How the CC Harmonizes with Secure Software Development Lifecycle

Sep. 11. 2013.

Jinseok Park(1st Author)

CIST(Center for Information Security Technologies), Korea University

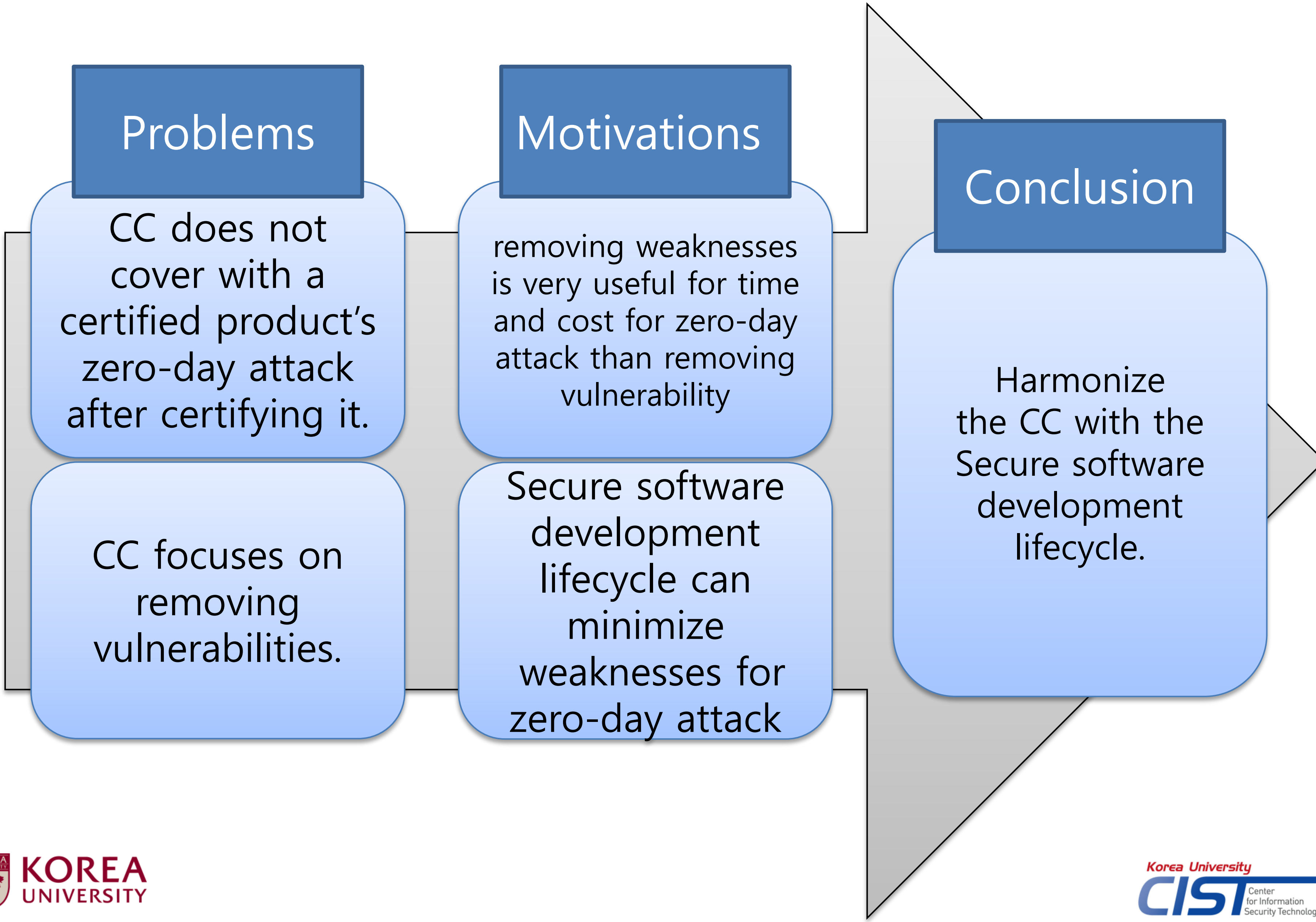
ilysoon7@korea.ac.kr

Seungjoo Kim (Corresponding Author)

CIST(Center for Information Security Technologies), Korea University

skim71@korea.ac.kr

Overview



Problems
CC does not cover with a certified product's zero-day attack after certifying it.
CC focuses on removing vulnerabilities.
Motivations
removing weaknesses is very useful for time and cost for zero-day attack than removing vulnerability
Secure software development lifecycle can minimize weaknesses for zero-day attack
Conclusion
Harmonize the CC with the Secure software development lifecycle.

Contents

Definitions

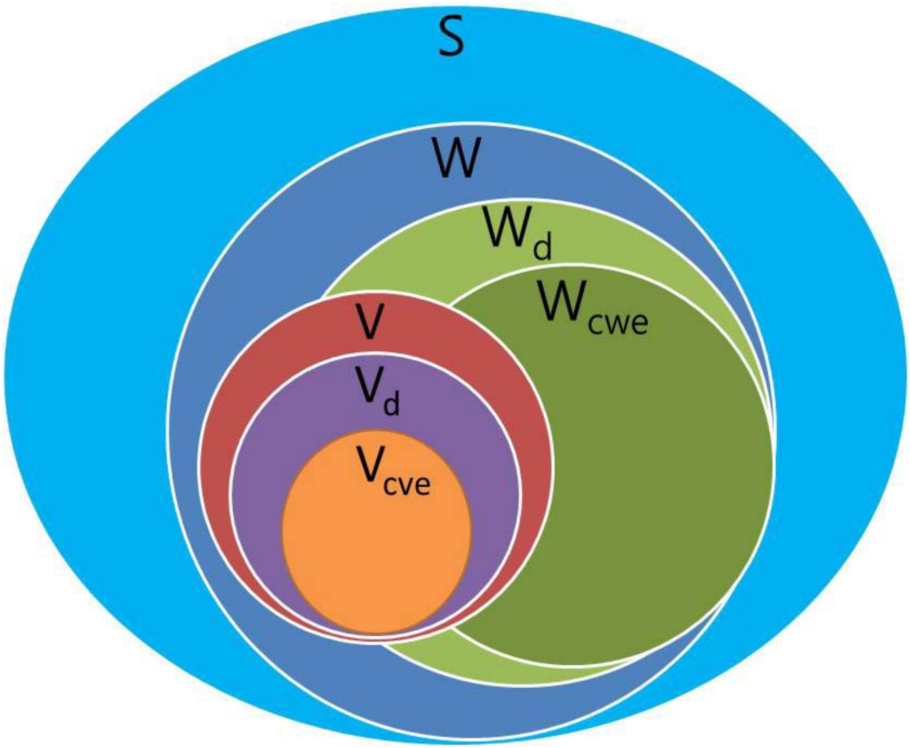
- **(Software Security) Weakness**
  - **A type of mistake in software**
  - **Bugs, Errors**
  - **Can be aggravated to (software security) vulnerabilities (i.e., Zero-day attacks)**

- **(Software Security) Vulnerability**
  - **An occurrence of a weakness (or multiple weaknesses) within software**

- **Zero-day attack**
  - **Weakness is exploited by hackers before the vendor becomes aware to fix it**

Definitions

<The relationship between weakness and vulnerability [8]>



- S : The set of all software in existence at some point in time
- W : The set of all instance of software weaknesses in S
- W_d : The set of discovered software weaknesses in W
- Wcwe : The set of Identified with a CWE
- V : The set of all vulnerabilities in W
- V_d: The set of all discovered Vulnerabilities in V
- Vcve : The set of Identified with a CVE

Motivations

- **Software bugs or errors are so detrimental that they cost the U.S economy an estimated $59.5 billion annually. (GDP 0.6％)**
- **Errors requirements/design stage cost 1X to fix. But if it is not found until the post-product release stage, it costs 30 times more to fix.**

Requirements Gathering and Analysis/ Architectural Design	Coding/Unit Test	Integration and Component/RAISE System Test	Early Customer Feedback/Beta Test Programs	Post-product Release
1X	5X	10X	15X	30X

Motivations

- The top 10 software vendors have a patch remedy rate of just over 94% of all vulnerabilities disclosed.
- But, 47% of all vulnerabilities disclosed in 2012 remain without a remedy.
- A zero-day attack can still be thwarted by properly-patched software.
  - But they are not cost and time effective!
- Economically, many researchers have tried to remove the vulnerability in software
  - To remove weaknesses is very useful for time and cost.

Motivations

- If we can remove weaknesses, vulnerabilities and zero-day attack can also be removed.

- Thus, we are interested in removing design stage's and implementation stage's weaknesses.
  - It is very useful for time and cost to remove weaknesses

Problems

- The CC philosophy is that the threats to security and organisational security policy commitments should be clearly articulated and the proposed security measures be demonstrably sufficient for their intended purpose. [9]

- CC focuses on removing vulnerabilities.

- CC does not cover with a certified product’s zero-day attack after certifying it.

How to Fix It in a Nutshell

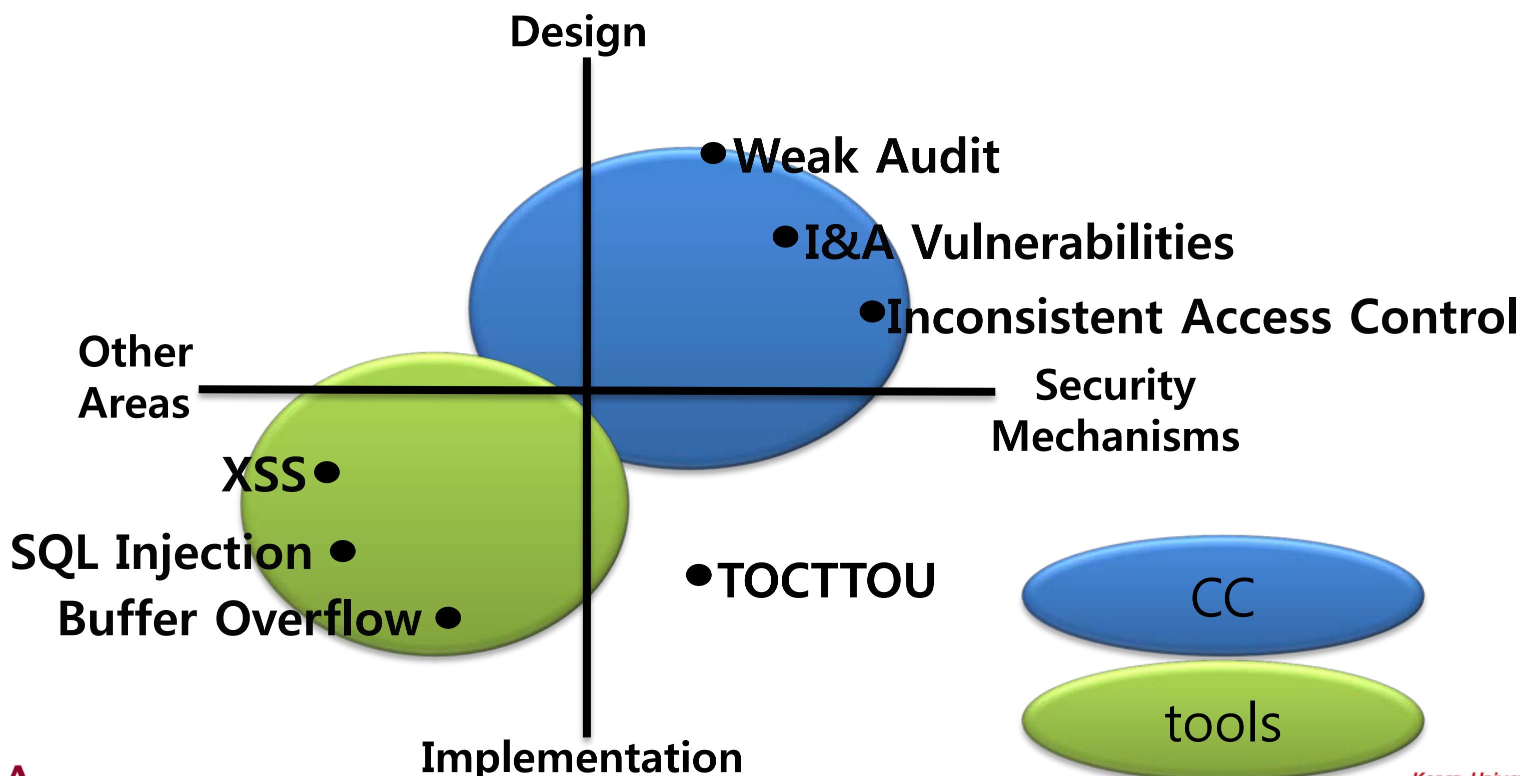
- **Software Assurance**
  - **The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the life cycle.**

- **Secure Software Development Lifecycle**
  - **Software Development Lifecycle + Software Assurance**

- **SSDLCs focus on removing weaknesses.**

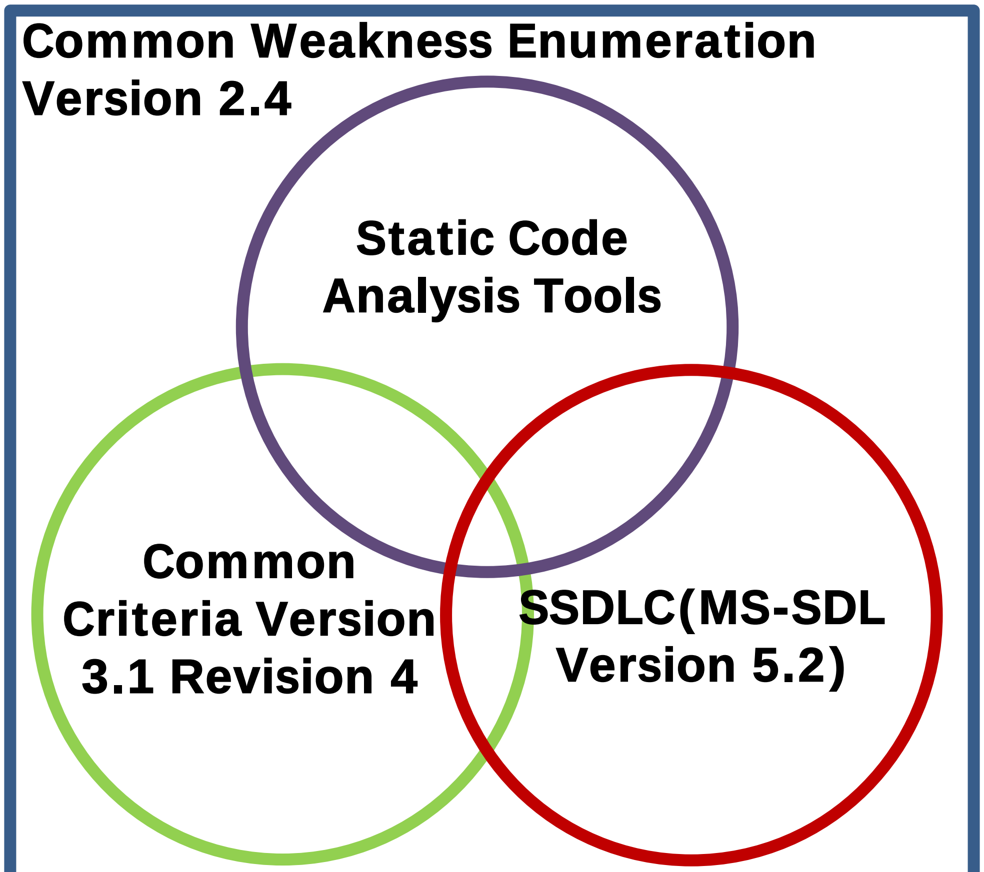
How to Fix It in a Nutshell

- CC and source code analysis tools are not rivals [11]
  - They find different types of vulnerabilities
  - If together, they can discover more common vulnerabilities types

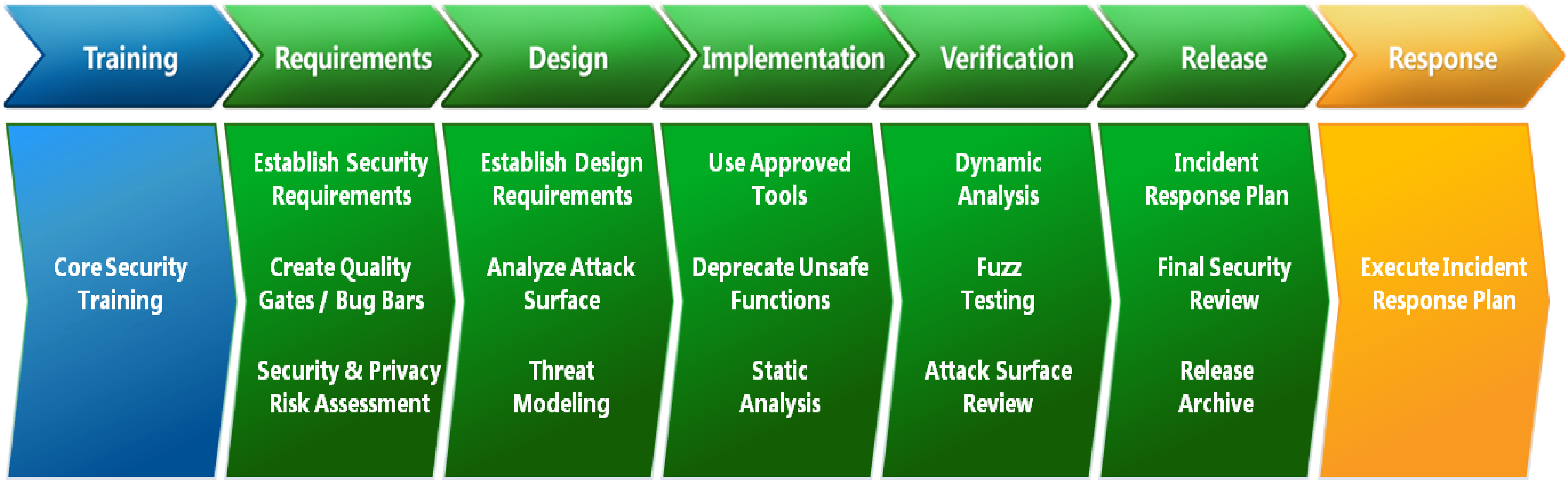


How to Fix It in a Nutshell

- Based on CWE v2.4, CC v3.1, MS-SDL(one of the famous SSDLCs), static code analysis tools.
  - Dynamic analysis tools can remove limited weaknesses. [12]



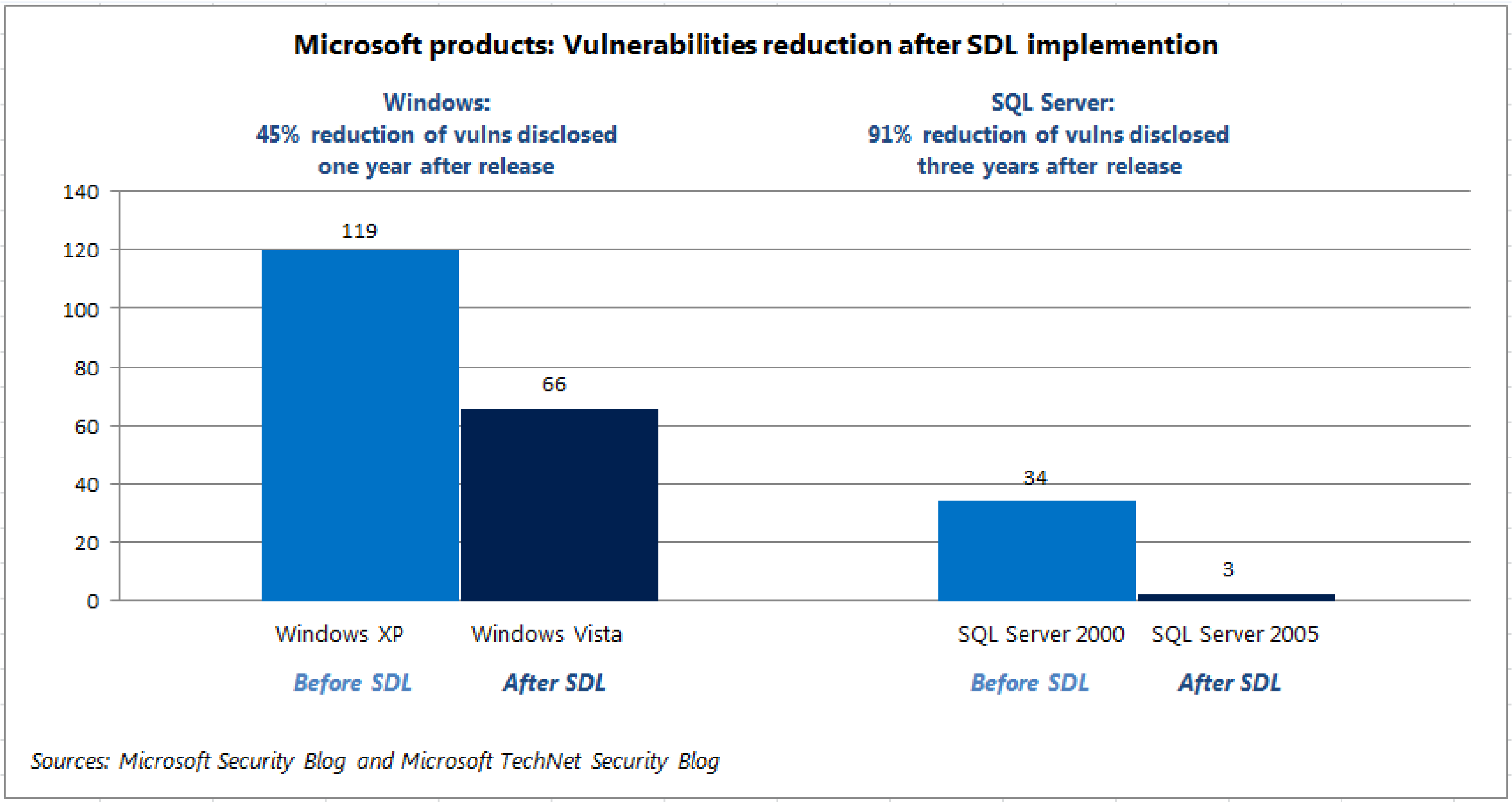
Our Methods in Detail



- MS-SDL(Microsoft-Security Development Lifecycle)
  - Software security assurance process
  - A mandatory policy since 2004

Our Methods in Detail

- MS-SDL helps you build software, that's more secure by reducing the number and severity of vulnerabilities in your code



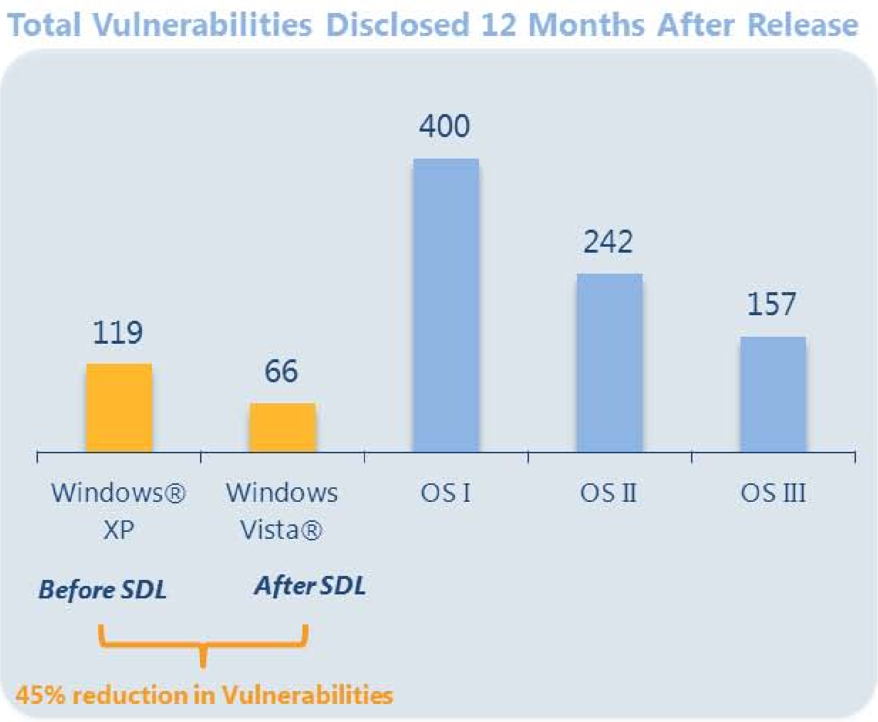
KOREA UNIVERSITY

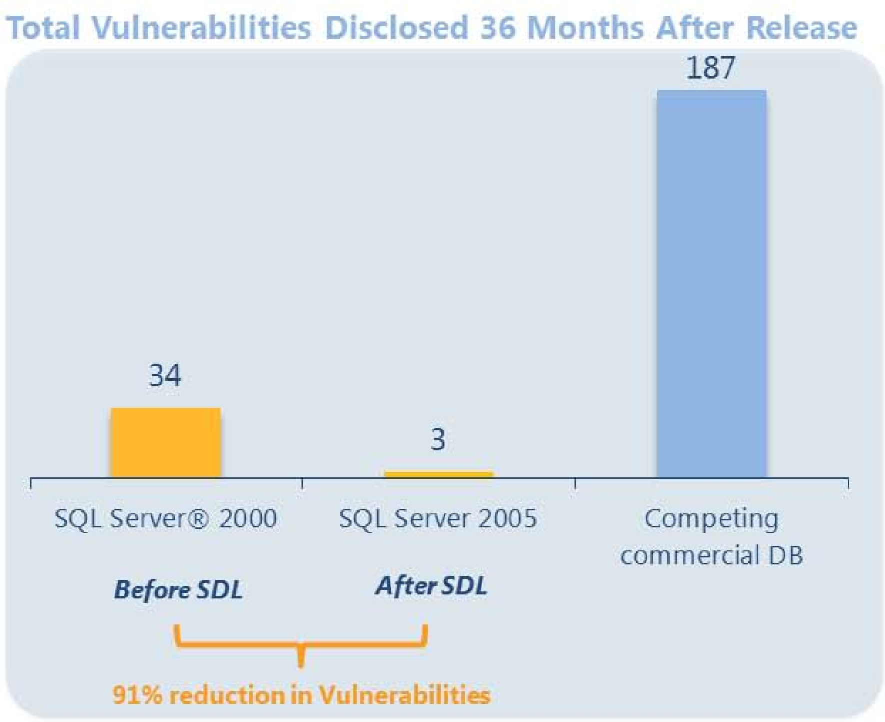
[Reference : 13]

Korea University CIST Center for Information Security Technologies

Our Methods in Detail

- **Consistent application of sound security practices during all phases of a development project will result in fewer vulnerabilities**

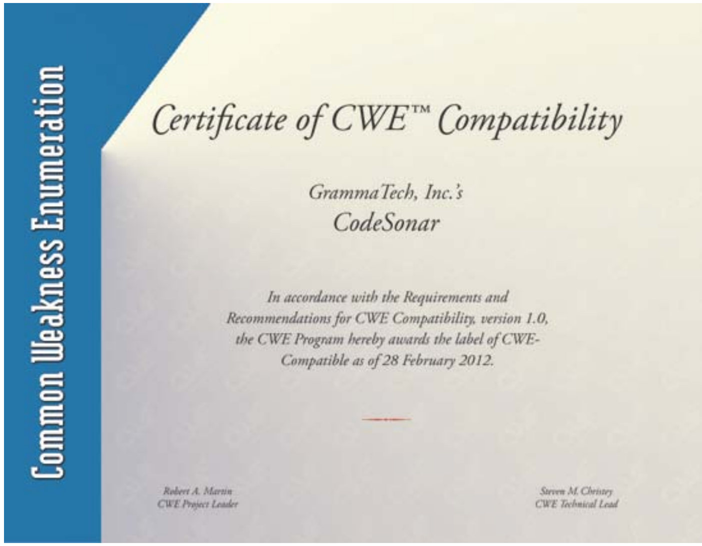




[Reference : 14]

Our Methods in Detail

- **Static code analysis tools**
  - **Analyze source code and/or compiled version of code in order to help find security flaws(weaknesses)**
- **Certificate of CWE compatibility (5 product) [15]**
  - **CodeSonar, Covertiy Quality Advisor/Security Advisor, HP Fortify Static Code Analyzer, Klocwork Insight**



KOREA UNIVERSITY

Korea University CIST Center for Information Security Technologies

Our Methods in Detail

- **Four different areas :**
  1. **Design(CWE-701)**
  2. **Implementation(CWE-702)**
  3. **Security mechanisms(CWE-254)**
  4. **Other parts(non-security mechanisms)**

Our Methods in Detail

- **Total weaknesses: 920 entries, 8 types**

View : 29 entries

Category : 176 entries

Selected 703 entries

Weakness – Class : 88 entries

Weakness – Base : 330 entries

Weakness – Variant : 276 entries

Compound Element – Composite : 6 entries

Compound Element – Named Chain : 3 entries

Deprecated : 12 entries

Our Methods in Detail

- ## For example, CWE/SANS TOP 25

Rank	CWE Type	CWE-ID : Name	Design	Implementation	Security Mechanisms	Static Code Analysis Tools	CVE Entry	MS-SDL			CC	
								Design	Implementation	Verification	SFR	SAR
1	Base	CWE-89 : SQL Injection	○	○		○	7	○	○	○		○
2	Base	CWE-78 : OS Command Injection	○	○		○	10	○	○	○		○
3	Base	CWE-120 : Classic Buffer Overflow		○		○	5	○	○	○		○
4	Base	CWE-79 : Cross-site Scripting	○	○		○	11	○	○	○		○
5	Variant	CWE-306 : Missing Authentication for Critical Function	○		○		3	○	○	○	○	○
6	Class	CWE-862 : Missing Authorization	○	○			19	○	○	○	○	○
7	Base	CWE-798 : Use of Hard-coded Credentials	○		○		10	○			○	○
8	Base	CWE-311 : Missing Encryption of Sensitive Data	○	○	○		20	○		○	○	○
9	Base	CWE-434 : Unrestricted Upload of File with Dangerous Type	○	○			10		○		○	○
10	Base	CWE-807 : Reliance on Untrusted Inputs in a Security Decision	○	○	○		5	○	○	○		○

[Reference : 16]

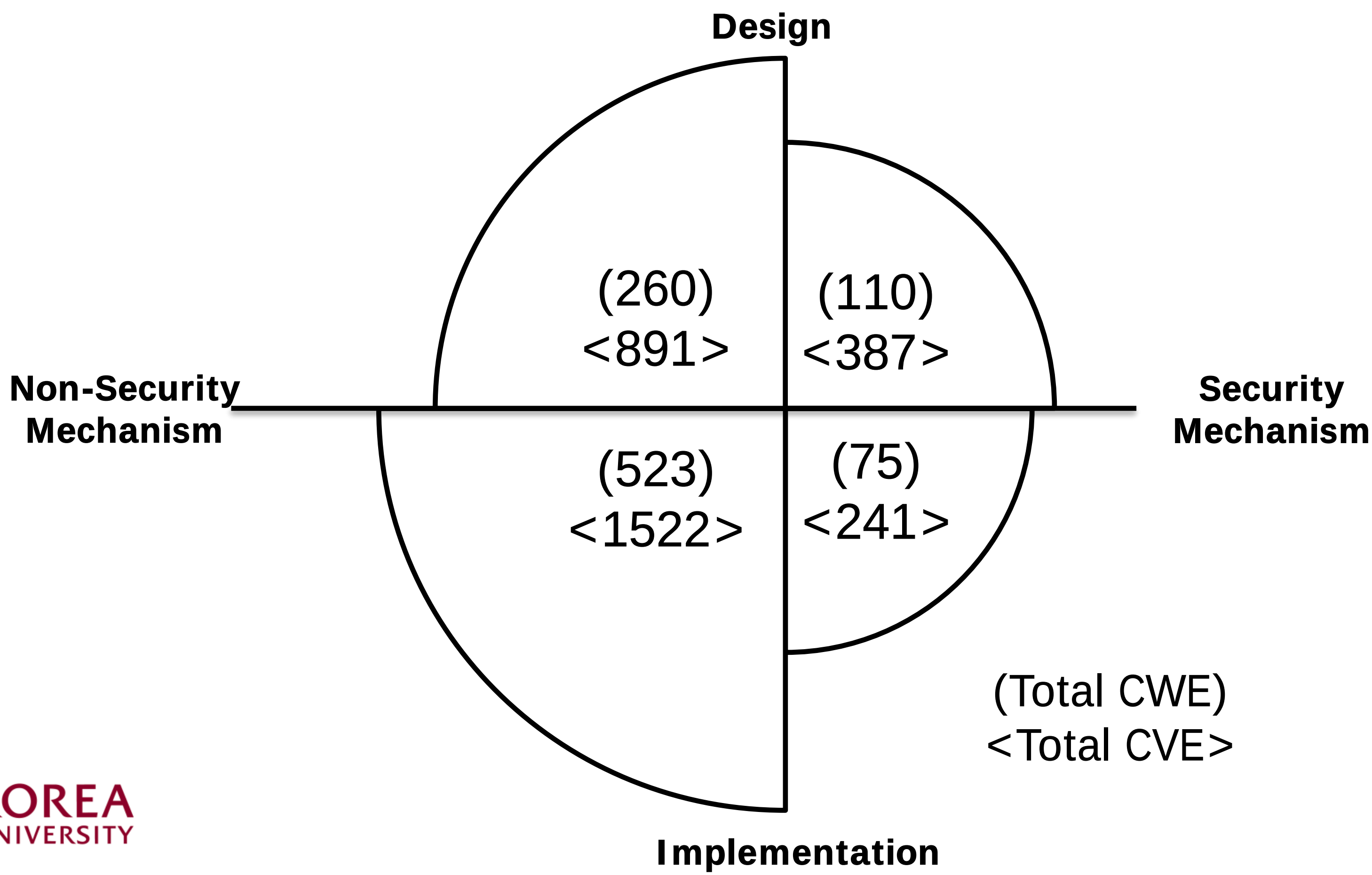
Our Methods in Detail

- ## For example, CWE/SANS TOP 25

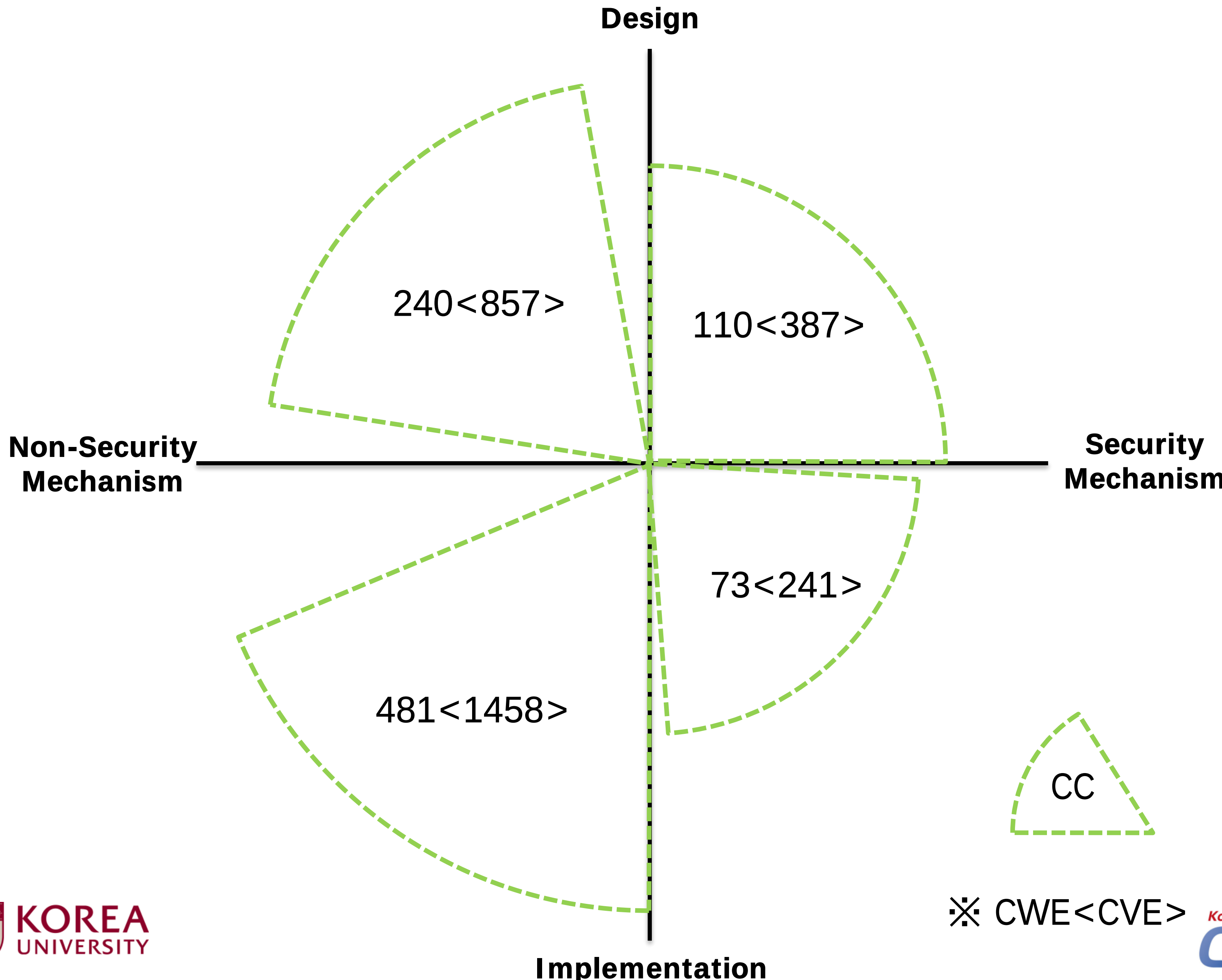
11	Class	CWE-250 : Execution with Unnecessary Privileges	○		○	○	7	○	○	○	○	○
12	Composite	CWE-352 : Cross-Site Request Forgery(CSRF)	○				10	○	○		○	○
13	Class	CWE-22 : Path Traversal	○	○		○	11	○	○	○		○
14	Base	CWE-494 : Download of Code Without Integrity Check	○	○			4	○	○		○	○
15	Class	CWE-863 : Incorrect Authorization	○	○			9	○	○	○	○	○
16	Class	CWE-829 : Inclusion of Functionality from Untrusted Control Sphere					20	○	○	○	○	○
17	Class	CWE-732 : Incorrect Permission Assignment for Critical Resource	○	○		○	17	○	○	○	○	○
18	Base	CWE-676 : Use of Potentially Dangerous Function	○	○		○	6		○			○
19	Base	CWE-327 : Use of a Broken or Risky Cryptographic Algorithm	○		○		8	○			○	○
20	Base	CWE-131 : Incorrect Calculation of Buffer Size		○		○	14	○	○	○		○
21	Base	CWE-307 : Improper Restriction of Excessive Authentication Attempts	○		○		6	○	○	○	○	○
22	Variant	CWE-601 : Open Redirect	○	○			3	○	○	○	○	○
23	Base	CWE-134 : Uncontrolled Format String		○		○	6	○	○	○		○
24	Base	CWE-190 : Integer Overflow or Wraparound		○		○	6	○	○	○		○
25	Base	CWE-759 : Use of a One-Way Hash without a Salt					2	○			○	○

Analyses

- Divided into four areas(Design, Implementation, Security mechanism, Non-Security mechanism)
- Distribution of weakness and vulnerabilities in each area

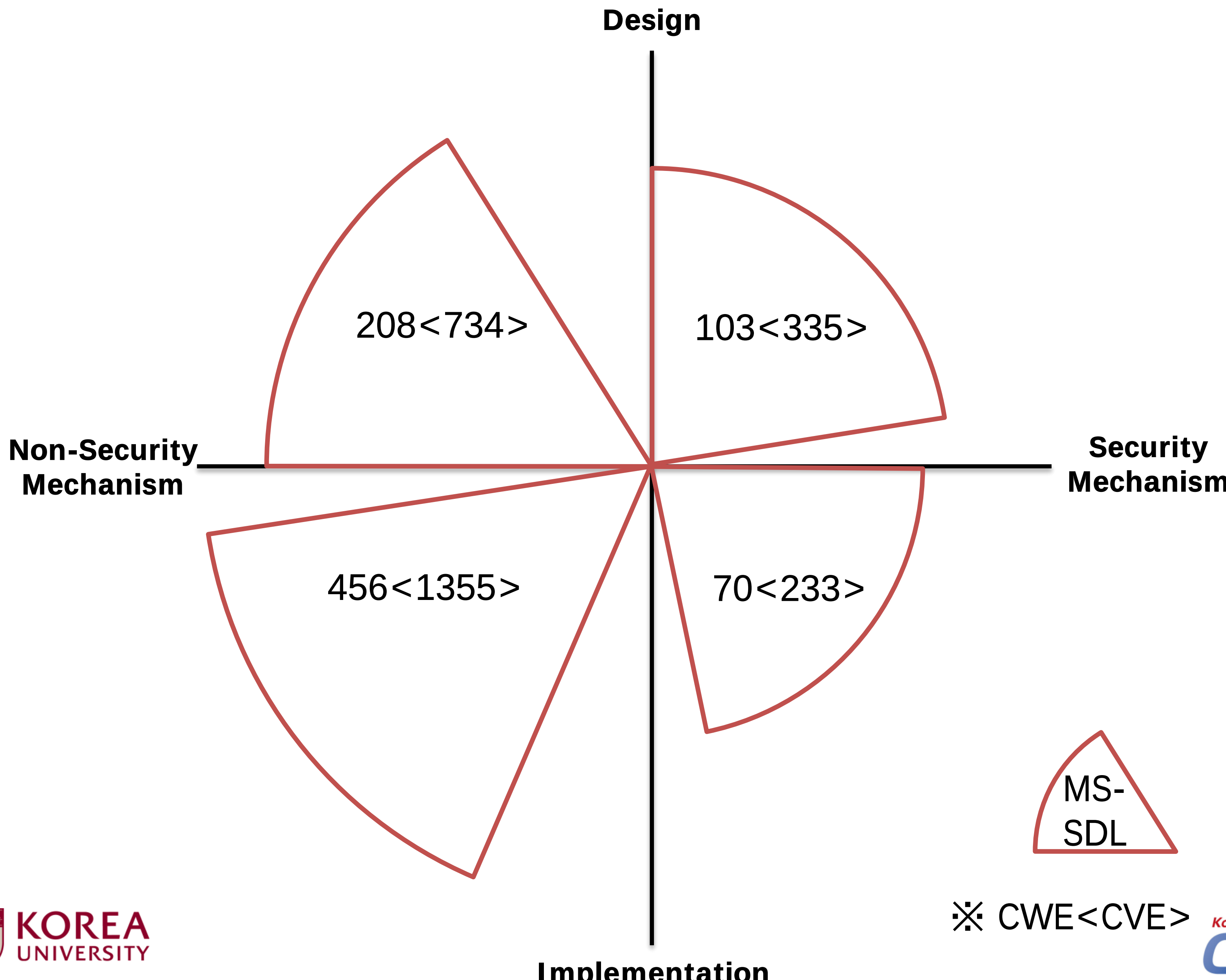


Analyses - CC -



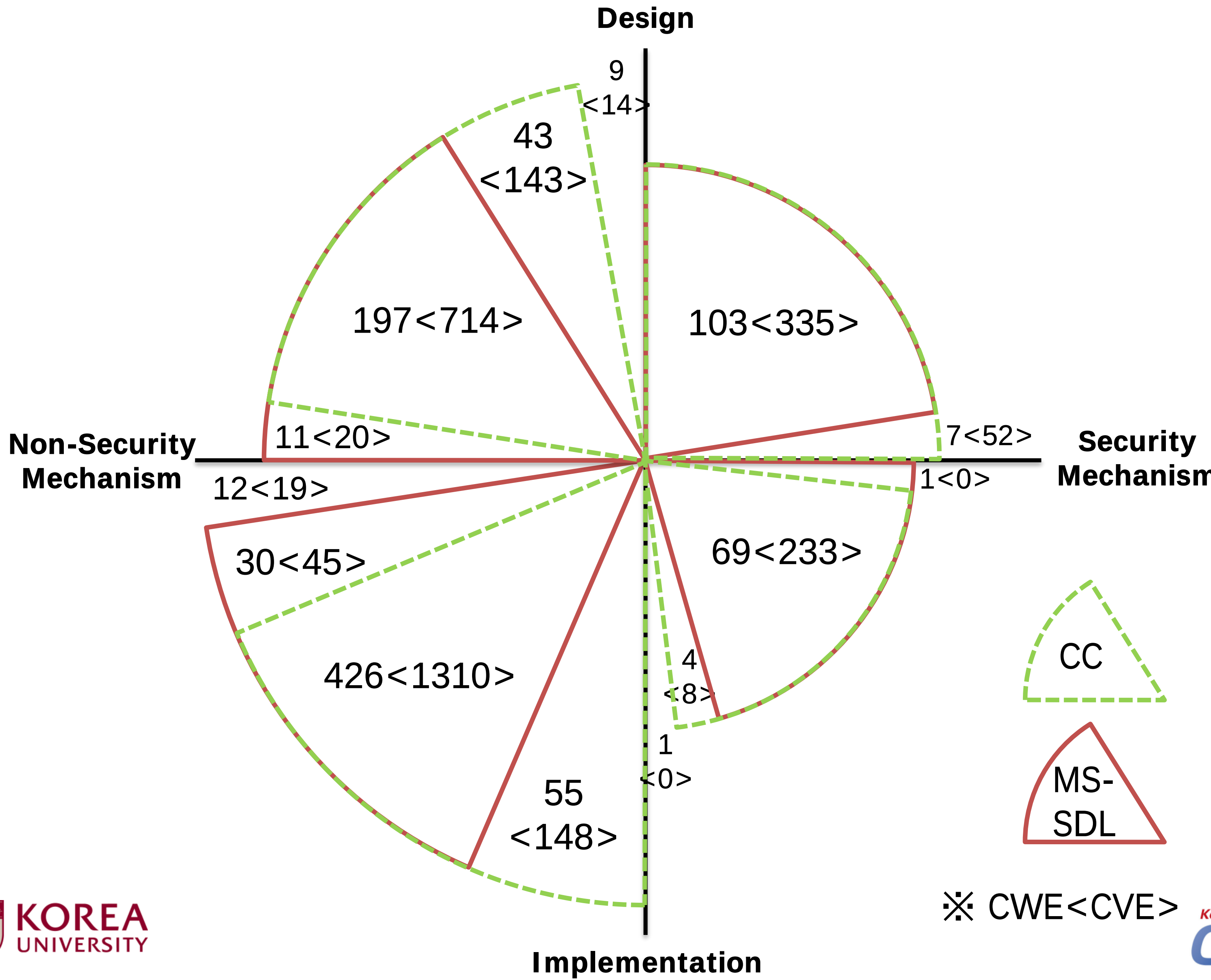
※ CWE<CVE>

Analyses - MS-SDL -



KOREA UNIVERSITY
Korea University CIST Center for Information Security Technologies

Analyses - CC and MS-SDL -

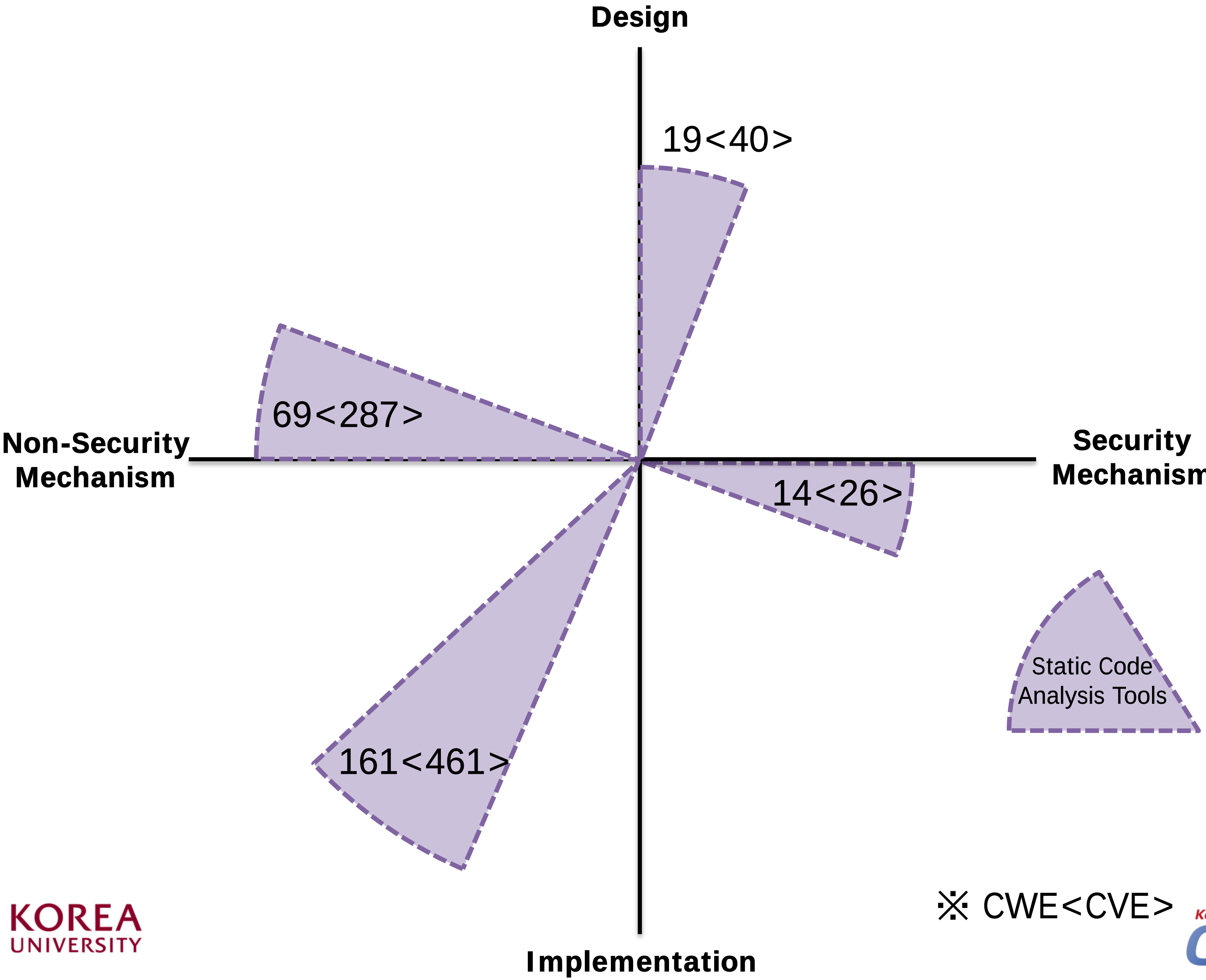


※ CWE<CVE>

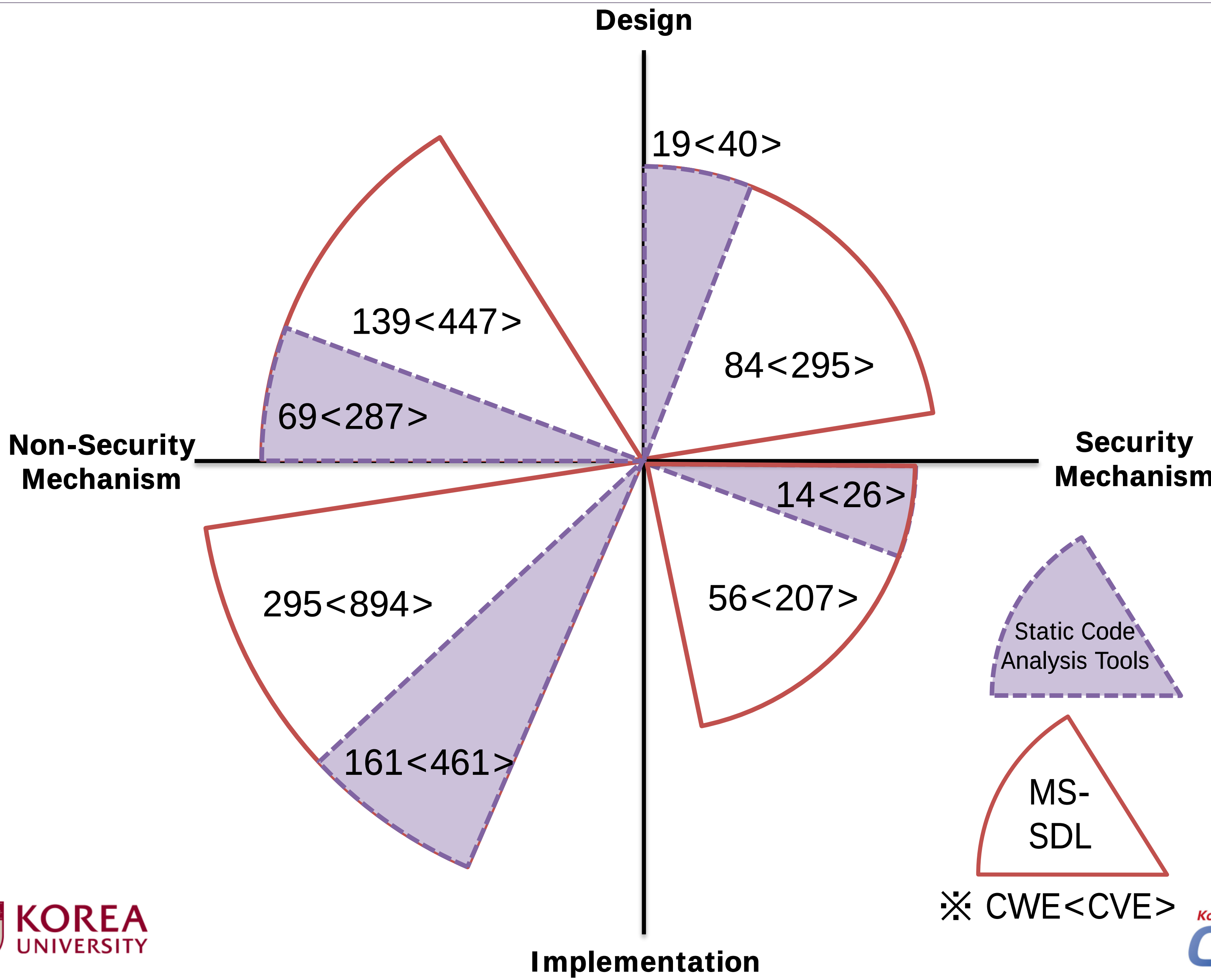
KOREA UNIVERSITY

Korea University CIST Center for Information Security Technologies

Analyses - Static Code Analysis Tools -



Analyses - CC & Static Code Analysis Tools -



How to Harmonize CC with SSDLC

- **Proposed Security Assurance Requirements(SAR)**
- **Now CC + SSDLC's practice**

SSDLC Process	Practice	CC Security Assurance Requirements
1.Training	Core Security Training	ALC_DVS
2.Requirements	Establish Security and Privacy Requirements, Create Quality Gates/Bug Bars, Perform Security and Privacy Risk Assessments	ASE
		ALC_TAT
		AVA
3.Design	Establish Design Requirements, Attack Surface Analysis/Reduction, Use Threat Modeling	ADV
		AVA
4.Implementation	Use Approved Tools, Deprecate Unsafe Functions, Perform Static Analysis	ATE
		ADV_IMP
5.Verfication	Perform Dynamic Analysis, Fuzz Testing, Attack Surface Review	ATE
6.Release/Response	Create an Incident Response Plan, Conduct Final Security Review, Certify Release and Archive, Execute Incident Response Plan	AGD
		ALC_CMC
		AVA

Conclusion

- The CC and the SSDLC are similar methodologies for removing vulnerabilities.
  - But they find different types of vulnerabilities.

- Static code analysis tools can help removing weaknesses in CC

- The CC and the SSDLC are not competitors. Rather, they are complements.

References

1. Gregory Tassey, Ph.D, "The Economic Impact of Inadequate Infrastructure for software Testing, Planning Report", 02-3, NIST, May 2002.
2. Paul Wood, "Closing The Window Of Vulnerability: Exploits And Zero-Day Attacks", Internet security Threat report", vol. 17, Symantec, Apr. 2012.
3. Brian McGee, "Vulnerabilities in enterprise software", IBM X-Force 2012 Mid-year Trend and Risk Report, Sep. 2012.
4. Gerhard Eschelbeck, "Systems and software threats", Security Threat Report, Sophos, Jan. 2012.
5. Theresa Lanowitz, "Now Is the Time for Security at the Application Level", Gartner, Dec. 2005.
6. Joe Jarzombek, "Software Assurance: Enabling Security and Resilience throughout the Software Lifecycle", MITRE, Nov. 2012.
7. U.S Department of Homeland Security Web page: https://buildsecurityin.us-cert.gov/swa/forums-and-working-groups/processes-and-practices/swa-capability-benchmarking
8. Richard Struse, "Software Assurance-Making the Software Ecosystem Rugged", ICSJWG, Oct. 2011.
9. Common Criteria Recognition Arrangement, "Common Criteria for Information Technology Security Evaluation Version 3.1 Revision 4", Sep. 2012.
10. U.S Department of Homeland Security Web page: https://buildsecurityin.us-cert.gov/swa/process-view/overview
11. Adam O'Brien, "Common Criteria and Source Code Analysis Tools: Competitors or Complement", International Common Criteria Conferences 9th, Sep. 2010.
12. B. Chess and C. McGraw, "Static analysis for security," IEEE Security & Privacy, vol. 2, no. 6, pp. 76~79, Nov. 2004.
13. Jeff Jones, "Microsoft products: Vulnerabilities reduction after SDL implementation", Microsoft Security Blog and Microsoft TechNet Security Blog, Jan. 2008.
14. "Basics of Secure Design Development Test : Secure Software Made Easier", Microsoft, 2008.
15. Common Weakness Enumeration(CWE) web page : Assessment and Remediation Tool, http://cwe.mitre.org/compatible/category.html, Jun. 2013.
16. MITRE, "2011 CWE/SANS Top 25 Most Dangerous Software Errors", Sep. 2011.

Thank you

ilysoon7@korea.ac.kr

Jinseok Park
E-mail : ilysoon7@korea.ac.kr
Facebook : @ilysoon7

Jinseok Park received his B.S. (2010) in computer science from Soongsil University in Korea. also, , He served as electronic signature and authentication Team of the Korea Information Security Agency (KISA) for 1 years. Now he is enroll in the M.S. at Korea University. His research interests include security evaluation, information security and online social network service security. He has many certificates(CISSP, CISA, Security Product Evaluator, CPPG, Information Processing Engineer)

Seungjoo Kim (Corresponding Author)
E-mail : skim71@korea.ac.kr
Homepage : www.kimlab.net
Facebook, Twitter : @skim71

Prof. Seungjoo Kim received his B.S. (1994), M.S. (1996), and Ph.D. (1999) in information engineering from Sungkyunkwan University (SKKU) in Korea. Prior to joining the faculty at Korea University (KU) in 2011, He served as Assistant & Associate Professor of School of Information and Communication Engineering at SKKU for 7 years. Before that, He served as Director of the Cryptographic Technology Team and the (CC-based) IT Security Evaluation Team of the Korea Information Security Agency (KISA) for 5 years. Now he is Full Professor of Graduate School of Information Security at KU, and a member of KU's Center for Information Security Technologies (CIST). Also, He has served as an executive committee member of Korean E-Government, and advisory committee members of several public and private organizations such as National Intelligence Service of Korea, Digital Investigation Advisory Committee of Supreme Prosecutors' Office, Ministry of Justice, The Bank of Korea, ETRI(Electronic and Telecommunication Research Institute), and KISA, etc. His research interests include cryptography, information security and information assurance.

The corresponding author (Seungjoo Kim) acknowledges the support of the IT R&D program (10043959, Development of EAL 4 level military fusion security solution for protecting against unauthorized accesses and ensuring a trusted execution environment in mobile devices) of KEIT/MOTIE and MSIP(Ministry of Science, ICT & Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (NIPA-2013-H0301-13-1003) supervised by the NIPA(National IT Industry Promotion Agency)