A Policy-Based Vulnerability Analysis Framework

Sophie J. Engle
sjengle@ucdavis.edu

Framework Goals

Build a repeatable and practical framework for vulnerability analysis

Framework Goals

Build a repeatable and practical framework for vulnerability analysis

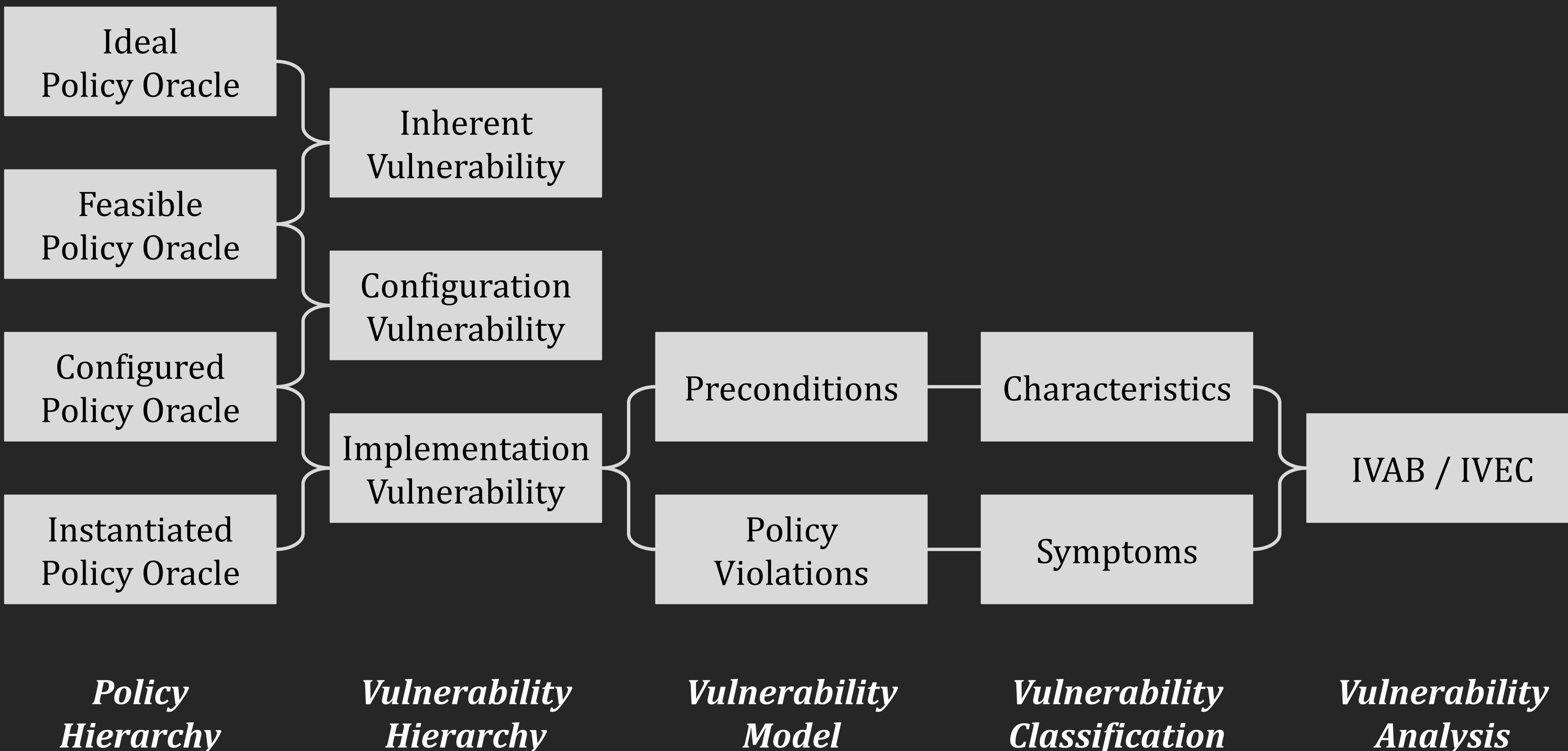
– Theoretical foundation

Framework Goals

Build a repeatable and practical framework for vulnerability analysis

- Theoretical foundation
- Practical levels of abstraction

Terminology Overview



Talk Outline

- Section 1: Security Policy
- Section 2: Vulnerability Hierarchy
- Section 3: Vulnerability Model
- Section 4: Vulnerability Classification
- Section 5: Vulnerability Analysis

Security Policy

Section 1

Terminology

- Policy Event
  - E = (*<u>s</u>ubject*, *<u>o</u>bject*, *<u>a</u>ction*, *<u>b</u>oolean condition*)
- Global Policy Event Space
  - Universe of policy events $\mathbb{E} = \mathbb{S} \times \mathbb{O} \times \mathbb{A} \times \mathbb{B}$
- Policy Oracle
  - Oracle function $\mathcal{P}(\mathrm{E}) = \{ \text{yes, no, unknown} \}$

Policy Hierarchy

- Ideal Policy Oracle
  - Which policy events *should be authorized* (ideally)

$\mathcal{P}_{id}$(Xander, control room, enter, true) = yes

Policy Hierarchy

- Ideal Policy Oracle
  - Which policy events *should be authorized* (ideally)
- Feasible Policy Oracle
  - Which policy events *are authorized* (realistically)

$$\mathcal{P}_{fe}(\text{ bid:14, room:21, enter, true }) = \text{yes}$$

Policy Hierarchy

- Ideal Policy Oracle
  - Which policy events *should be authorized* (ideally)
- Feasible Policy Oracle
  - Which policy events *are authorized* (realistically)
- Configured Policy Oracle
  - Which policy events *are allowed* (by configuration)

$$\mathcal{P}_{co}(\text{ bid:14, room:21, enter, true }) = \text{no}$$

Policy Hierarchy

- Ideal Policy Oracle
  - Which policy events *should be authorized* (ideally)
- Feasible Policy Oracle
  - Which policy events *are authorized* (realistically)
- Configured Policy Oracle
  - Which policy events *are allowed* (by configuration)
- Instantiated Policy Oracle
  - Which policy events *are possible* (by implementation)

$\mathcal{P}_{\text{in}}$(bid:14, room:21, enter, true) = yes

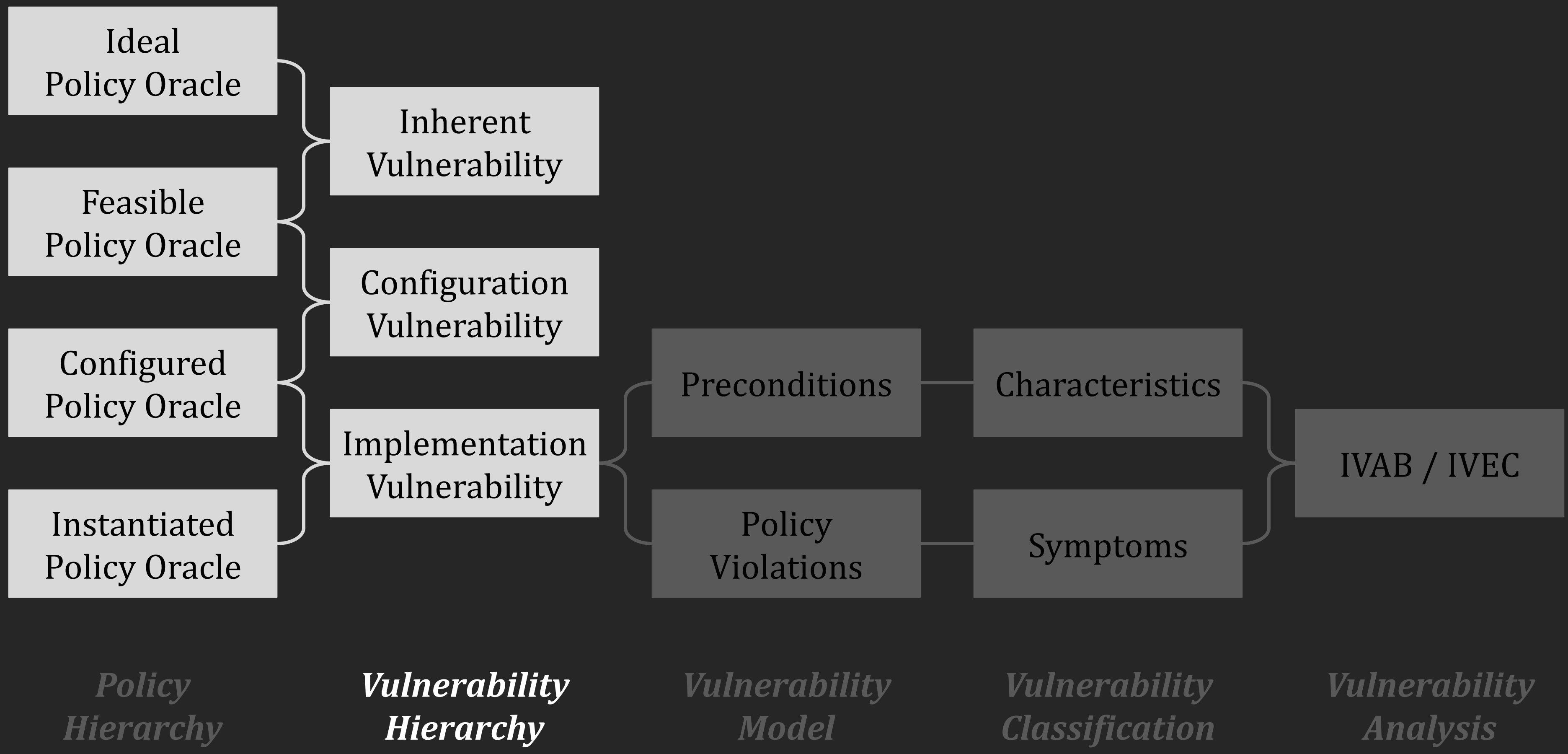
Policy Hierarchy

- Policy violations occur between oracles
  - $\mathcal{P}_{id}$(Xander, control room, enter, true) = yes
  - $\mathcal{P}_{fe}$(bid:14, room:21, enter, true) = yes
  - $\mathcal{P}_{co}$(bid:14, room:21, enter, true) = no
  - $\mathcal{P}_{in}$(bid:14, room:21, enter, true) = yes

Vulnerability Hierarchy

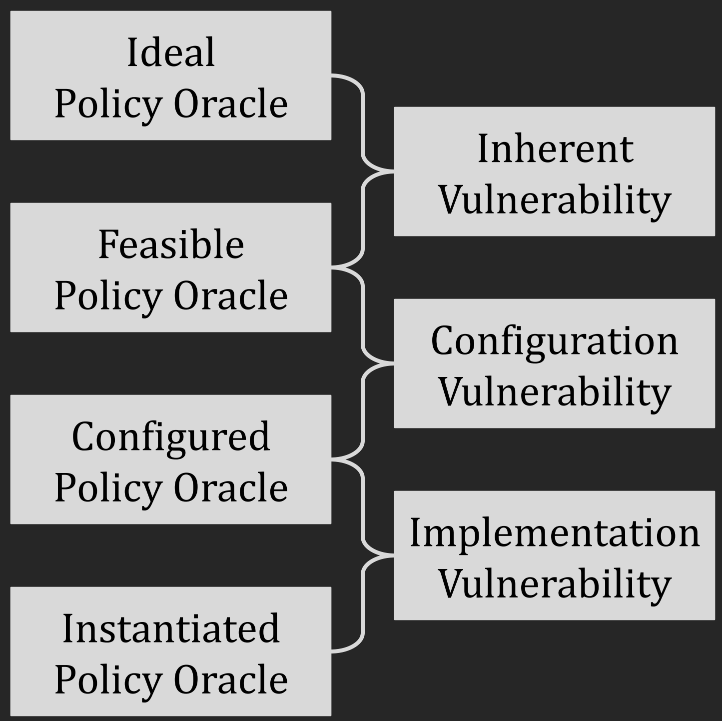
Section 2

Terminology Overview

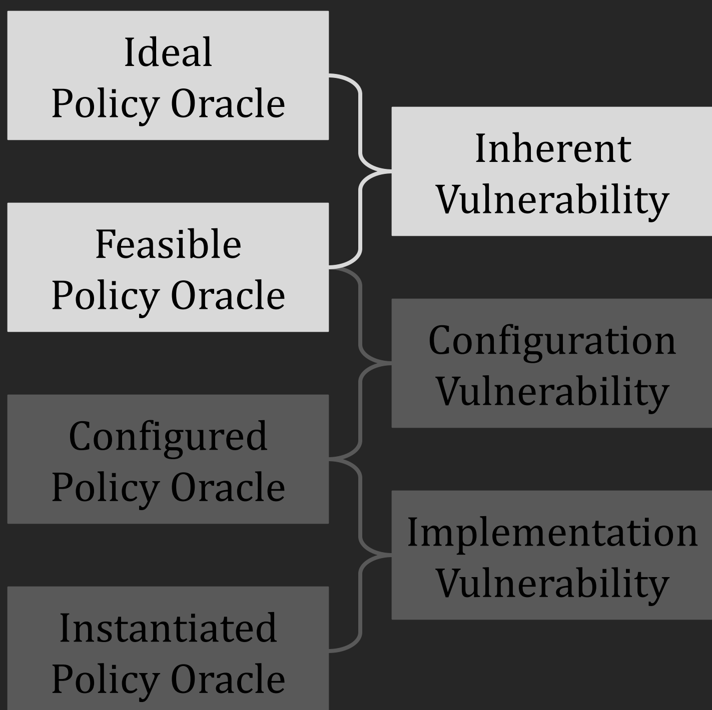


Vulnerability Hierarchy

- A *vulnerability* is the set of conditions that enable an unequivocal policy violation.

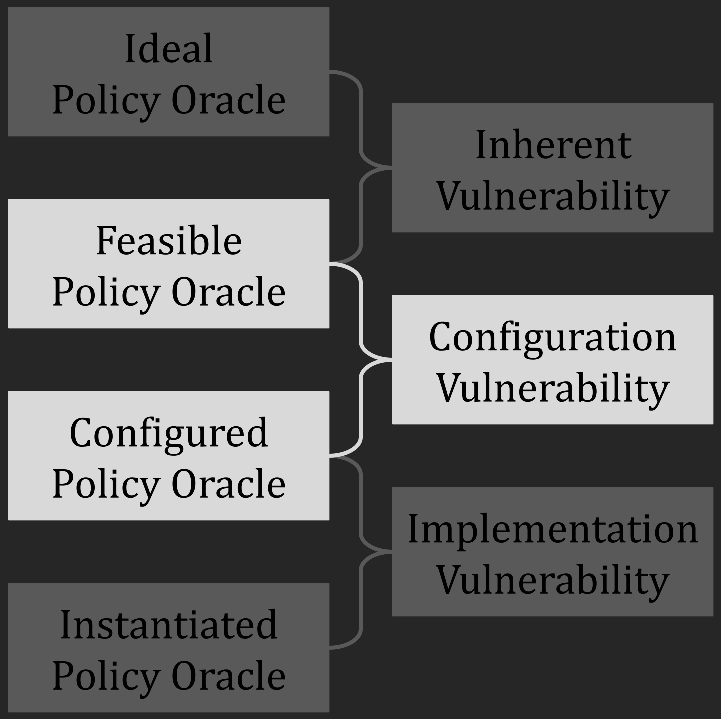


Inherent Vulnerabilities



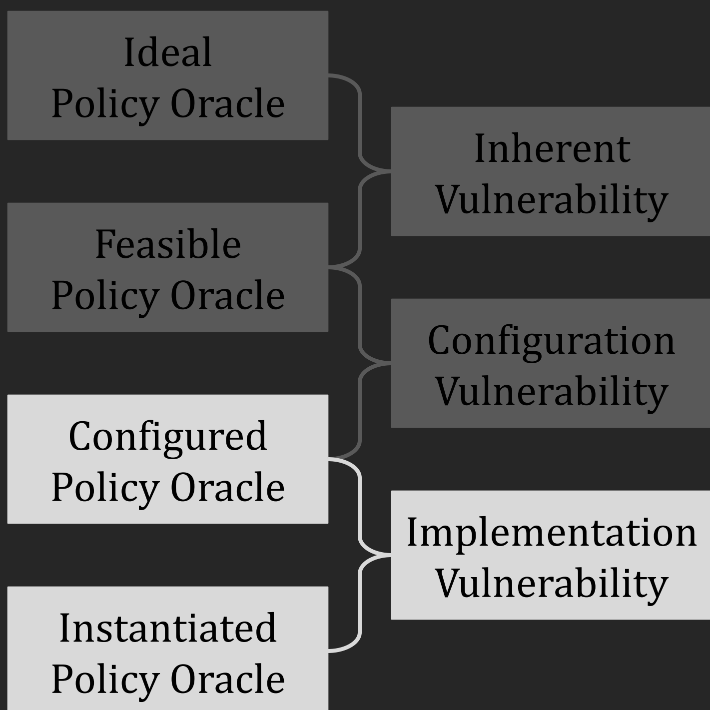
- Result of intentional compromises
- Indicates where functionality, configuration, manageability, or usability may be improved

Configuration Vulnerabilities



- Indicates that the policy as configured is incorrect
- Caused by difficult to configure or maintain security mechanisms, or poorly articulated policies

Implementation Vulnerabilities

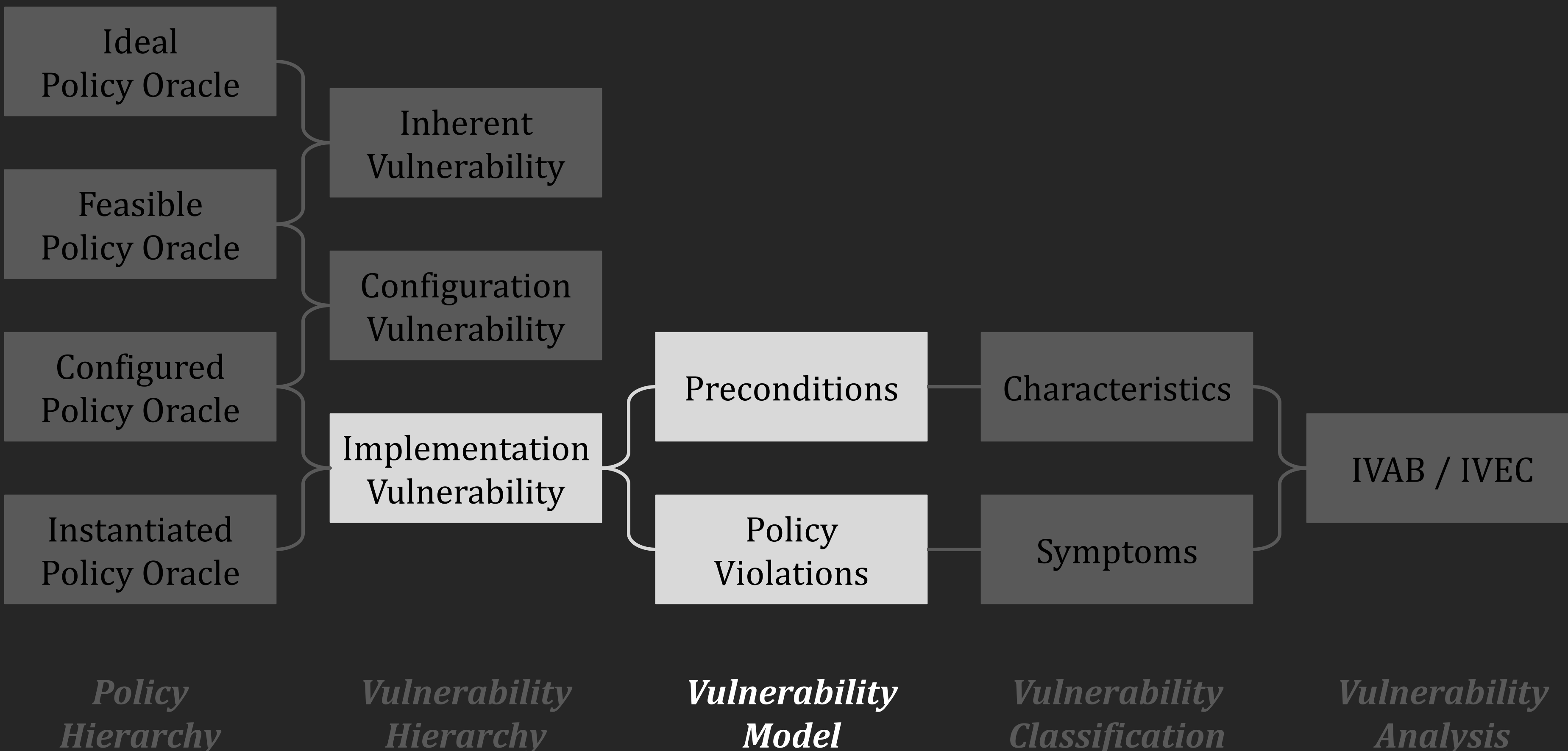


- Captures the traditional notion of a vulnerability
- Indicates that the mechanism’s implementation does not properly enforce the policy

Vulnerability Model

Section 3

Terminology Overview



Terminology

- Security Policy
  - Traditionally defined as a *partition of states*
  - Instead define as a *language of configurations*

Example: State q_i is authorized if w is on the tape.

- Policy as a partition:
  - Must design TM and split q_i into two states
- Policy as a configuration:
  - $\{ uq_iv : u \circ v \equiv w \}$

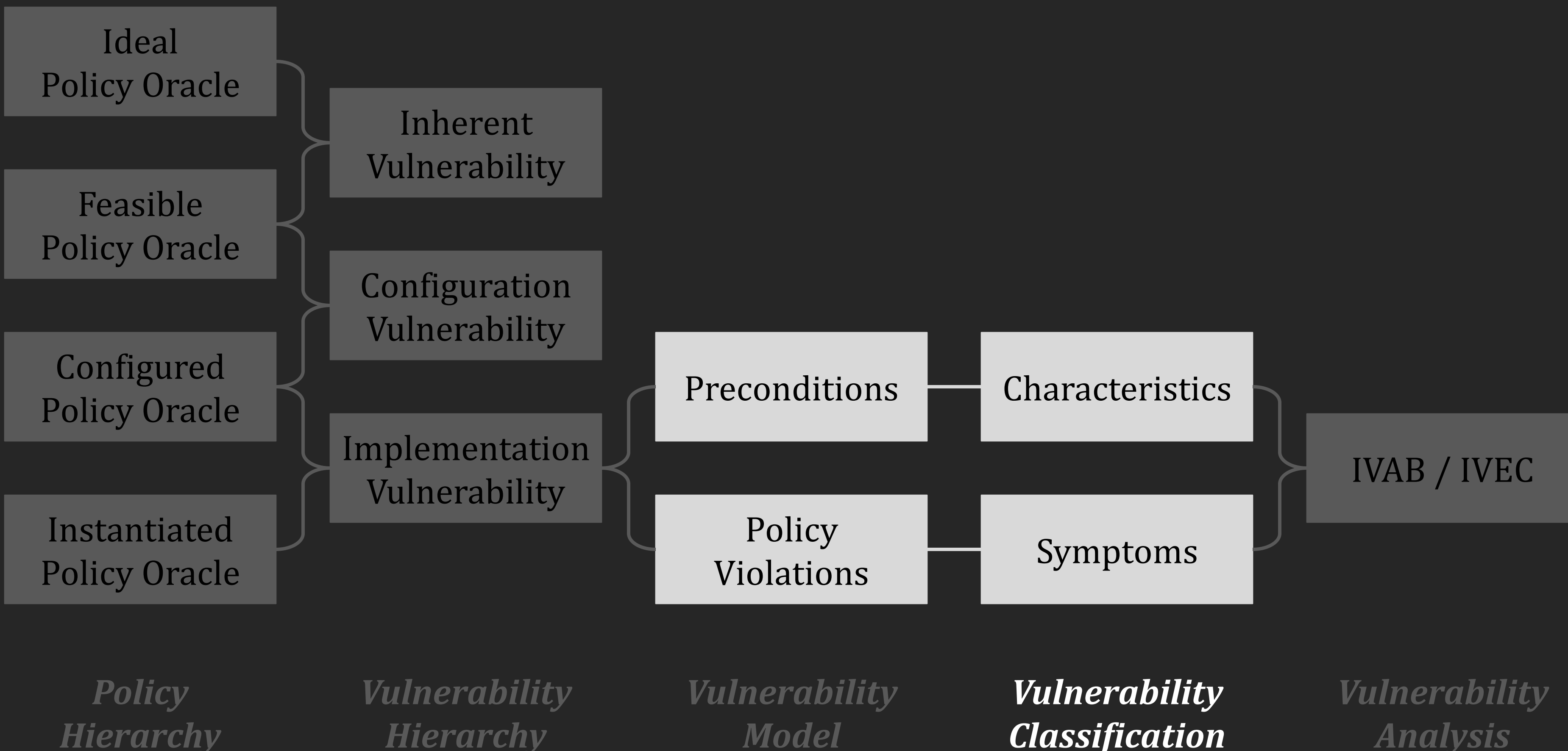
Terminology

- Policy Violation
  - A configuration that is either valid but unauthorized, or authorized but invalid
- Precondition
  - A language of configurations describing trace prior to the policy violation
- Implementation Vulnerability
  - A policy violation and its associated preconditions

Vulnerability Classification

Section 4

Terminology Overview



Perfect Knowledge Assumption

- Why is our formal model impractical?
  - Do not have the formal specification
  - Do not have access to computation trace
  - Do not have an explicit set of systems

Perfect Knowledge Assumption

- Why is our formal model impractical?
  - Do not have the formal specification
  - Do not have access to computation trace
  - Do not have an explicit set of systems

- End result:
  - Defining a precondition is impractical
  - Defining a policy violation is impractical
  - *Defining an implementation vulnerability is impractical*

Vulnerability Abstraction

- Characteristic
  - A set of similar known preconditions
  - Example: X_{null} = { t : t contains the null character \0 }

- Symptom
  - A set of similar known policy violations
  - Example: Y_{incr} = { u : VALID(M) \ L(P) }
    i.e. u is a valid configuration, but not authorized by policy

Vulnerability Abstraction

- Implementation Vulnerability: $V = (U, T)$
  - T is the set of policy violations
  - U is the set of associated preconditions

Vulnerability Abstraction

- Implementation Vulnerability: $V = (U, T)$
  - T is the set of policy violations
  - U is the set of associated preconditions

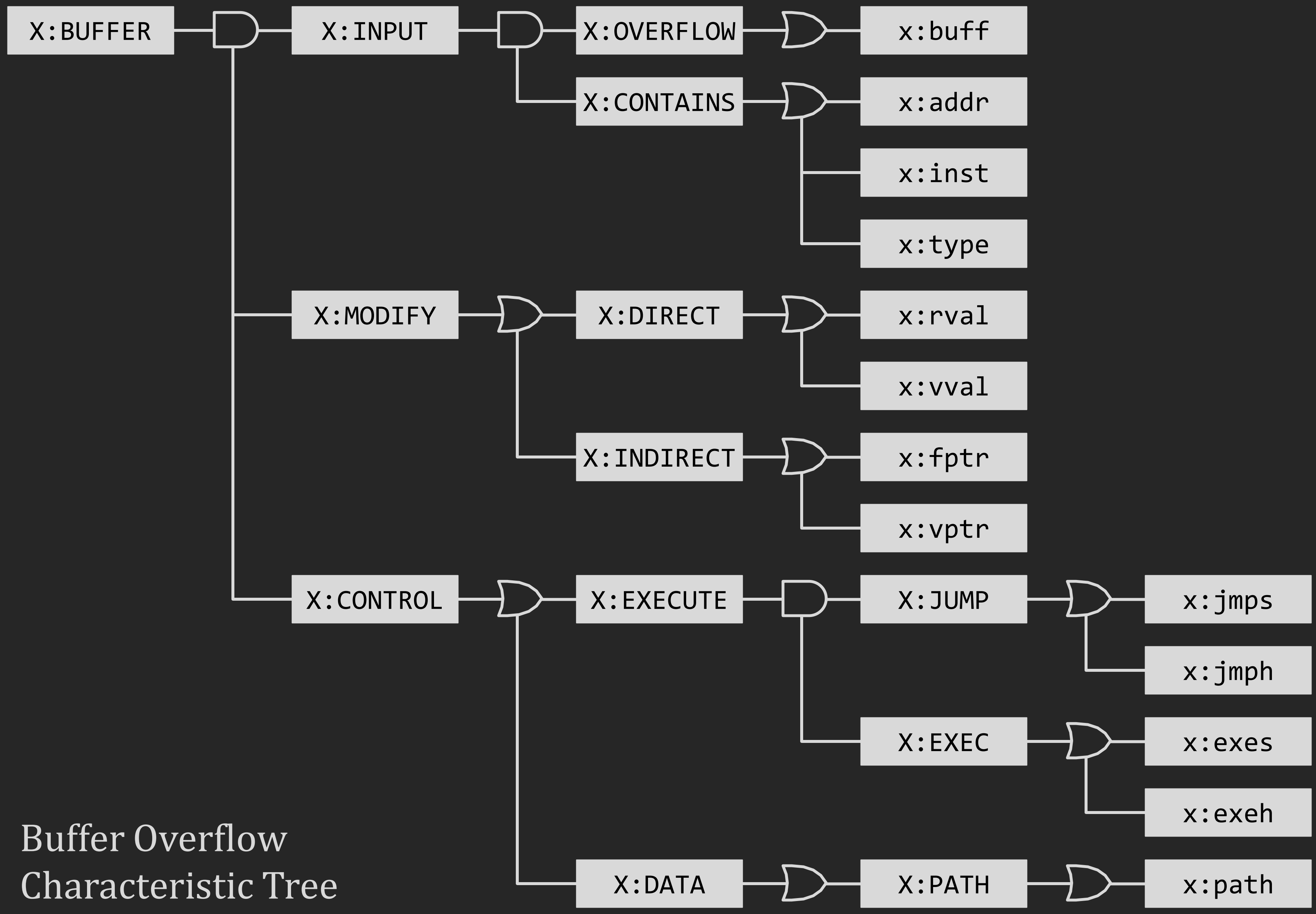
- Vulnerability Abstraction (IVAB): $Z = (X, Y)$
  - X is the basic characteristic set for U
  - Y is the basic symptom set for T

Vulnerability Abstraction

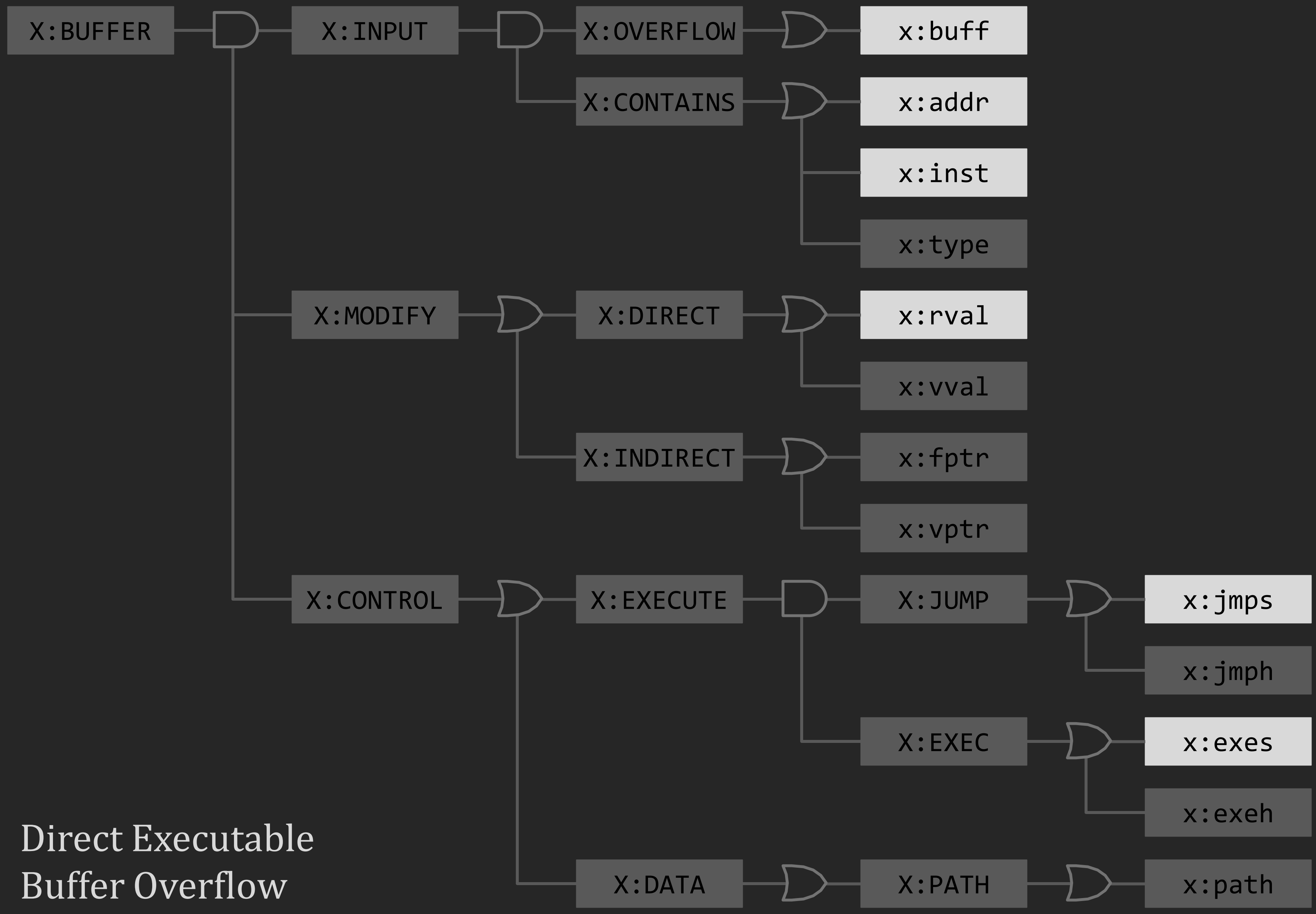
- Implementation Vulnerability: $V = (U, T)$
  - T is the set of policy violations
  - U is the set of associated preconditions
- Vulnerability Abstraction (IVAB): $Z = (X, Y)$
  - X is the basic characteristic set for U
  - Y is the basic symptom set for T
- Equivalence Class (IVEC): $Z = (X, Y)$
  - The set of equivalent IVABs

Vulnerability Classification

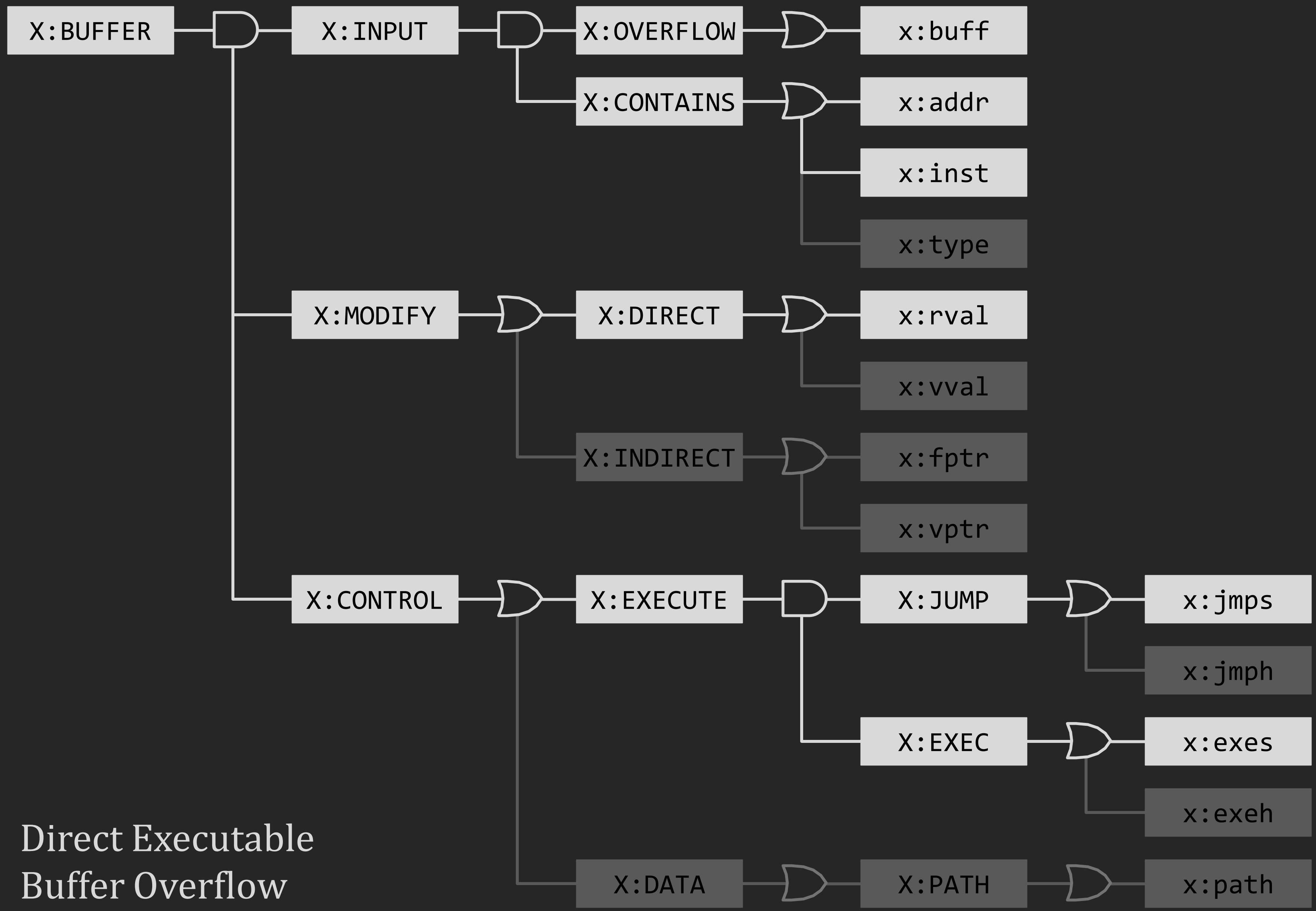
- Master Classification Tree
  - Characteristic Classification Tree
  - Symptom Classification Tree
- Vulnerability Classification Tree



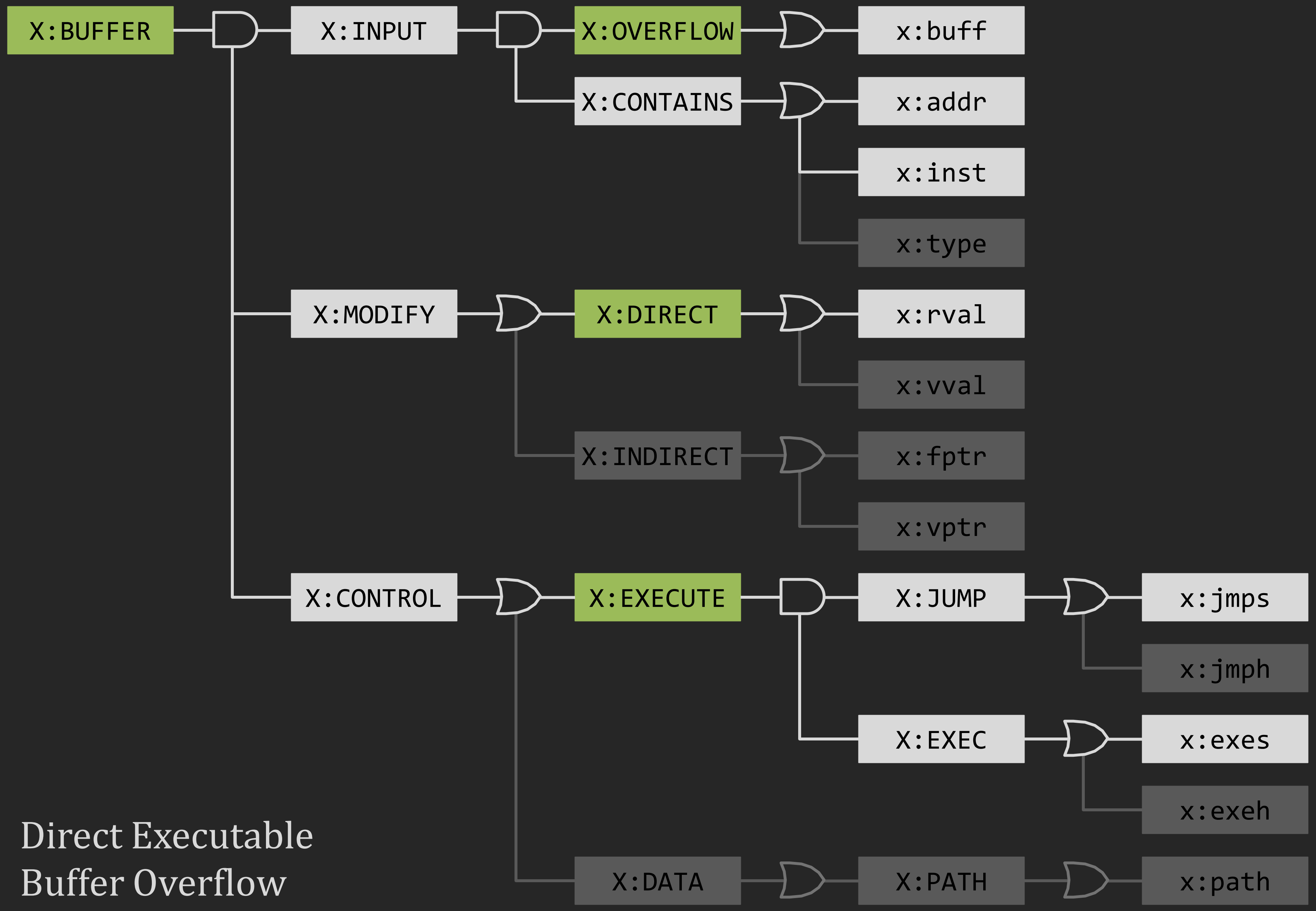
Buffer Overflow Characteristic Tree



Direct Executable
Buffer Overflow



Direct Executable
Buffer Overflow

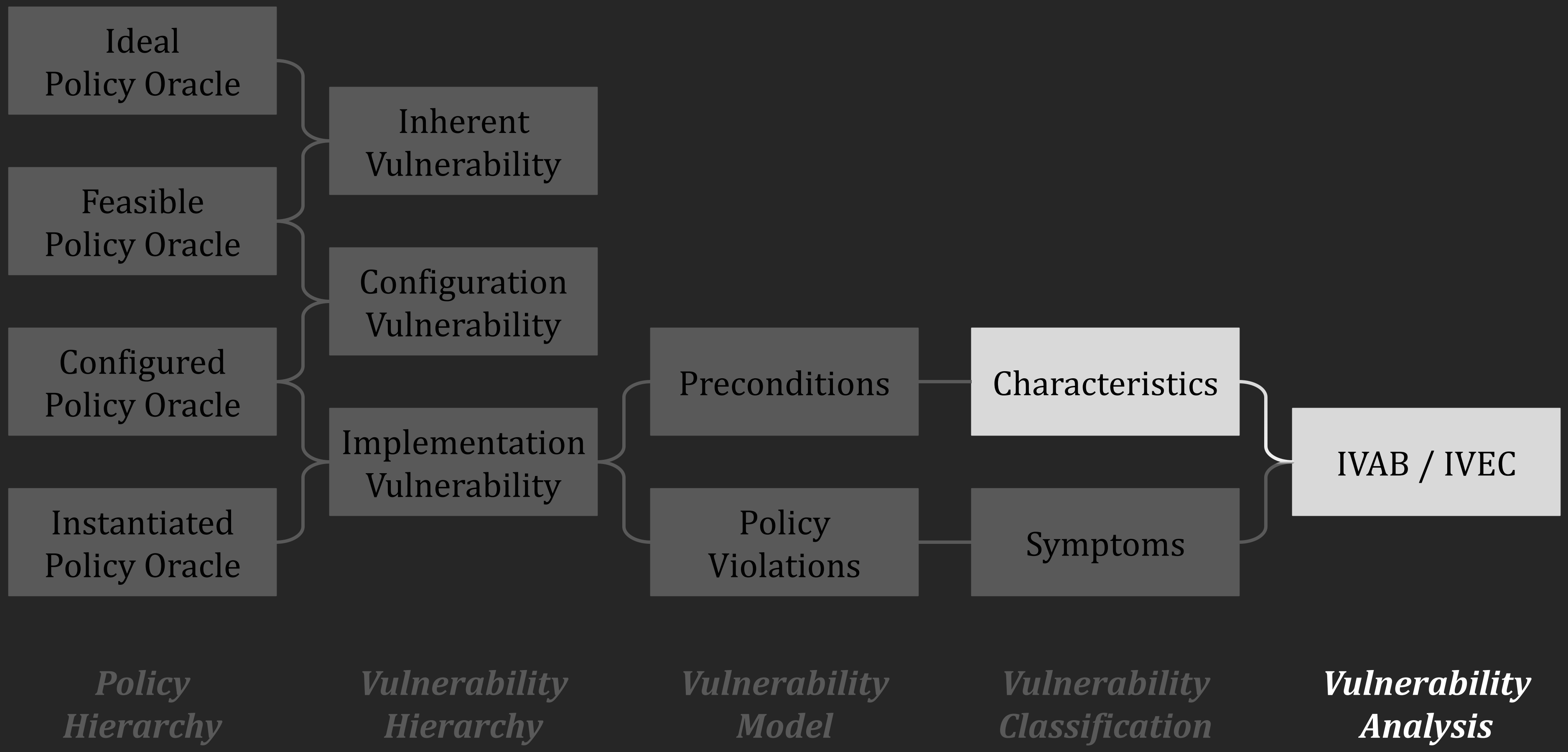


Direct Executable Buffer Overflow

Vulnerability Analysis

Section 5

Terminology Overview



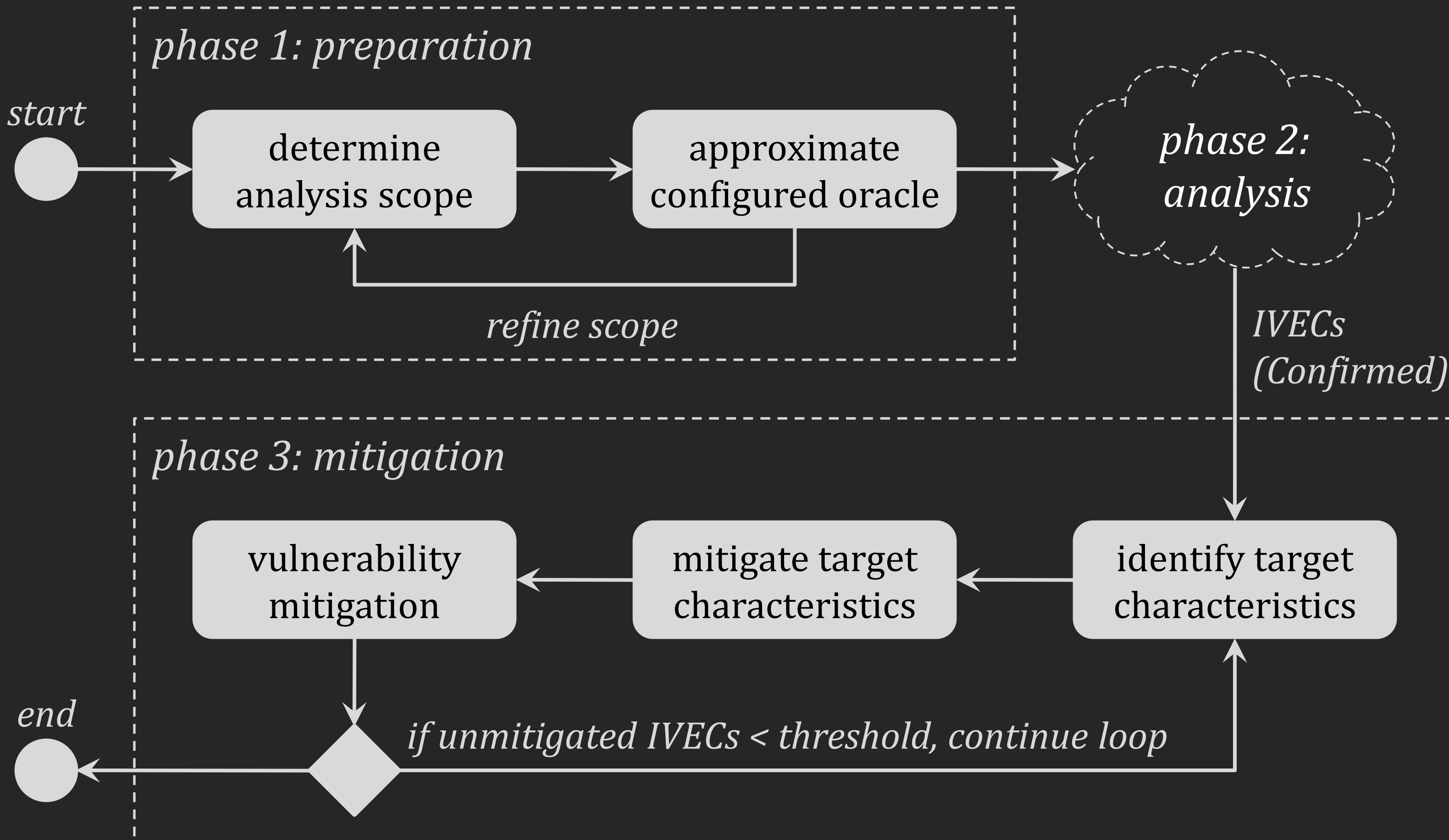
Analysis Goals

- Shift focus from *if* a system is secure to *when* a system is secure
- Locate and mitigate implementation vulnerability (equivalence classes) via characteristic-based analysis

Analysis Overview

- Phase 1: Preparation
  - Define global policy event space
  - Approximate configured oracle
- Phase 2: Analysis
  - Approximate instantiated oracle
  - Identify confirmed IVECs and characteristics
- Phase 3: Mitigation
  - Identify target characteristics
  - Disable target characteristics

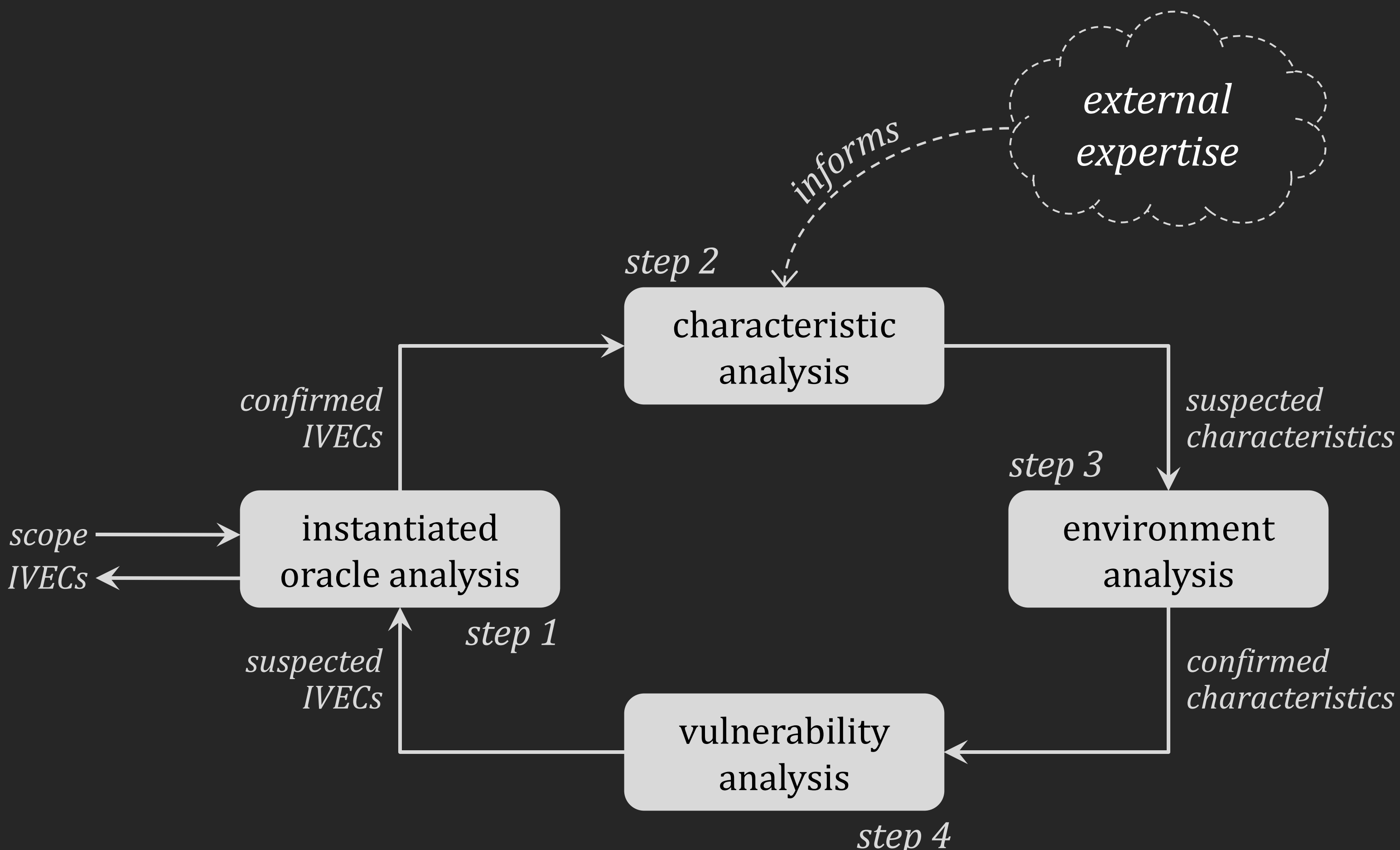
Analysis Overview



Phase 2 Analysis

- Characteristic Analysis
  - Develops set of suspected characteristics
- Environment Analysis
  - Determines if suspected characteristics exist
- Vulnerability Analysis
  - Develops set of suspected IVECs
- Instantiated Oracle Analysis
  - Determines if suspected IVECs exist

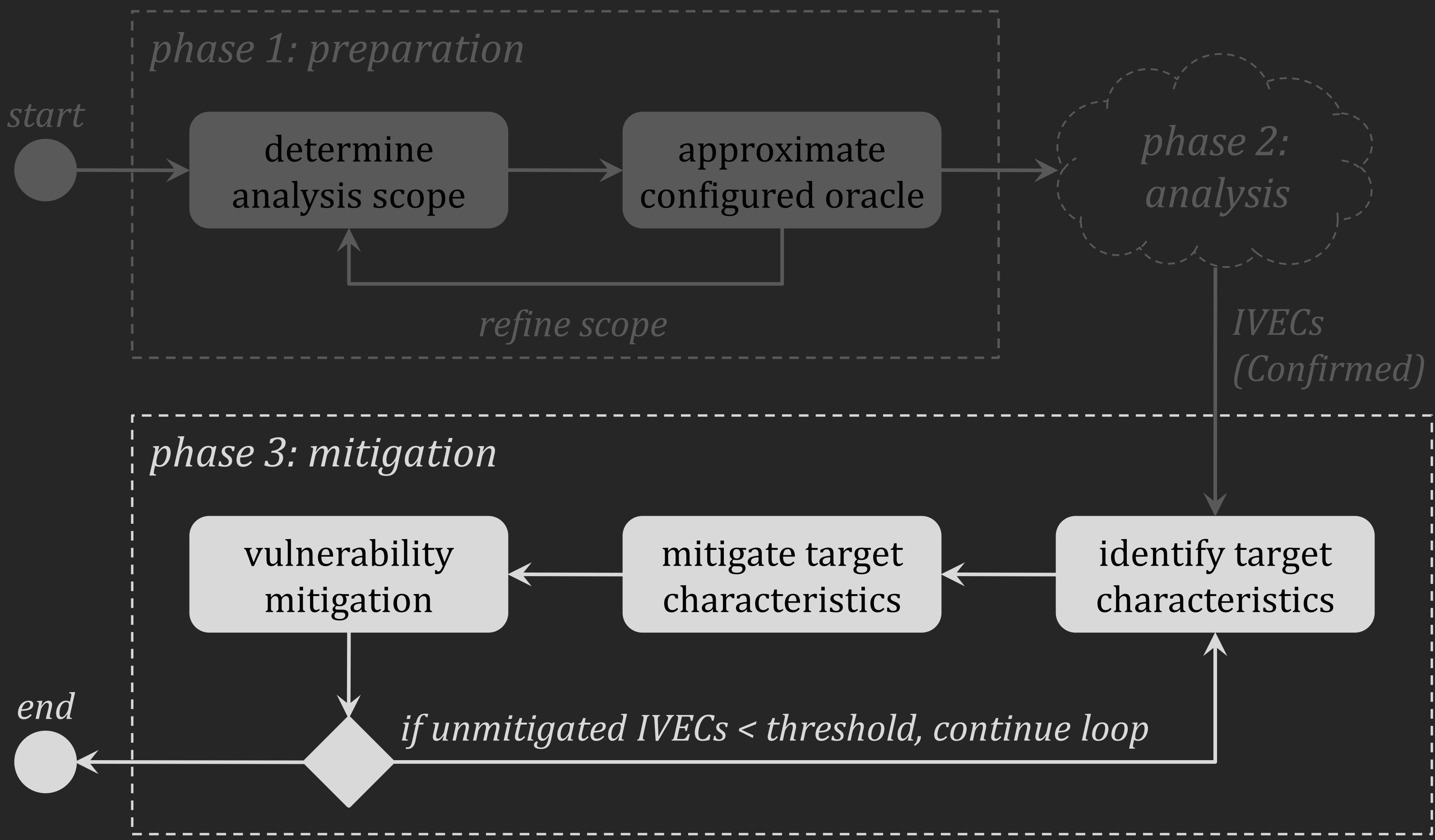
Phase 2 Overview



Phase 3 Mitigation

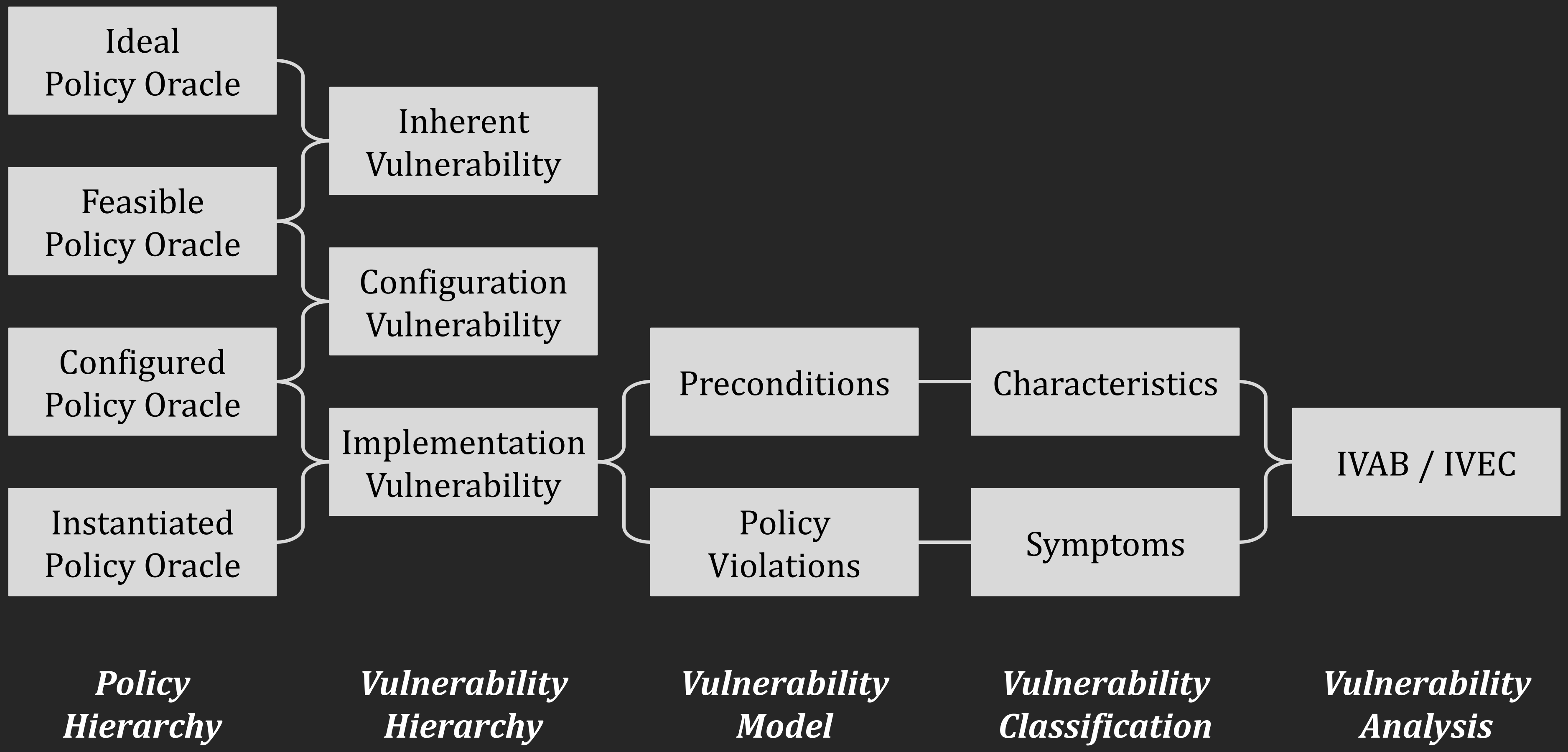
- Identify target characteristics
  - Frequent, i.e. associated with most IVECS
  - Dangerous, i.e. associated with worst symptoms
- Disable target characteristics
  - Some may be impossible or infeasible to fully disable
- Mitigate vulnerabilities
  - Compare confirmed IVECs with disabled characteristics
  - Update set of confirmed IVECs

Phase 3 Overview

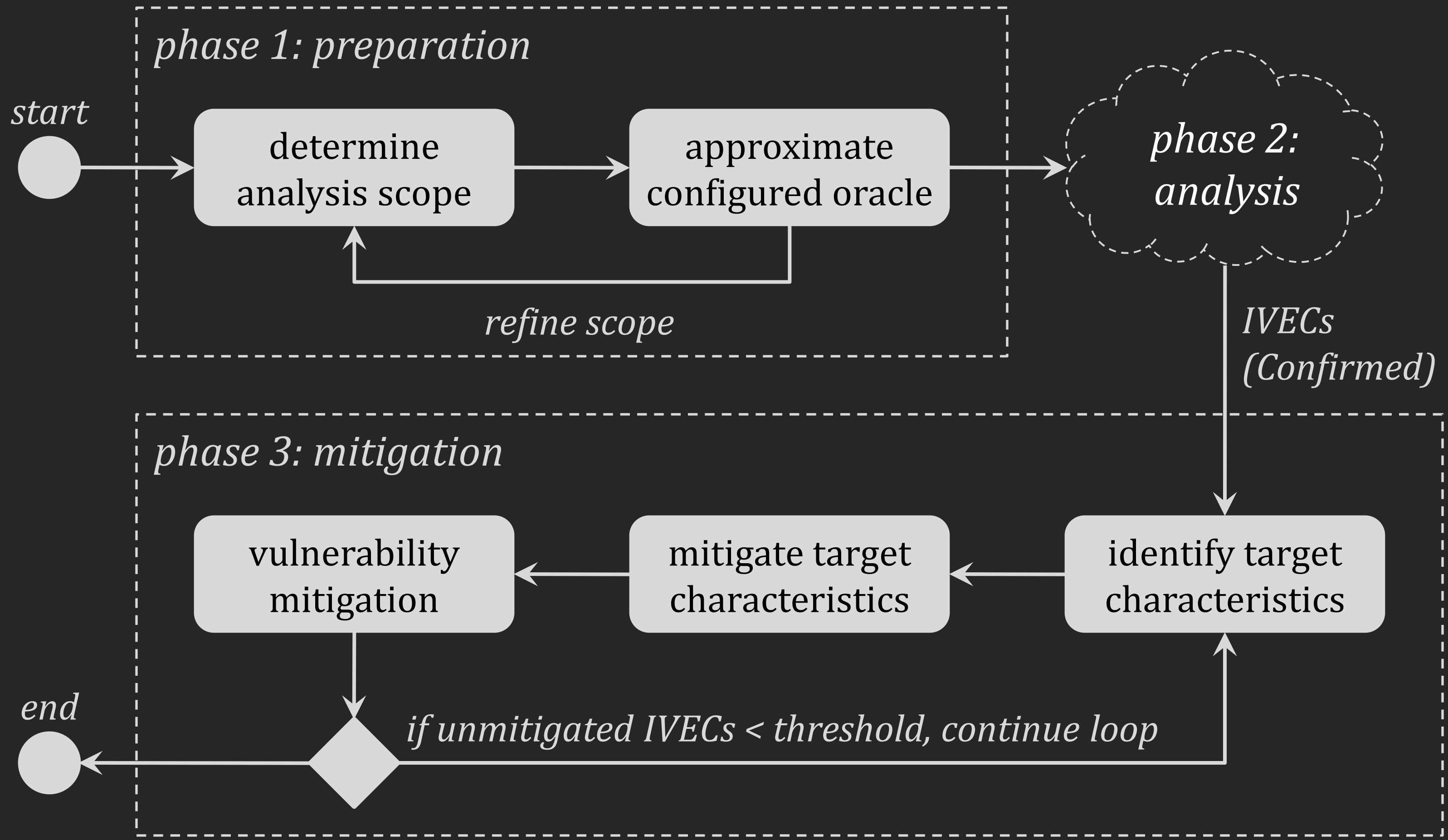


Conclusion

Terminology Recap



Framework Recap



Contributions

- Policy-Based Vulnerability Hierarchy
  - Can incorporate both security procedures and security mechanisms
  - Captures high-level and low-level vulnerabilities
- Formal Implementation Vulnerability Model
  - Policy as a language of configurations, instead of a partition of states
  - Theoretical foundation for classification scheme

Contributions

- Characteristic-Based Vulnerability Classification
  - Makes "perfect knowledge assumption" explicit
  - Provides reversible layers of abstraction

- Policy-Based Vulnerability Analysis Framework
  - Capable of repeatable vulnerability analysis results
  - Practical for stable, small-scale environments

Future Work

- Theoretical Results
  - Decidability of different security problems
- Vulnerability Database
  - Characteristic-based classification
  - Classification versus clustering
- Extended Case Study
  - Hypothetical electronic voting environment

Extended Case Study

- Four Analysis Teams
  - Environment: *Develops hypothetical environment*
  - Alpha: *Performs analysis using framework*
  - Beta: *Performs analysis using framework*
  - Control: *Performs ad-hoc analysis*

- Compare Results
  - Number of vulnerabilities found
  - Consistency of results across teams

Questions?

General Information

- Dissertation:
  - Sophie Engle, A Policy-Based Vulnerability Analysis Framework, Ph.D. Dissertation, Technical Report CSE-2010-06, Department of Computer Science, University of California, Davis, 2010.
- Committee:
  - Professor Matt Bishop (Chair)
  - Professor S. Felix Wu
  - Professor Karl Levitt
  - Professor Sean Peisert

Selected References

- Vulnerability Analysis: An Extended Abstract
  - Matt Bishop. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 1999, pages 125–136.
- We Have Met the Enemy and He is Us
  - Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen, and Carrie Gates. In *Proceedings of the 2008 New Security Paradigms Workshop (NSPW)*, September 2008, pages 1–12.
- A Taxonomy of Buffer Overflow Preconditions
  - Matt Bishop, Damien Howard, Sophie Engle, and Sean Whalen. *Technical Report CSE-2010-01*, Department of Computer Science, University of California, Davis, 2010.
- The Unifying Policy Hierarchy Model
  - Adam Carlson. *Master's Thesis*, Department of Computer Science, University of California, Davis, June 2006.
- Protocol Vulnerability Analysis
  - Sean Whalen, Sophie Engle, and Matt Bishop. *Technical Report CSE-2005-04*, Department of Computer Science, University of California, Davis, 2005.

Contact Information

Sophie Engle
sjengle@ucdavis.edu

Insider Threat Case Study

Supplemental Slides

Insider Threat Case Study

- Demonstrates vulnerability analysis using the Policy-Based Vulnerability Hierarchy
- Insider threat exists whenever:
  - Someone has more privileges at a lower policy level than at a higher policy level
  - The "*insiderness*" captures number of extra privileges
- Focus on identifying *potential for misuse* of privileges, not *potential for abuse* of any particular user

Insider Threat Case Study

- Two Primary Phases:
  - Inherent vulnerability analysis, such that $\mathcal{P}_{\mathrm{fe}}(E)$ = yes and $\mathcal{P}_{\mathrm{id}}$ (E) = no
  - Absolute vulnerability analysis, such that $\mathcal{P}_{\mathrm{in}}(E)$ = yes and $\mathcal{P}_{\mathrm{id}}$ (E) = no
- See dissertation for details

Electronic Voting Case Study

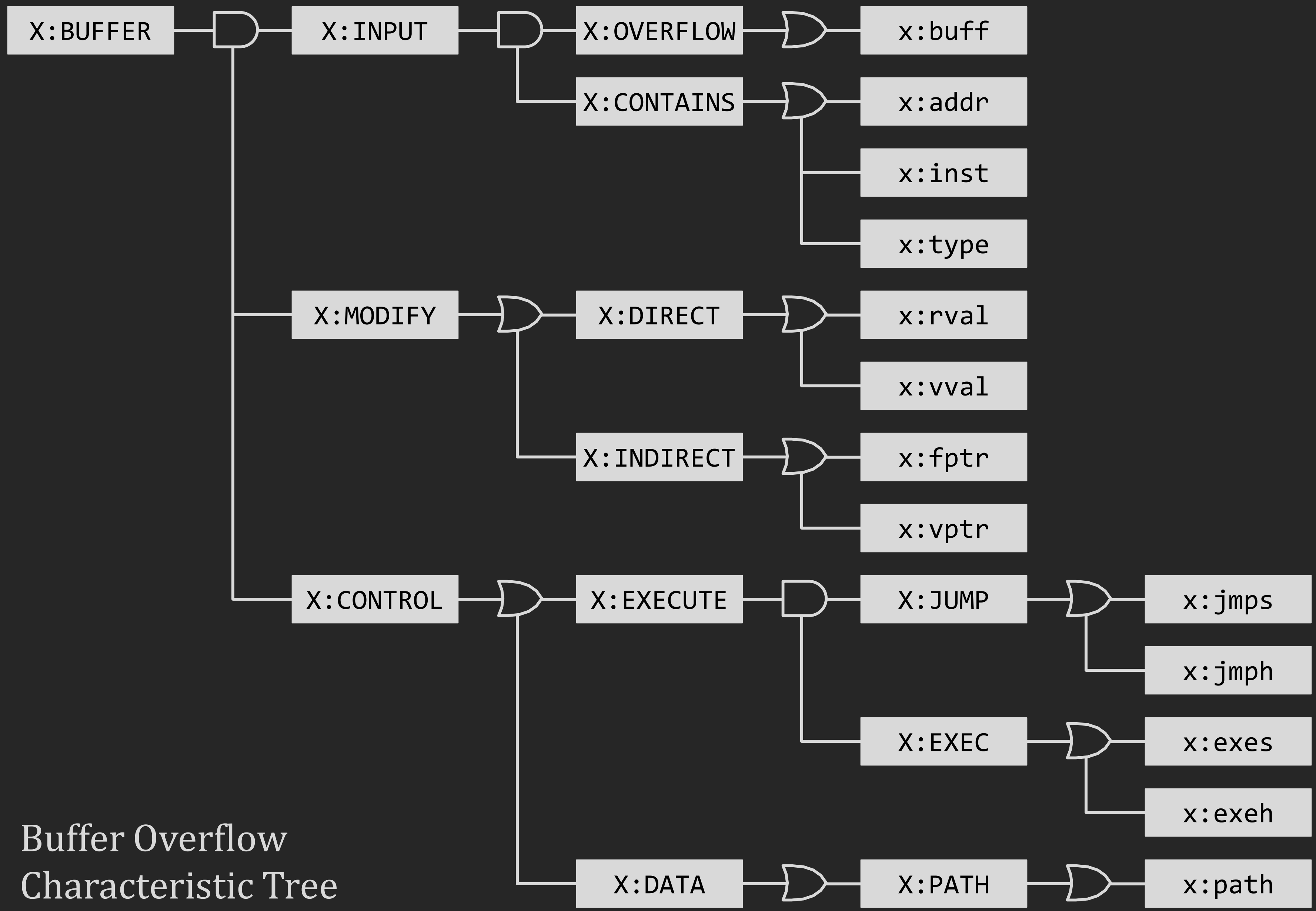
Supplemental Slides

Electronic Voting Case Study

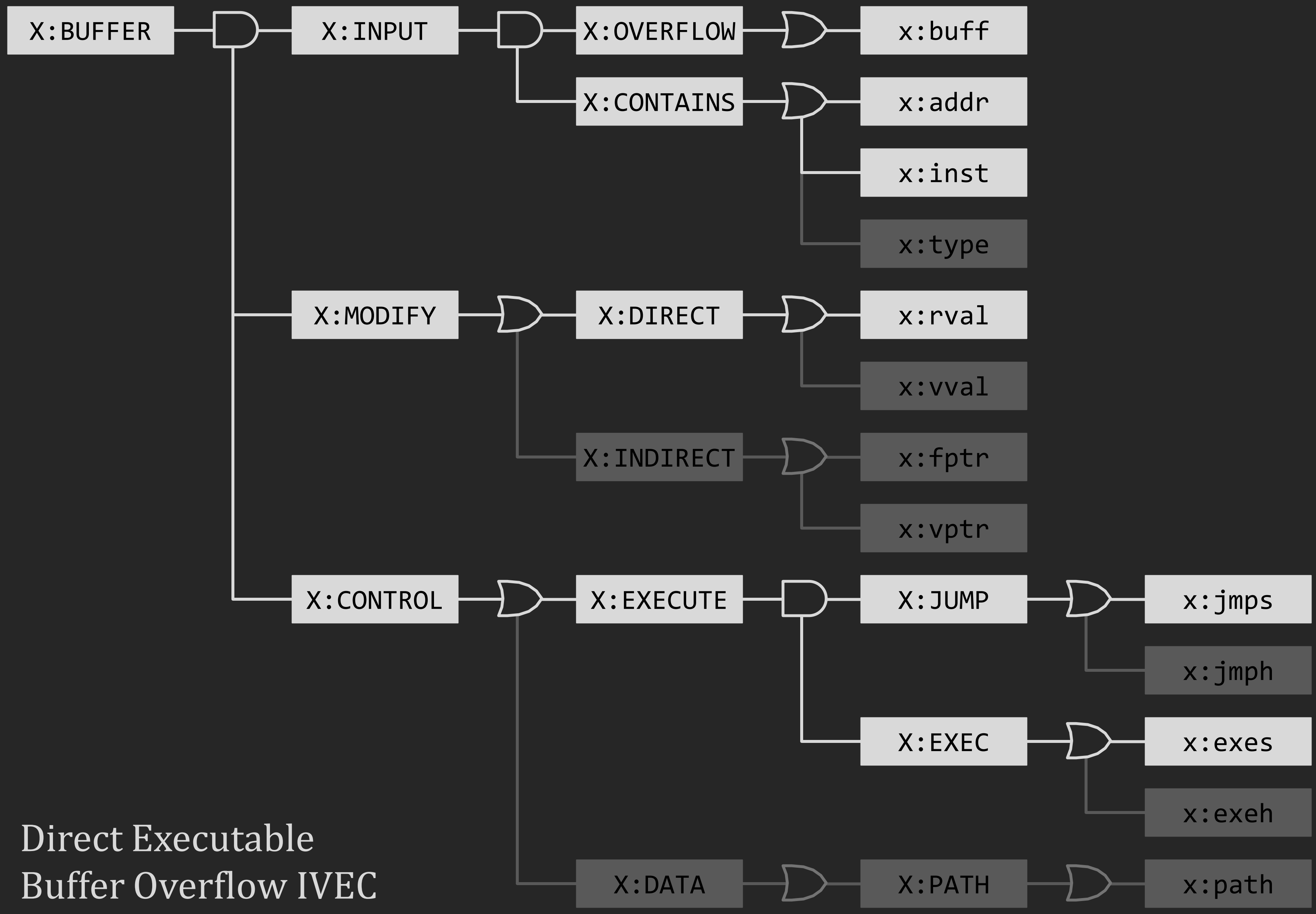
- Demonstrates the Policy-Based Vulnerability Analysis Framework
- Target Environment:
  - Electronic voting setup for a single precinct
  - Ideal due to precise set of systems and procedures
- See dissertation for details

Buffer Overflow Characteristics

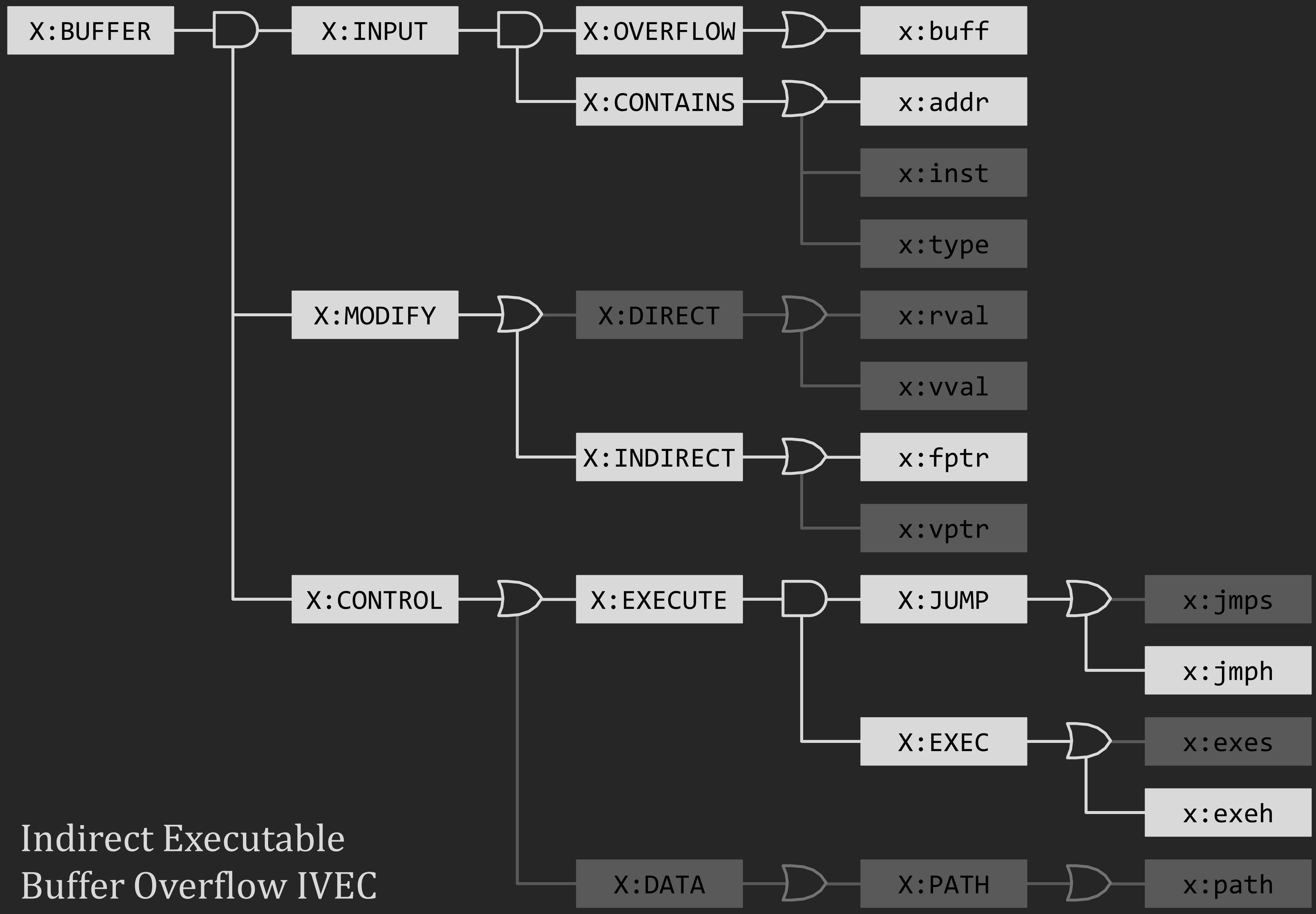
Supplemental Slides



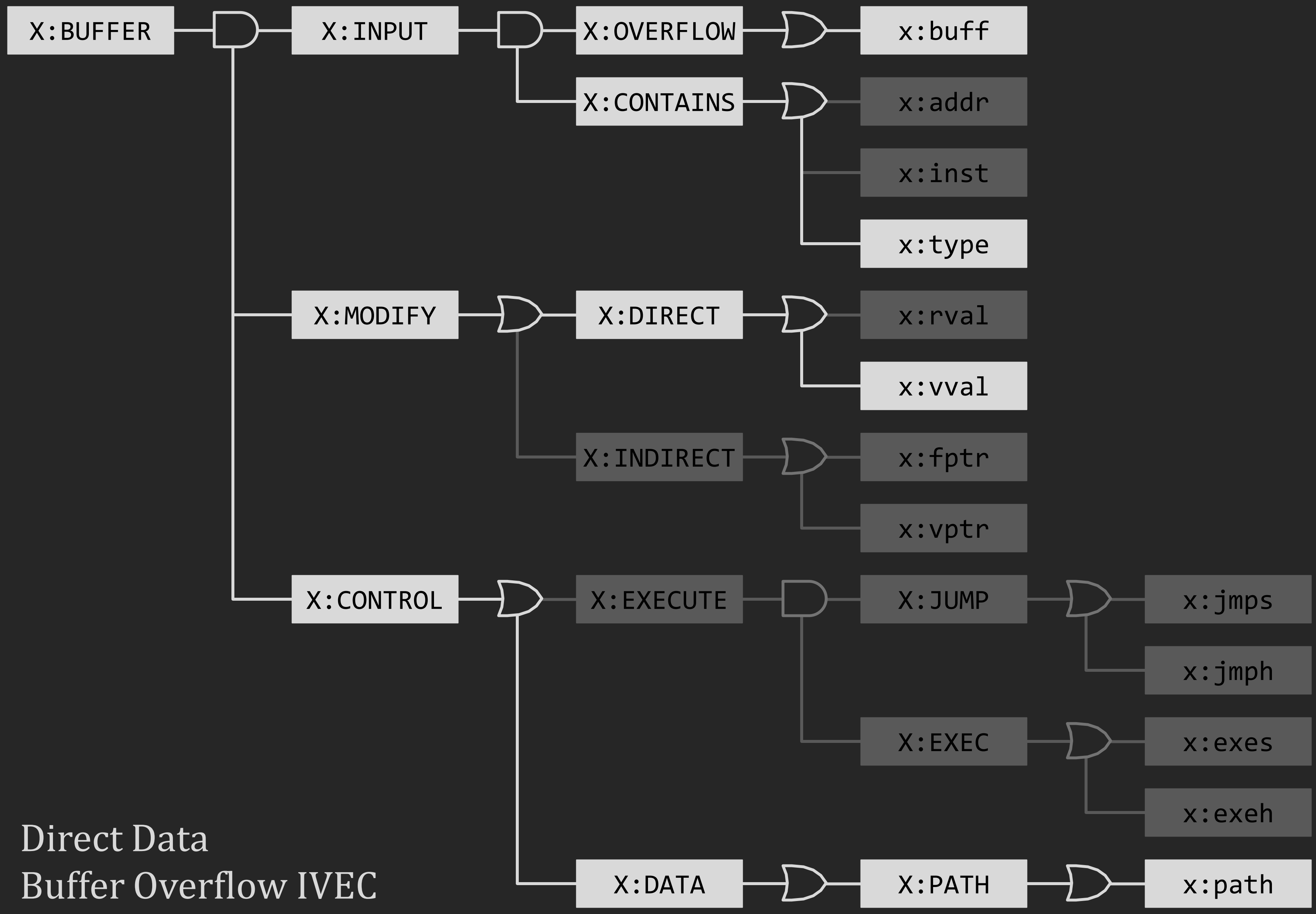
Buffer Overflow
Characteristic Tree



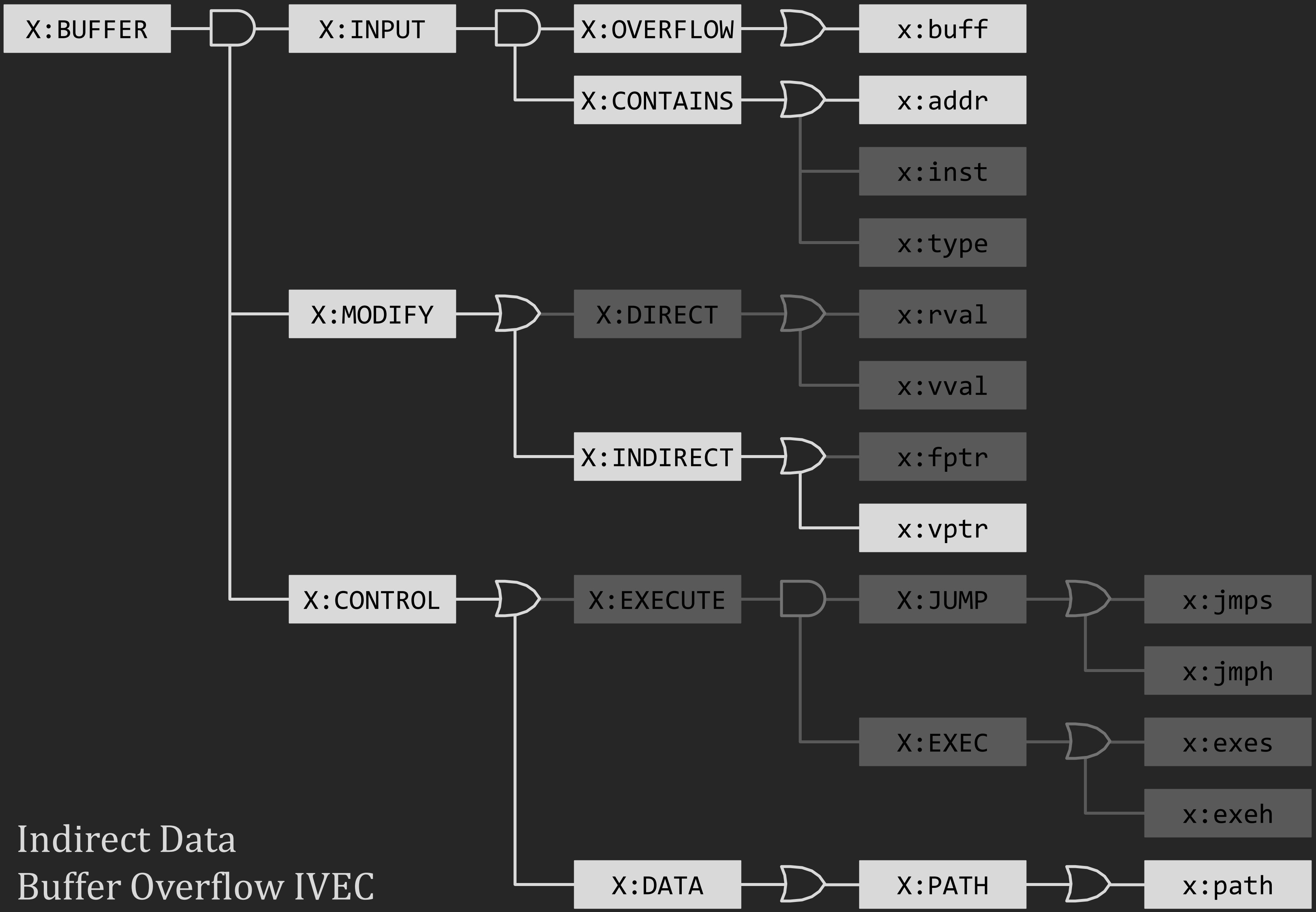
Direct Executable
Buffer Overflow IVEC



Indirect Executable
Buffer Overflow IVEC



Direct Data
Buffer Overflow IVEC



Indirect Data
Buffer Overflow IVEC