A General Definition of Malware

(Talk at Nanyang Technological University)

Simon Kramer
(j.w.w. Julian C. Bradfield, U Edinburgh)

University of Tsukuba, Japan

January 15, 2010

Malware example and impact

Conficker **worm** detected in November 2008—still active—can cause a computer under the Windows operating system to become a component of a remote-controlled **botnet** against the user's will—on an infected computer, it causes a buffer overflow in which harmful excess code is executed by the operating system—the excess code downloads more code that hijacks the server services of the operating system, in order to update and spread the worm via the network—variant code inhibits also the security services of the operating system and connections to anti-malware websites. . . *affected European military systems*. . .

Worldwide malware-induced damage in 2006 **$**$13.3 \cdot 10^9$ [Computer Economics Inc., 2007]

Motivation, Goal, and Methodology

Motivation An open problem [FHZ06]: find a general definition of **malware** (= **mal**icious soft**ware**), e.g., botnets, rootkits, Trojan horses, viruses, worms, etc.

Goal Obtain a formal solution to the problem.

Methodology Formulation of the solution as a *single sentence* in a **computational modal fixpoint logic.**

Malware as harmful software

What is malware?

- Informally,

  $\boxed{\text{malware} = \text{malicious software}}$

  - Malicious intention is not generally directly observable!
  - How to distinguish *unawareness* (juvenile hacking, accidental anti-hacking) from *malice*?
  - **Users don't care:** all that matters is (harmful) *effect*, not (malicious) intention.
  - **Malice is immaterial!**
  - psychological "definition"
- Intuitively,

  $\boxed{\textbf{malware} = \textbf{harmful software}}$

  - Harmful effect *is* observable!
  - scientific definition

Harm as incorrectness

- **doing harm** = causing that

  actual behaviour $\neq$ **intended** behaviour
- **actuality** $-$ **intention** = **incorrectness**
- **defining principle** for malware:

  $\boxed{\textbf{causation of incorrectness}}$
- **harmful attack** = falsification of a necessary condition for correctness
- **formal systems engineering**
  - correctness *intention* must be specified
  - we don't care how:

    $\boxed{\mathrm{correct}(s)}$

Example: Sorting

- **Given:** a program s for sorting an array A of l integers
- **Sought:** a correctness definition for s
  - $\mathsf{Pre} := A : \mathtt{Array}_{l\in\mathbb{N}}(\mathbb{Z})$
  - $\mathsf{Post} := \forall(1 \leq i \leq l)\forall(1 \leq j \leq l)(i \leq j \rightarrow A[i] \leq A[j])$
  - $\boxed{\mathrm{correct}(s) \ \text{:iff} \ \vdash_{\mathrm{Hoare}} \mathsf{Pre}\,\{s\}\,\mathsf{Post}}$
- **Variations:** add necessary conditions (e.g., exact *algorithmic complexity*), stipulate *proof-carrying code*, etc.

Prerequisites

Theorem (Knaster-Tarski fixpoint theorem)

Let $\langle L, \leq \rangle$ designate a **complete** *lattice*[1] *and $f : L \to L$ a* **monotonic** *map*[2] *on L. Then,*

$$g := \bigvee \{\ a \mid a \in L \text{ and } a \leq f(a)\ \}$$

is the **greatest** *fixpoint of f, and, dually,*

$$l := \bigwedge \{\ a \mid a \in L \text{ and } f(a) \leq a\ \}$$

is the **least** *fixpoint of f.*

[1] $\bigvee S$ (lub) and $\bigwedge S$ (glb) exist for **arbitrary** $S \subseteq L$

[2] for all $a, b \in L$, if $a \leq b$ then $f(a) \leq f(b)$

Preliminaries

Definition (Damaging software)

A software system s **damages** a correct software system s' by definition if and only if s (directly or indirectly) causes incorrectness to s'. Formally,

$s \text{ damages } s'$	:iff	correct(s') and not correct($s(s')$)	directly
$s \text{ damages}^0\ s'$	:iff	$s \text{ damages } s'$	
$s \text{ damages}^{n+1}\ s'$	:iff	there is s'' s.t. not $s'' \text{ damages}^\circ\ s'$ and $s(s'') \text{ damages}^n\ s'$	indirectly
$s \text{ damages}^\circ\ s'$	:iff	$\bigcup_{n\in\mathbb{N}} s \text{ damages}^n\ s'$.	

Preliminaries

Definition (Repairing software)

A software system s **repairs** an incorrect software system s' by definition if and only if s (directly or indirectly) causes correctness to s'. Formally,

$s \text{ repairs } s'$	:iff	not correct(s') and correct($s(s')$)	directly
$s \text{ repairs}^0\ s'$	:iff	$s \text{ repairs } s'$	
$s \text{ repairs}^{n+1}\ s'$	:iff	there is s'' s.t. not $s'' \text{ repairs}^\circ\ s'$ and $s(s'') \text{ repairs}^n\ s'$	indirectly
$s \text{ repairs}^\circ\ s'$	:iff	$\bigcup_{n\in\mathbb{N}} s \text{ repairs}^n\ s'$.	

Malware Logic

Definition (MalLog)

Let $\mathcal{M}$ designate a countable set of propositional variables M, and

$$\Phi \ni \phi ::= M \mid \neg\phi \mid \phi \wedge \phi \mid \forall\mathbf{D}(\phi) \mid \forall\mathbf{R}(\phi) \mid \boldsymbol{\nu} M(\phi)$$

the language Φ of MalLog where all free occurrences of M in ϕ of $\boldsymbol{\nu} M(\phi)$ are assumed to occur within an even number of occurrences of $\neg$ to guarantee the existence of (greatest) fixpoints (expressed by $\boldsymbol{\nu} M(\phi)$) [BS07].

Malware Logic (continued)

Then, given the (or only some sub-) class $\mathcal{S}$ of software systems (not just pieces of software) s and an interpretation $[\![\cdot]\!] : \mathcal{M} \to 2^{\mathcal{S}}$ of propositional variables, the interpretation $\|\cdot\|_{[\![\cdot]\!]} : \Phi \to 2^{\mathcal{S}}$ of MalLog-propositions is:

$$\begin{aligned}
\|M\|_{[\![\cdot]\!]} &:= [\![M]\!] \\
\|\neg\phi\|_{[\![\cdot]\!]} &:= \mathcal{S} \setminus \|\phi\|_{[\![\cdot]\!]} \\
\|\phi \wedge \phi'\|_{[\![\cdot]\!]} &:= \|\phi\|_{[\![\cdot]\!]} \cap \|\phi'\|_{[\![\cdot]\!]} \\
\|\forall\mathbf{D}(\phi)\|_{[\![\cdot]\!]} &:= \{\ s \mid \text{for all } s', \text{ if } s \text{ damages}^\circ\ s' \text{ then } s' \in \|\phi\|_{[\![\cdot]\!]}\ \} \\
\|\forall\mathbf{R}(\phi)\|_{[\![\cdot]\!]} &:= \{\ s \mid \text{for all } s', \text{ if } s \text{ repairs}^\circ\ s' \text{ then } s' \in \|\phi\|_{[\![\cdot]\!]}\ \} \\
\|\boldsymbol{\nu} M(\phi)\|_{[\![\cdot]\!]} &:= \bigcup\{\ S \mid S \subseteq \|\phi\|_{[\![\cdot]\!]_{[M\mapsto S]}}\ \}
\end{aligned}$$

where $[\![\cdot]\!]_{[M\mapsto S]}$ maps M to S and otherwise agrees with $[\![\cdot]\!]$.

Malware Logic (continued)

Further, $\phi \vee \phi' := \neg(\neg\phi \wedge \neg\phi')$, $\top := \phi \vee \neg\phi$, $\bot := \neg\top$, $\phi \rightarrow \phi' := \neg\phi \vee \phi'$, $\phi \leftrightarrow \phi' := (\phi \rightarrow \phi') \wedge (\phi' \rightarrow \phi)$, and

$$\begin{aligned} \exists\mathbf{D}(\phi) &:= \neg\forall\mathbf{D}(\neg\phi) \\ \exists\mathbf{R}(\phi) &:= \neg\forall\mathbf{R}(\neg\phi) \\ \boldsymbol{\mu} M(\phi(M)) &:= \neg\boldsymbol{\nu} M(\neg\phi(\neg M)). \end{aligned}$$

Finally,

- for all $\phi \in \Phi$ and $s \in \mathcal{S}$, $s \models \phi$:iff $s \in \|\phi\|_{[\![\cdot]\!]}$
- $\models \phi$:iff for all $s \in \mathcal{S}$, $s \models \phi$
- for all $\phi, \phi' \in \Phi$,
  - $\phi \Rightarrow \phi'$:iff for all $s \in \mathcal{S}$, if $s \models \phi$ then $s \models \phi'$
  - $\phi \Leftrightarrow \phi'$:iff $\phi \Rightarrow \phi'$ and $\phi' \Rightarrow \phi$.

Basic properties of MalLog

Fact

1. *$\models \phi \rightarrow \phi'$ iff $\phi \Rightarrow \phi'$*
   (By expansion of the definitions.)
2. *$\models \phi \leftrightarrow \phi'$ iff $\phi \Leftrightarrow \phi'$*
3. *MalLog is a member of the family of μ-calculi over the modal system $\mathbf{K}_2$, which is characterised by the validities of propositional logic and the modal laws $\models \Box(\phi \rightarrow \phi') \rightarrow (\Box\phi \rightarrow \Box\phi')$ and "if $\models \phi$ then $\models \Box\phi$", where $\Box \in \{\forall\mathbf{D}, \forall\mathbf{R}\}$.*

Basic properties of MalLog (continued)

Corollary

1. *If* $\mathrm{damages}^\circ$ *and* $\mathrm{repairs}^\circ$ *are* **decidable** *on a given software systems domain then the satisfiability problem for MalLog, i.e., "Given $\phi \in \Phi$, is there $s \in \mathcal{S}$ s.t. $s \models \phi$?", (and thus also the model-checking problem, i.e., "Given $\phi \in \Phi$ and $s \in \mathcal{S}$, is it the case that $s \models \phi$?") is decidable.*
2. *MalLog is* **axiomatisable** *by the following Hilbert-style proof-system:*
   1. *the axioms/rules of the modal system* $\mathbf{K}$ *for each* $\forall\mathbf{D}$ *and* $\forall\mathbf{R}$
   2. *the axiom* $\overline{\phi(\boldsymbol{\mu} M(\phi(M))) \rightarrow \boldsymbol{\mu} M(\phi(M))}$
   3. *the rule* $\dfrac{\phi(\phi') \rightarrow \phi'}{\boldsymbol{\mu} M(\phi(M)) \rightarrow \phi'}$.

Defining malware

Definition (Malware)

A software system s is **malware** by definition if and only if s damages non-damaging software systems (the *civil population* so to say) or software systems that damage malware (the *anti-terror force* so to say). Formally,

$$\mathrm{mal}(s) \quad \text{:iff} \quad s \models \boldsymbol{\nu} M(\exists\mathbf{D}(\forall\mathbf{D}(M))).$$

An iterative paraphrase

- Everything is malware (better be safe than sorry)
- except for (throw **out** what is clearly safe) the following systems:

0. non-damaging systems (CP)
1. systems that damage only systems that damage CP (ATF1)
2. systems that damage only systems that damage ATF1 (ATF2)
3. systems that damage only systems that damage ATF2 (ATF3)
4. etc.

Deriving benware

Definition (Benware)

A software system s is **benware** by definition if and only if s is non-damaging or damages only software systems that damage benware. Formally,

$$\mathrm{ben}(s) \quad :\mathrm{iff} \quad s \models \mu M(\forall \mathbf{D}(\exists \mathbf{D}(M))).$$

An iterative paraphrase

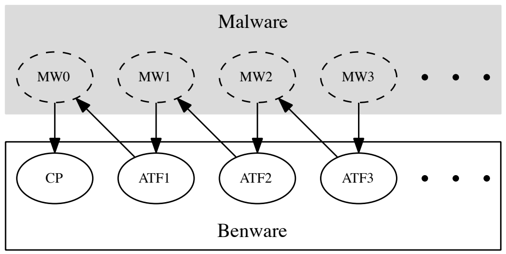
- Nothing is benware (again, better be safe than sorry)
- except for (throw **in** what is clearly safe) the following systems:

0. non-damaging systems (CP)
1. systems that damage only systems that damage CP (ATF1)
2. systems that damage only systems that damage ATF1 (ATF2)
3. systems that damage only systems that damage ATF2 (ATF3)
4. etc.

The Malware-versus-Benware **arms race**

Fact

$\mathrm{ben}(s)$ *if and only if not* $\mathrm{mal}(s)$



Good&Bad distinction induced by the existence of a population that is (perceived as) non-damaging

Anti-malware

Definition (Anti-malware)

A software system s is **anti-malware** by definition if and only if s damages no benware (safety)[3] and s neutralises[4] malware (effectiveness). Formally,

$$\begin{array}{lll}\mathrm{antimal}(s) & :\text{iff} & s \models \neg\exists\mathbf{D}(\mathrm{BEN}) \text{ and} \\ & & \text{there is } s' \text{ s.t. } \mathrm{mal}(s') \text{ and not } \mathrm{mal}(s(s'))\end{array}$$

where BEN $:= \mu M(\forall\mathbf{D}(\exists\mathbf{D}(M)))$.

[3] no friendly fire

[4] Damage is insufficient!

Medware

Definition (Medware)

A software system s is **medware** by definition if and only if s damages no benware (safety) and s repairs benware (effectiveness). Formally,

$$\mathrm{med}(s) \quad :\text{iff} \quad s \models \neg\exists\mathbf{D}(\mathrm{BEN}) \land \exists\mathbf{R}(\mathrm{BEN}).$$

Tasks, Tools, and Techniques for fighting Malware

Task	Tool	Technique
detection	satisfaction relation $\models$	**M**odel **C**hecking
comparison	language & bisimulation equivalence	**E**quivalence **C**hecking
classification	characteristic formulas	MC, EC

Malware Comparison (declarative)

Definition (Language equivalence)

For all $s_1, s_2 \in \mathcal{S}$,

- $s_1 \sqsubseteq_\Phi s_2$:iff for all $\phi \in \Phi$, if $s_1 \models \phi$ then $s_2 \models \phi$
- $s_1 \equiv_\Phi s_2$:iff $s_1 \sqsubseteq_\Phi s_2$ and $s_2 \sqsubseteq_\Phi s_1$.

Malware Comparison (operational)

Definition (Bisimulation equivalence)

- For all $s_1, s_2 \in \mathcal{S}$,
  - $s_1 \sqsubseteq s_2$:iff for all $s_1' \in \mathcal{S}$,
    1. if s_1 damages° s_1' then there is $s_2' \in \mathcal{S}$ s.t. s_2 damages° s_2'
    2. if s_1 repairs° s_1' then there is $s_2' \in \mathcal{S}$ s.t. s_2 repairs° s_2'.
- For all $S \subseteq \mathcal{S} \times \mathcal{S}$,

  $$\mathcal{O}_{\sqsubseteq}(S) := \{\ (s_1, s_2) \mid (s_1, s_2) \in S \text{ and } s_1 \sqsubseteq s_2 \text{ and } s_2 \sqsubseteq s_1\ \}.$$

- $\approx$:= the greatest fixpoint of (monotonic) $\mathcal{O}_{\sqsubseteq}$
  $= \bigcup\{\ S \mid S \subseteq \mathcal{O}_{\sqsubseteq}(S)\ \}$, by Knaster-Tarski

Malware Classification

Definition (Characteristic formula)

Let $S \subseteq \mathcal{S}$, $s \in S$, $D(S,s) := \{\ s' \mid s' \in S \text{ and } s \text{ damages}^{\circ}\ s'\ \}$, $R(S,s) := \{\ s' \mid s' \in S \text{ and } s \text{ repairs}^{\circ}\ s'\ \}$, and $M_s \in \mathcal{M}$. Then, the **characteristic formula** $\chi(s, S)$ of the software system s w.r.t. S is the solution of the equation system

$$M_s \stackrel{\nu}{=} \forall\mathbf{D}(\bigvee_{s' \in D(S,s)} M_{s'}) \wedge \forall\mathbf{R}(\bigvee_{s' \in R(S,s)} M_{s'}) \wedge [\bigwedge_{s' \in D(S,s)} \exists\mathbf{D}(M_{s'})] \wedge [\bigwedge_{s' \in R(S,s)} \exists\mathbf{R}(M_{s'})],$$

(where $\bigvee\emptyset := \bot$ and $\bigwedge\emptyset := \top$) obtained [BS07] by translating each equation $M^i \stackrel{\nu}{=} \psi^i(S)$ into a formula $\nu M^i(\psi^i(S))$ and recursively substituting these formulae for the corresponding free variables in the first formula $\nu M_s(\psi_s(S))$.

Characterisation result

Theorem

For all $s, s' \in \mathcal{S}$,

$$s \equiv_{\Phi} s' \quad \textit{iff} \quad s \approx s' \quad \textit{iff} \quad s \models \chi(s', \mathcal{S}).$$

Assessment

Our approach:

1. malware-versus-benware arms race confined to formal systems engineering
2. malware detection $\rightsquigarrow$ automated systems verification
3. system security $\rightsquigarrow$ system correctness
4. generic (predicate correct is a *plug-in*)
5. **hacker-safe:**

   $\boxed{\text{no recipe for malware construction derivable}}$

Related work

About viruses only, not hacker-safe (constructive):

1. Adleman: Gödel-numberings [Adl88]
2. Cohen: Turing-machines [Coh87]
3. Bonfante et al.: Kleene Recursion Theorem [BKM06]

Future work

Refinements:

- add **time** (temporal modalities): malware *evolution*
- add **measure:** *degrees* of damage, malware *cost*

Bibliography

F. Cohen.
Computer viruses: Theory and experiments.
Journal of Computers & Security, 6, 1987.

L. Adleman.
An abstract theory of computer viruses.
In *Proceedings of CRYPTO'88*, volume 403 of *LNCS*, 1988.

G. Bonfante, M. Kaczmarek, and J.-Y. Marion.
On abstract computer virology from a recursion theoretic perspective.
Journal in Computer Virology, 1(3–4), 2006.

Bibliography

E. Filiol, M. Helenius, and S. Zanero.
Open problems in computer virology.
Journal in Computer Virology, 1(3–4), 2006.

S. Kramer and J.C. Bradfield.
A general definition of malware.
Journal in Computer Virology, Online First, 2009.
`http://dx.doi.org/10.1007/s11416-009-0137-1`

J.C. Bradfield and C. Stirling.
Handbook of Modal Logic, volume 3 of *Studies in Logic and Practical Reasoning*, chapter Modal Mu-Calculi.
Elsevier, 2007.

質問

質問がありますか。

Contact

Email:
simon.kramer@a3.epfl.ch

Homepage:
http://www.cipher.risk.tsukuba.ac.jp/~kramer/