES (2)

- Heap management code is a machine, contents of heap (chunk metadata) are its programs
  - Programmed since “Once upon a free” & “Vudo malloc tricks” Phrack 57:8, 57:9
  - ISA includes “write four bytes X to location Y”
  - Configuration is performed via malloc() calls (e.g., Sotirov’s “Heap Feng-Shui”, 2007)

"Once upon a free": http://phrack.org/issues/57/9.html
"Vudo malloc tricks": http://phrack.org/issues/57/8.html
"Heap Feng-Shui in Javascript":
https://www.blackhat.com/presentations/bh-europe-07/Sotirov/Presentation/bh-eu-07-sotirov-apr19.pdf

"Weird Machine" Instruction Set Architectures?

- If code is a machine, what is its Instruction Set Architecture?
- "push ebp; mov esp, ebp; …; leave; ret"
- Format string exploitation: %n stack write
- Malloc exploitation: "aa4bmo" "chunk->flink->blink = chunk->blink"
- Sigreturn-oriented Programming
- …

"Advances in format string exploitation": http://phrack.org/issues/59/7.html
"Advanced Doug lea's malloc exploits": http://phrack.org/issues/61/6.html -- explains aa4bmo primitive
"Heap Exploitation Abstraction by Example": http://census-labs.com/media/heap-owasp-appsec-2012.pdf
"Sigreturn-oriented Programming": https://www.cs.vu.nl/~herbertb/papers/srop_sp14.pdf

Co-evolution of Defense & Offense

- "If you shame attack research, you misjudge its contribution. Offense and defense aren't peers. **Defense is offense's child**."
  -- John Lambert @JohnLaTwC
- Offense provides the computational model for defense regarding its areas of engagement
- Would you use cryptographic primitives that have not been attacked?
  - How is software different from crypto?

"BRING YOUR OWN LINKER" (BYOL)

- Turing completeness of "weird machines" (emergent architectures for unexpected computation) is a nice theoretical property, but largely a distraction in practice
- Except when a non-trivial computation is needed to effect composition of exploit program & the target
- "Must compute hidden/obfuscated symbols" (Cesare 1998, Grugq, ERESI 2001, …)

A sketch of composition patterns used by exploits can be found in:
"Composition Patterns of Hacking", Bratus et al., The First International Workshop on Cyberpatterns, Abingdon, UK,
http://www.cs.dartmouth.edu/~sergey/drafts/hacker-composition.pdf
(also in "Cyberpatterns, Unifying Design Patterns with Security and Attack Patterns",
Blackwell, Clive, Zhu, Hong (Eds.), Springer, 2014, XI, 264 p.]

VALIDATION IS VERIFICATION

- What we do with input is “validate” it
  - “Sanitizing” data to “remove badness” is an anti-pattern. Avoid it and the modes of design as the plague they are!
- What we should do with code is **verify** it
- We can only trust “data” inputs if we know what it does **as a program**

INPUTS MUST BE VERIFIED!

"**AEG as a program-verification task but with a twist** [..]. Traditional verification takes a program and a specification of safety as inputs and verifies the program satisfies the safety specification. The twist is we replace typical safety properties with an "exploitability" property, and the **"verification" process becomes one of finding a program path where the exploitability property holds.** Casting AEG in a **verification framework** ensures AEG techniques are based on a **firm theoretic foundation**."

"Automatic Exploit Generation",
Avgerinos et al, Vol. 57 No. 2, Pages 74-84, 2014

http://cacm.acm.org/magazines/2014/2/171687-automatic-exploit-generation/abstract

VERIFICATION IS HARD

- Static analysis quickly runs into undecidability
- Liveness: Halting problem
- Non-trivial code properties: Rice's theorem
- Security in any real sense is unattainable if respective verification task is undecidable
  - Statically deciding what a program does
  - Statically deciding if two parsers interpret the same message equivalently

THE UNDECIDABILITY CLIFF

- Verification tasks related to data languages become **undecidable** after certain language complexity is reached
  - Liveness of Turing-complete languages
    - "Weird Machines" in ELF: A Spotlight on the Underappreciated Metadata", https://www.usenix.org/conference/woot13/workshop-program/presentation/shapiro
  - “Benighness” of the above
    - "Scriptless Attacks: Stealing the Pie Without Touching the Sill", Heiderich et al., https://www.nds.rub.de/media/emma/veroeffentlichungen/2012/08/16/scriptlessAttacks-ccs2012.pdf
  - Equivalence of parsers (i.e. languages recognized): undecidable for non-deterministic pushdown automata & above
    - "PKI Layer Cake: New Collision Attacks Against the Global X.509 Infrastructure", Kaminsky, Patterson, Sassaman, https://www.cosic.esat.kuleuven.be/publications/article-1432.pdf

Shifting the focus of verification

- The philosophers have only **interpreted** the world, in various ways; the point is to **change** it" – K. Marx (add hammers & sickles!)
- Verification efforts tend to regard computing tasks—including data-related ones—as set. The point is to make the **data languages simpler** to make tasks of verifying its code more **tractable**.

Simpler data formats → simpler recognition tasks

- Simplifying languages of data formats pays off:
  - *Regular* if no arbitrary depth nesting needed
  - *Context-free* if recursive nesting **is** needed
  - *Weakly context-sensitive* if possible
    - Good luck defending that ;)
- **Never** Turing-complete input languages!

RECOGNIZERS ARE EVERYWHERE

- Any code that receives input is driven by that input
- Parser bugs still dominate Pwnies
  - In 2013 and 2014

Heartbeat sent to victim

SSLv3 record:

Length
4 bytes

HeartbeatMessage:

Type	Length	Payload data	
TLS1_HB_REQUEST	65535 bytes	1 byte	

Victim's response

SSLv3 record:

Length
65538 bytes

HeartbeatMessage:

Type	Length	Payload data	
TLS1_HB_RESPONSE	65535 bytes	65535 bytes	

HEARTBLEED IS A PARSER BUG!

Two length fields must agree for heartbeat message to be valid. Never checked.

Android Master Key(s): A Parser Differential

- Java crypto signature verifier: unzip without unsigned integers
- C++ installer: unzip with unsigned integers
- Different contents “verified” vs. installed
- Finally fixed right: one parser applied to package files, not two expected to be equivalent

"Android Bug Superior to Master Key": http://www.saurik.com/id/18
see also: http://www.saurik.com/id/17, http://www.saurik.com/id/19

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  /* MISTAKE! THIS LINE SHOULD NOT BE HERE */
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
```

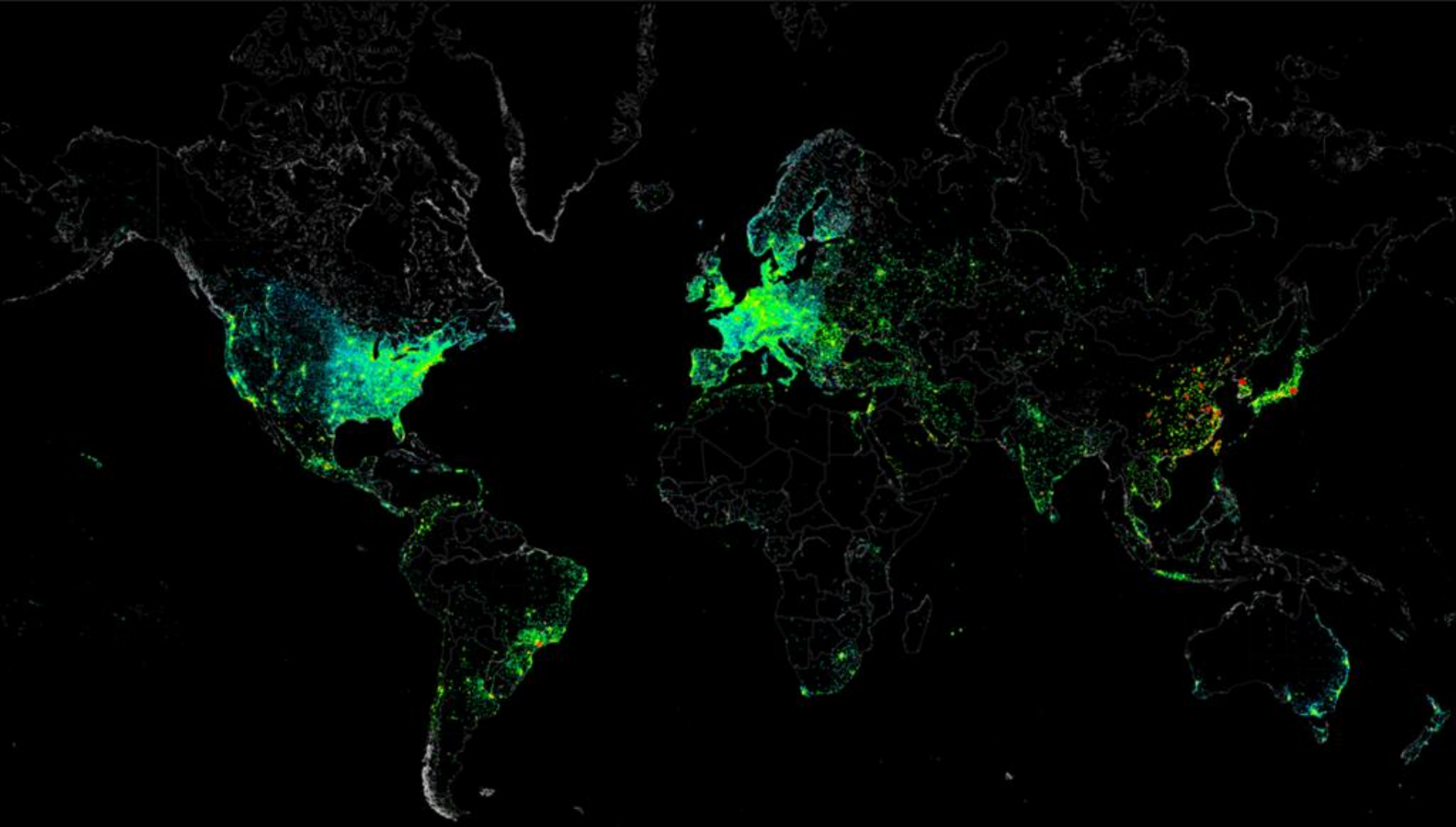
http://nakedsecurity.sophos.com/2014/02/24/
anatomy-of-a-goto-fail-apples-ssl-bug-explained-plus-an-unofficial-patch/

GOTO FAIL;

State machine done wrong: code must be generated

NGINX CHUNKED ENCODING BUG

- State machine done wrong: states and inputs from all terms and productions mixed together
  - 57 `switch` statements
  - 272 single-char `case` clauses
  - 2400+ SLOC
- Parser combinator style would have exposed the issue immediately, not 10 years after the same bug in Apache
  - CVE-2013-2028 vs. CVE-2002-0392



The First IEEE S&P Workshop: http://spw14.langsec.org/

LANGSEC WORKSHOP

Map: Internet Census 2012 / Carna


THANK YOU
WELCOME TO THE LANGSEC CONSPIRACY