An Attack Surface Metric Suitable for Heterogeneous Redundant System with the Voting Mechanism

Liqun Wang, Zheng Zhang, Weichao Li, Zhenwu Liu and Hao Liu
State key Laboratory of Mathematical Engineering and Advanced Computing, MEAC, Zhengzhou, Henan, 450001, China
*Corresponding author’s e-mail: standoutking@126.com

Abstract. With the development of cyberspace security “asymmetry in attack and defense”, heterogeneous redundancy design is gradually being widely used. Heterogeneous redundancy design can enhance the robustness and security of the system through the differentiation of heterogeneous components. But the complex structural design of the system makes it difficult to apply the existing attack surface metric to measure the security of heterogeneous redundant system. In this paper, based on the attack surface metric proposed by Manadhata, we provide a new attack surface metric suitable for heterogeneous redundant system with voting mechanism. In this new metric, we define a new notion named attack surface arbitration, and also propose a new method for quantifying result of arbitration. The experiment result shows that the new attack surface metric can properly describe the attack surface of a heterogeneous redundant system with voting mechanism, which changes with the adjustment of voting algorithm.

1. Introduction
The in-depth development of the “Internet plus” era has made the software industry extend to other industries and has radiated to every corner of society. Software industry brings a lot of conveniences to people as well as security challenges. Software security has conducted a profound impacts on many aspects, from personal life and business operations to national interests. Therefore, measurement of software security, both qualitatively and quantitatively, is a long-standing challenge to the research community and is of practical significance to software industry today[1-3].

In this paper, based on the attack surface metric proposed by Manadhata, we provide a new attack surface metric suitable for heterogeneous redundant system (HRS) with voting mechanism. In this new metric, we define a new notion named attack surface arbitration (ASA), and also propose a new method for quantifying result of arbitration.

2. Related work
2.1 Heterogeneous redundant system
Redundancy design is divided into homogeneous redundancy and heterogeneous redundancy. Its origins can be traced back to1834. Lardner proposed that an instruction is executed simultaneously by multiple identical computers can ensure the correctness of the calculation results, and an instruction is executed simultaneously by multiple different computers can also enhance the reliability of the calculation results[4]. With the advancement of technology and the requirement of reliability, heterogeneous redundancy design gradually replaces homogeneous redundancy design in order to effectively reduce the probability of occurrence of common mode failure. However, heterogeneous
redundancy design is generally not applied separately, but combined with voting mechanism. A
typical HRS with voting mechanism is just dissimilar redundant system (DRS), which is widely used
in flight control computer system design[5-8]. In addition, heterogeneous redundancy design is also
the basis of the architecture of some security defense systems, such as intrusion tolerance system
(SITAR[9], MAFTIA[10], SCIT[11]) and mimic defense system (DHR[12]).

2.2 Voting mechanism
The voting mechanism is introduced to judge the response results of all heterogeneous redundant
subsystems according to some voting algorithms, so that the results of external output are more
accurate and appropriate. The implementation of the voting mechanism relies on the voting algorithms.
The sample voting algorithms mainly include multi-mode voting, large-number voting, and full-
consistent voting, etc. The complex voting algorithms mainly include maximum appropriate voting,
weighted voting, large-number voting based on historical information, and Byzantium. Voting, etc.
The complex voting algorithms are on the basis of multi-mode voting, and usually introduce weighted
parameters such as current security situation, system resource status, attack frequency, historical
performance of each executive entity, etc. The research work in this paper is mainly based on the
simple voting algorithms, and the complex voting algorithms will be studied in the future work.

2.3 Attack surface
A system’s attack surface is the subset of its resources that an attacker can use to attack the system[13].
The informal notion of attack surface was first introduced by Michael Howard[14]. Howard proposed
a measurement method for the Windows operating system’s attack surface. And Manadhata et al., who
were inspired by Michael Howard’s Relative Attack Surface Quotient measurements, who were
motivated by their practice’s findings, defined a systematic attack surface measurement method[13].
Manadhata’s method identifies the attack surface resources by the entry point and exit point
framework and introduces the notion of the damage potential-effort ratio (DPER) to estimate the
weights of each attack surface resource. However, for the dynamic shifting of the moving target
defense approach, Manadhata et al. proposed the notion of attack surface shifting (ASS) and
introduced a method to quantify the shift[15]. Therefore, inspired by the ASS, we propose the notion
named the ASA according to the voting mechanism, and introduce a new method for quantifying
result of arbitration.

![Diagram](image)

Figure 1. N functionally equivalent heterogeneous redundant subsystem

3. Attack surface metric
In this section, we discuss that an attack surface metric suitable for HRS. The HRS includes N
functionally equivalent heterogeneous redundant subsystems, which operate independently each other,
and the key components of each subsystem are heterogeneous, as shown in figure 1. Therefore, firstly,
we use Manadhata’s attack surface model to formalize each subsystem’s attack surface, then, define the notion of ASA and measure a HRS’s attack surface by the ASA.

3.1 Formalizing each subsystem’s attack surface
The attack surface of a functionally equivalent heterogeneous redundant subsystem $A_i$ is formally described below.

We define a scene, $w_i$, which include a system set, $G_i$, an attacker set, $U_i$, and a data set, $D_i$. For a given system, $A_i$, $A_i \in G_i$, we define its environment as a three-tuple: $E_{A_i} = \{U_i, D_i, T_i\}$, among, $T_i = G_i \setminus \{A_i\}$, a set of systems that does not include $A_i$; $U_i$, a set of attackers who launch cyberspace attacks; $D_i$, a set of data which is shared between $G_i$ and $U_i$. Thus, the attack surface of subsystem $A_i$ is formalized as:

$$surf_{A_i} = \left\{M^{E_{A_i}}, C^{E_{A_i}}, I^{E_{A_i}}\right\}$$

among, $M^{E_{A_i}} = \{m^A_{1,k}, m^A_{2,k}, \ldots, m^A_{k,n} | n \geq k \geq 1, n \in N^+\}$

(2)

where, $m^A_k$ represents a system method of $A_i$ and is the component unit of the set of its entry points and exit points.

$C^{E_{A_i}} = \{c^A_{1,k}, c^A_{2,k}, \ldots, c^A_{k,n} | n \geq k \geq 1, n \in N^+\}$

(3)

where, $c^A_k$ represents a system channel of $A_i$ and is the component unit of the set of its channels.

$I^{E_{A_i}} = \{d^A_{1,k}, d^A_{2,k}, \ldots, d^A_{k,n} | n \geq k \geq 1, n \in N^+\}$

(4)

where, $d^A_k$ represents a untrusted data of $A_i$ and is the component unit of the set of its untrusted data.

Based on the formalized expression above, the attack surface resource set (ASRS) of the $A_i$ is defined as:

$$R_{A_i} = M^{E_{A_i}} \cup C^{E_{A_i}} \cup I^{E_{A_i}}$$

(5)

Other functionally equivalent heterogeneous redundant subsystems can also do the above analogy.

3.2 Defining the notion of attack surface arbitration and measuring a HRS’s attack surface
In the above, we have formalized the attack surface of each functionally equivalent heterogeneous redundant subsystem. We need to extend Manadhata’s attack surface model and apply it to HRS.

Given a scene $W$, including $N$ sub-scenes, which can be formalized like $A_i$’s scene. And $W$ is still composed of a system set, $G = \{G_1, G_2, G_3, \ldots, G_n\}$, an attacker set, $U = U_1$, and a data set, $D = \{D_1, D_2, D_3, \ldots, D_n\}$. For a given system $A = \{A_1, A_2, A_3, \ldots, A_n\}$, $A \in G$, therefore, its environment is represented by:

$$E_{A} = \{E_{A_1}, E_{A_2}, E_{A_3}, \ldots, E_{A_n} | n \geq i \geq 1, n \in N^+\}$$

(6)

among, $E_{A_i} = \{U_i, D_i, T_i\}$.

Through the above expression, it can be concluded that if $A_i$ is a HRS with voting mechanism, then its ASRS is equal to the union of the ASRSs of all subsystems before voting, and can be expressed as:

$$R_{A_i} = R_{A_1} \cup R_{A_2} \cup R_{A_3} \cup \cdots \cup R_{A_n} \cup \cdots \cup R_{A_i} \cup \cdots \cup R_{A_n}, n \geq i \geq 1, n \in N^+$$

(7)

among, $R_{A_i}$ is the ASRS of $A_i$.
Definition 1. Given a HRS with voting mechanism, \( A = \{ A_1, A_2, A_3, \ldots, A_n \} \), its environment, \( E_A \), the ASRSs of its subsystems respectively are \( R_{A_1}, R_{A_2}, \ldots, R_{A_n} \), if there exists a attack surface resource, \( r \), such that (i) \( r \in R_{A_i} \cap R_{A_j} \), we call that \( r \) is 2-common resource between \( A_i \) and \( A_j \) (2-CR\(_{ij}\)); (ii) \( r \in R_{A_i} \cap R_{A_j} \cap R_{A_k} \), we call that \( r \) is 3-common resource among \( A_i \), \( A_j \) and \( A_k \) (3-CR\(_{ijk}\)); (iii) \( r \in R_{A_i} \cap R_{A_j} \cap \cdots \cap R_{A_n} \), we call that \( r \) is n-common resource among \( A_1, A_2, \ldots, A_n \) (n-CR\(_{12\ldots n}\)); (iv) Other situations do similar reasoning.

Theorem 1. Given a HRS with voting mechanism, \( A = \{ A_1, A_2, A_3, \ldots, A_n \} \), its environment, \( E_A \), the ASRSs of its subsystems respectively are \( R_{A_1}, R_{A_2}, \ldots, R_{A_n} \), if attack surface resource \( r \) is \( p\)-CR, then \( r \) is also \( (p-1)\)-CR, for example, \( r \) is \( 3\)-CR\(_{ijk}\), then \( r \) is also \( 2\)-CR\(_{ij}\), \( 2\)-CR\(_{ik}\) and \( 2\)-CR\(_{jk}\).

Proof (Theorem 1.) If attack surface resource \( r \) is \( p\)-CR, then from Definition 1, there is \( p \) subsystems, such that \( B_1, B_2, B_3, \ldots, B_p \), and \( r \in B_1 \cap B_2 \cap B_3 \cap \cdots \cap B_p \). Hence there exists a
expression, \( r \in B_1 \cap B_2 \cap B_3 \cap \cdots \cap B_{p-1} \), and again from Definition 1, we can call that \( r \) is \((p-1)-\)common resource among \( B_1, B_2, \cdots, B_{p-1} \) \((-CR\)).

The definition of \( k-CR \) is a prerequisite for the notion of the attack surface arbitration, and \( k-CR \) is an important indicator for the ASA to filter the elements of ASRS, as shown in figure 2. However, the voting algorithm of HRS decides the threshold \( k \) of \( k-CR \) \((n \geq k \geq 1, n \in N^* \), \( n \) is the number of the all subsystems). When the voting algorithm is full-consistent voting, the threshold \( k \) of \( k-CR \) is equal to \( n \). When the voting algorithm is large-number voting, the threshold \( k \) of \( k-CR \) is greater than or equal to \( \lceil n/2 \rceil \). When the voting algorithm is multi-mode voting (assuming \( t \) mode, \( n \geq t \geq 2, t \in N^* \)), the threshold \( k \) of \( k-CR \) is greater than or equal to \( t \).

**Definition 2.** When the response results of the various subsystems are transmitted to the voting device, some attack surface resources of the various subsystems will be invalid, which can be not exploited by attackers. This phenomenon is alluded to as the ASA. After the attack surfaces arbitrate, the set of attack surface resources that may still be exploited by the attacker is called the arbitrated attack surface (AAS). Given a HRS with voting mechanism, \( A = \{A_1, A_2, A_3, \cdots, A_n\} \), its environment, \( E_d \), the ASRSs of its subsystems respectively are \( R_{A_1}, R_{A_2}, \cdots, R_{A_n} \). When the voting algorithm is full-consistent voting, every element of \( A \)'s AAS is \( n-CR \). When the voting algorithm is large-number voting, every element of \( A \)'s AAS is at least \( 2n-CR \). When the voting algorithm is multi-mode voting (assuming \( t \) mode, \( n \geq t \geq 2, t \in N^* \)), every element of \( A \)'s AAS is at least \( t-CR \).

### 3.2.2 Measuring a HRS’s attack surface

In figure 2, we can find that the process of HRS’s attack surface metric mainly includes the arbitrating process and the quantifying process. The arbitrating process: under a certain voting algorithm, the elements of the AAS are identified from the ASRSs of HRS’s subsystems. The quantifying process: the quantitative result of the AAS is calculated by the DPER of each resource in the AAS.

According the notion of DPER proposed by Manadhata, we give a similar definition as followed.

**Definition 3.** Given an attack surface resource, \( r \), there exists a function, \( \text{der}(r) : \text{resource} \rightarrow Q \), that maps each resource to its damage potential-effort ratio belonging to the set \( Q \) of rational number. In practice, we assign a proper numeric values as resource’s DPER by system security personnel.

**Definition 4.** Given a HRS with voting mechanism, \( A = \{A_1, A_2, A_3, \cdots, A_n\} \), its environment, \( E_d \). If the AAS of \( A \) is expressed as:

\[
\text{AAS}_A = \{r_i, r_2, \cdots, r_m \mid m \geq i \geq 1, m \in N^* \}
\]

among, \( r_i \) is \( k-CR \), \( m \geq k \geq 2, k \in N^* \). And \( k-CR \) is decided by the voting algorithm. Thus, the measurement of the HRS’s AAS is expressed as:

\[
\sum_{i=1, i \in k-CR}^{n} \text{der}(r_i)
\]

among, \( \text{der}(r_i) \) represents the DPER of \( r_i \).

### 4. Experimental evaluation

In this section, firstly, we take a web application system with heterogeneous redundancy design as an experimental object. Then, we analyze the attack surface of each functionally equivalent heterogeneous redundant subsystem. Meanwhile, we identify the common mode situation of each attack surface resource. Finally, we assign a proper numeric values as its DPER according every attack
surface resource’s efficiency in the attack process, and compute the measurements of the HRS’s AAS with multi-mode voting algorithm, large-number voting algorithm and full-consistent voting algorithm.

4.1 Experimental object

The experimental web application system is composed of 9 functionally equivalent heterogeneous redundant subsystems, which all exist file upload and SQL injection vulnerability. And these subsystems’ configurations are as shown in table 1.

| Subsystem | Application | Web Server | Database | OS       |
|-----------|-------------|------------|----------|----------|
| A1        | PHP (No-label)\(^a\) | Nginx      | MySQL (No-label)\(^c\) | CentOS   |
| A2        | PHP (No-label) | Nginx      | MySQL (No-label) | Ubuntu   |
| A3        | PHP (No-label) | Nginx      | MySQL (No-label) | Windows server |
| A4        | PHP (No-label) | Apache     | MySQL (label 3)\(^d\) | RedHat   |
| A5        | PHP (No-label) | Apache     | MySQL (label 3) | CentOS   |
| A6        | PHP (label 1)\(^b\) | Apache     | MySQL (label 3) | Ubuntu   |
| A7        | PHP (label 1) | Lighttpd   | MySQL (label 3) | CentOS   |
| A8        | PHP (label 1) | Lighttpd   | MySQL (label 4) | Windows server |
| A9        | PHP (label 2) | IIS        | MySQL (label 4) | Windows server |

\(^a\) The key functions and keywords of PHP are not labelled.
\(^b\) The key functions and keywords of PHP are labelled by ‘1’.
\(^c\) The key functions and keywords of SQL are not labelled.
\(^d\) The key functions and keywords of SQL are labelled by ‘3’.

4.2 Analysis of attack surface resources

We can understand the attack surface of each subsystem by analyzing the process of file upload and SQL injection vulnerability exploitation. Attackers firstly upload the PHP Trojan file by file upload vulnerability, next, get the encrypted name of Trojan file through SQL injection vulnerability, then, trigger the uploaded Trojan file via URL, finally, remote control the target system by socket communication. Thus, the ASRSs of all subsystems are as shown table 2.

| Subsystem | Attack surface resources |
|-----------|--------------------------|
| A1        | Form PHP SQL URL socket command\(_{\text{Linux}}\) |
| A2        | Form PHP SQL URL socket command\(_{\text{Linux}}\) |
| A3        | Form PHP SQL URL socket command\(_{\text{Windows}}\) |
| A4        | Form PHP SQL\(^b\) URL socket command\(_{\text{Linux}}\) |
| A5        | Form PHP SQL\(^b\) URL socket command\(_{\text{Linux}}\) |
| A6        | Form PHP\(^a\) SQL\(^b\) URL socket command\(_{\text{Linux}}\) |
| A7        | Form PHP\(^a\) SQL\(^b\) URL socket command\(_{\text{Linux}}\) |
| A8        | Form PHP\(^a\) SQL\(^b\) URL socket command\(_{\text{Windows}}\) |
| A9        | Form PHP\(^a\) SQL\(^b\) URL socket command\(_{\text{Windows}}\) |

\(^a\) The key functions and keywords of PHP are labelled by ‘1’.
\(^b\) The key functions and keywords of SQL are labelled by ‘3’.
The key functions and keywords of SQL are labelled by ‘3’.

Before voting, the experimental web application system’s ASRS is expressed as:

$$ASRS_{\text{before}} = \{ \text{Form}, \text{PHP, PHP}_1, \text{PHP}_2, \text{SQL, SQL}_3, \text{SQL}_4, \text{URL}, \text{socket, command}_{\text{Linux}}, \text{command}_{\text{Windows}} \}$$

(10)

Then we can get the common mode situation of each attack surface resource according the table 2 and the equation (10), as shown in table 3.

| Number | Resources | $CR$ |
|--------|-----------|------|
| 1      | Form      | 9-CR_{1,2,3,4,5,6,7,8,9} |
| 2      | PHP       | 5-CR_{1,2,3,4,5} |
| 3      | PHP$_1$   | 3-CR_{6,7,8} |
| 4      | PHP$_2$   | —— |
| 5      | SQL       | 3-CR_{1,2,3} |
| 6      | SQL$_3$   | 4-CR_{4,5,6,7} |
| 7      | SQL$_4$   | 2-CR_{8,9} |
| 8      | URL       | 9-CR_{1,2,3,4,5,6,7,8,9} |
| 9      | socket    | 9-CR_{1,2,3,4,5,6,7,8,9} |
| 10     | command$_{\text{Linux}}$ | 6-CR_{1,2,4,5,6,7} |
| 11     | command$_{\text{Windows}}$ | 3-CR_{3,8,9} |

4.3 Measurement of the arbitrated attack surface

According the table 3, we give the DPERs of the all attack surface resources in $ASRS_{\text{before}}$, based on their the difficulty of exploited by attackers and the damage to the system. We divide them into three categories, including input vectors, active content, and access control [16], as shown in table 4.

| Categories    | Resources | DPER     |
|---------------|-----------|----------|
| Input vectors | Form      | {0, 1}   |
|               | URL       | {0, 1}   |
|               | socket    | {0, 1}   |
|               | PHP       | {0, 8}   |
|               | PHP$_1$   | {0, 6}   |
|               | PHP$_2$   | {0, 6}   |
|               | SQL       | {0, 5}   |
|               | SQL$_3$   | {0, 3}   |
|               | SQL$_4$   | {0, 3}   |
|               | command$_{\text{Linux}}$ | {0, 5, 10} |
|               | command$_{\text{Windows}}$ | {0, 5, 10} |

For the input vectors, we consider the DPER of Form, URL and socket as the value with weight 1, because these resources can’t cause direct damage, but they are used to match other attack mediums. For the active content, we think PHP’s DPER is higher than SQL, because Trojan file’s language is
PHP. And the DPERs of labelled PHP and SQL are lower than normal PHP and SQL, because labelled PHP and SQL are more difficult to be exploited. For the access control, The DPERs of command\textsubscript{Linux} and command\textsubscript{Windows} depend on user permissions of Linux and Windows, such as unauthenticated, authenticated and root.

According the table 3 and table 4, we can get that \( z = 11 \), in figure 2, when the voting algorithm is full-consistent voting, \( k = 9 \); when the voting algorithm is large-number voting, \( k = 5 \); when the voting algorithm is multi-mode voting, \( 2 \leq k \leq 9, k \in N^+ \). Therefore, we give the quantitative results of all subsystems’ attack surface, as shown in figure 3. And the trend of quantitative results of HRS’s attack surface with \( k \) is as shown in figure 4.

According the figure 3, the minimum attack surface for all subsystems is 17 in the authenticated access control. And the minimum attack surface for all subsystems is 22 in the root’s access control.

According the figure 4, when \( 9 \geq k \geq 5 \), the HRS’s attack surface is less than or equal to 21 in the authenticated access control. And HRS’s attack surface is less than or equal to 16 in the root’s access control. Moreover, when \( 9 \geq k \geq 7 \), the HRS’s attack surface is 3, which is the smallest, which is significantly less than the attack surface of every subsystem. That means that when \( 9 \geq k \geq 5 \), HRS is safer than every subsystem, when \( 9 \geq k \geq 7 \), the security of HRS is the highest.
5. Summary and future work
In this paper, we define the notion of the ASA, provide a method for quantify result of arbitration, introduce an attack surface metric suitable for HRS with the voting mechanism. Experimental results show that the new attack surface metric can properly describe the attack surface of a heterogeneous redundant system with voting mechanism. And when the voting algorithm is full-consistent voting, the security enhancement is largest. When the voting algorithm is large-number voting, the security enhancement is less. When the voting algorithm is multi-mode voting and the number of mode is greater or equal to \( \lceil n/2 \rceil \), the security enhancement will gradually approach the security enhancement under applying full-consistent voting as the number of mode increases.

Currently our attack surface metric can only apply to the sample voting algorithm, including multi-mode voting, large-number voting, and full-consistent voting, but it can’t support the complex voting algorithm. So we will analyze the characteristics of complex algorithms and improve existing metric to support the complex voting algorithms in the future work.

Acknowledgments
The authors thank Jiexin Zhang and Bolin Ma for their assistance and encouragement in this research. And this research is financial supported by the National Key R&D Program of China (Grant No. 2017YFB0803201, Grant No. 2016YFB0800104); and the Scientific Research Projects of Shanghai Science and Technology Committee (Grant No.16DZ1120502).

References
[1] Lin H S, Spector A Z, Neumann P G, et al. (2007) Toward a safer and more secure cyberspace. Communications of the ACM, 50(10):128-128.
[2] Computing Research Assoc. (2003) Four Grand Challenges in Trustworthy Computing. http://www.cra.org/reports/trustworthycomputing.pdf
[3] Mcgraw G. (2003) From the Ground Up: The DIMACS Software Security Workshop. Security & Privacy IEEE, 1(2):59-66.
[4] Prevost H. (1982) Babbage's calculating engines :. Tomash.
[5] Rea J. (1993) Boeing 777 high lift control system. Aerospace & Electronic Systems Magazine IEEE, 8(8):15-21.
[6] Briere D, Traverse P. (1993) AIRBUS A320/A330/A340 electrical flight controls - A family of fault-tolerant systems. In: International Symposium on Fault-Tolerant Computing. pp. 616-623.
[7] Yeh Y C. (1998) Design Considerations in Boeing 777 Fly-By-Wire Computers. In: IEEE International Symposium on High-Assurance Systems Engineering. pp. 64-72.
[8] Collinson R P G. (1999) Fly-by-wire flight control. Computing & Control Engineering Journal, 10(4):141-152.
[9] Wang, F., Jou, F., Gong, F., Sargor, C., Gosevapostojanova, K., & Trivedi, K. (2002) SITAR: A Scalable Intrusion-Tolerant Architecture for Distributed Services. In: DARPA Information Survivability Conference and Exposition. pp.153-155.
[10] Neves N F, Verissimo P. (2003) Complete Specification of APIs and Protocols for the MAFTIA Middleware. TECHNICAL REPORT SERIRS-UNIVERSITY OF NEWCASTLE UPON TYNE COMPUTING SCIENCE.
[11] Bangalore, A. K., & Sood, A. K. (2009) Securing Web Servers Using Self Cleansing Intrusion Tolerance (SCIT). In: International Conference on Dependability. pp. 60-65.
[12] Tong Q, Zhang Z, Zhang WH, Wu JX. (2017) and implementation of mimic defense web server. Ruan Jian Xue Bao Journal of Software, 28(4).
[13] Manadhata, P. K., & Wing, J. M. (2011) An attack surface metric. IEEE Transactions on Software Engineering, 37(3), 371-386.
[14] M. Howard, (2003) Fending off Future Attacks by Reducing Attack Surface, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure02132003.asp.

[15] Manadhata, P. K. (2013) Game Theoretic Approaches to Attack Surface Shifting. In: Moving Target Defense II. Springer New York.

[16] Heumann, T., Türpe, S., Keller, J. (2010) Quantifying the Attack Surface of a Web Application. In: Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. Berlin. pp. 5-7.