

Global Adversarial Capability Modeling

Jonathan Spring*, Sarah Kern*[†], Alec Summers*

*CERT® Program, Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA, USA

Email: netsa-contact@cert.org

[†]School of Information Sciences
University of Pittsburgh

Abstract—Intro: Computer network defense has models for attacks and incidents comprised of multiple attacks after the fact. However, we lack an evidence-based model the likelihood and intensity of attacks and incidents.

Purpose: We propose a model of global capability advancement, the adversarial capability chain (ACC), to fit this need. The model enables cyber risk analysis to better understand the costs for an adversary to attack a system, which directly influences the cost to defend it.

Method: The model is based on four historical studies of adversarial capabilities: capability to exploit Windows XP, to exploit the Android API, to exploit Apache, and to administer compromised industrial control systems.

Result: We propose the ACC with five phases: Discovery, Validation, Escalation, Democratization, and Ubiquity. We use the four case studies as examples as to how the ACC can be applied and used to predict attack likelihood and intensity.

Keywords-incident response, intrusion detection, intelligence, computer network defense, CND, modeling, security, cybersecurity

I. INTRODUCTION

The number of adversaries, and their skill level, attacking a network defender are key informational components to defensive planning. However, estimating the extent of adversarial capability has historically been difficult, especially in the cyber domain. This paper describes a model to better estimate and reason about global adversarial capability and the landscape of computer network attacks (CNA). The model identifies and clarifies trends in the progression of adversary capabilities and provides a clearer view of the global threat landscape.

Inspired by the intrusion kill chain of individual CNA [1], the Adversarial Capability Chain (ACC) model describes the progression of expertise in conducting CNA much more generally. The ACC models adversarial competency in a specific capability against a particular system. Understanding the state of the capability against a particular technology can help guide risk-based assessments about the wisdom of deploying that particular technology and decide what defenses can be considered adequate to resist an adversary's

capabilities. The ACC model defines “adversarial capability” and “system” carefully to optimize intelligibility, usefulness, and reliability.

The ACC model organizes adversarial capability into five phases: Discovery, Validation, Escalation, Democratization, and Ubiquity. Since adversarial capability is not directly observable, the analyst tracks status by observing symptoms that correspond to the phases. The symptoms have been derived both heuristically from cybersecurity expert opinion and analytically from economic and criminological principles.

Since each software system is idiosyncratic and there are relatively few such globally important systems, correlations cannot be derived experimentally. Therefore, this paper derives the ACC model from a series of case studies of existing systems. Four such case studies are included: Windows XP, Android user-facing application programming interface (API), Apache web server, and industrial control systems (ICS). The first three case studies explore and model the adversarial capability of system exploitation to gain unauthorized control over the software system in question. The ICS case study does not focus on a particular software system; instead, it focuses on the adversarial capability to leverage cyber-physical systems more generally rather than focusing on the exploitation capability of a particular software system.

The ACC is a chain in the same way that the intrusion kill chain is a chain—adversarial progress per system is unidirectional (i.e., monotonically increasing). Discovery of vulnerabilities in a particular system is inherently so; exploits are never forgotten. This one-way movement makes the capability chain easier to interpret. The rankings given are minimums until further observations ratchet the adversarial capability up the chain. The complex state of software systems muddies this issue somewhat. More than one system protects any individual information item, so multiple adversarial capabilities against multiple systems impact how well a target is protected. The model and framework presented here can be used to clarify this complex ecosystem; however, the model is not an automatic solution to this highly complex

problem. The model provides a tool to improve analyst conclusions and help reduce error.

The expected gains from employing ACC modeling of threats to defender systems is primarily to extend the time window in which accurate predictions can be made about cyber events and capabilities. This is a critical computer network defense (CND) planning function; if an expected system lifetime is five years, it should be deployed with the ability to defend against the *expected* adversarial capabilities in that time frame, not the *present* adversarial capabilities. At present, reliable predictions about global adversarial capabilities are much shorter than expected system lifetimes, and this makes such forward-thinking planning impractical. The ACC model helps reverse that trend.

II. MOTIVATION

There exists a theme in the historical progression of attacks against software systems. It is not a matter of *if* any particular attack capability becomes widely accessible after it is first developed, but *when*. Different software systems are usually attacked in idiosyncratic ways, so extracting the consistent elements from the examples is challenging. Yet if the patterns within the historical progression could be extracted, analysts could improve their understanding of how adversarial capability progresses and better inform decision makers in deploying technologies and defenses. Adversarial capability to attack a software system is proportional to the amount of effort a defender must spend to prevent the system from being compromised.

The purpose of modeling is to provide a useful representation of reality that improves analyst performance [2]. In doing so, some simplifications are necessary; otherwise the model becomes as useless as a 1:1 map. Models involving engineered mechanisms such as computer systems are difficult due to their dynamic and complex nature [3]. The Adversarial Capability Chain (ACC) model is coarse grained, so some stark simplifications must be made. However, one benefit of abstraction is to move far enough from specific software systems to avoid the changeable nature of software and model something more constant: human expertise, expertise growth, human communities, and economic drivers. These concepts are roughly what the ACC models as *adversarial capability* when paired with a specific capacity, such as system exploitation. (See Section IV-A for a more precise definition of adversarial capability.)

Although the ACC models human capability, it is intimately tied to a software system. The adversarial capability to affect a particular software system does not directly transfer to another system. Therefore, the definition of what is considered one “system” is critically important. The rough heuristic the model uses defines a system as the unit of code that can be exploited. This definition is made more precise in Section IV-C.

Once concepts are adequately defined, the remainder of Section IV fleshes out the model. Section IV-D explains the organization of the model and specifies the five phases of adversarial capability: Discovery, Validation, Escalation, Democratization, and Ubiquity. Section IV-E introduces and defines the method for visualizing the ACC.

There exist processes for creating a resilient risk-management capability within an organization [4]. There are many different definitions of risk, and evaluating and communicating risk can be challenging [5]. In general an adversary needs both access and capability to execute an attack; a traditional risk assessment estimates risk of attack as a function of both. With the global connectivity of the Internet, any addressable system can access any other. Thus the risk of some attack is primarily a function of only adversarial capability, since all relevant adversaries have access. Thus the ACC helps start an accurate process for a principled risk assessment of an Internet-connected system.

The ACC model has been derived both heuristically from cybersecurity expert opinion and analytically from economic and criminological principles, including our prior models of malicious Internet behavior [6, 7, 8]. These derivation methods are informed by detailed case studies of adversarial capability against different software systems. Section VI includes several case studies using the ACC model; these case studies are presented to demonstrate the validity of the model and provide an example of how to apply the ACC to a given software system. The systems subject to case study are Windows XP, Android user-facing application programming interface (API), Apache web servers, and industrial control systems (ICS).

III. RELATED WORK

Several models of attacker behavior and purported attack prediction exist. They range widely in the subject matter used to characterize attacks, and come from both academic and industry sources. Watters et al. developed a Cyber Attacker Model Profile (CAMP) to characterize and predict cyber attacks via social characteristics within different cultures [9]. Qin and Lee presented an approach to predict short-term attack strategies by clustering and correlating IDS alerts [10]. The anti-virus company Sophos has researched the exploit skills of specific malware author groups, providing insight into individual attacker groups [11]. The ACC offers a complementary view of attack prediction by modeling global adversary capability that is at once technically grounded and broad-scope.

The ACC model enhances the existing analytical tool box. The current tool box for a computer network operations (CNO) analyst includes a common language for computer security incidents [12], intrusion kill chain models [1], and the diamond model of intrusion analysis [13]. The existing models provide fine-grained or narrowly scoped intrusion and attribution analysis. However, there is no wide-lens,

context-based perspective model of the whole adversarial ecosystem. The ACC proposed here fills this gap. The results of ACC modeling provide higher level conclusions appropriate for decision makers as well as context for analysts using the more fine-grained analytical tools.

IV. ADVERSARIAL CAPABILITY CHAIN

The ACC describes, analyzes, and models the state of global capabilities for CNO against a particular software system. The ACC provides a method for articulating global security policy goals as well as organizational technical controls. For example, national security interests may make it important to delay or prevent the development of certain capabilities against certain technologies that would be especially pernicious if available to particular kinds of adversaries.

The state of global adversarial capability to accomplish any particular task is a complex information item. The ACC model breaks down this complex concept with specific descriptions of what is being modeled by “actors,” “adversarial capability,” and “software systems.” The model draws on existing definitions as much as possible; however these terms do not have satisfactory existing definitions, so they are elaborated in Sections IV-A through IV-C.

Section IV-D describes the ACC model, which is organized into five phases that indicate a certain level of adversarial capability: Discovery, Validation, Escalation, Democratization, and Ubiquity. Each phase is indicated and established by the observation of particular symptoms. Section IV-E explains the method of visualizing the progression of these phases and symptoms over time for a particular software system. Example graphics are included in the appendix.

The ACC was informed by heuristic expert opinion developed through analysis of years of public reporting about computer security. After hypothesizing the model based on expert opinion, it was applied to several case studies to test its robustness and usefulness. These varied case studies have generally upheld the model and seem to indicate it is not biased to any particular software system (see Section VI). However, we offer a note about potential bias: Windows XP holds a special place in this model’s development. Since so much of the security history of the last decade revolves around Windows XP machines, that system directly and indirectly influences expert opinion. Model development addresses this by explicitly considering XP as a key case study. However, the goal of the ACC is to find insights that are constant across software systems, not just to model what happened to XP. Therefore, additional case studies were vital in demonstrating the external validity and generalizability of the ACC.

A. Actors

In the model, the word *adversary* does not refer to any particular actor. The goal of the model is to be agnostic in terms of particular adversaries. Since attribution of cyber-attacks is so difficult, this approach helps a defender devise defenses according to a principled risk assessment without needing to know a specific adversary, and with only having to know the state of adversarial expertise globally.

The model accounts for the creation of expertise and the transfer of expertise to other actors groups. For example, when expertise becomes accessible to rogue nation-states or organized crime, a progression in the adversarial capability model reflects that. “Adversarial capability” is defined in the following section. When new adversaries gain access to particular capabilities, it changes the global character of attacks using that capability (such as the capability to exploit a particular software system).

Although it is not the primary goal of the model, the ACC can assist attribution. In some cases, actor intent or identity may be inferred from the existing global capability. The model can identify when only well-funded, well-organized actors have a particular capability, for example. Therefore, if a defender notes an attack utilizing such a capability, it helps identify the adversary as well resourced.

B. “Adversarial Capability”

A capability is the synthesis of expertise and physical resources. A mechanic only has the capability to fix a vehicle if she has both the knowledge and the tools to fix it, not just one or the other. Adversaries may have many capabilities, and each is separable from the others. The ability to exploit a particular system does not necessarily include the ability to covertly administrate that system or to use it to extract and launder money. Therefore, to maintain clarity and intelligibility, the ACC only models one adversarial capability at a time.

For an analyst to investigate correctly, one must know the capability benchmarks for which to look. The ACC can model any particular adversarial capability that is properly specified, but it is tailored for one in particular: the capability to exploit and gain unauthorized control of a given software system, as defined in Section IV-C.

The ACC is tailored toward system exploitation for two reasons. First, unauthorized control and use of a software system (i.e., an “unauthorized result”) is a necessary component of any computer security incident [12], and it is what the intrusion kill chain identifies as the critical step in any incident [1]. Therefore, a model of exploitation capability will be relevant to all computer security events. Second, there are many different software systems, and the capability to exploit each is separable; that is, if an adversary has the capability to exploit Windows XP it does not directly impact the ability to exploit Apache or Android. Therefore, the seemingly simple capability of system exploitation is

frequently reused and re-specified for different software systems. It is sensible to tailor the model to such a frequently exercised use case.

For example, the case studies in Section VI examine four distinct global adversarial capabilities: exploiting Windows XP, exploiting Android user-facing API, exploiting Apache web servers, and exercising remote control of cyber-physical systems such as ICS. Note the fourth capability does not address exploitation, but addresses control of a class of system after exploitation of the specific software system has occurred, demonstrating the flexibility of the ACC model.

C. “Software Systems” and Scope

Defining what counts as a single software system has been the most difficult modeling choice. This difficulty is exacerbated by the lack of consensus on a precise definition. Some examples considered include:

- “A system of intercommunicating components based on software forming part of a computer system (a combination of hardware and software) [14, 15].”
- A software system “consists of a number of separate programs, configuration files, which are used to set up these programs, system documentation, which describes the structure of the system, and user documentation, which explains how to use the system [16].”

However, none of these definitions are suitable in describing what software system means as the target of adversarial exploit and control. To capture the relevant modeling aspects, the ACC model uses the following definition for software system: *the set of software instructions that executes in an environment with a coherent function and set of permissions.*

A couple key aspects of this definition are essential to successful modeling within the ACC. A software system executes; that is, it takes actions. Execution is a critical component because these actions are what the adversary manipulates. The software system resides in a particular environment; this helps scope the system down to a manageable size and place. A software system has a coherent functionality, or particular purpose; this purpose is often what the adversary is attempting to co-opt. Most importantly, a software system executes with a logical set of permissions that mark the system off as a related bundle of code that is permitted to act in a specified space. In the context of adversarial exploitation, this emphasis on coherent permissions means that if an adversary escalates privileges through an attack, they have moved into a new software system.

Examples of different software systems include the Windows XP operating system, the Android user-facing API, the Android kernel, the programmable logic controllers (PLC) on microcontrollers in cyber-physical systems, Cisco IOS running on networking devices, the Linux OS, Firefox or Chrome web browsers, standardized Internet protocols such as TLS or DNS, database user-space, database administrator space, and applications such as Apache or BIND.

The peculiar status of Windows XP as a unitary system deserves comment. Since XP is an important case study, any oddities within it could inadvertently corrupt the model. However, unlike most other operating systems, the compromise of an application with user privileges for all intents and purposes also compromises the operating system. That is, all functions on an XP machine run with the same set of permissions. This permission configuration is not true on other modern operating systems. Therefore the analysis of capability to exploit XP is oddly simplified in that any common user-space application feeds in to expertise to compromise the OS; this is not the case for most systems. A second peculiarity is that Internet Explorer, though a web browser, “is a Windows feature, [so] you can’t uninstall it [17].” Thus, successful attacks against IE are simultaneously attacks against the operating system itself. This fits with the definition of *software system* used above and throughout the model, but it may be counterintuitive or surprising for analysts that whether a web browser is part of the software system depends on which operating system and browser are involved.

Systems on the Internet are rarely independent, and so advances against one system may also create advances against other systems. These interdependencies make it difficult to tease apart an accurate adversarial capability model. The focus is on a system at the appropriate level of granularity, “Windows XP,” not “Windows.” In some cases, Windows has had defensive systems introduced, such as Address Space Layout Randomization (ASLR) [18] and Data Execution Prevention (DEP) [19], and one could view the procession of adversarial capability against these particular system overlays as distinct from that of Windows itself. Although this adds some complexity in multiple, overlapping, concurrent systems being tracked, it creates additional explanatory power because it clearly demonstrates the one-way, incremental increases against each system. Instead of modeling that introducing ASLR rolls back the adversarial capability against Windows, the adversarial capability against ASLR starts fresh and develops as any other system. The success of a single adversary against an information target is a function of the adversarial capability against all software systems in between the adversary and the information target. Estimating this function is an area for future work in risk assessment.

D. Phases of Adversarial Capability

The ACC consists of a series of five *phases*: Discovery, Validation, Escalation, Democratization, and Ubiquity. The phases indicate what kinds of actors and expertise are available in the world with the capability in question. The phases should begin in sequence; this has been observed in each case study completed so far, and the model posits that this sequencing will hold for the analyses of other software systems. Each earlier phase beginning is a prerequisite for the following phase to begin.

To better track and predict time to the next phase, the model uses multiple *symptoms* as indicators of a phase. These are common observables that indicate that a specific adversarial capability exists at a certain point in time. A phase *begins* once one symptom is observed. After all symptoms in a phase are observed, the phase is *established*. The prior phase need not be established before the next phase begins; however, if a phase is established, the following phase is more likely to begin sooner than would otherwise be the case—if it has not started already.

The phases do not end per se; the expertise signaled by a particular phase remains when the next phase begins. The subsequent phases indicate an expansion of expertise and actors with it. The time between the beginning of any particular phase and the next one is elastic and may change depending on the software system and incentives for the adversary community.

There are several possible explanations for failure to observe a symptom. Simply, the symptom may not have occurred. More likely, the lack of observation may be due to the covert nature of adversaries and the difficulties of detection. Moreover, the date of symptomatic actions may only become known well after the fact. Symptoms are imperfect indicators, as knowing a symptomatic event may depend on defender detection and public reporting, both of which are imperfect.

One goal of providing several past case studies is to improve future prediction of the rate of adversarial capability growth. Using past history of the rate at which phases are established, an analyst can predict future progression of symptoms and phases. Although such predictions of future development are difficult, the ACC provides a framework for the evidence-based estimation that is currently lacking.

Table I summarizes the phases and their symptoms. The detailed definitions and implications of each phase follow along with recommendations for detecting their symptoms.

1) *Discovery*: The initial phase of the ACC model is characterized by the observance of two symptoms:

- First Published Vulnerability/Exploit of the Software System
- Targetability

There is typically little to no economic gain in successful exploitation at the Discovery stage, however reputation gain is a common incentive. Defenders should expect target systems to be chosen opportunistically.

Since computer security has industrialized and information technology has become more widespread, this phase tends to be short. Further shortening this phase in exploitation-related capabilities, researchers have enumerated the classes of information system weaknesses [20], making discovery easier for adversaries.

The first symptom is self-explanatory; the first known vulnerability and the first known exploit are clearly a symptom

of discovery of adversarial capability to exploit the system. The other symptom in this phase is the extent to which a software system is a target. The targetability of a system is the ability of adversaries to target a system with the developed exploits. Targetability covers the general concept that a vulnerability is not a threat unless an adversary has a genuine ability to make use of the vulnerability [21].

Targetability is a flexible concept that can apply usefully across different software systems. For example, Apache web servers must listen for web requests from the general Internet and therefore can be targeted via automated scanning. Alternatively, the Android user API is targeted via leveraging user action since it is primarily user-facing.

Targetability is related to the number of users of a system: if a system is used by few, then there is little to target. Targetability also relates to whether the software system is deployed in an environment to which an adversary can effectively deliver the available exploits.

2) *Validation*: The validation phase is concerned with demonstrations that developing the capability in question is worthwhile or material (i.e., “of serious or substantial import; significant, important, of consequence” [22]). The following phase is partly indicated by the formation of organized, well-funded actors and efforts to deploy the capability for profitable endeavors. Validation is indicative of adversarial capability development because material benefit from developing a capability, such as exploiting a particular software system, needs to be demonstrated and validated before adversaries will spend the effort to develop and plan the capability.

Sometimes, dates of occurrence can only be estimated in retrospect due to the secretive nature of development in this phase. The first public disclosure may be both the first time evidence has come to light and contain all the information to the signal the phase is established. Defenders should expect “important” assets to be targeted, though unimportant assets are likely not targeted at all. Politically sensitive or otherwise high-value targets will be targeted persistently by highly motivated adversaries during this phase. This phase consists of three symptoms:

- Ability to Inflict Electronic or Physical Damage or Disruption
- Monetary Gains
- Ability to Conduct Economic or Government Espionage

3) *Escalation*: In this phase, various actors recognize the value of the pursuit of exploits sufficiently such that well-funded organizations may advertise their intent to acquire expertise in a particular software system. While there may be evidence of increased reporting of CNA against a system from an increased number of actors, the results of any development done during this pursuit are generally not public. Moreover, techniques and methods of exploitation are difficult to obtain outside of well-funded and organized

Table I
PHASES AND SYMPTOMS IN THE ADVERSARIAL CAPABILITY CHAIN (SYMPTOMS ARE NOT EXPECTED TO HAPPEN SEQUENTIALLY.)

Phase	Symptom I	Symptom II	Symptom III
Discovery	First Vulnerability/Exploit	Targetability	N/A
Validation	Disruption Abilities	Monetary Gain	Espionage Abilities
Escalation	Indefensible Attacks Observed	Remote/Automated Attacks	Well-Funded, Organized Actors
Democratization	Cheap Exploits & Attack Delivery	Cheap Control Software	Cheap Infected Infrastructure
Ubiquity	Open Source Attacks	Open Source Control	End of Support

actors. The economic viability of earning profit from exploitation of the software system is tested during the Escalation phase, and defenders should expect “important” assets to be targeted relentlessly. Economic assets will be targeted opportunistically. There are three symptoms observed in this phase:

- Observation of Indefensible Attacks (e.g., zero-day exploits)
- Remote/Automated Control Available (at any cost)
- Existence of Well-Funded, Organized Actors

These symptoms are not always indicated by a single event. What makes an actor group “well-funded” and “organized” is a matter of degree. There are several other symptoms that are a matter of degree in the following phases. Such symptoms should be supported by multiple corroborating events, rather than a single point event.

4) *Democratization*: The technology for CNO against a software system becomes widely available during this phase, although its possession is generally illegal or otherwise discouraged. Exploitative technology is of sufficient value that it must be purchased, but the price has dropped to what is feasible for a middle-class individual. The social structures from which to acquire the resources are sufficiently permeable that they can be found and integrated into CNO without much prior knowledge. CNO against the software system in question can range from aiming to settle petty disputes to directed use for more wanton acts. There are no longer relevant central points of control for dissemination and development of exploits and expertise. The economic viability of profit via CNO against the software system is proven, and defenders should expect economic assets to be targeted relentlessly. There are three symptoms in this phase, and they are defined by falling costs in three areas:

- “Cheap” Exploits & Attack Delivery
- “Cheap” Control Software
- “Cheap” Infected Infrastructure

5) *Ubiquity* : In this phase, proven economic value of CNO against the target software system leads to widespread exploitation and viable economies in support of activity. Specialization occurs within the adversary community as management of assets for CNO grows beyond the scope of individuals. Malicious software is freely available while specialized support and management services are not; this is analogous to any other economic growth. Defenders should expect the software system to be compromised simply for

the sake of being compromised; the actors often seek to simply spread their malicious software and sell access to the compromised hosts as a general purpose service in their own right. The defender will likely find it impossible to resist any well-funded adversary, and even poorly resourced actors will provide significant challenges. There are three symptoms detected in this phase:

- Open Source (i.e., free and easily accessible) Attacks
- Open Source (i.e., free and easily accessible) Control or Bot Software
- End of Support & Security Updates

E. Visualization

For a better understanding and application of the model, the symptoms and phases of capability are illustrated in a timeline. A visual representation of the ACC provides a clear reference for understanding the trajectory of capability growth, as well as the scaffolding nature of incremental developments against a particular software system.

Each symptom in the model is visualized by a box that transitions from white (no observed evidence) to dotted gray (moderately observed) to solid gray (significant evidence of a symptom), where most of the symptoms are (1) significantly represented by a single event and (2) skip the moderately observed period altogether. The final symptom observed in a phase is colored dark gray, signaling that the phase is established. After this time, all symptoms of that phase are colored dark gray.

The charts also display the market share held over time for the XP, Android, and Apache models. The market share data is represented as a percentage of the total market as reported in public sources over time. The charts include the market data to indicate the popularity and deployment of the software systems in question. Popularity and deployment are tangentially related to some symptoms and phases. For example, it is necessary but not sufficient for targetability that targets are deployed.

At this time, we have not found a direct correlation between market share and advancing phases of adversarial capability reached. However, it is also not clear that market share or total deployed instances does not influence progression. Therefore, market share is included in the visualizations to inspire questions and future work. Global market share data for Windows XP machines comes from Tech Talk [23], OneStat [24], NetMarketShare [25], and

Net Applications [26]. Android smartphone global market share data comes from Canalys [27], Gartner [28], International Data Corporation (IDC) [29], and Strategy Analytics [30]. Apache market share for all sites is gathered from Netcraft [31].

The appendix contains visualizations of the heuristic assessment for each software system assessed.

V. BENEFITS OF THE ACC

The ACC provides a new perspective on the threat landscape by focusing on trends in adversarial capability growth throughout the lifetime of a software system. The ACC contributes a wide-lens, context-based perspective model of the whole adversarial ecosystem. Actor capability is shown to develop in a pattern of phases and competency passes from skilled and well-financed adversaries to those with fewer resources.

A new context is developed that supports CND decision making by viewing these developments as trends instead of a series of individual efforts from specific actors. The patterns within the historical progression improves analysts' understanding of how adversarial capability progresses and in turn, their capacity to inform decision makers in deploying technologies and defenses. Evidence-based predictions improve risk-based assessments in defensive planning by extending the time frame of reliable predictions about adversary capabilities. Integrating the analysis results and timeline of actor development from the ACC model improves analysts' abilities to forecast capability growth, and thus the threats they will face in the near and long term.

The ACC model is intended for application in the evaluation of expected adversarial capability. While it is helpful to understand the present adversarial capability related to a technology, the level of capability cannot be reversed. A technology may be patched to stop exploitation of a specific vulnerability, but it does not reverse the capability level reached, and the effects of an exploit cannot be erased. This continual adversary improvement makes defense difficult, but at least defensive measures should be based on the foreseen adversarial capability for the lifetime of a technology.

To reduce an adversary's abilities to target defender resources, additional technologies can be deployed. As discussed in Section IV-C, ASLR and DEP are changes to the Windows operating system that helped resist adversary attacks. This is modeled by considering DEP and ASLR as different software systems that the adversary must build capability against. So while the adversaries' capability against Windows XP is very advanced, if the system properly implements DEP and ASLR then the adversary's ability to achieve their ultimate goal is reduced. The likelihood that adversaries will achieve their objectives¹ is a function of the minimum of the capability against any of the software in the actual attack

path. Therefore, the likelihood an adversary succeeds may be reduced by adding defense in depth; however the adversarial capability level cannot be reversed or erased for any specific software system. Thus introducing defenses and removing highly targeted software are both effective methods for increasing CND readiness. ACC analysis improves the time horizon at which these predictions can be made reliably, thus increasing the time window for preparing a defense.

VI. CASE STUDIES

By utilizing the ACC model, assessments can be developed regarding the state of the threat landscape against various software systems. The following assessments are qualitative and are supported by observational evidence in reputable public reporting. Although the model can apply to the capability to exploit all software systems, space is limited, so case studies have been carefully selected to provide key insights. The four software systems selected provide insight into contrasting states of actor capability, from widely accessible to actively developing to highly specialized. This broad spectrum provides many key strategic examples on which the model is informed and demonstrates the external validity and generalizability of the ACC model. The case studies present a study of exploitation capability against Windows XP, Android user-space applications, and Apache web servers; and a study of control capability of embedded cyber-physical systems in ICS.

A. Windows XP

For the majority of its supported lifetime, Windows XP was the market leader in desktop operating systems. In 2006, five years after its initial release, it controlled over 85% of the market [32]. XP is examined in detail because it has completed its entire life cycle and it has been highly targeted by adversaries. This case study provides a clear picture of the ACC framework in its entirety and can provide clues to the path that other systems will take as adversarial capability increases.

1) *Discovery*: Windows XP, released by Microsoft to the public in October 2001, initially shipped with vulnerabilities that carried over from code shared with its predecessors. The first example of this was the Universal Plug and Play (UPnP) vulnerability that was reported in December 2001 [33] and was followed quickly by the public release of an exploit for it in January 2002 [34]. These events signify the "First Vulnerability & Exploit" symptom.

In December 2001, the Goner worm targeted Windows machines, including XP [35]. While no significant damage was done, Goner shows evidence of "Targetability" of XP because so many devices were able to be targeted and infected automatically. Evidence of this symptom established the Discovery phase.

¹We follow Howard and Longstaff [12] in our definition of *objectives*.

2) *Validation*: In August and November of 2002, reports revealed the first evidence of sophisticated actors pursuing CNO against Windows XP for monetary gain. The rise of Shadowcrew, a criminal gang focusing on the stealing and selling of customer credit data [36], occurred in August 2002, though it was not well reported until after it was taken down by the United States Secret Service (USSS) in October 2004. The disclosure of this criminal gang shows partial evidence of the “Monetary Gains” symptom of the Validation phase.

The Russian Business Network (RBN), a malicious ISP with alleged ties to the Russian government [37, 38], began operations in November 2002. The operation of multiple criminal actors around the Windows XP exploitation ecosystem provides strong proof of worth in attacking the system, thus existence of the RBN further evidences the “Monetary Gains” symptom.

In August 2003, the Blaster Worm demonstrated adversaries’ ability to build a botnet that could disrupt system availability [39]. The functionality of this worm indicates the “Damage” symptom. There is no evidence that the Blaster Worm was deployed for more sophisticated attacks than denial of service, thus the assessment of its significance is limited to proof of disruption.

In September 2003, the first evidence surfaced detailing a Chinese espionage campaign [40]. The adversaries infiltrated the U.S. Department of Defense and several of their major contracting partners. According to a source list from a U.S. Army memorandum, the National Security Agency (NSA) had version 1.1 of a guide to securing Windows XP in December 2003 [41]. Due to the sensitive nature of the incident, the systems involved are not certain; however, since the military was using XP, it is plausible and likely that XP was compromised in this attack. Evidence of this campaign represents the “Espionage” symptom, and establishes the Validation phase.

3) *Escalation*: The growth of the Zombie King botnet (via XP vulnerabilities) and its sale on underground e-markets in mid-2004 illustrates specialization of actor expertise and a specialized economy [42, 43]. The launch of the ZeuS malware in June 2006 further demonstrated this as well [44]. These two events evidence the symptom “Remote/Automated Control.”

In August 2005, open source reporting showed the spread of a keylogging malware via a zero-day exploit of Windows Internet Explorer versions 6, 7, and 8 (version 6 was standard with XP) [45]. The software used in this attack is believed to have been professionally designed. Since Internet Explorer is built in to Windows systems (see Section IV-C) and XP controlled the majority of PC market share at the time, it is deduced that this event represents the “Indefensible Attacks Observed” symptom in this phase.

The beginning of APT1 in January 2006 shows arising evidence of the symptom, “Well-Funded, Organized Actors.”

This adversary was shown to have targeted XP [46] and exhibited a high level of sophistication [47].

In October 2007, it was reported that the RBN had been advertising hosting services for \$600 (versus the industry standard at the time: \$60) [48]. This illustrates the willingness by malicious individuals/organizations to pay more for hosting services that disregarded nefarious activity, and thus shows additional observation of the “Well-Funded, Organized Actors” symptom, establishing the Escalation phase.

4) *Democratization*: In May 2008, open source reporting showed black market forums were offering the ZeuS code for hire [49]. This provided middle-class actors the tools needed to perform advanced attacks, thus allowing for widespread commercialization of expertise against XP [50]. In addition, the SpyEye banking Trojan emerged in January 2009, with over half of the computers it infected running XP [51]. These two events in conjunction with one another represent evidence of “Cheap Control Software.”

In October 2008, the Conficker Worm gathered any victim machines that could possibly contribute to a botnet [52]. This gathering of infected machines shows evidence of the symptom “Cheap Infected Infrastructure.” Furthermore, Conficker indicated a shift to low-value targets being pursued relentlessly, as opposed to opportunistically.

In August 2010, the Blackhole Exploit Kit (BEK) surfaced and could be leased to deliver malicious payloads toward Windows systems and applications [53, 54]. For a relatively low price, the toolkit provided an easy way to carry out an attack and offered capability for customization. Thus, the existence of the BEK evidences “Cheap Exploits & Attack Delivery,” establishing the Democratization phase.

5) *Ubiquity* : The Ubiquity phase is characterized by freely available malicious software and widespread exploitation. In May 2011, the ZeuS code was first made public [55, 56]. This event indicates the “Open Source Control” symptom.

Seven months later, the Metasploit framework was released publicly [57], allowing for non-skilled and low-funded actors to carry out sophisticated attacks easily. This event indicates the symptom, “Open Source Attacks.”

The phrase “End of Support” refers to when a technology’s maintenance and support system stops operation. In the case of XP, Windows stopped supporting it in April 2014 [58]. At this point, CNA against the system completely lack response, and updates for zero-day exploits remain unpatched. The Ubiquity phase is completely established. Adversarial capability is essentially complete against Windows XP.

B. *Android*

Android user-facing API malware has become well developed in the past few years, and is probably tracking to Windows XP PCs circa early 2009. Many exploits are

opportunistic attacks that require the user to be fooled into installing something malicious. This approach is fruitful, and the adversaries appear to profit from the endeavor. Since the Android platform has proven some profit potential in the underground economy, it is probable that it will experience increased focus by adversaries, just as XP did, until the point where Android user-space malware is much more pernicious. This development of global adversarial capability can probably be expected within one or two years, since the progression of CNA capability against Android user-space has generally progressed faster than it did against Windows XP PCs.

1) Discovery: In October 2008, within a week of the Android OS launch, the first vulnerability was discovered. When exploited, the buffer overrun flaw afforded adversaries the capability to hijack a mobile device's web browser [59]. This flaw exhibits "First Vulnerability & Exploit."

The targetability of the Android user-space software system is multi-faceted, although it was established in each of these facets rather early. These targeting methods include using the cell network, messaging services, and legitimate apps in the application store.

In May 2009, two separate DoS issues exhibited capability to disconnect devices from the cellular network and suddenly restart devices, respectively [60]. These attacks represent the ability to target phones remotely over the cell network to cause the DoS condition.

Other phone software systems have been leveraged to spread malware via MMS without human interaction, such as Symbian in 2005 [61]. There is no evidence that Android messaging apps were vulnerable in this way. In this case, the software system is known to be targetable using the messaging service; however, there is no known vulnerability in this facet of targetability.

Android apps present a requested set of permissions to the user on installation. This is problematic because applications may be granted excessive permissions by naïve users. Further, the app store approval process did not pre-screen apps for malicious content until 2012 [62]. In November 2010, researchers demonstrated a flaw in the Android permission granting process [63] that allows an application with justifiable permissions to covertly download arbitrary apps with any permissions without requiring any user interaction [64]. The new apps then have ability to steal contacts, send premium rate SMSs, and more. This proof-of-concept app demonstrates the Android user-facing system is targetable via apps and the app store.

The combination of these "Targetability" symptoms demonstrate that the system was thoroughly targetable by November 2010, though it had been targetable in some ways for much longer. This combination of exploits and targetability evidence indicates the Discovery phase was firmly established by late 2010.

2) Validation: When exploited, the previously mentioned DoS flaws demonstrated ability to damage Android devices [65]. In addition, the launch of a DoS tool utilizing Android phones in October 2011 showed that defenders could simulate these types of attacks on web servers via phones as well [66]. These two events are sufficient for the "Damage" symptom.

In August 2010, the first SMS trojan was found on Android phones [67]. Disguised as a media player app, the malware surreptitiously sent SMSs to premium rate numbers, transferring money from the user to the cybercriminals. This trojan represents "Monetary Gains."

In March 2011, the first major malware was found inside the Android Marketplace [68, 69]. Embedded within seemingly legitimate apps, was malicious software capable of stealing sensitive information and relaying it back to command and control servers. In addition, in June 2011, it was discovered that malware on alternative app markets showed escalation of privilege capability for gaining root access on Android phones and also showed bot-like functionality [70, 71]. These two events exhibit "Espionage" and establish the Validation phase.

3) Escalation: In March 2011, it was reported that Android devices were being compromised with a mobile version of the SpyEye malware [72]. This malware partially represents the "Well-Funded, Organized Actors" symptom. In addition, the Zitmo Trojan, a mobile variant of Zeus, was found on Android devices in July 2011 [73, 74]. Zitmo shows evidence of "Remote/Automated Control" and combined with SpyEye, evidences the "Well-Funded, Organized Actors" symptom for Android.

In November 2011, zergrush, a zero-day buffer overflow exploit, was discovered and exclusively targeted the Android user-space [75]. While a patch was quickly released, the exploit was posted online for others to target unpatched devices. This event indicates "Indefensible Attacks," establishing the Escalation phase.

4) Democratization: AndroRAT, the first reported open source RAT (Remote Access Trojan) for Android, was released in November 2012 [76, 77]. AndroRAT allowed less capable actors to automate the infection process and control Android devices remotely. AndroRAT represents "Cheap Control Software."

Also in November 2012, the Android Framework for Exploitation (AFE) was launched [78, 79]. Like Metasploit for Windows XP, AFE was a framework for compromising a technology in an open source manner. Initially, no exploits came with the framework; however, automated exploits have been publicly uploaded since. While AFE will certainly lead to symptoms of "Open Source Attacks" in the Ubiquity phase, at the time, it indicates "Cheap Exploits & Attack Delivery."

As of yet, there has not been public reporting of the sale or rental of botnets made up of Android phones. At that

occurrence, the Democratization phase will be established.

5) *Ubiquity*: Android user-space API has not evidenced any symptoms of this phase.

C. Apache

The Apache web server is the longest running technology for which we performed a case study. It has been the world's most popular web server for the majority of its lifetime, hosting more websites than any other system every year since 1996. While its market dominance has begun to shrink in recent years [31], it remains a viable target for CNA. This case study explores the development of adversarial capability against Apache web servers, and estimates that its life cycle is tracking to Windows XP around 2007.

1) *Discovery*: In March 1996, the first publicly reported Apache web server vulnerability and exploit emerged [80]. Due to an input validation error, the vulnerability allowed unauthorized read access to files on the server [81]. This event indicates the symptom, "First Vulnerability & Exploit."

Web servers are fundamentally designed to be searchable and to "listen" for incoming requests. With the presence of known vulnerabilities, the development of tools to quickly scan for vulnerable web servers would indicate the capability to target the software system. In a 2007 paper, researchers showed a dramatic increase of scanning traffic in early 1998 in their data set of web logs collected at the border of a national laboratory address space between 1995 and 2007 [82]. Nmap, released in September 1997, was an attempt to collate the efforts of various developers of port scanners around this time into a flexible, open-source scanning tool [83]. We judge that these events indicate "Targetability" of Apache web servers, and thus establish the Discovery phase.

2) *Validation*: From June to August 2002, two malicious worms were discovered: Scalper and Slapper [84, 85]. Each malware family exploited a vulnerability in Apache web servers and sought to establish remote control capability. Adversary intentions are mixed, and these efforts show evidence of two symptoms in this phase: "Damage" and "Espionage." Moreover, the exploit code to capitalize on the Apache vulnerability that Scalper leveraged was posted online by a security group after they grew frustrated with the lack of response on their discovery [86, 87]. While this is a single event of cheap exploits, it does not show advancement of adversarial capability typical of the Democratization phase.

We judge that Scalper and Slapper indicate the symptom of "Monetary Gains," although not directly as observed and reported. It has been reliably reported that web servers are commonly compromised to host illegal content for profit [88, 89]. Unfortunately, due to both the relatively large amount of elapsed time since the late 1990s and probable reluctant victim reporting, there is no reliable, concrete public reporting on this subject from this time period. However, it is reasonable to impute that illegal hosting was ongoing in the late 1990s and early 2000s. There

certainly was adult content for sale on the Internet in the late 1990s. There is no indication that the "business model" of hosting questionable or illegal content on compromised web servers is a new invention. Therefore, we judge that this behavior started sometime in the late 1990s, with the rest of the dotcom bubble, and thereby corroborated the monetary value of exploiting Apache.

Direct observation of several worms exploiting Apache web servers combined with reasonable after-the-fact reasoning about illegal content hosting determine that the Validation phase was established by the end of the 2002 calendar year.

3) *Escalation*: Public reporting in June 2002 revealed that the Scalper worm had the potential to set up a back-door component that allowed for remote control of the worm [90, 84]. There are no reports of widespread attacks by the worm, so it is deduced that this is partial evidence of "Remote/Automated Control."

In January 2008, reports emerged citing a mass breach of Apache web servers [91, 92]. A watering-hole attack took place, where the altering of web server processes had infected websites to spread malware to site visitors. In addition, April 2011 marked the estimated start of the Darkleech campaign of attacks [93]. These two examples show "Well-Funded, Organized Actors."

Efforts of the adversaries behind the Darkleech campaign targeted Apache web servers numerous times. In August 2011, their Apache Killer malware was discovered, which included a zero-day exploit [94]. This event shows evidence of "Indefensible Attacks."

In March 2013, another watering-hole style delivery of attack was reported under the name Cdorked.A [95]. The malware had remote command and control capability, so it indicates additional evidence of "Remote/Automated Control," establishing the Escalation phase.

4) *Democratization*: The state of adversarial capability against Apache web servers is approaching the Democratization phase; however, at this time, no symptoms have been observed. At the same time, open source attacks are currently taking place that leverage the way Apache handles certain processes to capitalize on vulnerabilities in other technologies, as in the case of Shellshock exploits [96]. For these reasons, we predict that multiple symptoms of the Democratization phase for Apache web servers will occur before the end of 2016.

D. Industrial Control Systems

The case analysis for capability growth against industrial control systems (ICS) focuses on the control of a class of system after exploitation has occurred, as opposed to modeling the capability of exploitation itself. Given this, the final case study represents a departure from the previous three models and demonstrates the flexibility of the ACC model.

Global adversarial capability against ICS and programmable logic controllers (PLCs) is considerably less advanced than that against Android user-facing API and Windows PCs. ICS life cycle is similar to Windows PCs around 2003. ICS malware is still largely the purview of nation-state adversaries, and quality exploits are expensive. Moreover, expertise is difficult and expensive to obtain. Still, adversarial capability has grown, but it is difficult to measure since activity is hidden, less common, and rarely publicly reported. Unfortunately, ICS devices are generally hopelessly vulnerable if targeted, and the patch rate is slow to nonexistent [97]. Thus, adversaries are expected to progress in much the same manner, developing specialized capabilities against particular targets of value. The expertise for attacking ICS will grow slowly until it becomes more publicly accessible, potentially as nation-state employees move into private enterprise and disseminate their expertise to the highest bidder. The ICS capability chain may be presently turning this corner, and as such vulnerability marketplaces for ICS are coming into existence [98].

1) *Discovery*: January 2008 marked the first reported vulnerability in an ICS: a buffer overflow vulnerability in ICS monitoring and control software with potential for disruption to operations [99]. This event evidences the “First Vulnerability & Exploit” symptom.

Reports suggest that the Stuxnet attack was in operation as far back as June 2010. This event spans more than one phase, given its specific focus and sophistication. In *Discovery*, it indicates “Targetability” because of its ability to infect a variety of remote systems automatically, and thus indicates genuine threats to ICS [100].

2) *Validation*: The intent and result of the Stuxnet attack was to damage machines at a nuclear material processing facility. Thus, this event also evidences the “Damage” symptom in the *Validation* phase. A report in May 2014 highlighted two cases of compromise to public utilities that occurred in the first quarter of that year [101]. Both incidents indicated compromise to control systems’ networks, and together they reveal the symptom of “Espionage.”

3) *Escalation*: ICS control has not evidenced any symptoms of the *Escalation* or subsequent phases.

VII. CONCLUSION AND FUTURE WORK

While Windows XP has completed its life cycle and Android user-space adversarial capability is rapidly approaching the Ubiquity phase, the majority of technologies for which the ACC model can be applied are at a point where adversarial capability has been publicly demonstrated. Yet malicious software is not open source or commonly available. As new technologies emerge, the accessibility and popularity of a technology directly relates to the speed of travel through the model. The heuristic correlation between market share and progression through the ACC in the

diagrams in the appendix demonstrates this. However more empirical work is needed.

The ACC is an important advancement in tools for planning CND capabilities. Not only does the ACC help inform current defense resourcing needs, but it helps CND plan for one, two, and five years into the future as we begin to get a feel for adversarial capability against a software system—where it is now, and how quickly it can progress. Understanding these broad-scope trending predictions of the future has been a sore point for CND analysis. The ACC can help address this in a principled and evidence-based manner.

More research is needed to complete assessments of the state of the ACC for other software systems to have a more thorough understanding of the landscape.

ACKNOWLEDGMENT

This material is based upon work funded and supported by Department of Homeland Security under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

This material has been approved for public release and unlimited distribution.

Carnegie Mellon® and CERT® are registered marks of Carnegie Mellon University. DM-0002036

REFERENCES

- [1] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, “Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains,” *Leading Issues in Information Warfare & Security Research*, vol. 1, p. 80, 2011.
- [2] R. N. Giere, “How models are used to represent reality,” *Philosophy of science*, vol. 71, no. 5, pp. 742–752, 2004.
- [3] E. Hatleback and J. M. Spring, “Exploring a mechanistic approach to experimentation in computing,” *Philosophy & Technology*, vol. 27, no. 3, pp. 441–459, 2014.
- [4] R. A. Caralli, J. H. Allen, and D. W. White, *CERT Resilience Management Model: A Maturity Model for Managing Operational Resilience*. Addison-Wesley Professional, 2010.
- [5] M. G. Morgan, *Risk communication: A mental models approach*. Cambridge University Press, 2002.
- [6] J. M. Spring, “Modeling malicious domain name take-down dynamics: Why eCrime pays,” in *IEEE eCrime Researchers Summit*, Anti-Phishing Working Group, September 17, 2013.

- [7] J. M. Spring, "Toward realistic modeling criteria of games in internet security," *Journal of Cyber Security & Information Systems*, vol. 2, no. 2, pp. 2–11, 2014.
- [8] L. B. Metcalf and J. M. Spring, "Blacklist ecosystem analysis update: 2014," Tech. Rep. CERTCC-2014-82, Software Engineering Institute, CERT Coordination Center, Pittsburgh, PA, December 2014.
- [9] P. A. Watters, S. McCombie, R. Layton, and J. Pieprzyk, "Characterising and predicting cyber attacks using the cyber attacker model profile (camp)," *Journal of Money Laundering Control*, vol. 15, no. 4, pp. 430–441, 2012.
- [10] X. Qin and W. Lee, "Attack plan recognition and prediction using causal networks," in *Computer Security Applications Conference, 2004. 20th Annual*, pp. 370–379, IEEE, 2004.
- [11] G. Szappanos, "Exploit this: Evaluating the exploit skills of malware groups," 2015.
- [12] J. D. Howard and T. A. Longstaff, "A common language for computer security incidents," Tech. Rep. SAND98-8667, Sandia National Laboratories, October 1998.
- [13] S. Caltagirone, A. Pendergast, and C. Betz, "The diamond model of intrusion analysis," tech. rep., Center for Cyber Intelligence Analysis and Threat Research, 2013.
- [14] S. Dalal and R. S. Chhillar, "Case studies of most common and severe types of software system failure," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 8, pp. 341–347, 2012.
- [15] V. S. Chomal and J. R. Saini, "Cataloguing most severe causes that lead software projects to fail," *International Journal on Recent and Innovation Trends in Computing and Communication*, pp. 1143–1147, May 2014.
- [16] I. Sommerville, *What Is Software*. International Computer Science Series, 2007.
- [17] Microsoft, *Install or Uninstall Internet Explorer*.
- [18] Y. Yuval, "Method of relocating the stack in a computer system for preventing overrate by an exploit program," Sept. 7 1999. US Patent 5,949,973.
- [19] Microsoft, "A detailed description of the data execution prevention (dep) feature in windows xp service pack 2, windows xp tablet pc edition 2005, and windows server 2003," 2013. Article ID: 875352.
- [20] MITRE, "Common weakness enumeration: A community-developed dictionary of software weakness types." <http://cwe.mitre.org>, April 2, 2014.
- [21] R. S. Ross, "Guide for conducting risk assessments," tech. rep., National Institute of Standards and Technology, 2012.
- [22] Oxford English Dictionary, "material." <http://www.oed.com/view/Entry/114923>, 2014.
- [23] Lyle, "Windows xp market share." <http://techtalk.pcpitstop.com/2005/12/27/windows-xp-market-share>, 2005.
- [24] OneStat, "Microsoft's global usage share." http://www.onestat.com/html/aboutus_pressbox58-microsoft-windows-vista-global-usage-share.html, 2008.
- [25] Gartner, "Desktop operating system market share." <http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0>, November 2014.
- [26] "Net applications." <http://netapplications.com>, 2011.
- [27] Canalys, "Google's android becomes the world's leading smart phone platform." <http://www.canalys.com/newsroom/google/%E2%80%99s-android-becomes-world%E2%80%99s-leading-smart-phone-platform>, January 2011.
- [28] Gartner, "Gartner says worldwide mobile device sales grew 12.8 percent in second quarter of 2010." <http://www.gartner.com/newsroom/id/1421013>, August 2010.
- [29] J. Yarow, "It's official: Apple is just a niche player in smartphones now." <http://www.businessinsider.com/android-market-share-2012-11>, November 2012.
- [30] Strategy Analytics, "Wireless smartphone strategies reports." <http://www.strategyanalytics.com/default.aspx?mod=saservice&a0=91&m=5#1>, 2014.
- [31] Netcraft, "May 2014 web server survey." <http://news.netcraft.com/archives/2014/05/07/may-2014-web-server-survey.html>, 2014.
- [32] R. MacManus, "Microsoft has 97% of os market, says onestat.com." <http://www.zdnet.com/blog/web2explorer/microsoft-has-97-of-os-market-says-onestat-com/262>, August 2006.
- [33] S. V. Hernan, "Vulnerability Note VU#411059: Microsoft windows universal plug and play (upnp) fails to limit the data returned in response to a notify message," tech. rep., Software Engineering Institute, 2001.
- [34] SecuriTeam, "Upnp exploit code released." <http://www.securiteam.com/exploits/SSP011560G.html>, January 2002.
- [35] F-Secure, "Threat description: Worm: W32/goner." <https://www.f-secure.com/v-descs/goner.shtml>, 2001.
- [36] S. Hilley, "Case analysis of the shadowcrew carding gang," *Computer Fraud & Security*, vol. 2006, no. 2, p. 5, 2006.
- [37] R. Howard, *Cyber Fraud: Tactics, Techniques and Procedures*. CRC Press, 2009.
- [38] B. Krebs, "Mapping the russian business network," *Washington Post*, October 2007.
- [39] C. Dougherty, J. Havrilla, S. Hernan, and M. Lindner, "W32/blaster worm," 2003.





- [40] N. Thornburgh, "The invasion of the chinese cyberpies (and the man who tried to stop them)," *Time Magazine*, 2005.
- [41] United States Department of the Army, "Memo: Army golden master waiver process," 2006.
- [42] FBI, "The case of the "Zombie King": Hacker sentenced for hijacking computers for profit," 2006.
- [43] R. Lemos, "Suspected bot master busted." <http://www.securityfocus.com/news/11353/1>, November 2005.
- [44] B. Stone-Gross, "The lifecycle of peer-to-peer (gameover) zeus," tech. rep., Dell Secure Works, 2012.
- [45] B. Krebs, "Hacking made easy," *Washington Post*, March 2006.
- [46] D. Shick and A. Horneman, "Investigating advanced persistent threat 1 (apt1)," tech. rep., CERT Division Software Engineering Institute, 2014.
- [47] Mandiant, "Apt1: Exposing one of china's cyber espionage units," 2013.
- [48] B. Krebs, "Shadowy russian firm seen as conduit for cybercrime," *Washington Post*, October 2007.
- [49] J. Noble, "Media alert: Rsa afcc detects 'All-in-one' zeus trojan package for sale," *CIO Magazine*, 2008.
- [50] L. Dignan, "The next big thing? crimeware-as-a-service," *ZDNet*, April 2008.
- [51] J. E. Dunn, "Spyeye trojan stole \$3.2 million from us victims," *Tech World*, September 2011.
- [52] K. Burton, "The conficker worm," 2010.
- [53] A. Gololobov, "Blackhole exploit kit," *Web Sense*, February 2011.
- [54] A. Larson and E. Gonzalez, "Six months after blackhole: Passing the exploit kit torch," 2014.
- [55] P. Kruse, "Complete zeus sourcecode has been leaked to the masses," *CSIS*, 2011.
- [56] D. Fisher, "Zeus source code leaked," *Threat Post*, May 2011.
- [57] jcran, "20111205000001." <https://github.com/rapid7/metasploit-framework/releases/tag/20111205000001>, December 2011.
- [58] K. O'Flaherty, "The unlocked door: End-of-support for windows xp," *SC Magazine*, May 2014.
- [59] C. Boulton, "Google scrambles to patch buffer overrun exploit in android g1," *EWeek*, October 2008.
- [60] R. Naraine, "Google patches android dos vulnerabilities," *ZDNet*, October 2009.
- [61] Y. Liu, "Symbos.comwarrior.a," 2005.
- [62] A. Greenberg, "Google gets serious about android security, now auto-scans app market for malware," *Forbes*, February 2012.
- [63] A. Greenberg, "When angry birds attack: New android bug lets spoofed apps run wild," *Forbes*, 2010.
- [64] ShmooCon11, *Team JOCH vs. Android: The Ultimate Showdown*, 2011.
- [65] C. Miller and C. Mulliner, "#2009-014 android denial-of-service issues," 2009.
- [66] M. Kumar, "Andosid the dos tool for android," *Hacker News*, 2011.
- [67] Kaspersky Lab, "First sms trojan detected for smartphones running android," 2010.
- [68] Info Security, "Droiddream trojan is a nightmare for thousands of android users," *Info Security*, March 2011.
- [69] T. Strazzere, "Update: Android malware droiddream: How it works," March 2011.
- [70] X. Jiang, "Security alert: New android malware gold-dream found in alternative app markets," 2011.
- [71] X. Jiang, "Security alert: New sophisticated android malware droidkungfu found in alternative chinese app markets," 2011.
- [72] F-Secure, "Trojan: Symbos/spitmo.a," 2011.
- [73] P. Roberts, "Zeus banking trojan comes to android phones," *Threat Post*, July 2011.
- [74] P. Roberts, "Zitmo hits android," July 2011.
- [75] "Issue 21681: Cve-2011-3874 -libsutils rooting vulnerability (zergrush)," 2011.
- [76] A. Lelli, "Remote access tool takes aim with android apk binder," *Symantec*, 2013.
- [77] A. Neville, "Android.dandro," 2013.
- [78] A. Gupta, "Android framework for exploitation," *Club Hack Mag*, June 2013.
- [79] XYSEC Labs, "Android framework for exploitation, is a framework for exploiting android based devices," January 2013.
- [80] mudge, "test-cgi vulnerability," April 1996.
- [81] CERT-SEI, "Vulnerability in ncsa/apache cgi example code."
- [82] P. V. Allman, M. and J. Terrell, "A brief history of scanning," 2007.
- [83] G. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide To Network Discovery And Security Scanning*. Nmap Project, 2011.
- [84] F-Secure, "Threat description: Scalper," 2002.
- [85] F-Secure, "Threat description: Worm: Linux/slapper," 2002.
- [86] Gobbles, "apache-nosejob.c -now with freebsd & netbsd targets," 2002.
- [87] B. McWilliams, "Gobbles releases apache exploit," *Security Focus*, June 2002.
- [88] T. Moore and R. Clayton, "Evil searching: Compromise and recompromise of internet hosts for phishing," *Financial Cryptography and Data Security*, pp. 256-272, 2009.
- [89] J. Deahl, "Websites' servers hacked to host child abuse images," *BBC News*, August 2013.
- [90] McAfee for Consumer, "Virus profile: Bsd/scalper.worm," 2002.
- [91] J. Barr, "Mystery infestation strikers linux/apache web sites," January 2008.

- [92] D. Jackson, "Dell SecureWorks Discovers Protection Against Massive Website Attack Infecting 10,000 Linux/Apache Servers," 2008.
- [93] Symantec, "Trojan.apmod," 2011.
- [94] M. Prince, "Apache Killer Killed: Zero Day Exploit, Zero Day Fix," August 2011.
- [95] P.-M. Bureau, "Linux/cdorked.a: New apache back-door being used in the wild to serve blackhole," April 2013.
- [96] D. Desaie, "Shellshock attacks spotted in wild," September 2014.
- [97] Jackson Higgins, Kelly, "The scada patch problem," *Dark Reading*, January 2013.
- [98] A. Manion, "A survey of vulnerability markets," 2014.
- [99] C. Taschner, "Vulnerability Note VU#308556," 2008.
- [100] N. Falliere, L. OMurchu, and E. Chien, "W32.stuxnet dossier," 2011.
- [101] ICS-CERT, "Internet accessible control systems at risk." https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_\%20Jan-April2014.pdf, 2014.




APPENDIX

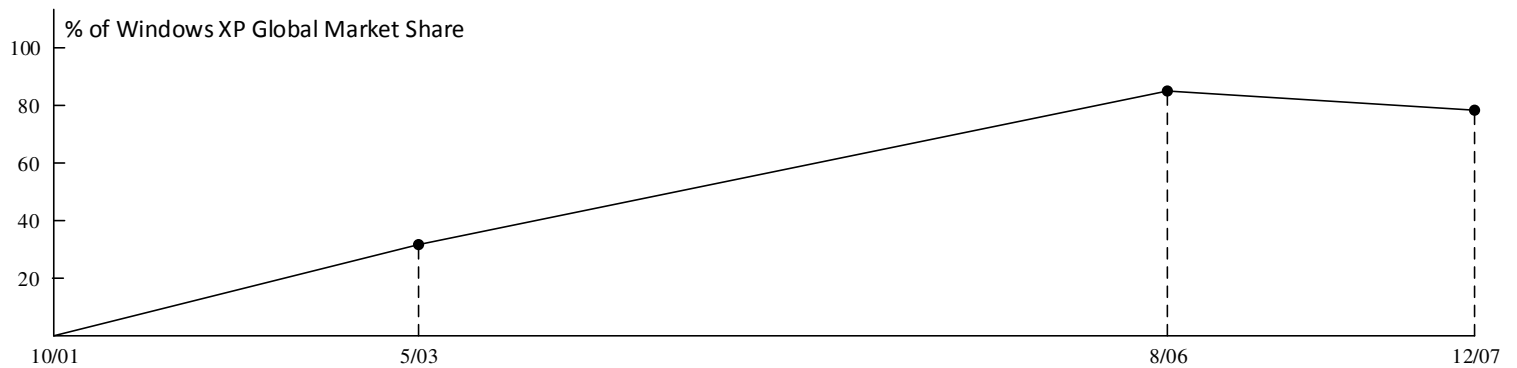
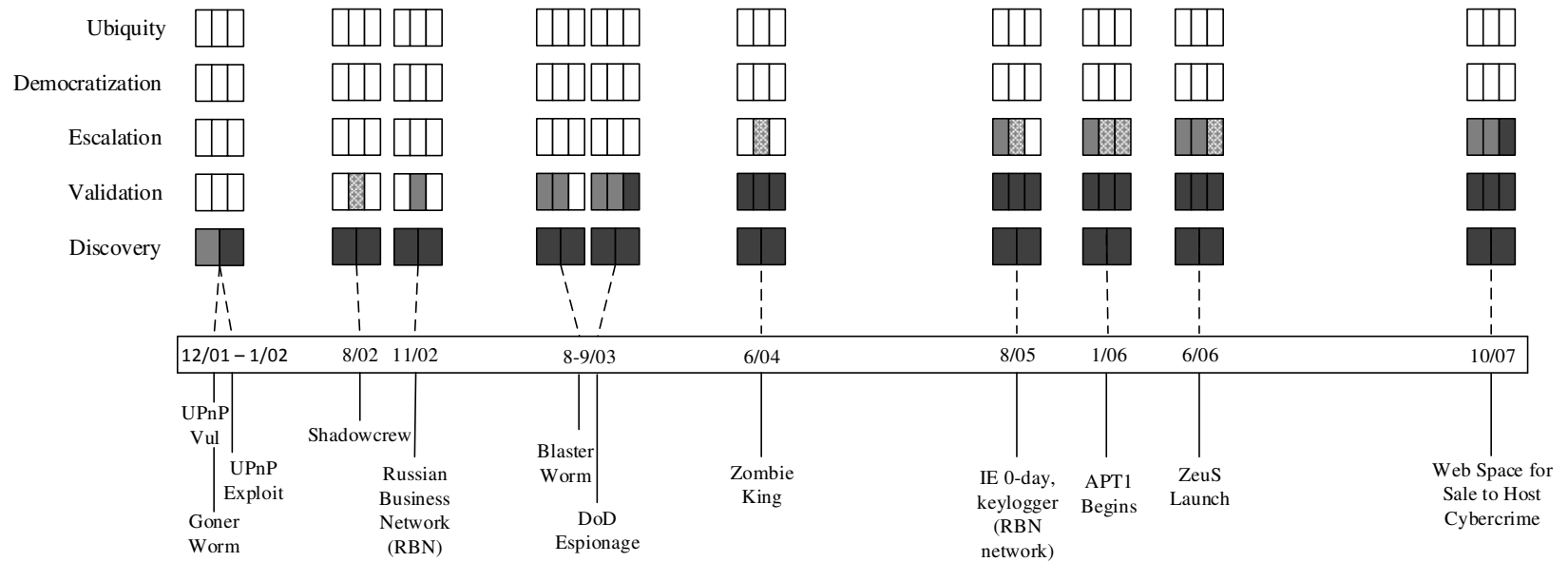
The following diagrams illustrate the phases and symptoms of the Adversarial Capability Chain models for Windows XP, Android user-space, Apache web servers, and ICS. The symptoms for each phase are illustrated as follows:

Table II
COLOR-CODING KEY FOR TIMELINES OF ACC SYMPTOMS




-  White fill indicates that evidence of this symptom has not been observed yet.
-  Dotted gray fill indicates partial evidence of this symptom. (This is often skipped.)
-  Solid gray fill indicates significant evidence of this symptom.
-  Dark gray fill indicates that there is significant evidence of this symptom, and all other symptoms in the phase have been clearly observed as well. All symptoms of an established phase then have dark gray fill.

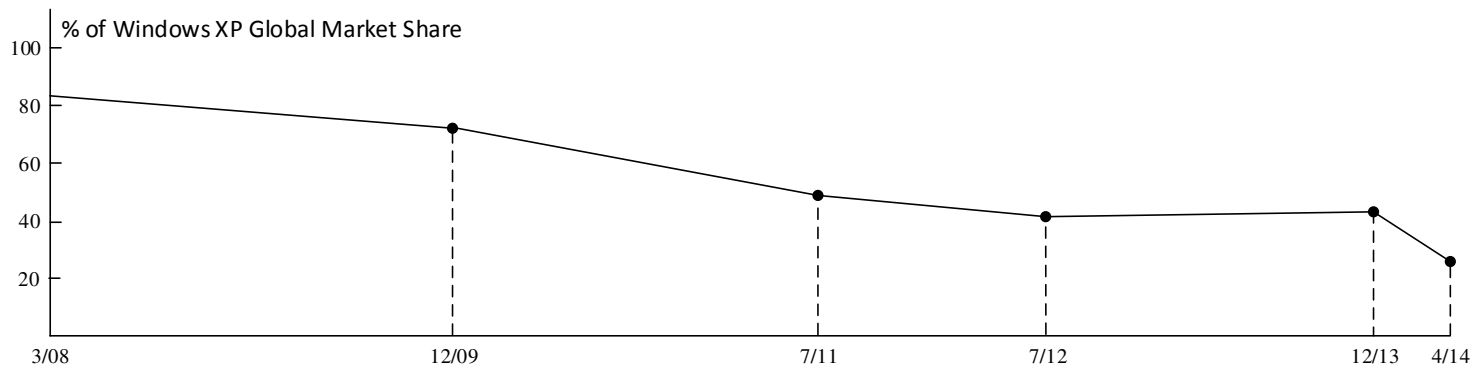
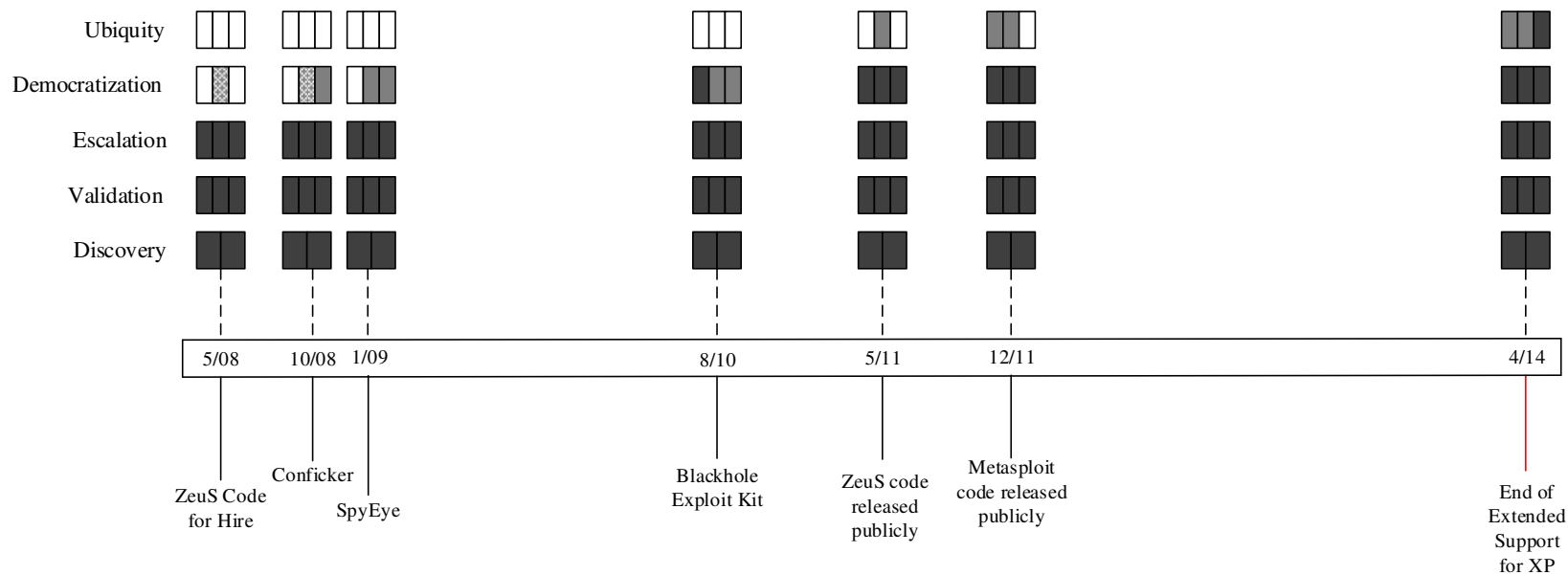
XP Adversarial Capability Timeline 2001 - 2008

Key  Partial evidence  Significant evidence  Phase established






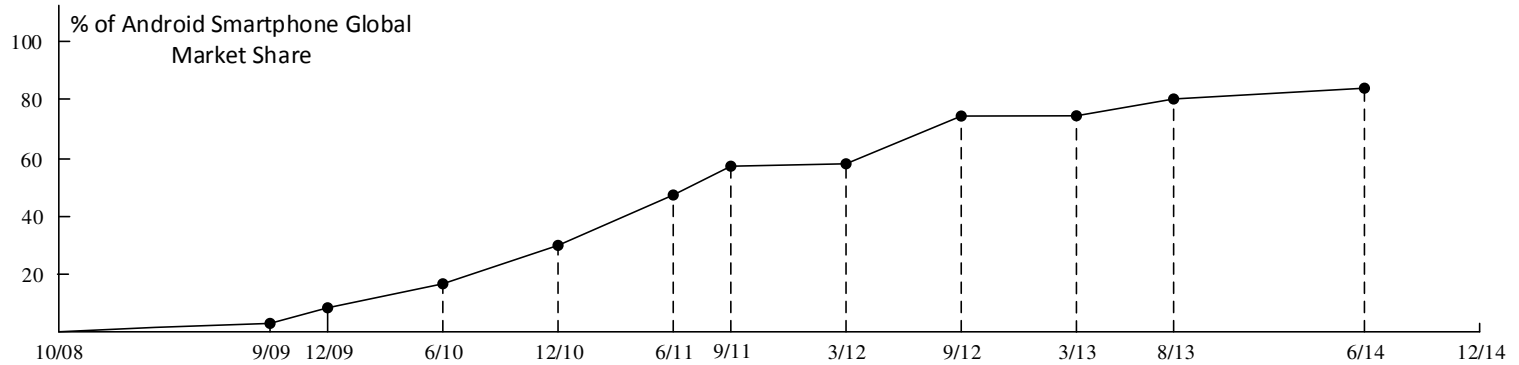
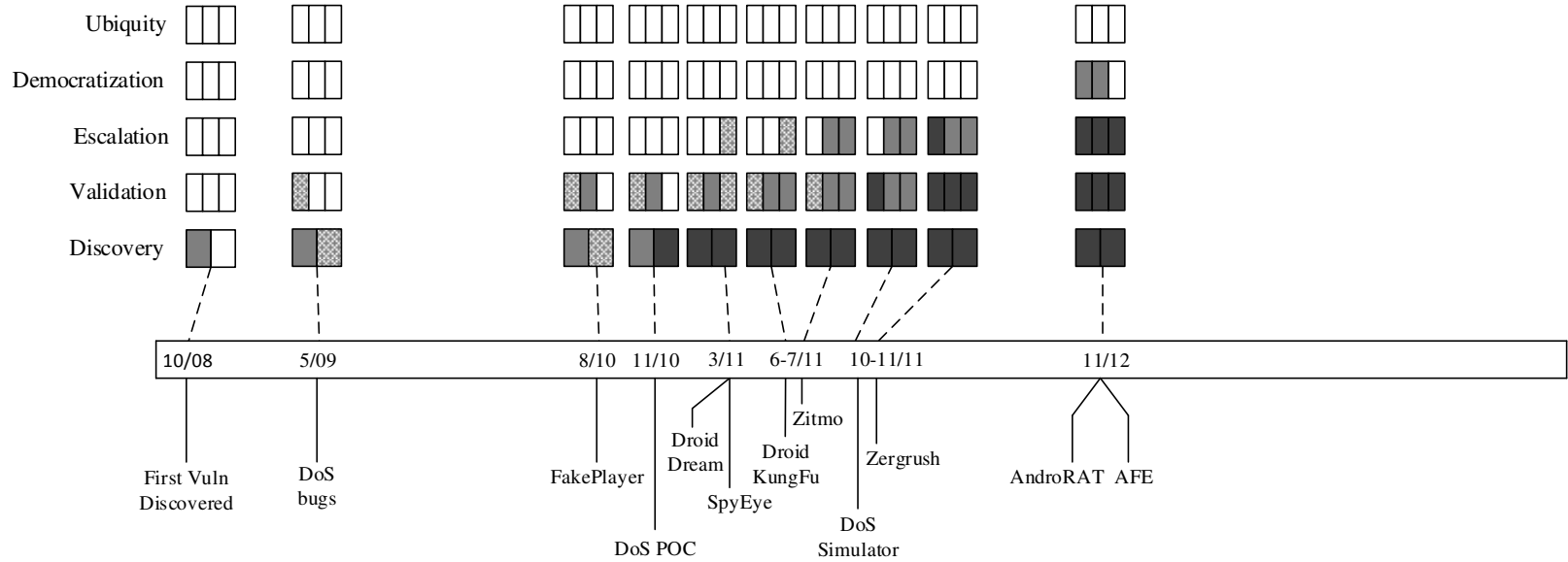
XP Adversarial Capability Timeline mid 2008 – mid 2014

Key  Partial evidence  Significant evidence  Phase established






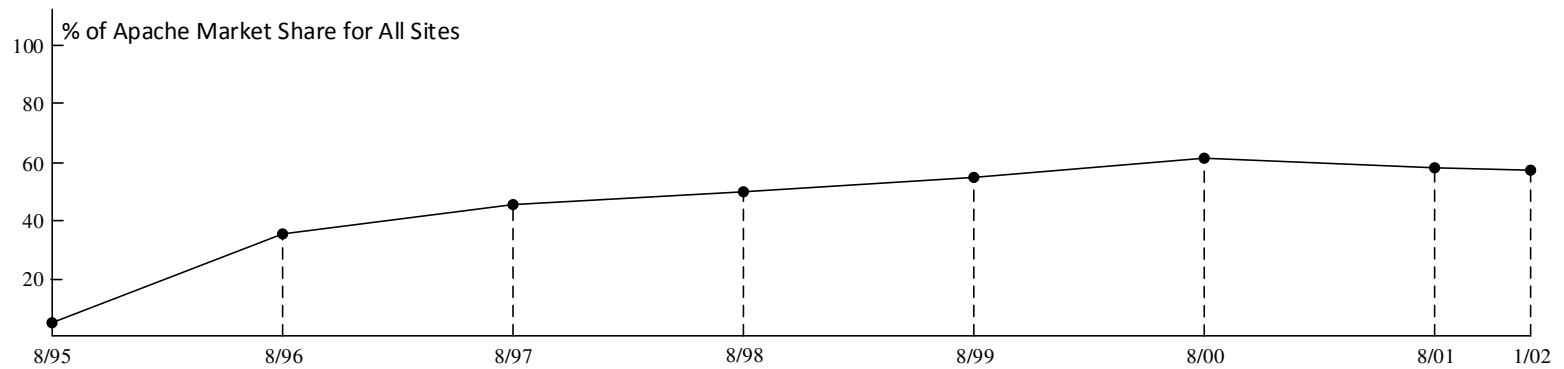
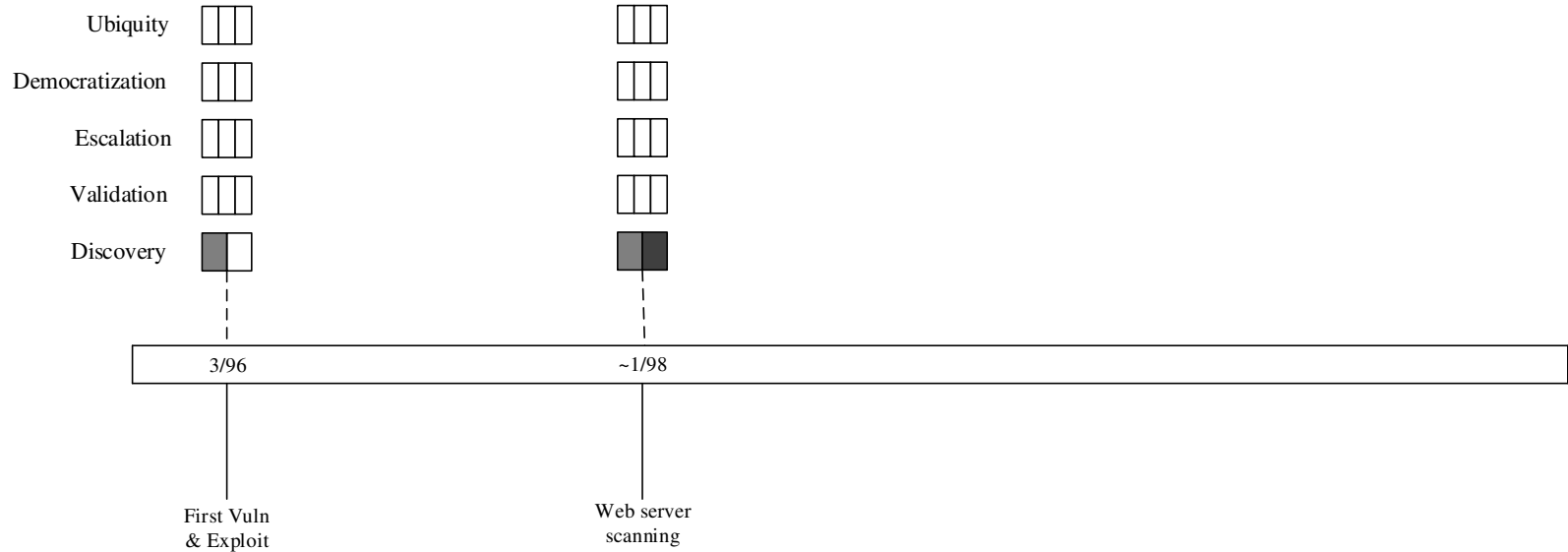
Android Adversarial Capability Timeline 2008 – mid 2014

Key  Partial evidence  Significant evidence  Phase established






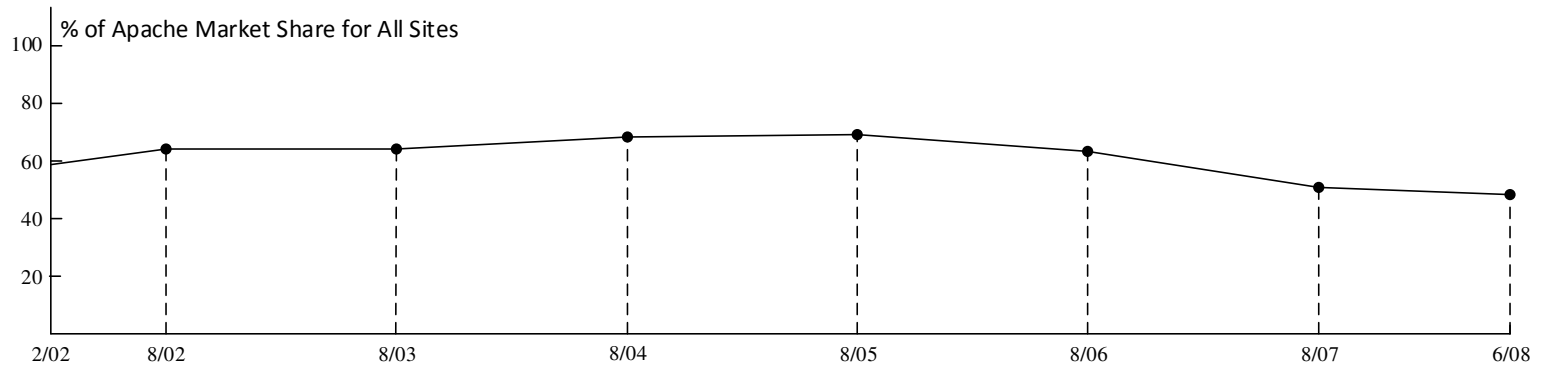
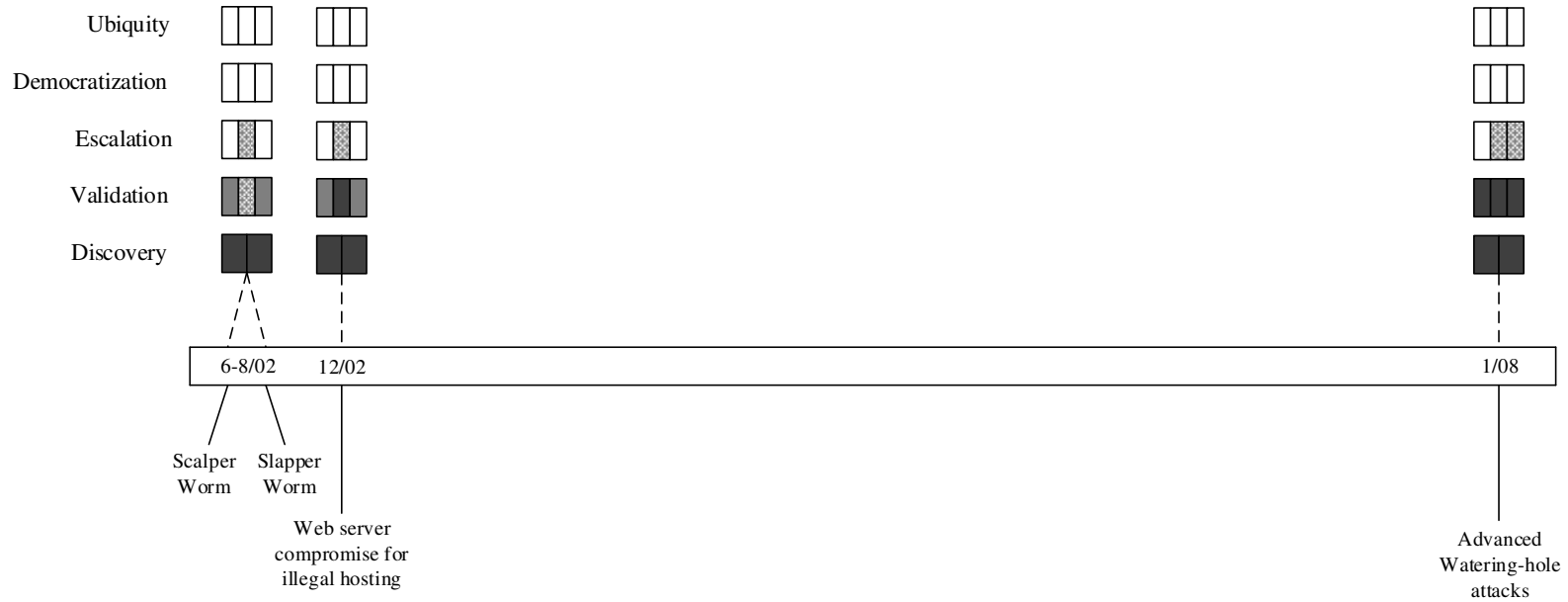
Apache Adversarial Capability Timeline mid 1995 - 2002

Key  Partial evidence  Significant evidence  Phase established






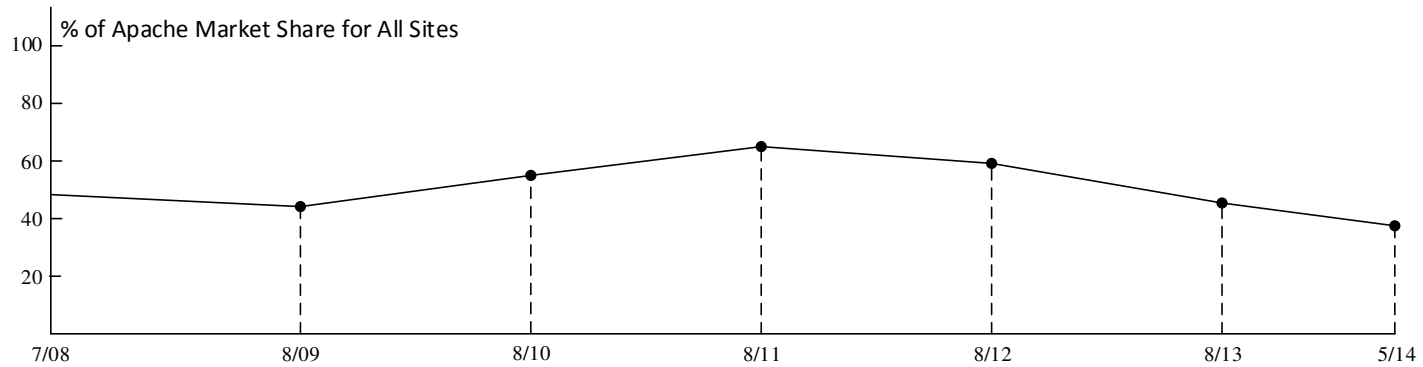
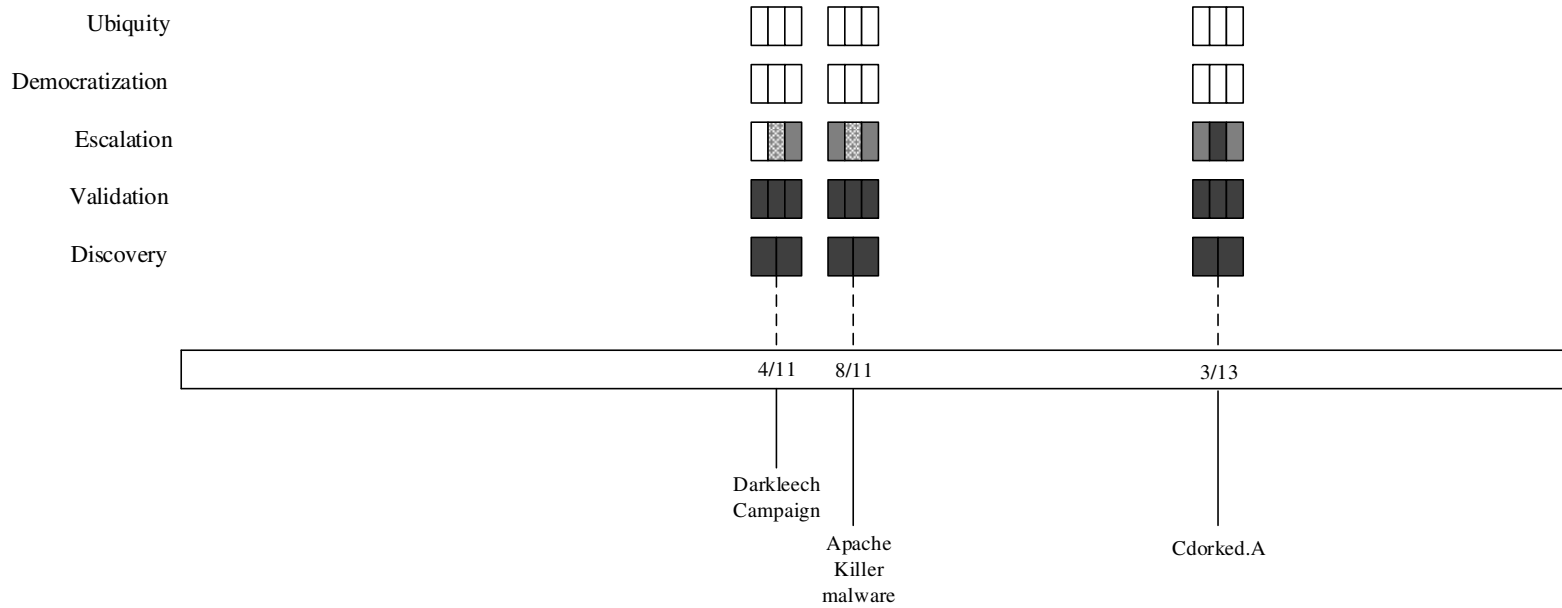
Apache Adversarial Capability Timeline 2002 – mid 2008

Key  Partial evidence  Significant evidence  Phase established






Apache Adversarial Capability Timeline mid 2008 – mid 2014

Key  Partial evidence  Significant evidence  Phase established



ICS Adversarial Capability Timeline 2008 – mid 2014

Key  Partial evidence  Significant evidence  Phase established

